IEEE SA
STANDARDS
ASSOCIATION

IEEE Standard for an Architectural Framework for the Internet of Things (IOT)

STANDARDS

IEEE Computer Society

Developed by the
IEEE SA Board of Governors/Corporate Advisory Group (BoG/CAG)

IEEE Std 2413™-2019

IEEE

IEEE Standard for an Architectural Framework for the Internet of Things (IoT)

Developed by the

IEEE SA Board of Governors/Corporate Advisory Group (BoG/CAG)
of the
IEEE Computer Society

Approved 21 May 2019

IEEE SA Standards Board

Abstract: An architecture framework description for the Internet of Things (IoT) which conforms to the international standard ISO/IEC/IEEE 42010:2011 is defined. The architecture framework description is motivated by concerns commonly shared by IoT system stakeholders across multiple domains (transportation, healthcare, Smart Grid, etc.). A conceptual basis for the notion of things in the IoT is provided and the shared concerns as a collection of architecture viewpoints is elaborated to form the body of the framework description.

Keywords: architectural framework, IEEE 2413™, Internet of Things (IoT)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-1-5044-5886-3 STD23719
Print: ISBN 978-1-5044-5887-0 STDPD23719

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/ipr/disclaimers.html.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and

adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore at http://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website at http://standards.ieee.org.

Errata

Errata, if any, for IEEE standards can be accessed via https://standards.ieee.org/standard/index.html. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore: https://ieeexplore.ieee.org/browse/standards/collection/ieee/. Users are encouraged to periodically check for errata.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this IEEE standard was completed, the Internet of Things (IoT) Architecture Working Group had the following membership:

Oleg Loginov, *Chair*
Ludwig Winkel, *Vice Chair*

Siby Abraham
Yoshiaki Adachi
Chuck Adams
Roberto Aiello
Jim Allen
Wilson Ang
Masayuki Ariyoshi
Marilyn Arndt
Yasuo Bakke
Martin Bauer
Rajdeep Bhowmik
Zhen Bin
Josef Blanz
Pat Brett
Paul Brooks
Dennis Brophy
Phillip Brown
Timothy Carey
Joseph Carr
Samita Chakrabarti
Rabindra Chakraborty
Lihao Chen
Penny Chen
Howard Choe
Koh Lian Chong
Sim Bak Chor
Andrew Cofler
Graham Cunliffe
Nicolas Damour
Chen De
Jean-Pierre Desbenoit
Wael Diab
Jeff Drake
Sophie Dumas
Ed Eckert
Rouzbeh Farhoumand
Jean Philippe Faure
Jeff Fedders
Ray Forbes
Ingo Friese
Rob Gillan
Tim Godfrey
Tinlin Gug
Sebastian Hans
Xuan He
Bob Heile
Juergen Heiles
SeonMeyong Heo
Betsy Hawkinson
Martin Hoglinger
Thomas Hott
Joel Huloux
Paul Hunkar
Daisuke Ishii
Naoki Ito
Shogo Iwaki
Antonio Jara
Emery Jou
Rajah Kalipatnapu
Jay Kalra
Jeffrey Katz
David Kaufman
Anil Keshavamurthy
Taizo Kinoshita
Douglas Knisely
Soumitri Kolavennu
Semen Kort
Bruce Kraemer
Victor Kueh
Sue Leight
Gadi Lenz
Jidong Li
Jinhui Li
Min Liu
Mark Lynass
Santosh Madathil
Brenda Mancuso
Hiroshi Mano
Hirofumi Masukawa
Tetsushi Matsuda
Toshiko Matsumoto
John Messina
Pascal Moniot
Masaki Nakano
Haewoon Nam
Philippe Nappey
Lee Neitzel
Vien Nguyen
Lei Ning
Hiroaki Nishi
Shinji Oda
Marie-Paule Odini
Ken Ogiso
Nobuyuki Ogura
Richard Oh
François Oudot
Harirajan Padmanabhan
Jing Pan
Sylvain Paineau
Andreas Pfaff
Harish Pillay
Nathan Pocock
Benoit Ponsard
Daniel Popa
Steve Pridie
Buaey Qui
Demir Rakanovic
Imir Rashid
Tobin Richardson
Mika Rissa
Sandra Roggeveen
Eric Rotvold
Ekaterina Rudina
Francesco Russo
Manas Saksena
Rashid Sangi
Jun Sato
Maik Seewald
Yang Shan
Mayank Sharma
Jie Shen
Masahiro Shimohori
Hitoshi Shirakabe
Eric Simmon
Rodney Sloan
Matthew D. Smith
Daniel Smolinski
Joe So
Alexandru Soceanu
Praveena Sridhar
Klaus Starnberger
Gary Stuebing
Padmakumar Subramani
Stefan Svensson
Masaki Tanaka
Mukesh Taneja
Shigeyuki Tani
Antonio Tassone
Pat Thaler
Gilles Thonet
Larry Wagoner
Yasushi Wakayama
Joachim Walewski
Andy Wang
Dun Wang
Guichun Wang
Xiaofeng Wang
Cliff Whitehead
Euntae Won
Chen Wu
Tao Xing
Dayin Xu
Daoyan Yang

Mike Yao
Wei Yong
Nie Yongfeng
Kohei Yoshida
Tao Zhang
Tengji Zhang
Yongjing Zhang
Slava Zolotnikov
Juan Carlos Zuniga

The following members of the entity balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Beckhoff Automation
BII Group Holdings Ltd.
Chaincomp Technologies Co., Ltd.
Cisco Systems, Inc.
Electric Power Research Institute, Inc. (EPRI)
Ericsson AB
Huawei Technologies Co., Ltd.
IoTecha Corp.
Kaspersky Labs Limited
National Taiwan University
Nokia
Panasonic Corporation of North America
Power Plus Communications AG
Rockwell Automation
Schneider Electric
Siemens Corporation
STMicroelectronics
Wi-SUN Alliance
Yokogawa Electric Corporation

When the Standards Board approved this standard on 21 May 2019, it had the following membership:

Gary Hoffman, *Chair*
Ted Burse, *Vice Chair*
Jean-Philippe Faure, *Past Chair*
Konstantinos Karachalios, *Secretary*

Masayuki Ariyoshi
Stephen D. Dukes
J. Travis Griffith
Guido Hiertz
Christel Hunter
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick
David J. Law
Joseph Levy
Howard Li
Xiaohui Liu
Kevin Lu
Daleep Mohla
Andrew Myles
Annette D. Reilly
Dorothy Stanley
Sha Wei
Phil Wennblom
Philip Winston
Howard Wolfman
Feng Wu
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 2413-2019, IEEE Standard for an Architectural Framework for the Internet of Things (IoT).

This document has four logical parts:

- **Part 1: IoT domains**
  - Description of behaviors and applications of IoT domains (see Clause 4)
- **Part 2: Domain commonalities and shared concerns**
  - Discussion of the core characteristics across IoT domains (see Clause 5)
- **Part 3: IoT architecture framework**
  - ISO/IEC/IEEE 42010:2011 conformant description of the architecture framework (see Clause 6)[1]
- **Part 4: Examples of IoT achitectures**
  - Domain-specific ISO/IEC/IEEE 42010:2011 conformant architectural description

[1] Information on references can be found in Clause 2.

Contents

IEEE Standard for an Architectural Framework for the Internet of Things (IoT)

1. Overview

1.1 Scope

This standard defines an architecture framework description for the Internet of Things (IoT). The architecture ontology and methodology of the framework architecture conforms to the international standard ISO/IEC/IEEE 42010:2011.

The architecture framework description is motivated by concerns commonly shared by IoT system stakeholders across multiple domains (transportation, healthcare, Smart Grid, etc.). This standard provides a conceptual basis for the notion of things in the IoT and then elaborates the shared concerns as a collection of architecture viewpoints that form the body of the framework description.

1.2 Purpose

The IoT is predicted to become one of the most significant drivers of growth in many technology markets. The architectural framework defined in this standard will promote cross-domain interaction, aid system interoperability and functional compatibility, and further fuel the growth of the IoT market. The standard will address stakeholder concerns to help them build a connected world that can interoperate while meeting both the needs of enterprises and society for trustworthiness of IoT systems. The adoption of a unified approach to the development of IoT systems will reduce industry fragmentation and create a critical mass of multi-stakeholder activities around the world.

The motivation for the standard as informed by stakeholder concerns and desired outcome is to build stakeholder confidence through the following:

a) Provide a framework for vendors to build conformant, interoperable, secure, IoT systems that can span multiple application domains.

b) Provide a framework for buyers to make comparisons and assessments of such systems.

c) Provide a framework for system designers to accelerate design, implementation, and deployment processes of such systems.

Business goals trigger and frame system concerns and thus the system architecture. In fact, the U.S. National Institute of Standards and Technology (NIST) CPS Public Working Group generated a list of concerns for cyber-physical systems (CPS) as listed in the Framework for Cyber-Physical Systems (NIST SP 1500-201 [B88]). The framework includes a business aspect that defines "concerns about enterprise, time to market, environment, regulation, cost, etc." The concerns in the present document as derived from the NIST concerns incorporate business elements that acknowledge that systems need to be integrated into business processes to successfully meet the business goals. There is no common approach to addressing the business viewpoint and, therefore, this standard does not include a business architecture viewpoint. The relevance of the business goal of value creation and the desire for differentiable value propositions in the market are implied throughout the standard. Business goals are important for any viable system architecture.

This document provides both a normative part (Clause 6) with the architecture viewpoints intended to be the basis for describing the architecture for an IoT system, and an informative part (Clause 7) with examples of architecture descriptions associated with architecture views that conform to the architecture viewpoints.

1.3 Introduction to the Internet of Things (IoT)

Subclause 1.3 provides a high-level discourse of IoT systems. More information on the core concepts of IoT systems are provided in the conceptual viewpoint (see 6.6.2).

In the context of this standard, a *thing* is an IoT component or IoT system that has functions, properties, and ways of information exchange (see Figure 1). Furthermore, the information maintained within the thing, or when being exchanged, is appropriate secured. Note that the information exchange could be very limited and one way (e.g., bar code, visual recognition).

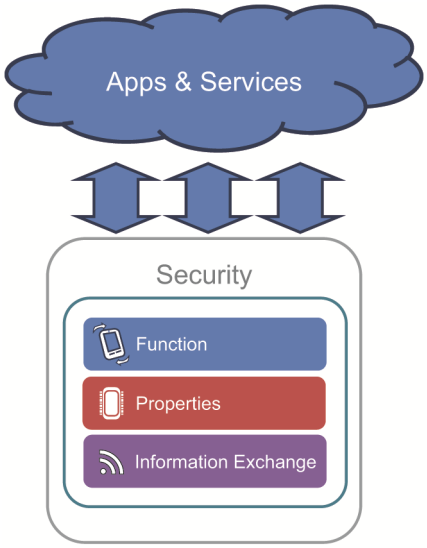


Figure 1—The meaning of *thing* in the context of this standard

It is important to understand what the IoT is and what the difference is between the IoT environment and an IoT system. A simple definition of an IoT system is "a system of entities (including cyber-physical devices, information resources, and people) that exchange information and interact with the physical world by sensing, processing information, and actuating."

The set of IoT components available to be composed into IoT systems, the networks connecting the components, and any associated services that provide the mechanisms for discovery, composition, and orchestration can be called an *IoT environment*. Although an IoT environment contains the IoT components that may be used to create IoT systems, it does not necessarily contain any functioning IoT systems (if no IoT components have been instructed to interact as a system). On a similar note, an IoT environment may be used to create non-IoT systems (conventional IT systems) by excluding IoT components that have sensing or actuating capabilities. In its current state, the Internet may be considered an IoT environment.

At the most fundamental level, IoT systems address the interaction of computing resources with physical entities through sensors and actuators, while an IoT environment is an environment of connected components that can be combined to form IoT systems.

Core to the idea of IoT is this interaction with physical entities. An entity of interest is a physical entity that is of interest to a human or organization for the completion of a goal. A physical entity is a discrete, identifiable part of the physical environment. Physical entities can be any physical object: humans, animals, cars, store or logistics-chain items, mechanical devices, computing devices, etc. These entities range from solid items to bulk matter such as liquids or gases (e.g., air). The sensors and actuators required depend on the type of physical entity and the property of concern.

Interconnected and integrated IoT systems can provide new functionalities to improve the quality of life and to enable technological advances in areas such as personalized healthcare, emergency response, traffic-flow management, manufacturing, defense and homeland security, and energy supply and use. The impacts of IoT will be revolutionary and pervasive; this is already evident in emerging technologies such as autonomous vehicles, Smart Transportation, Smart Logistics, intelligent buildings, Smart Mining, Smart Energy Systems, Smart Manufacturing, multi-purpose robots, Smart Agriculture, Smart Forestry, and Smart Medical Devices.

An IoT system is composed of components that interact with each other to achieve a set of goals.[2] Typically, components are distributed. A component that is a technical artifact and that has an element of computation and interacts with the physical world through sensing and actuation of physical entities with a focus on the physical entity itself, is referred to as *cyber-physical device* (CPS Public Working Group [B20]). Actuation, sensing, and control (using sensing and actuation together) of entities of interest are the fundamental interaction patterns in IoT systems. Examples of other types of devices include dedicated storage devices and networking equipment such as routers, switches, and transceivers.

Cyber-physical devices can be understood as “information transducers,” in that they mediate the translation of physical properties into information by using a function, see function viewpoint in 6.6.7, and vice versa. These devices are part of a common trend toward the “dematerialization” of interactions. These information transducers include both sensors for observing the physical world and actuators for changing the physical world.

The sensing capability of a cyber-physical device captures specific physical states of an entity of interest and translates these states into an estimation of a property. Consider a cyber-physical device that senses light emissions—for example, a light sensor (such as a photodiode or photo resistor) with a bandpass filter. In the presence of an entity giving off light centered at a wavelength of 670 nm, the sensor translates this emission into the observation “*the color of the entity of interest is red*.” The properties of an entity can also be prescribed in a common data dictionary and be used by a normative name to describe aspects of an IoT system and its context, see IEC 61360 [B39].

Information in IoT systems is usually captured in digital form. While that information is necessary for an IoT system to fulfill its goal, it is not sufficient. What is needed is knowledge, i.e., information with context and significance. Definitions of data, information, and knowledge, and how the three relate to each other,

[2] A component may also be considered a system.

are provided in the information viewpoint (see 6.6.6) and the compatibility-level viewpoint (see 6.6.3) to help classify the kind of communication. Information exchange (see Figure 1) demands semantic information of the data on the transmitter and receiver side to interpret the data as information. Some network components, which have the function of forwarding information, do not need to interpret the data in a frame. They are interconnectable in the sense of no need to interpret the frame, but are also interworkable in the sense interpreting the header and tail around a data frame in a communication network.

It is important to understand that digital representations are representations of properties of the physical world, based on some simplification (model) of the world. As such, a digital representation is only an estimation of the property and has errors that may be expressed as an uncertainty (Taylor and Kuyatt [B136]). While in many cases the uncertainty of a given system is small enough that it can be ignored, information uncertainty should be evaluated and understood for every system due to the consequences of failure in an IoT system. This exercise feeds into ascertaining the significance of gathered information. The properties of a thing (see Figure 1) are part of the properties of the physical world.

Once a property has been observed using a sensor and converted into a digital representation, the data/information is processed to create useful knowledge. This is done by processors that combine the sensor data with data from other sensors and *a priori* knowledge stored in information archives or other sources. In the case of complex IoT systems, this is likely a combination of information from a variety of heterogeneous sources. To effectively combine this information, a thorough understanding of the information is required. This includes meta-information about the physical context (time, place, and property) and uncertainties related to the information.

This process of generating knowledge from data/information is complex, and in IoT this often falls under the category of Big Data and analytics. The processing may be done with one processor or may be distributed among many processors.

The action or *actuation* capability converts information into action on a physical entity in the physical world, changing the state of the physical entity. Actuation is the process of effecting change in the physical world. On the one hand, this could be interpreted as the end goal of all IoT observations and knowledge generation, and on the other hand it may be out of the scope of a given IoT system (some knowledge may be stored for later use). As with digital representations, there is a disconnection between the desired change and the actual change. This error shall be accounted for and is often referred to as *tolerance*. In many cases the error is well below a required threshold and can be considered insignificant, but it is important to verify this through measurement or calculation. The confidence about the IoT system's ability to consistently produce the same values and behaviors across different copies of the same system and time is assurance. Depending upon the impact of an IoT system not fulfilling the intended behavior, the justification needed to support that the claimed behavior can be achieved will vary.

For closed-loop control systems, the result of actuation can be sensed to create a feedback based on the error between the observed change and the desired change creating a negative feedback control loop.

The components in an IoT system are typically distributed and can interact by aid of communication networks. While IoT does not have an explicit set of communication protocols, IoT communication is usually characterized by packet-based, end-to-end connectivity that occurs, by default, over ad hoc paths through the network.

Humans (people) assume several roles in IoT systems. They may be developers/builders, defining the goals that an IoT system has to enable. They may be users, using the IoT system to accomplish those defined goals. In addition, they may participate as an element of the IoT system, interacting with other components of the system to achieve the defined goals. Finally, they can be entities of interest themselves.

The person as an element of an IoT system is particularly interesting. People have the ability to sense, act, process, store data, and communicate. In other words, people can be considered a biological part of IoT systems. While people can perform these functions, they do so in a manner significantly different from engineered electronics-based IoT systems. Not only do people sense, process, store data, and act differently

than computer-based IoT, they also exchange information in a different manner. An IoT system with people participating within the system needs to be able to convert machine-type information into a form people can process, and vice versa.

Automobile brakes can be used to illustrate how cyber-physical systems have evolved from traditional mechanical systems. Early automobile brakes used mechanical systems with Bowden cables or linkages between the brake pedal and the brake itself. In most contemporary systems, the mechanical system is replaced by a hydraulic system. However, cyber-physical solutions for braking are currently developed and also in use. In the cyber-physical solution, the engagement of the brake pedal is recorded by a sensor that converts the brake-pedal pressure into an electrical current or voltage. This current or voltage is then converted into digital data through some type of analog-to-digital converter. These data are translated into information, which is communicated to the electronic controller for the brake. The controller then converts the digital information into an analog signal through a digital-to-analog converter that actuates a servo causing the brake to engage. The communication can take place via dedicated communication links or via on-board communication networks. In the case of such an electronic brake system, two information transducers are used in order to achieve the same function as in a purely mechanical brake: the transducer at the pedal senses the engagement of the pedal and turns it into information, while the actuator at the brake turns the received information into action, i.e., the engagement of the brake. Sensing produces data that is translated into information while actuation in the controller is driven by data that is derived from information. This example is focused on an embedded approach, but the approach could as easily be distributed across a network.

Important characteristics of IoT systems include:

- Modularity/interoperability—each component can be connected to other components to form systems
- Agility/dynamic—a system can be quickly modified once it is built and components can be quickly added or removed
- Shared—components can be shared among systems
- Trustworthiness
- Resilience
- Assurance—inspiration of confidence in the performance of the system

These characteristics along with example application domains are addressed within the viewpoints provided in this document.

1.4 Architecture method

This standard adheres to the architecture ontology and methodology outlined in ISO/IEC/IEEE 42010:2011.

ISO/IEC/IEEE 42010:2011 architecture descriptions use viewpoints (framing a group of stakeholders concerns), which are addressed by architecture views. The views are work products that express the architecture of a system from the perspective of a viewpoint. Architecture viewpoints establish “conventions for the construction, interpretation and use of architecture views to frame specific system concerns” in ISO/IEC/IEEE 42010:2011. According to ISO/IEC/IEEE 42010:2011, an architecture framework, as provided in this document, is the collection of viewpoints that frame the concerns of an identified group of stakeholders (see Clause 4, Clause 5, and Clause 6). Architecture examples for a certain system within an application domain are described in Clause 7. Accordingly, the architecture examples in Clause 7 describe specific framed concerns and their stakeholders versus the generic viewpoints in Clause 6. Architecture views that provide specific implementation descriptions are provided in Clause 7.

Figure 2 illustrates the process how the present document was built.

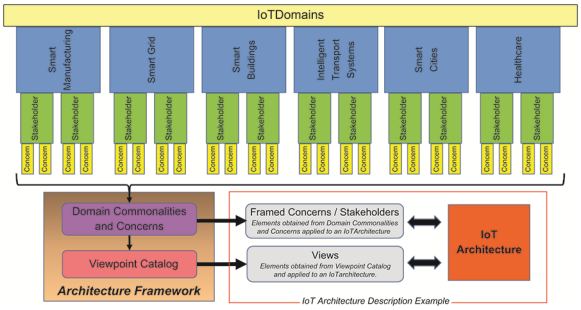


Figure 2—Architecture development process

2. Normative reference

The following referenced document is indispensable for the application of this document (i.e., it must be understood and used, so the referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ISO/IEC/IEEE 42010:2011, Systems and software engineering— Architecture description.[3, 4]

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[5]

Italicized terms are defined within 3.1.

adversary: Individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.

NOTE—See CNSSI No.4009 [B17].[6]

application: Functional unit that is specific to the solution of a problem. Note that an application may be distributed among resources, and may communicate with other applications.

NOTE—See [B44] IEC 61804-2:2016.

[3] The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.
[4] IEEE publications are available from the Institute of Electrical and Electronics Engineers (http://standards.ieee.org/).
[5] *IEEE Standards Dictionary Online* is available at: http://dictionary.ieee.org.
[6] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

architectural framework: See *architecture framework*.

architecture: "Fundamental concepts or properties of a *system* in its environment embodied in its elements, relationships, and in the principles of its design and evolution."

NOTE—See ISO/IEC/IEEE 42010:2011.

architecture description: Work product used to express an *architecture.*

NOTE—See ISO/IEC/IEEE 42010:2011.

architecture framework: Conventions, principles, and practices for the description of *architectures* established within a specific *domain* of application and/or community of stakeholders.

NOTE—See ISO/IEC/IEEE 42010:2011.

architecture view: Work product expressing the architecture of a system from the perspective of specific system concerns.

NOTE—See ISO/IEC/IEEE 42010:2011.

architecture viewpoint: Work product establishing the conventions for the construction, interpretation, and use of architecture views to frame specific system concerns.

NOTE—See ISO/IEC/IEEE 42010:2011.

assurance: Grounds for justified confidence that a claim has been or will be achieved.

NOTE—See ISO/IEC 15026 [B60].

Attack: An attempt to destroy, expose, alter, disable, steal, gain unauthorized access to, or make unauthorized use of, an asset.

NOTE—See ISO/IEC 27000:2016 [B64].

attribute: Property or characteristic of an entity.

NOTE—See IEC 61499-1:2012 [B41].

availability: Property of being accessible and usable upon demand by an authorized entity.

NOTE—See 2.9 of ISO/IEC 27000:2016 [B64].

coexistence: Ability of two or more devices to operate independently of one another in the same network respecting the common rules for sharing the same medium.

NOTE—See IEC 61804-2:2016 [B44].

concern: Interest in a system relevant to one or more of its *stakeholders*.

NOTE—See ISO/IEC/IEEE 42010:2011.

confidentiality: Property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

NOTE—See ISO/IEC 27000:2016 [B64].

configuration (of a system or device): Selecting functional units, assigning their locations, and defining their interconnections.

NOTE—See IEC 61499-1:2012 [B41].

cyber-physical device: "A device that has an element of computation and interacts with the physical world through sensing and actuation."

NOTE—See CPS Public Working Group [B136].

data: Representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human beings or by automatic means.

NOTE—Based on IEC 61499-1:2012 [B41].

data type: A set of values together with a set of permitted operations.

NOTE— See IEC 61499-1:2012 [B41].

dependability: Ability to perform as, and when, required to meet specified communication and operational requirements.

NOTE—See IEC 61907 [B46].

device: Independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces.

NOTE—See IEC 61499-1:2012 [B41].

domain: A particular area of interest based on a one or more viewpoints. In the context of this document the following domains are considered:

- Application domains which define areas of related applications like Smart Grid, Smart Home, and Smart Manufacturing.
- Technology domains which define a specific technology area, such as IoT.

Domains of responsibility which cover areas that are under the responsibility of a specific actor/user such as an end customer, a distribution network operator, or a factory.

entity: A particular thing, such as a person, place, process, object, concept, association, or event.

NOTE—See IEC 61499-1:2012 [B41].

event: Instantaneous occurrence that is significant to scheduling the execution of an algorithm.

NOTE 1—See IEC 61499-1:2012 [B41].

NOTE 2—The execution of an algorithm may make use of variables associated with an event. Occurrence or change of a particular set of circumstances. (See ISO/IEC 27000:2016 [B64])

NOTE 3—An event can be one or more occurrences, and can have several causes.

NOTE 4—An event can consist of something not happening.

NOTE 5—An event can sometimes be referred to as an "incident" or "accident."

exchangeability: Ability of one device to fulfill exactly the same role of another device in the physical process and distributed application as required by the process design

NOTE—See IEC 61804-2:2016 [B44].

function: Intended purpose of an entity or its characteristic action.

NOTE—Based on IEC 61499-1:2012 [B41].

hardware: Physical equipment, as opposed to programs, procedures, rules, and associated documentation.

NOTE—See IEC 61499-1:2012 [B41].

interconnectivity: Ability of two or more devices to operate with one another using the same communications protocols and communication interface.

NOTE—See IEC 61499-1:2012 [B41].

interface: Shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate.

NOTE—See IEC 60050-351:2013 [B37].

Internet: The single, interconnected, worldwide system of commercial, governmental, educational, and other computer networks that share (a) the protocol suite specified by the Internet Architecture Board (IAB) and (b) the name and address spaces managed by the Internet Corporation for Assigned Names and Numbers (ICANN).

NOTE—See CNSSI No.4009 [B58].

Internet of Things (IoT): A system of entities (including cyber-physical devices, information resources, and people) that exchange information and interact with the physical world by sensing, processing information, and actuating.

NOTE—See 1.3.

interoperability: Ability of two or more devices to work together in one or more distributed applications with the shared knowledge of the data types and the semantics of the data transmitted.

NOTE—See IEC 61804-2:2016 [B44].

interworkability: Ability of two or more devices to support the transfer of data between devices with the shared knowledge of the data types of the data transmitted.

NOTE 1—The content of the data is unknown, e.g., a text string that can be displayed but not interpreted by the receiver device itself, or the content is a value but the unit is unknown.

NOTE 2—See IEC 61804-2:2016 [B44].

lifecycle (also: life cycle): Evolution of a system, product, service, project, or other human-made entity from conception through retirement.

NOTE—See ISO/IEC/IEEE 15288 [B66].

model: Representation of a real world process, device, or concept.

NOTE—Based on IEC 61499-1:2012 [B41].

performance: Quantitative or qualitative level of a property at any point in time considered.

NOTE—Based on ISO 15686-1:2011 [B58].

physical entity: "A physical entity is a discrete, identifiable part of the physical environment that is of interest to the user for the completion of her goal. Physical entities can be almost any physical object or environment; from humans or animals to cars; from store or logistics chain items to computers; from electronic appliances to closed or open environments."

NOTE—See Carrez [B14].

privacy: Right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

NOTE—See ISO TS 17574 [B67].

property: Characteristic common to all members of an object class.

NOTE—See ISO/IEC 11179-1 [B63].

Reference model: A "reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete *architectures* using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists."

NOTE—See OASIS [B91].

reliability: The ability of an item to perform a required function under given conditions for a given time interval.

NOTE—See IEC TR 62511 [B51].

resilience: Ability of a system or component to maintain an acceptable level of service in the face of disruption.

NOTE—See IIC Vocabulary 2.0 [B57].

safety: The condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment.

NOTE—See ISO Guide 51 [B65].

security: A condition that results from the establishment and maintenance of protective measures that enable an enterprise to perform its mission or critical functions despite risks posed by threats to its use of information systems. Protective measures may involve a combination of deterrence, avoidance, prevention, detection, recovery, and correction that should form part of the enterprise's risk management approach.

NOTE—See CNSSI No.4009 [B17].

semantics: Relationships between the symbolic elements and their meanings, interpretation, and use.

NOTE—Based on [B38].

service: The means by which the needs of a consumer are brought together with the capabilities of a provider.

NOTE—See OASIS [B91].

stakeholder: "An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system" (ISO/IEC/IEEE 42010:2011).

system: Set of interrelated elements considered in a defined context as a whole and separated from its environment.

NOTE 1—Such elements may be both material objects and concepts as well as the results thereof (e.g., forms of organization, mathematical methods, and programming languages).

NOTE 2—The system is considered to be separated from the environment and other external systems by an imaginary surface, which can cut the links between them and the considered system (based on [B37]).

technical artifact: "Entity that is designed by humans and that has a composition. The composition of the technical artifact enables functions that are accessible to humans and other technical artifacts."

NOTE—See De Vries [B144].

Thing: Any *physical entity* in combination with its digital representation.

NOTE—See Carrez [B14].

threat: Threat is potential cause of an unwanted incident, which may result in harm to a system or organization.

NOTE—See ISO/IEC 27000:2016 [B64].

trustworthiness: Degree of confidence that the system performs as expected with characteristics including safety, security, reliability, and resilience.

NOTE—See IIC Vocabulary 2.0 [B57].

view: See *architecture view*.

viewpoint: See *architecture viewpoint*.

virtual entity: "Computational or data element representing a *physical entity*."

NOTE—See Carrez [B14].

vulnerability: Weakness of an asset or control that can be exploited by one or more threats.

NOTE—See ISO/IEC 27000:2016 [B64].

3.2 Acronyms and abbreviations

ABAC	attribute-based access control
AC	access control

ACL	access control list
ADS	autonomous decentralized system
ADS-net	autonomous decentralized system-network
API	application programming interface
AS	authorization server
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CA	certifying authority
CNSSI	Committee on National Security Systems Instructions
CoAP	Constrained Application Protocol
CPS	cyber-physical system
CPU	central processing unit
CR	correspondence rule
CRC	cyclic redundancy check
DAC	discretionary access control
DIKW	data, information, knowledge, wisdom
DSCP	differentiated services codepoint
EMI	electromagnetic interference
ENISA	European Union Agency for Network and Information Security
EPC	event-driven process chain
EU	European Union
EV	electrical vehicle
EVWPT	electric vehicle wireless power transfer
FB	function block
FoF	factories of the future
GDPR	General Data Protection Regulation
GIS	geography information system
GPS	global positioning system
HMI	human-machine interface
HTTP	hypertext transfer protocol
H-ARC	hardening—adaptivity, responsivity, cooperativity
IAB	Internet Architecture Board
ICANN	Internet Corporation for Assigned Names and Numbers
ICT	information and communication technologies
IdP	identity provider
IDS	intrusion detection system
IEC	International Electrotechnical Commission

IIC	Industrial Internet Consortium
IOC	Intelligent Operation Center
IOI	item of interest
IoT	Internet of Things
IP	Internet Protocol
IPS	intrusion prevention system
ISA	International Society of Automation
ISO	International Organization for Standardization
IT	information technology
ITS	Intelligent Transportation Systems
ITU	International Telecommunication Union
MAC	media access control
MIT	Massachusetts Institute of Technology
MQTT	message queue telemetry transport
MTTR	mean time to repair
NIST	National Institute of Standards and Technology
NPE	non-person entity
OASIS	Organization for the Advancement of Structured Information
OCL	Object Constraint Language
OIX	Open Identity Exchange
OODA	observe, orient, decide, act
OPC	Open Platform Communications
OPC UA	OPC Unified Architecture
OSI	open systems interconnection
OT	operational technology
OWASP	Open Web Application Security Project
OWL	Web Ontology Language
PAP	policy administration point
PCN	Process Control Network
PDCA	plan, do, check, act
PDP	policy decision point
PEP	policy enforcement point
PIN	personal identification number
PIP	policy information point
PKI	public key infrastructure
QoS	quality of service
RADIUS	remote authentication dial-in user service

RBAC	role-based access control
REST	representational state transfer
RFC	Request for Comments
RFID	radio frequency identification
R&D	research and development
SAL	security assurance level
SCADA	supervisory control and data acquisition
SELinux	Security Enhanced Red Hat Linux
SI	International System of Units
SMS	short message service
SNMP	simple network management protocol
SoS	system of systems
SSL	secure sockets layer
ST	structured text
SysML	System Modeling Language
S-ADS	symbiotic-autonomous decentralized system
TACACS	terminal access controller access control system
TCP	transmission control protocol
TOGAF	The Open Group Architecture Framework
TR	technical report
TS	technical specification
TSN	time-sensitive networking
UDP	user datagram protocol
UMA	user-managed access
UML	Unified Modeling Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
UUID	universally unique identifier
WWW	World Wide Web
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

4. IoT domains

4.1 Abstract IoT Domain

The Abstract IoT Domain captures the common characteristics and behaviors across IoT systems.

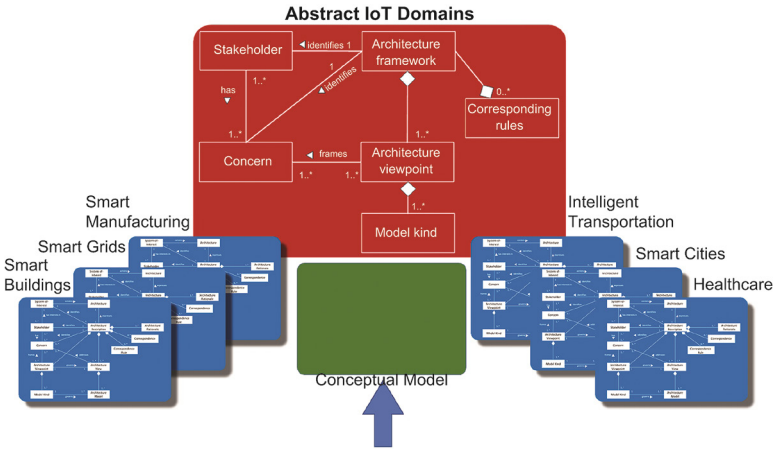


Figure 3—Example of IoT domains

As shown in Figure 3, the Abstract IoT Domain is based on the conceptual model which is the foundation of this architecture framework. The conceptual model informs a key architecture viewpoint, the conceptual viewpoint (see 6.6.2), which clarifies how the IoT is to be understood in the context of the present document.

In the present document, architecture viewpoints are elaborated based on commonalities identified across concrete IoT domains.

As shown in Figure 2, six representative domains are considered: *Smart Manufacturing, Smart Grid, Smart Buildings, Intelligent Transport, Smart Cities,* and *Healthcare*.

The six domains were informally examined with reference to the aspects and concerns of cyber-physical systems identified in NIST SP 1500-201 [B88][7] and the outcomes of that process were used to motivate the architecture viewpoints described in Clause 6.

4.2 Smart Manufacturing

There are several terms commonly used in this area which overlap in scope: Smart Automation, Smart Factory, Intelligent Factory, Smart Manufacturing, and Industry 4.0. Here, the term *Smart Manufacturing* refers to the shared vision of enhanced intelligence, flexibility, and dynamics over entire manufacturing processes and production chains.

According to Bulik [B11]:

The Smart, and mostly digital, factory concept focuses on an integrated planning and monitoring process that includes product design, process planning, and overall integration and implementation of the manufacturing operation, making the manufacturing process more efficient and responsive to change.

All aspects of production from suppliers, logistics, through to the lifecycle management of products are impacted: factory design, production planning, logistics, enterprise resource planning, manufacturing machinery and systems, control technologies, and devices in the field.

[7] The numbers in brackets correspond to those of the bibliography in Annex E.

Further, the devices and computing elements distributed throughout the process chains and production lines have degrees of intelligence and autonomy, are capable of analytics and decision making, and can dynamically modify, guide, and optimize operations in flight.

As with other domains, stakeholders in Smart Manufacturing are a broad and diverse group, including: shareholders, board members, executives, production teams, customers, suppliers, and industry regulators.

Equally broad is the range of benefits that are derived from the approach, including: process optimizations, production optimizations, reduced downtime, reduced material waste, leaner operations, shorter cycle times, reduced energy consumption, reduced greenhouse emissions, and increased worker safety.

Among the key drivers of Smart Manufacturing (and all overlapping terms and initiatives) are energy and resource optimization. Consequently, there are broad sustainability implications which also bring the environment and society in general into the stakeholder group.

Smart Manufacturing and Factories of the Future are components of the European Commission's digital single market strategy (European Commission [B25]):

The European strategy on smart specialization aims to strengthen the competitive advantages of EU regions in terms of ICT skills, R&D capability, industrial output and infrastructures while linking up similar strategies at regional & national levels, and offering incentives for growth and differentiation.

Similar policies are emerging in developed economies globally.

4.3 Smart Grid

The term *Smart Grid* refers to an electricity power infrastructure in which digital computing and communications technologies are deployed to enhance the ability of stakeholders to better understand and control the energy being pushed into the grid, the energy being drawn from the grid, and the state of the grid itself.

The smartness enhances insight into both the grid as a power network and the grid as a system of systems. Enhanced insight improves controllability and predictability, both of which drive improved operation and economic performance and both of which are prerequisites for the sustainable and scalable integration of renewables into the grid and the potential transition to new grid architectures.

The U.S. Department of Energy provides the following definition [B140]:

An automated, widely distributed energy delivery network, the Smart Grid will be characterized by a two-way flow of electricity and information and will be capable of monitoring everything from power plants to customer preferences to individual appliances. It incorporates into the grid the benefits of distributed computing and communications to deliver real-time information and enable the near-instantaneous balance of supply and demand at the device level.

The fundamental intersection of the Smart Grid and IoT is derived from a view of the domain as a collection of interconnected networks of independently addressable objects capable, to varying degrees, of processing, sensing, actuating, and communicating.

Overviews of Smart Grid networks can be found in Garner [B28] and Al-Omar, Al-Ali, Ahmed, and Landolsi [B95]. Smart Grid networks operate over a wide range of technologies and protocols (Al-Ali and Aburukba [B2]) and typically span: home area networks, neighborhood area networks, access networks, backhaul networks, and core and external networks.

In developed economies, the Smart Grid is a ubiquitous infrastructure and most households, commercial concerns, governmental, regulatory, and societal organizations will be stakeholders with dependencies on the grid. Stakeholders vary over regions and between countries depending on the form of regulation, the market, and the level of development of the local physical infrastructure.

Emerging economies share the global drivers of a Smart Grid rollout. More specifically, a Smart Grid helps enable access to electricity for all, reduces transmission and distribution losses, improves quality of power supply, helps peak load management, promotes prosumer concepts, and integrates renewable energy.

Smart Grid benefits spread across a broad spectrum, but generally include improvements in: power reliability and quality, grid resiliency, power usage optimization, operational insights, renewable integration, insight into energy usage, safety, and security.

4.4 Smart buildings

According to RCR Wireless News [B121]:

A smart building is any structure that uses automated processes to automatically control the building's operations including heating, ventilation, air conditioning, lighting, security and other systems. A smart building uses sensors, actuators and microchips, in order to collect data and manage it according to a business' functions and services.

Smart buildings drive benefits for a range of stakeholders including owners, tenants, visitors, caretakers, and the environment. Multiple aspects of such buildings (lighting, heating, air conditioning, safety, security, access, systems, etc.) are managed by autonomous agents (from simple sensors and actuator to sophisticated subsystems) monitoring, analyzing, collaborating, coordinating, and ultimately acting to deliver stakeholder benefits.

As described in Building Efficiency Initiative [B10]:

Smart buildings use information technology during operation to connect a variety of subsystems, which typically operate independently, so that these systems can share information to optimize total building performance. Smart buildings look beyond the building equipment within their four walls. They are connected and responsive to the smart grid, and they interact with building operators and occupants to empower them with new levels of visibility and actionable information.

4.5 Intelligent Transport Systems

4.5.1 General

Intelligent Transport, Smart Transport, and *Smart City Transport* are overlapping terms which refer to the intelligent coordination of assets and information within transport networks. This includes networks and assets which are in public, private, and commercial ownership. This domain also provides an example of the cross-domain nature of IoT as it includes elements of other domains.

Embedding and connecting intelligence across transport systems (both within the environment and within vehicles) will open up possibilities for significant benefits:

— Greater insight for citizens and authorities into the state of traffic and transport within a locality

— Enhanced ability for the transport system to self-heal in response to scheduled and unscheduled events

— More effective exploitation and usage of supporting infrastructure (parking, park-and-ride systems)

Greater transparency on the mobility of citizens and their usage of transport asset drives serious concerns over data confidentiality, data usage, anonymization, and long-term persistency of data. Conversely, it also raises concerns over the ability of such intelligent ecosystems to be subverted by injection of false information and the derivation of misleading analytics.

Intelligent transport is about the networking of assets which are capable of sensing, rationalizing, communicating, and acting on acquired and derived information to provide improved levels of efficiency which deliver, economic, and environmental benefits across society (United Nations Economic Commission for Europe [B139]).

In Intelligent Transportation Systems (ITS), IoT plays a central role in providing passive safety and traceability services for vehicles, bicycles, and pedestrians. The ultimate objective of ITS is to attain sustainable mobility. The system was originally a combination of vehicle navigation systems, electric road pricing, systems for supporting safe driving, traffic management, roadway management, and effective operation of public transportation, emergency vehicles, and pedestrians. In order to achieve these systems and services, a holistic system that integrates environmental and positioning sensors, wireless communication devices, and low-power computational nodes shall be connected to the Internet. In other words, the integration of IoT devices is indispensable. In fact, some such integration has already been established, affecting our daily lives. However, there are still many elements for discussion in the IoT domain of ITS, especially from the viewpoint of introducing electric vehicles (EVs).

4.5.2 EVs as an element of the ITS domain

EVs are more than just a simple transportation means. They are strongly expected to help build strong linkages with roads, grids, and communication systems as illustrated in Figure 4.

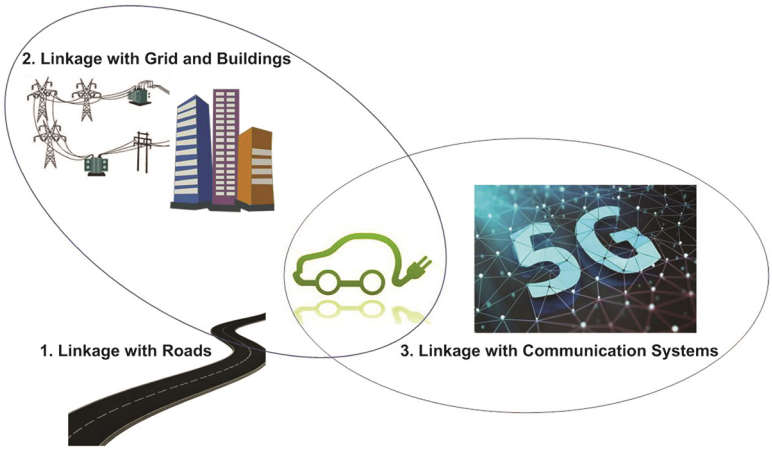


Figure 4—Intelligent Transport Systems overview

First, linkages with roads can support electrical vehicles (EVs) from several different perspectives. Electric vehicle wireless power transfer (EVWPT) is a cutting-edge technology for charging and, in some cases, discharging EV batteries even when traveling along a road, especially in an EV lane. Unfortunately, alignment errors might occur, degrading the efficiency of power transmission and leading to inefficient EV charging and discharging. In these situations, communication and sensing using the IoT can offer one solution for making EV charging and discharging more efficient.

As for the installation cost of EVWPT, battery replacement is considered to be one method of reducing cost because a replaceable battery can leverage IoT sensors for managing battery charging and lifecycle control.

The connection between EVs and roads in ITS can be applied to automatic fare rate management of EV lanes, as this comes with an Internet environment and cloud services. If a floating exchange rate system is applied to electricity prices for EVs, the changing rate would control the timing of EV charging, giving rise to new services.

This linkage also provides various information-based services. Generally, emergency vehicles should be prioritized over others in traffic, while, in general, effective navigation considering driver comfort, energy efficiency, and traffic control is crucial for sustainable mobility in transportation. In meeting these requirements, ITS can control traffic signals as well as navigation systems aboard EVs based on the information obtained by IoT devices. In that sense, IoT devices will certainly play a pivotal role in system control and optimization.

In cases of disasters, EVs with IoT features are of great importance. For example, the remaining battery power of EVs can be used for communication systems, such as smartphones. This can also be an energy resource for any other electrical devices. In evacuation centers, the management of EV discharging can be conducted using IOT devices aboard EVs.

Secondly, linkage with the power grid and micro-grids engenders peak shifting of electricity usage, spinning reserve for quick power recovery from grid failures, and voltage and frequency control for grid stabilization. Additionally, linkage with buildings leads to peak shifting and spinning reserve. The above applications call for automated measurement of electricity usage and device control, which are effectively performed by IoT technologies.

Finally, regarding linkage with communication systems, ITS can be an integral part of the overall system. Cars were once just a means of transportation; however, they are gaining considerable attention currently as communication devices. Communication for cars begins with ITS, as seen in traffic control, in an attempt to reduce congestion, accidents, road closures, and travel time. Furthermore, ITS shall provide other services, such as congestion information on parking lots and rest areas, or tourism information, such as sightseeing and events of interest to tourists. Vehicle-to-vehicle (V2V) and vehicle-to-road (V2R) communication, both of which are based directly on communication with EVs, have recently emerged. Both V2V and V2R enable traffic accident prevention, especially near intersections where pedestrians and bicycles may be involved. These communication forms can also lend themselves to sending control signals to EVs in order to give priority to emergency vehicles. Another form of communication, vehicle-to-home (V2H) and vehicle-to-building (V2B) communication are very promising. Although EV battery deterioration and initial cost should be taken into account, EV batteries can be used for “buying low and selling high” when EVs are connected to a home or building’s energy management system. Another type of application could be a car-sharing service that shares cars for users. IoT technologies can inform users of locations and wait times for each car in a car park. All of these linkages and services with communication systems are supported by IoT itself.

4.6 Smart Cities

4.6.1 General

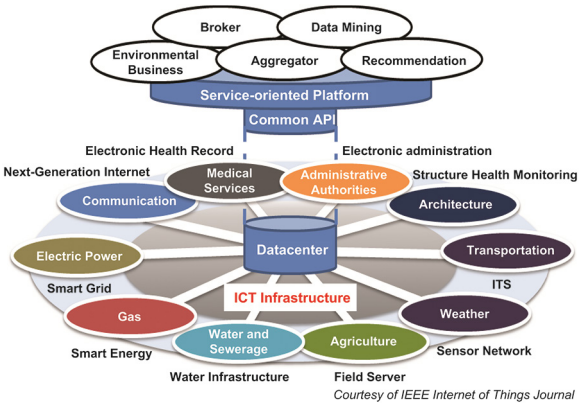
According to Zanella et al. [B147], Smart Cities are about making better use of the public resources, increasing the quality of the services offered to the citizens, and reducing the operational costs of the public administrations. This domain also provides an example of the cross-domain nature of IoT as it includes elements of other domains.

In typical urban environments, such resources and services include transport, parking, lighting, security and surveillance, air quality, maintenance of public areas, preservation of cultural heritage, garbage collection, and many more. Smart Cities therefore have strong synergies and dependencies on other "smart" verticals, such as Intelligent Transport Systems, Smart Buildings, Smart Grid, etc.

There is an expectation that Smart City concepts will alleviate pressures on administrative and operational resources by simplifying access to services, increasing transparency, and enabling efficiencies. Further, there is significant potential for new value to emerge as social networks and city services blend and cross-pollinate to build new digital communities, for example, for education, self-help, and entertainment, to enhance the quality of urban life.

Over the last couple of decades, there has been a trend of strong growth in information, communication, and computation technologies. Modern high-performance computing systems date back to the mid-1990s with cluster computing, the structure of which was one of the main topics of discussion. As its successor, grid computing, has been devised in an attempt to extend its usability. Recent years have seen the development of cloud computing, which has been discussed from the perspective of service provision. On the other hand, the electric power grid, as a typical example of all infrastructure, stems from bulk generation. The micro-grid as a distributed grid added to the debate over grid structure in the early 2000s. The Smart Grid, developed in the 2010s, extends its application range to services for both generators and consumers. Along with the advent of service-aware cloud computing and Smart Grids, the field of communication technology has also advanced without exceptions. Sensor networks were independently constructed in the 1990s for collecting sensing data and issuing commands for device manipulation. They technically form the foundation of IoT, which itself includes the idea of providing services. By means of IoT, the infrastructure in medical services, transportation, agriculture, water and sewerage, and other industries are now transforming into Smart Infrastructure, including Smart Medical Services, Smart Transportation, Smart Agriculture, and Smart Water Infrastructure. Therefore, discussion on IoT could be divided into industrial domains from the service perspective.

The Smart City is one of the IoT domains, and concurrently, it is the integration of Smart Infrastructures in and of itself. The IoT in a Smart City is one of the most important issues of discussion. Currently, a large volume of data is collected and used for services in Smart Infrastructure; however, these data and services are confined within their target Smart Infrastructure. For instance, the data of an electric power grid will not be used in medical infrastructure. The quintessence of a Smart City is the interaction among different types of Smart Infrastructure. Figure 5 typically describes the situation. The integration of Smart Infrastructure creates new services, such as data brokerage, data mining, and recommendation. The IoT in each type of Smart Infrastructure has to work in conjunction with each other when applied to Smart City services, which requires both unified naming and identification method.

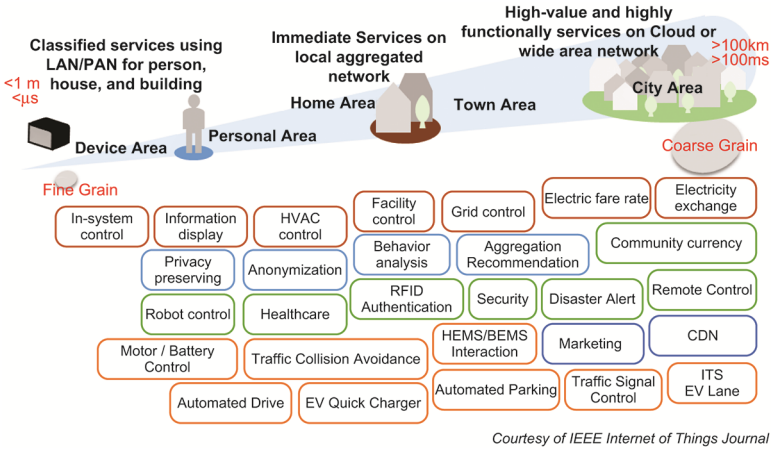


Courtesy of IEEE Internet of Things Journal

Figure 5—Smart City Subdomains

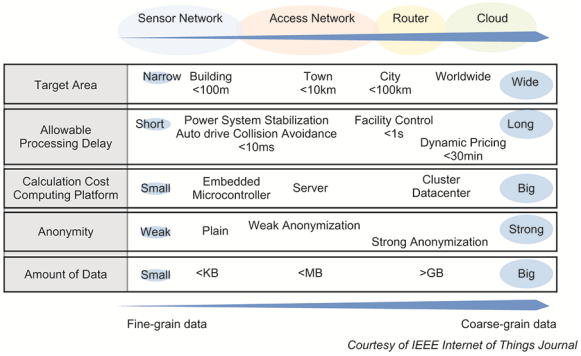
The Smart City is a sustainable living entity whose brain, nervous system, and organs collaborate with each other to sense and improve the physical world. New information and communication technology (ICT)—such as cloud computing, IoT, Big Data, and mobility—will make all things sensible, connected, and intelligent. The technologies aggregate, share, and converge city-wide resources to provide real-time, efficient, and intelligent information services.

To deliver these Smart City services, the IoT system should be integrated into the existing network and computing systems. Figure 6 illustrates the relationship between the typical examples of Smart City services and service providing positions. As shown in Figure 6, each service should be executed in the correct location. In other words, the most suitable processing device should be chosen to execute application software intended for the service. A simple idea is to execute the software in the cloud, but the figure is contrary to this idea and indicates the necessity of rebuilding the network architecture for the IoT era. Figure 7 depicts the relationship between network locations and attributes of service applications. Every service has its optimal execution platform in terms of data size, privacy control, latency, and other attributes defined by network locations. Motor or battery control should be achieved in a dedicated device or machine. Heating, ventilation, and air conditioning (HVAC) control should be executed in each household. If HVAC control is offered as a cloud service, a nonintrusive load-monitoring problem has to be solved, which implies that all the households should be warned of the risks of using cloud services. Electricity exchange service is a kind of aggregation service in which data integration is indispensable. Therefore, it should be offered as a cloud service. All the above things considered, the IoT system needs sharing for its effective use in the coming years. It should also be noted that an IoT system shall be independently shared in order to execute each service separately at its optimized location.



Courtesy of IEEE Internet of Things Journal

Figure 6—Smart City services and data sources



Courtesy of IEEE Internet of Things Journal

Figure 7—Network locations and attributes of service applications

Enabled by the Smart Government, residents can submit service requests through government self-service terminals, service halls, websites, and mobile apps. The Intelligent Operation Center (IOC) can aggregate a wide range of data to visualize the city running status, enable efficient emergency collaboration across agencies, and facilitate decision-making based on Big Data.

Smart Cities need viable platforms to constantly grow and evolve. Ultra-wideband networks, cloud computing, Big Data, and IoT technologies can be used to build open platform for Smart City construction. Building a better Smart City is a common aspiration of governments, telecom operators, telecom equipment suppliers, independent software vendors (ISVs), and residents.

Normally the Smart City ICT infrastructure has the following networks, cloud centers, and platforms as shown in Figure 8.

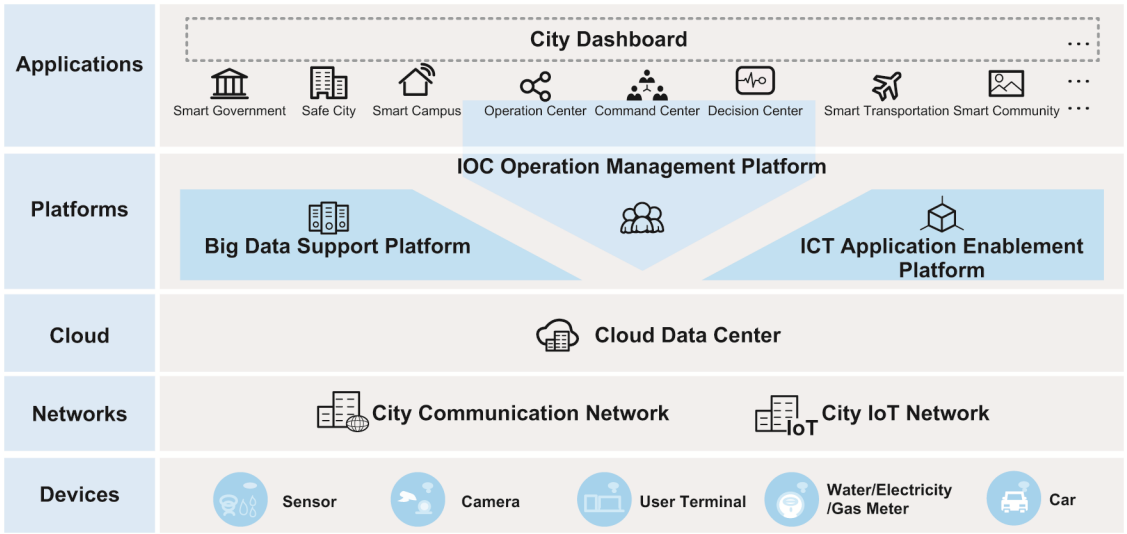


Figure 8—Smart Cities ICT infrastructure

Two networks: The urban communications network and the urban IoT. It combines wired and wireless broadband networks to enable ubiquitous broadband access in a city. In the IoT field, a diversity of access gateways and wireless access technologies are used, adapting to complex access environments, including high-density, large-volume, ground and underground scenarios.

One cloud center: The cloud data center which includes a rich portfolio of cloud products and service solutions, such as cloud computing and cloud storage.

— Three platforms: The Big Data Support Platform, ICT Application Enablement Platform, and Intelligent Operation Center (IOC)

— Big Data Support Platform: Enables data sharing and exchange, and provides data support for intelligent applications

— ICT Application Enablement Platform: Allows partners to develop intelligent applications

— IOC: Aggregates a wide range of data to visualize the city running status, enable efficient emergency collaboration across agencies, and facilitate decision-making based on Big Data.

The IOC will be described in detail in 4.6.2.

4.6.2 Intelligent Operation Center as an element of a Smart Cities domain

With the increase of urban population and the limitation of resources, cities around the world are facing more and more problems about the day-to-day city operation. In order to help ensure public safety and to provide water, electricity, transport, and other services more efficiently, the city government needs to collect and analyze more and more information by real-time communication and collaboration among different industries and departments.

For most cities, routine information and critical information are often distributed in different systems of functional departments. It is difficult for the city governments to make the correct and on-time decisions since they cannot have a clear view of the whole situation. An Intelligent Operation Center (IOC) is needed to make the cities operation and public service more efficient.

The Intelligent Operation Center acts like a brain and central nervous system for Smart Cities, integrating and interconnecting information and processes. And it also provides a platform for technology, operation, and management:

— Monitors city operations so professional staff can stay informed of conditions

— Accelerates emergency response with cross-agency collaboration

— Simulates city operations and facilitates intelligent decision-making with unified city development planning and data mining and analytics

— An open source data system offers personalized, smart data services for citizens

Figure 9 illustrates the IOC-based IoT, and Figure 10 illustrates the Intelligent Operation Center architecture.

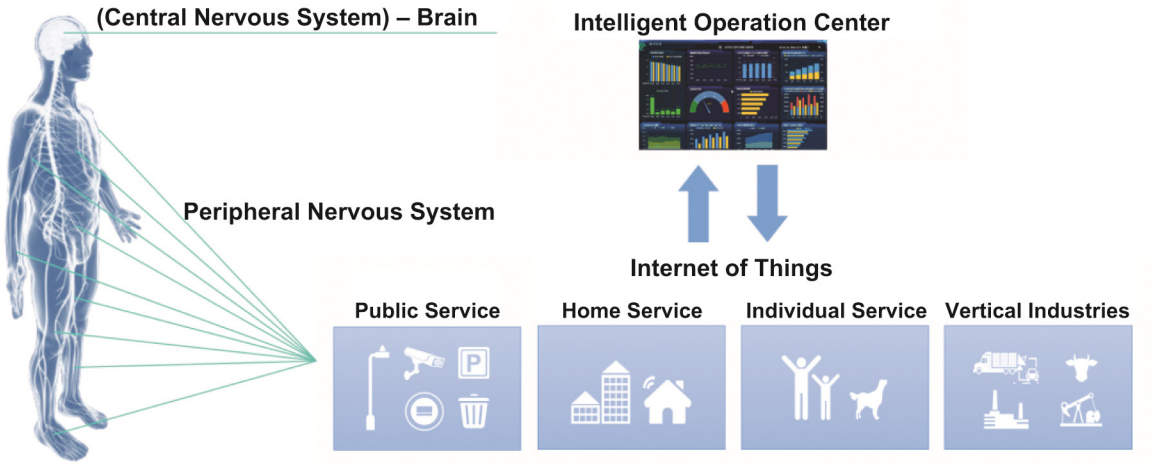


Figure 9—IOC Based on IoT

The Intelligent Operation Center is normally based on the cloud-based information and communication technology (ICT) infrastructure. What's more, an Intelligent Operation Center normally has comprehensive features such as city management databases, data support and resource services, and cross-agency applications. The Intelligent Operation Center may include:

— ICT infrastructure: A Cloud Platform centrally allocates and manages IT infrastructure and data resources such as computing, storage, and network resources.

— City management databases: Population, housing, macro economy, and geographic information support flexible data sharing and decision-making.

— Data support and resource services: Exploit the full benefits of Big Data by converging data across geographically-dispersed industries.

— Cross-agency service applications: City operations status displays for monitoring and alerts, collaborative handling and command of emergencies, simulation and unified planning, open city data and information services, and cyber security management.

— City operation center: Large-screen display, digital conferencing, sound reinforcement, video and audio switching, central control systems, and other facilities.

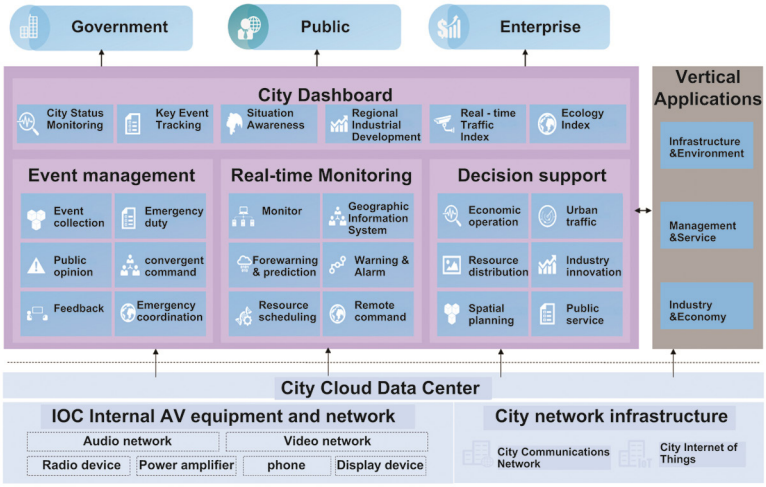


Figure 10—Intelligent Operation Center architecture

The benefits of deployment of Intelligent Operation Center include:

— Comprehensive monitoring

— Collaborative emergency handling and command

— Unified planning

— Data openness for open-source cities and personalized, Smart Citizen Services

4.7 Healthcare

Healthcare has a history of maintaining pace with technology. Patient records have been tracked electronically for many years, mobile solutions are commonly used to capture, monitor, and report patient data and to provide access to healthcare providers. As more affordable, intelligent connected devices become available, the healthcare sector is taking advantage of them.

Healthcare is evolving from an e-health perspective, in which technology was used to provide a level of automation, to a Smart Health perspective, in which healthcare systems exploit the advantages that supporting networks of context aware, autonomous intelligent agents can offer (Russo [B128]).

Connected devices which embed context aware, autonomous intelligent agents have the potential to provide significant support to patients—for example, for those patients self-managing chronic conditions. They can also support performance and lifestyle improvements (note the success of personal activity trackers worn as wrist-bands) and potentially help address behavior modification (such as reducing alcohol intake and stopping smoking).

Given the sensitivities around health data and the potential repercussions that the leaking of such data could have, confidentiality of data is a key concern.

Somewhat in contrast to that, there is potential value in enabling cross-pollination with social networks and Smart City services to build digital communities for patients, community care-givers, and health professionals.

5. Domain commonalities and common concerns

5.1 Stakeholders

Architecture descriptions are framed around the concerns of stakeholders in a system-of-interest (ISO/IEC/IEEE 42010:2011). Here, the system-of-interest is a framework which captures the commonalities across IoT domains and the initial stakeholders are a set of generalized roles shown in Table 1.

Table 1—IoT common stakeholders

Stakeholder	Role
Acquirers	Oversee the procurement of the IoT system or product
Assessors	Oversee the IoT system's conformance to standards and legal regulation
Builders	Construct and deploy the IoT system from specifications (or lead the teams that do this)
Communicators	Explain the IoT system to other stakeholders via its documentation and training materials
Developers	Construct and deploy the IoT system from specifications (or lead the teams that do this)
Maintainers	Manage the evolution of the IoT system once it is operational
Operators	Run the IoT system once it has been deployed
Owners	Derive the benefits of the IoT system when in use
Production engineers	Design, deploy, and manage the hardware and software environments in which the IoT system will be built, tested, and run
Regulators	Define the legal, regulatory, and domain-specific norms of practice/rules that the IoT system shall fulfill with regards to safety, reliability, security, privacy, and resilience
Suppliers	Build and/or supply the hardware, software, or infrastructure on which the IoT system will run
Support staff	Provide support to users for the IoT product or system when it is running
System administrators	Run the IoT system once it has been deployed
Testers	Test the IoT system to help ensure that it is suitable for use
Users	Define the IoT system's functionality and ultimately make use of it

Each viewpoint may provide unique stakeholder names that interpret and can be cross-referenced to the stakeholders in Table 1. The cross-reference is provided within each viewpoint description and in Annex B.

5.2 Common concerns

In the context of this standard, common concerns represent stakeholder concerns which have been:

— Identified in multiple IoT domains or

— Identified in a single IoT domain, but are considered to have broader applicability

A list of common concerns was derived from consideration of stakeholder concerns across the representative domains presented in 3.2. This list was then compared to those compiled by the NIST CPS Public Working Group in the Framework for Cyber-Physical Systems (NIST SP 1500-201 [B88]). An IoT system can be considered a type of cyber-physical system as it has one or more sensors and/or actuators. The resultant list of common IoT domain concerns is shown in Table 2 (without prioritization or weighting).

Table 2—IoT concerns

Concern	Description	Source[a]
Actuation	Concerns related to the ability of the cyber-physical system (CPS) to effect change in the physical world.	NIST SP 1500-201 [B88]
Adaptability	Concerns related to the ability of the CPS to achieve an intended purpose in the face of changing external conditions such as the need to upgrade or otherwise reconfigure a CPS to meet new conditions, needs, or objectives.	NIST SP 1500-201 [B88]
Assurance	The level of confidence that a CPS is free from vulnerabilities, either intentionally designed into it or accidentally inserted during its lifecycle, and that the CPS functions in the intended manner.	NIST SP 1500-201 [B88]
Autonomy	Concerns related to optimal functionality under independent operating conditions in a changing environment.	IEC White Paper 2015 [B52], Nonaka et al. [B89]
Behavioral	Concerns related to interdependence among behavioral domains. Concerns related to the ability to successfully operate a CPS in multiple application areas.	NIST SP 1500-201 [B88]
Collaboration	Concerns related to formation of two or more systems that form an ecosystem that guarantees value delivery.	IEC White Paper 2015 [B52], Nonaka et al. [B89]
Communication	Concerns related to the exchange of information internal to the CPS and between the CPS and other entities.	NIST SP 1500-201 [B88]
Complexity	Concerns related to our understanding of the behavior of CPS due to the richness and heterogeneity of interactions among its components, such as existence of legacy components and the variety of interfaces.	NIST SP 1500-201 [B88]
Conceptual	Concerns related to requirements engineering and concept engineering (including architecture development).	The present document
Confidentiality	Property that information is not made available or disclosed to unauthorized individuals, entities, or processes	ISO/IEC 270000:2016 [B64]
Constructivity	Concerns related to the ability to combine CPS modular components (hardware, software, and data) to satisfy user requirements.	NIST SP 1500-201 [B88]
Controllability	Ability of a CPS to control a property of a physical thing. There are many challenges to implementing control systems with CPS including the non-determinism of cyber systems, the uncertainty of location, time, and observations or actions, their reliability and security, and complexity. Concerns related to the ability to modify a CPS or its function, if necessary.	NIST SP 1500-201 [B88]
Cost	Concerns related to the direct and indirect investment or monetary flow or other resources required by the CPS throughout its lifecycle.	NIST SP 1500-201 [B88]
Data semantics	Concerns related to the agreed and shared meaning(s) of data held within, generated by, and transiting a system.	NIST SP 1500-201 [B88]
Data velocity	Concerns related to the speed with which data operations are executed.	NIST SP 1500-201 [B88]
Data volume	Concerns related to the volume or quantity associated with a CPS's operation.	NIST SP 1500-201 [B88]
Deployability	Concerns related to the ease and reliability with which a CPS can be brought into productive use.	NIST SP 1500-201 [B88]
Discoverability	Concerns related to the ease and reliability with which a CPS component can be observed and understood (for purposes of leveraging the component's functionality) by an entity (human, machines). Concerns related to the ease and reliability with which a CPS component's functions can be ascertained (for purposes of leveraging that functionality) by an entity (human, machines).	NIST SP 1500-201 [B88]
Disposability	Concerns related to the impacts that may occur when the CPS is taken physically out of service.	NIST SP 1500-201 [B88]
Engineerability	Concerns related to the ease and reliability with which a CPS design concept can successfully be realized via a structured engineering process.	NIST SP 1500-201 [B88]
Enterprise	Concerns related to the economic aspects of CPS throughout their lifecycle.	NIST SP 1500-201 [B88]

Concern	Description	Source[a]
Environment	Concerns related to the impacts of the engineering and operation of a CPS on the physical world.	NIST SP 1500-201 [B88]
Evolvability	Concerns related to the system's ability to evolve and to be functional with newer technology (backward compatibility).	IEC White Paper 2015 [B52]
Functionality	Concerns related to the function that a CPS provides.	NIST SP 1500-201 [B88]
Human factors	Concern about the characteristics of CPS with respect to how they are used by humans.	NIST SP 1500-201 [B88]
Identity	Concerns related to the ability to accurately recognize entities (people, machines, and data) when interacting with or being leveraged by a CPS.	NIST SP 1500-201 [B88]
Integrity	Property of accuracy and completeness.	ISO/IEC 270000:2016 [B64]
Logical time	Concerns related to the order in which things happen (causal order relation) or event driven.	NIST SP 1500-201 [B88]
Maintainability	Concerns related to the ease and reliability with which the CPS can be kept in working order.	NIST SP 1500-201 [B88]
Manageability	Concerns related to the management of CPS function. For example, managing timing in a complex CPS or SoS is a new issue with CPS that did not exist before.	NIST SP 1500-201 [B88]
Measurability	Concerns related to the ability to measure the characteristics of the CPS.	NIST SP 1500-201 [B88]
Monitorability	Concerns related to the ease and reliability with which authorized entities can gain and maintain awareness of the state of a CPS and its operations. Includes logging and audit functionality.	NIST SP 1500-201 [B88]
Networkability	Concerns related to the ease and reliability with which a CPS can be incorporated within a (new or existing) network of other systems.	NIST SP 1500-201 [B88]
Operability	Concerns related to the operation of the CPS when deployed.	NIST SP 1500-201 [B88]
Operations on data	Concerns related to the ability to create/read/update/delete system data and how the integrity of CPS data and behaviors may be affected.	NIST SP 1500-201 [B88]
Optimization	Concerns related to the ability of systems or resources to perform optimally.	IEC White Paper 2015 [B52], Nonaka et al. [B89]
Performance	Concerns related to the ability of a CPS to meet required operational targets.	NIST SP 1500-201 [B88]
Physical	Concerns about purely physical properties of CPS including seals, locks, safety, and EMI.	NIST SP 1500-201 [B88]
Physical context	Concerns relating to the need to understand a specific observation or a desired action relative to its physical position (and uncertainty). While this information is often implied and not explicit in traditional physical systems, the distributed, mobile nature of CPS makes this a critical concern.	NIST SP 1500-201 [B88]
Policy	Concerns related to the impacts of treaties, statutes, and doctrines on a CPS throughout its lifecycle.	NIST SP 1500-201 [B88]
Privacy	Concerns related to the ability of the CPS to prevent entities (people, machines) from gaining access to data stored in, created by, or transiting a CPS or its components such that individuals or groups cannot seclude themselves or information about themselves from others. Privacy is a condition that results from the establishment and maintenance of a collection of methods to support the mitigation of risks to individuals arising from the processing of their personal information within or among systems or through the manipulation of physical environments.	NIST SP 1500-201 [B88]
Procureability	Concerns related to the ease and reliability with which a CPS can be obtained.	NIST SP 1500-201 [B88]
Producibility	Concerns related to the ease and reliability with which a CPS design can be successfully manufactured.	NIST SP 1500-201 [B88]

Concern	Description	Source[a]
Quality	Concerns related to the ease and reliability of assessing whether a CPS meets stakeholder (especially customer) expectations.	NIST SP 1500-201 [B88]
Regulatory	Concerns related to regulatory requirements and certifications.	NIST SP 1500-201 [B88]
Relationship between data	Concerns related to how and why sets of data shall, may, or may not be associated with each other and the value or harm that can be derived from those associations.	NIST SP 1500-201 [B88]
Relevance	Concerns related to interpretation of fit based on situational context and significance.	IEC White Paper 2015 [B52], Murdock et al. [B78]
Reliability	Concerns related to the ability of the CPS to deliver stable and predictable performance in expected conditions.	NIST SP 1500-201 [B88]
Resilience	Concerns related to the ability of the CPS to withstand instability, unexpected conditions, and gracefully return to predictable, but possibly degraded, performance.	NIST SP 1500-201 [B88]
Responsibility	Concerns related to the ability to identify the entity or entities authorized to control the operation of a CPS.	NIST SP 1500-201 [B88]
Safety	Concerns related to the ability of the CPS to help ensure the absence of catastrophic consequences on the life, health, property, or data of CPS stakeholders and the physical environment.	NIST SP 1500-201 [B88]
Security	Concerns related to the ability of the CPS to help ensure that all of its processes, mechanisms (both physical and cyber), and services are afforded internal or external protection from unintended and unauthorized access, change, damage, destruction, or use. Confidentiality: Preserving authorized restrictions on access and disclosure. Integrity: Guarding against improper modification or destruction of a system, and includes ensuring non-repudiation and authenticity. Availability: Ensuring timely and reliable access to and use of a system.	NIST SP 1500-201 [B88]
Sensing	Concerns related to the ability of a CPS to develop the situational awareness required to perform its function.	NIST SP 1500-201 [B88]
Standardization	Concerns related to the availability and applicability of standards and the need for standards. Standards can help to achieve compatibility,[b] safety, security, repeatability, and quality. Standards can also facilitate commoditization of technologies and processes.	The present document
States	Concerns related to the states of a CPS. For example, the functional state of a CPS is frequently used to allow for variation in the CPS response to the same set of inputs. Variation in response based on state is sometimes referred to as functional modes.	NIST SP 1500-201 [B88]
Synchronization	Concerns for synchronization are that all associated nodes have timing signals traceable to the same time scale with accuracies as required. There are three kinds of synchronization that might be required: time, phase, and frequency synchronization, although frequency synchronization is also called *syntonization*.	NIST SP 1500-201 [B88]
Time awareness	Concerns that allow time correctness by design. The presence or absence of time explicitly in the models used to describe, analyze, and design CPS and in the actual operation of the components. This is a lifecycle concern as well as a concern for the ability to build devices without the need for extensive calibration of the timing properties.	NIST SP 1500-201 [B88]
Time to market	Concerns related to the time period required to bring a CPS from need realization through deployment.	NIST SP 1500-201 [B88]

Concern	Description	Source[a]
Time interval and latency	Specifying requirements for timing generally involves requirements for time intervals between pairs of events. A time interval is the duration between two instants read on the same time scale. CPS timing requirements are generally expressed as constraints on the time intervals (TI) between pairs of system significant events. These can be categorized in terms of bounded TIs or latency, deterministic TIs, and accurate TIs.	NIST SP 1500-201 [B88]
Uncertainty	Managing the effects of uncertainties is a fundamental challenge in CPS. Sources of uncertainty in CPS can be grouped into statistical (aleatoric) uncertainty, lack of knowledge (epistemic) uncertainty, or systematic uncertainty. In CPS, statistical uncertainty is caused by randomness of accuracy of sensing and actuation, often caused by uncertainty of manufacturing processes. Systematic uncertainty is caused by incomplete knowledge either due to limits of acquired knowledge or due to simplification in modeling. Typical manifestations of epistemic uncertainty are limited validity of models of physical processes or limits of computability of properties of mathematical models.	NIST SP 1500-201 [B88]
Usability	Concerns related to the ability of a CPS to be used to achieve its functional objectives effectively, efficiently, and to the satisfaction of users (see ISO 9241-210). The combination of physical and cyber into complex systems creates challenges in meeting usability goals. Complexity is a major issue. The diversity of interfaces creates a significant learning curve for human interaction.	NIST SP 1500-201 [B88]
Utility	Concerns related to the ability of a CPS to provide benefit or satisfaction through its operation. Utility reflects a business concern, especially when considered as the numerator when computing value, which equals utility divided by costs.	NIST SP 1500-201 [B88]

[a] If coming from an external source.
[b] Interoperability is a level of compatibility.

6. Architecture framework

6.1 General information

Figure 11 shows a model of the architecture framework being described in the present document. Subclause 3.2 and Clause 5 have provided background in terms of representative domains, stakeholders, and views of the concerns to be addressed. Clause 6 now uses that background to construct the body of the architecture description as described in ISO/IEC/IEEE 42010:2011.

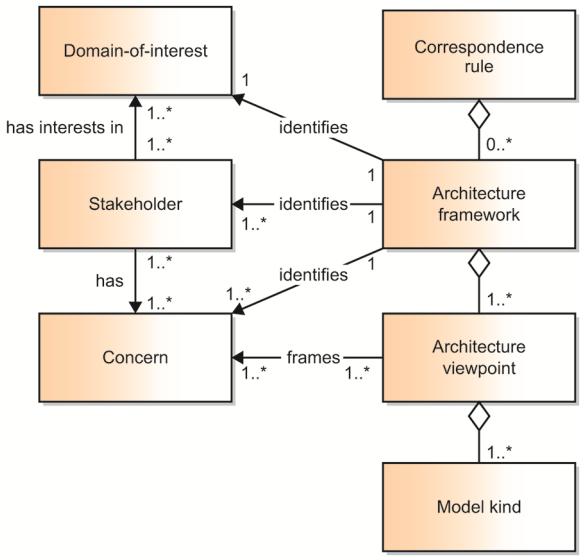


Figure 11—Conceptual model of an architecture framework (based on ISO/IEC/IEEE 42010:2011)

6.2 Viewpoints and model kinds

Viewpoints and model kinds are the foundation of the architecture framework. Subclause 6.6 presents a catalogue of viewpoints which frame the concerns described in 5.2. The viewpoints and model kinds govern the conventions for constructing, interpreting, using, and analyzing architecture views and models.

The following are different viewpoints abbreviations used for correspondence rule (CR) numbers:

- 6.3.2 CNC: Conceptual viewpoint
- 6.3.3 CMP: Compatibility viewpoint
- 6.3.4 LFC: Lifecycle viewpoint
- 6.3.5 COM: Communication viewpoint
- 6.3.6 INF: Information viewpoint
- 6.3.7 FUN: Function viewpoint
- 6.3.8 THM: Threat model viewpoint
- 6.3.9 SSM: Security and safe monitoring viewpoint
- 6.3.10 ACA: Access control viewpoint
- 6.3.11 ADS: Adequate design for required security viewpoint
- 6.3.12 PTR: Privacy and trust viewpoint
- 6.3.13 COL: Collaboration viewpoint
- 6.3.14 CPR: Computing resources viewpoint

An overview of the viewpoints provided in this standard, including pertinent correspondence rules, is provided in Table 3.

Table 3—Correspondence rules

Viewpoint	Subclause	Correspondence rule	Addressed viewpoint	Comment
Conceptual	6.6.2	CNC-CR-1	Compatibility	
		CNC-CR-2	Threat model	
		CNC-CR-3	Security and safety monitoring	
		CNC-CR-4	Access control architecture	
		CNC-CR-5	Adequate design for architecture	
		CNC-CR-6	Communication	
		CNC-CR-7	Collaboration	
		CNC-CR-8	Lifecycle	
Compatibility	6.6.3	CMP-CR-1	Conceptual	
		CMP-CR-2	Compatibility	
		CMP-CR-3	Lifecycle	
		CMP-CR-4	Communication	
		CMP-CR-5	Information	
		CMP-CR-6	Function	
		CMP-CR-7	Threat model	
		CMP-CR-8	Security and safety monitoring	
		CMP-CR-9	Access control architecture	
		CMP-CR-10	Adequate design for required security	
		CMP-CR-11	Privacy and trust	
		CMP-CR-12	Collaboration	
Lifecycle	6.6.4	—	All other viewpoints	The lifecycle is used to pattern all other architecture views.
Communication	6.6.5	COM-CR-1	Conceptual	
		COM-CR-2	Compatibility	
		COM-CR-3	Lifecycle	
		COM-CR-4	Communication	
		COM-CR-5	Information	
		COM-CR-6	Functional	
		COM-CR-7	Threat model	
		COM-CR-8	Security and safety monitoring	
		COM-CR-9	Access control architecture	
		COM-CR-10	Adequate design for required security	
		COM-CR-11	Privacy and trust	
		COM-CR-12	Collaboration	
Information	6.6.6	—	—	Influences the expression of model kinds in the viewpoints lifecycle, communication, collaboration, and information.
Functional	6.6.7	FUN-CR-1	Conceptual	
		FUN-CR-2	Compatibility	
		FUN-CR-3	Lifecycle	
		FUN-CR-4	Communication	
		FUN-CR-5	Information	
		FUN-CR-6	Threat model	
		FUN-CR-7	Security and safety monitoring	

Viewpoint	Subclause	Correspondence rule	Addressed viewpoint	Comment
		FUN-CR-8	Access control architecture	
		FUN-CR-9	Adequate design for required security	
		FUN-CR-10	Privacy and trust	
		FUN-CR-11	Collaboration	
Threat model	6.6.8	THM-CR-1	Functional and information viewpoint	
		THM-CR-2	All security-related viewpoints	
Security and safety monitoring	6.6.9	SSM-CR-1	Conceptual	Threat and security policy viewpoints should govern monitoring system sensor deployment.
		SSM-CR-2	Conceptual	All significant security and safety events and possible attacks should be monitored by monitoring system sensors.
		SSM-CR-3	Functional and informational	The functional and informational viewpoints govern endpoint IoT device interface monitoring. Signature detection should be provided for all user-visible device interfaces.
		SSM-CR-4	Conceptual	The deployment model should include monitoring system components.
Access control architecture	6.6.10	ACA-CR-1	Functional	
		ACA-CR-2	Threat model and functional	
		ACA-CR-3	Security and safety monitoring	
Adequate design for required security	6.6.11	ADS-CR-1	Security-related viewpoints	
		ADS-CR-2	Security-related viewpoints	
		ADS-CR-3	Treat model	
		ADS-CR-4	Security-related viewpoints	
		ADS-CR-5	Security-related viewpoints	
		ADS-CR-6	Security-related viewpoints	
		ADS-CR-7	Collaboration	
Privacy and trust	6.6.12	—	—	
Collaboration	6.6.13	COL-CR-1	Conceptual	
		COL-CR-2	Adequate design for required security	
		COL-CR-3	All other viewpoints	
Computing resources	6.6.14	CPR-CR-1	Communication	
		CPR-CR-2	Functional	

6.3 Architecture development

The architects select appropriate viewpoints to help them develop views addressing specific concerns based on the target system. It is not required to use all the viewpoints provided in 6.6. It is anticipated that architects may develop additional viewpoints and model kinds to apply in concert with those in 6.6 when addressing concrete systems of interest (Hilliard [B32]).

Guidance on the use of the framework is given in ISO/IEC/IEEE 42010:2011.

6.4 Rationale for key decisions

The rationale for the key decisions for an architecture of a system, with the framed concerns and their stakeholders and the architecture view, shall be specified according to the application needs.

6.5 Stakeholders and concerns

The stakeholders and concerns are presented in Clause 5.

6.6 Viewpoint catalogue

6.6.1 Overview

Subclause 6.6 describes the different detailed viewpoints of the IoT architecture framework. The viewpoint definition based on ISO/IEC/IEEE 42010:2011.

6.6.2 Conceptual viewpoint

6.6.2.1 General information and key features

The conceptual viewpoint is intended to assist architects in the design, analysis, and expression of system architectures employing a shared language. It frames a number of architecture concerns related to the concepts within architecture description and the use of such descriptions. It is important to address the conceptual view as a part of architecture to ensure that stakeholders are provided with a common vocabulary and semantics when talking about IoT systems.

The definition of conceptual models, i.e., of entities and their relationships, stimulates coherence and consistency of architecting products (architecture descriptions, built systems, etc.). Conceptual models foster communication over application-domain borders, organizational borders (e.g., across teams), and cultural borders (e.g., world regions). Conceptual models also foster communication across stakeholder "generations," e.g., successive employee generations. Also, instantiating the conceptual view fosters an architecture description language that remains stable during revisions and during reuse, e.g., when creating architecture descriptions for new systems of interest. Conceptual models are especially helpful for stakeholders interested in more than one system. This interest can manifest over time or at once. An example for the former is a builder who designs new product generations.

The conceptual viewpoint employs six model kinds: the entity model kind, the system model kind, the intent model kind, the IoT component model kind, the IoT component capability model kind, and the representation model kind.

6.6.2.2 Stakeholders and concerns

Stakeholders for this viewpoint are as defined in 5.1. An architect is an additional stakeholder of this viewpoint.

Table 4 shows the concerns of the conceptual viewpoint.

Table 4—Conceptual viewpoint; concerns

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Assurance	How can we convince stakeholders we have met our obligations for being safe, reliable, resilient, secure, and meeting privacy expectations?
Concept	How can we make system concepts reusable, e.g., over product generations and/or across engineering teams?
Concept	How can we make architecture descriptions—and the concepts they employ—comparable to each other?
Concept, cost	How can we improve the communication between teams?[a]
Constructivity	How can we increase the standardized composition of component interfaces?
Engineerability	How can we aid architects and implementers with the design of component interfaces?
Standardization	How can we support the clarification where IoT standardization is needed?
Standardization	How can we support gap analyses of IoT standards?
Procureability	How can system offers be made more comparable?

[a] For a thorough discussion of the architectural implications of communication borders see Conway [B19].

6.6.2.3 Anti-concerns for this viewpoint

This viewpoint does not address implementation details, e.g., how an IoT component is implemented, nor how interfaces between IoT components shall be implemented.

6.6.2.4 Model kinds

6.6.2.4.1 General

The main model kinds for any conceptual view intended to describe an IoT system architecture include:

- Entity model kind
- System model kind
- Intent model kind
- IoT component model kind
- IoT component capability model kind
- Representation model kind

6.6.2.4.2 Entity model kind

6.6.2.4.2.1 Objective

In order to differentiate distinguishable manifestations from others and how they are described, e.g., distinguishing a human from a sensor, we introduce the entity model kind.

6.6.2.4.2.2 Conventions

6.6.2.4.2.2.1 General

An entity has a physical manifestation and/or a conceptual representation. Note that a conceptual representation has at least one physical manifestation, for instance an XML file on a hard disk, but the physical manifestation of a conceptual representation is likely abstracted out of view.

The concept of an entity is not used in its wider ontological sense (see, for instance, Hoehndorf [B33]), i.e., that anything perceivable is an entity. Rather, the entity concept is used in a narrow sense: Only manifestations and representations pertaining to an IoT system are modeled as entities, while related categories—for instance attributes of these manifestations—are not. Attributes are abstractions that represent the interpreter's understanding of an entity. Examples for attributes are color, temperature, and spatial dimensions.

6.6.2.4.2.2.2 Model kind languages or notations

This model is presented in UML 2.2 [B94], particularly the class-diagram pattern. Other modeling languages that are capable of representing ontologies are also feasible, including natural languages.

6.6.2.4.2.2.3 Model kind meta-model

The ontology of the concept entity is shown in Figure 12.

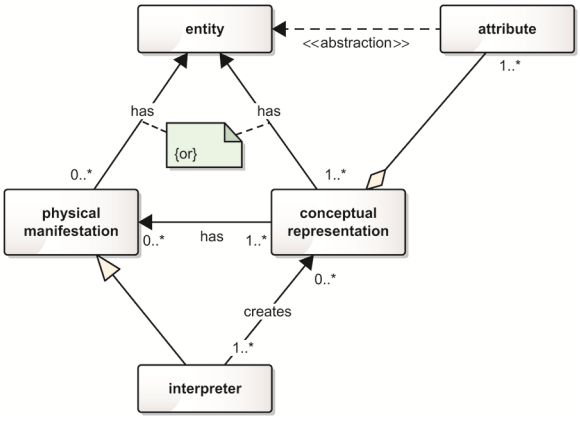


Figure 12—Ontology of the concept entity (class diagram)

6.6.2.4.2.3 Correspondence rules

The entity concept is used in all other model kinds in this viewpoint.

6.6.2.4.3 System model kind

6.6.2.4.3.1 Objective

One of the central concepts used in this standard is that of a system. This model kind provides the reader with a list of axioms that all systems fulfill. Subclause 6.6.2.4.3 also refers to an open list of propositions tied to each of the axioms and it discusses the most relevant propositions. Note that some of these propositions are addressed in other model kinds (see 6.6.2.4.7.3).

6.6.2.4.3.2 Conventions

The following axioms apply to all designed systems, and thus to all IoT systems (Whitney et al. [B146]):

- The centrality axiom states that central to all systems are two pairs of propositions: emergence and hierarchy, and communication and control.
- The contextual axiom states that system meaning is informed by the circumstances and factors that surround the system.
- The design axiom states that system design is a purposeful balancing of resources and relationships.
- The goal axiom states that systems achieve specific goals through purposeful behavior using pathways and means.
- The information axiom states that systems create, possess, transfer, and modify information.

An open list of propositions aligned with these axioms can be found elsewhere in the literature (Whitney et al. [B146]). Every IoT system adhering to this standard adheres to the axioms above, but it does not need to adhere to all of the aforementioned propositions. Let us next address a proposition that all IoT systems fulfill.

Hierarchy (centrality axiom): “Entities meaningfully treated as wholes are built up of smaller entities which are themselves, wholes. In a hierarchy, emergent properties denote the levels” (Whitney et al. [B146]). This proposition implies that emergent levels can be used when identifying hierarchy levels within an IoT system. Note that the hierarchy can be “flat,” for instance, for peer-to-peer systems.

Other relevant propositions are addressed in 6.6.2.4.3.3 and 6.6.2.4.3.4.

6.6.2.4.3.3 Operations

The following propositions shall be heeded when instantiating this model kind.

Boundary (contextual axiom): “The abstract, semi-permeable perimeter of the system defines the components that make up the system, segregating them from environmental factors and may prevent or permit entry of matter, energy and information.” (Whitney et al. [B146]). This proposition implies that the IoT architecture description shall define the system boundary. Note that this “physical” boundary is not the only one that can be defined for a system. Other boundaries can be drawn, for instance, for contingent parts of the system that can be accessed by users, for all parts information components that utilize encryption, etc.

Requisite parsimony (design axiom): "The capacity of human short-term recall is no greater than seven plus or minus two items." (Whitney et al. [B146]). This limitation is relevant when compiling architecture descriptions that can readily be parsed by the stakeholders.

6.6.2.4.3.4 Correspondence rules

This model kind applies everywhere in the standard where the concept *system* is used.

The following propositions have implications for other model kinds in this viewpoint and also other viewpoints. These propositions are ordered by the system axioms.

Centrality axiom

Emergence: "Whole entities exhibit properties and patterns that are meaningful only when they are attributed to the whole, not its parts." (Whitney et al. [B146]). This has, among others, implications for the sensors and actuators to be chosen (see 6.6.2.4.6) and for the information viewpoint. The implication for the latter is that meaning and thus knowledge is generated through synthesis. For instance, if an IoT system tracks the movement of cows on a pasture, one cannot infer whether a herd is in panic from the movement of a single cow. Rather, the IoT system needs to track the movement of several cows in order to infer this judgment. Interestingly, the fusion of different types of sensor data can enhance the reliability of this inference. In the cow herd example, the IoT system can, for instance, also record audio and listen for lowing.

Communication: "Communication is a transaction between the information source terminal and the destination terminal, with the sole aim of generation and reproduction of symbols. Information is transmitted as a selection along possible alternative states." (Whitney et al. [B146]). This proposition implies a central role for communication in IoT systems. Communication is addressed in the communication viewpoint (see 6.6.5).

Contextual axiom

The choice of the system boundary has implications on the intent model kind (see 6.6.2.4.4). For instance, are the physical entities, which the IoT system interacts with, chosen to be part of the IoT system or not? This choice has repercussions for what constitutes the environment of the IoT system (see, for instance, the core conceptual model kind in 6.6.2.4.6).

The goal axiom

Purposeful behavior: "Purposeful behavior is meant to denote that the act or behavior may be interpreted as directed to the attainment of a goal—i.e., to a final condition in which the behaving object reaches a definite correlation in time or in space with respect to another object or event." (Whitney et al. [B146]). This implies the centrality of the goal for the IoT system (see the "intent model kind" in 6.6.2.4.4).

The information axiom

Redundancy of potential command: "Effective action is achieved by an adequate concatenation of information." (Whitney et al. [B146]). This implies that information synthesis is important for IoT systems and needs thus to be reflected in the information view of an IoT system. Note that information synthesis also plays a role for emergent properties (see the proposition *Emergence* above).

6.6.2.4.4 Intent model kind

6.6.2.4.4.1 Objective

The intent model kind links IoT stakeholder goals to IoT systems, as well as to the entities of interest with which the IoT system interacts. The importance of the system goal is implied by the purposive-behavior system proposition (see 6.6.2.4.3.4). A stakeholder's intent has implications for—among others—what is considered to be the entity of interest and what not (Floridi [B27]). For instance, for the user of an IoT-based fitness trainer that tracks pulse rate during training, the user itself is the entity of interest. For a person responsible for the maintenance of the fitness trainer, e.g., for calibration purposes, the IoT device itself, i.e., the fitness trainer, is the entity of interest.

6.6.2.4.4.2 Conventions

6.6.2.4.4.2.1 General

This model kind adopts the stakeholder concept promoted by ISO/IEC/IEEE 42010:2011 (see Clause 5). It also adopts the role concept promoted by TOGAF (The Open Group [B119]).

6.6.2.4.4.2.2 Model kind languages or notations

This model is presented in UML 2.2 [B94], particularly the class-diagram pattern. Other modeling languages that are capable of representing ontologies are also feasible, including natural languages.

6.6.2.4.4.2.3 Model kind meta-model

This model kind describes the properties and roles of humans and stakeholders and how they relate to IoT systems. The intermediate between the former and the latter is goals and intent. It is represented by two class diagrams (see Figure 13 and Figure 14).

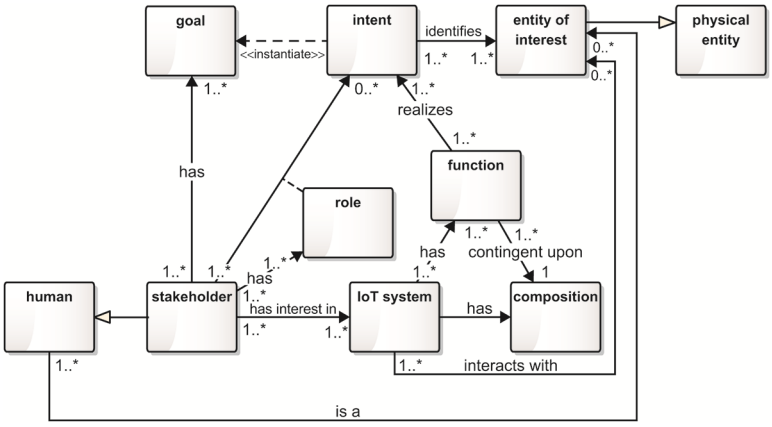


Figure 13—Class diagram of the intent model kind

A stakeholder is a human that has goals, which identify entities of interest. A human can be an entity of interest. An IoT system is a cyber-physical system, which interacts with the physical world through sensors

and actuators. The stakeholder goals are instantiated in function-centric intent, which are embodied by the IoT system. The IoT system interacts with the identified entities of interest.

Besides framing what entity of interests are within the scope of an IoT system, the intent also frames vantage points against which an entity can be represented. This aspect of the intent model kind is depicted in Figure 14.

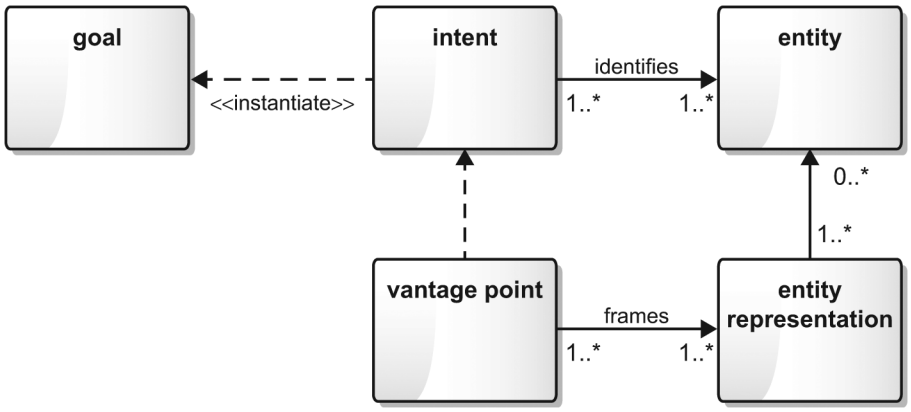


Figure 14—Class diagram of the vantage point model kind

6.6.2.4.4.3 Operations

Pertinent architecture models are created by instantiating the entities in this model kind for the system of interest. Note that we do not prescribe whether the entity of interest is part of the IoT system or not (see the proposition *boundary* in 6.6.2.4.3.3). Also, suitable vantage points and representations of the IoT system and the environment shall be chosen.

6.6.2.4.4.4 Correspondence rules

IoT systems are also addressed in the “IoT component capability model kind” (see 6.6.2.4.6). Entities of interest are also addressed in the “IoT component capability model kind” (see 6.6.2.4.6) and the representation model kind (see 6.6.2.4.7).

6.6.2.4.5 IoT component model kind

6.6.2.4.5.1 Objective

This model kind introduces components relevant to IoT systems and the IoT environment.

6.6.2.4.5.2 Conventions

6.6.2.4.5.2.1 General

This model kind does not have any pre-described conventions.

6.6.2.4.5.2.2 Model kind languages or notations

This model is presented in UML 2.2 [B94], particularly the class-diagram pattern. Other modeling languages that are capable of representing ontologies are also feasible, including natural languages.

6.6.2.4.5.2.3 Model kind meta-model

Figure 15 displays an UML representation of the IoT component model kind. This class diagram depicts the basic functional components of an IoT system. This diagram does not represent the implementation details of systems that provide these functions; rather it introduces the basic types of functions that make up any complex IoT component or system. In particular, the composition, orchestration, and security details are not shown as they vary depending on the specific implementation.

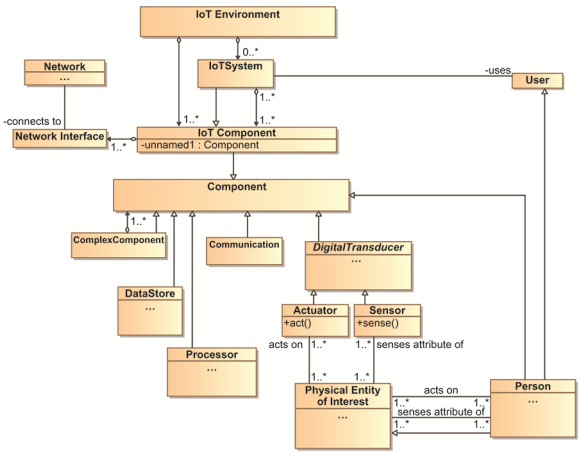


Figure 15—Class diagram of the IoT component model kind

Details of each class in Figure 15 are provided below.

IoT environment: The IoT environment is composed of IoT components that can be built into IoT systems (which are also part of the IoT environment).

IoT component: An IoT component is a type of component that provides a network interface and therefore may be composed into IoT systems. IoT components are the basic building blocks of IoT systems.

Component: Components are entities that can interact with other components to form systems that can achieve goal(s). Components in this model are the functional building blocks for cyber-physical systems. There are six distinct types of components, plus the concept of a complex component that is composed of some combination of the other six component types. The six distinct types are: DataStore, Processor, Communication, Actuator, Sensor, and Person. Actuator and Sensor are of the abstract type DigitalTransducer.

IoT system: The IoT system interacts with a physical entity of interest through sensors and actuators. IoT systems may also be IoT components if the IoT system provides a network interface.

Processor: Processors are the CPUs of the IoT world. The defining characteristic of a processor is the ability to modify data. Processors accept data as input and provide modified data as output.

Data store: Data stores (or data storages) are components that can store data for some period of time. The defining characteristic of a data store is the ability to place data in the data store and retrieve it at some later time. Data stores accept data as input and provide data as output.

Communication: Communication components are the networking components within IoT. Their defining characteristic is the ability to move data from one logical location to another. The logical locations are often (but not always) associated with different physical locations. The communication component is the glue that connects and binds an IoT system together.

Complex component: The complex component represents the type of component that is a combination of any of the other six functional component types. In practice, most IoT components are complex components.

6.6.2.4.6 IoT component capability model kind

6.6.2.4.6.1 Objective

While the IoT component model kind (see 6.6.2.4.5) is useful for equipment manufacturers and system component builders, the inherent white-box approach used in the IoT component model kind (showing some details of the construction of an IoT component) is not very useful from the perspective of an IoT system builder. From an IoT perspective, a black-box viewpoint of each component is much more useful, since an IoT system builder may not have access to any details of the internal workings of an IoT component. In fact, the internal workings of a component may change over time. This is especially relevant for IoT components that are themselves composed of other IoT components. The fact that one cannot access details of the inner workings is not important if the capabilities of a component are accurately described with the details necessary for the system builder to compare capabilities to system requirements. If an IoT component uses standardized interfaces, through which these capabilities can be exposed, configured, and accessed, the IoT components can be combined into systems regardless of internal implementations. Therefore, instead of focusing on the internal details of an IoT system (see 6.6.2.4.5), this model kind focuses on the capabilities a component can provide. A system builder can combine these capabilities with the capabilities of other components in order to create a system that can achieve a set of goals. By understanding each component as a set of capabilities, a system builder can match these capabilities to their system requirements.

Using this capabilities model kind, an IoT component can be understood by the set of capabilities it provides (a simple definition of capability is “quality of being able to perform a given activity”).

6.6.2.4.6.2 Conventions

6.6.2.4.6.2.1 General

6.6.2.4.6.2.2 Model kind languages or notations

This model is presented in UML 2.2 [B94], particularly the class-diagram pattern and the flow-diagram pattern. Other modeling languages that are capable of representing ontologies are also feasible, including natural languages.

6.6.2.4.6.2.3 Model kind meta-model

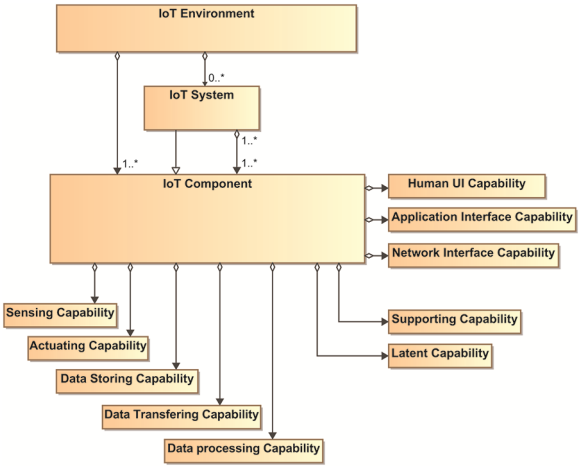


Figure 16—Class diagram of the capabilities of an IoT component

The capability classes in Figure 16 are discussed in more detail below.

Sensing capability: Provides an observation of a property of the physical entity of interest in the form of digital data. Information from sensor observations may be provided to other IoT components through the component's network interface for processing and storage. Sensing is "read only"; any change to the physical state is a side effect. Uncertainties in the observation of the property may be introduced by the physical environment between the physical system and the sensor transducer, in the sensor transducer itself, in the analog electrical circuit, in the analog-to-digital (A/D) converter, and in the digital logic of the sensor. There is a time delay between the sensing function and the data becoming available at the component output. Examples include temperature sensing, spatial sensing, optical sensing, and audio sensing.

Actuating capability: Provides the ability to make a change in the physical world, based on a digital input to the component. Errors may be introduced in the digital logic, the digital-to-analog converter, the analog electrical circuit, and the actuator transducer. There is a time delay between the input data arriving at the component and the change being made to the environment. Examples of actuating capabilities include heating coils, electric shock delivery, electronic door locks.

Data storing capability: Provides the ability to store and retrieve data and information over time. Data persist for a finite period of time. There is a time delay between the input and output. Some examples of data-storing capabilities include databases, block storage, and data brokers (such as a message queue telemetry transport [MQTT] broker).

Data-transferring capability: Provides the ability to transmit data from one physical or logical location to another. The data-transferring capability provides the ability to "black box" a communication network and provide information about the network without having to understand the specific network topology. As the interactions of an IoT system with the physical world require the data-transferring network to meet latency, reliability, and security requirements, it is useful to be able to describe the network characteristics in this manner, so the capability is explicitly called out by the IoT general model. Some examples of specific data-transferring capabilities include data networks based on Ethernet, IEEE 802.11™, and LTE.

Data-processing capability: Provides the ability to transform data based on a defined algorithm. There is a time delay between the input and output that should be accounted for. The transformation may be very simple, with a single input variable and a single output, or it may be complex with multiple inputs and outputs. Control algorithms are an important type of processing that take the output of sensor(s) and

actuator(s) or pre-processor(s) and provide an output that can be fed into an actuator or post-processor. These control algorithms often are used within negative feedback loops, but not always. A proportional-integral-derivative (PID) control algorithm is an example of such a control algorithm. Some examples of processing include data aggregation, data analytics, and predictive analysis.

Supporting capability: Provide additional functionality that supports the functioning of the IoT component or IoT system and does not directly affect the process data. Note that some IoT components may only provide a supporting capability, such as orchestration, and not offer any transducer or data capabilities. Examples of supporting capabilities include time synchronization, data encryption, authentication, orchestration, and remote component management.

Network interface capability: Provides the ability to interface with a communication network. Every IoT component shall have at least one network interface capability and may have more than one. Some examples of network interface capability include: Ethernet adapter interface capability (includes IEEE 802.1™ and IEEE 802.3™, but not the frame interpretation), LTE radio interface capability, and IEEE 802.15™ radio interface capability.

Human UI capability: Provides the ability for the component to exchange information directly with people. Not all IoT components will have a human UI capability (i.e., a processor component may only provide data through the network interface, while a cell phone has several human interface capabilities in the form of the touchscreen, audio, and camera. Examples of human UI capabilities include displays, touch screens, audio speakers, and microphones.

Latent capability: Capabilities that the IoT component could potentially provide, but are not currently enabled for access externally from the IoT component. For example, a component may have a USB port, but nothing is plugged into that port. In that state, the USB port is considered a latent capability. It has the potential to be used at any time, and if someone attaches something to it, that could enable any of the other capabilities. For example, if someone plugs a IEEE 802.11 adapter into the USB port, the IoT component would then have an additional network interface capability.

Application interface capability: Provides the ability for other IoT components (components, systems, etc.) to communicate with a given IoT component through an IoT component application. A widely-used type of application interface is an application programming interface (API).

An overview of the atomic capabilities of an IoT component is provided in Table 5.

Table 5—Atomic capabilities of an IoT component

Atomic capability type	Total arity	In arity	Out arity	Input type	Transform	Output type	Assumptions
Sensing	N	1	1	Physical energy	Aspect of physical system state to representation of aspect of physical entity of interest	Digital data	Intent is to observe a property of the physical entity of interest. Is "read only"—any change to the physical state is an undesired side effect. Measurement errors are introduced by the physical environment between the physical system and the sensor transducer, in the sensor transducer itself, in the analog electrical circuit, in the A/D converter, and in the digital logic of the sensor. There is also a time delay between the sensing and the data becoming available at the component output.
Actuating	N	1	1	Digital data	Representation of change in aspect of physical entity of interest to actual change in aspect of physical entity of interest	Physical energy	Intent is to effect change of state in the physical entity of interest. Errors may be introduced in the digital logic, the D/A converter, the analog electrical circuit, and the actuator transducer. There is a time delay between the input data arriving at the component and the change being made to the environment.
Data processing	N	N	N	Digital data	Set of information to new set of information	Digital data	Intent is to transform digital data. There is no fundamental "lossiness" in digital processing. There is a time delay between the input and output.
Data storing	N	N	N	Digital data	Set of information to set or subset of information available over time	Digital data	Intent is to store data for later use. Data are persistent. Data may be pushed out by the component or provided based on an external request. There is a time delay between the input and output and between a response to a data request and the initial request.
Data transferring	N	N	N	Digital data	Set of information to same set of information available over distance	Digital data	Intent is to move data from one location to another. Location is understood in a logical sense rather than purely physical. There is a time delay between the input and output.

6.6.2.4.7 Representation model kind

6.6.2.4.7.1 Objective

The representation model kind introduces the digital projection, which is an informational representation of entities of interest. It also shows how the digital projection is linked to interactions with the entity of interest and the dependency of the digital projection on other models.

6.6.2.4.7.2 Conventions

6.6.2.4.7.2.1 General

6.6.2.4.7.2.2 Model kind languages or notations

This model is presented in UML 2.2 [B94], particularly the class-diagram pattern and the flow-diagram pattern. Other modeling languages that are capable of representing ontologies are also feasible, including natural languages.

6.6.2.4.7.2.3 Model kind meta-model

Figure 17 models the fundamental relationship between an entity of interest and the digital projection: Digital projections represent entities of interest.

Figure 17—Class diagram of the representation relationship between digital projection and physical entity

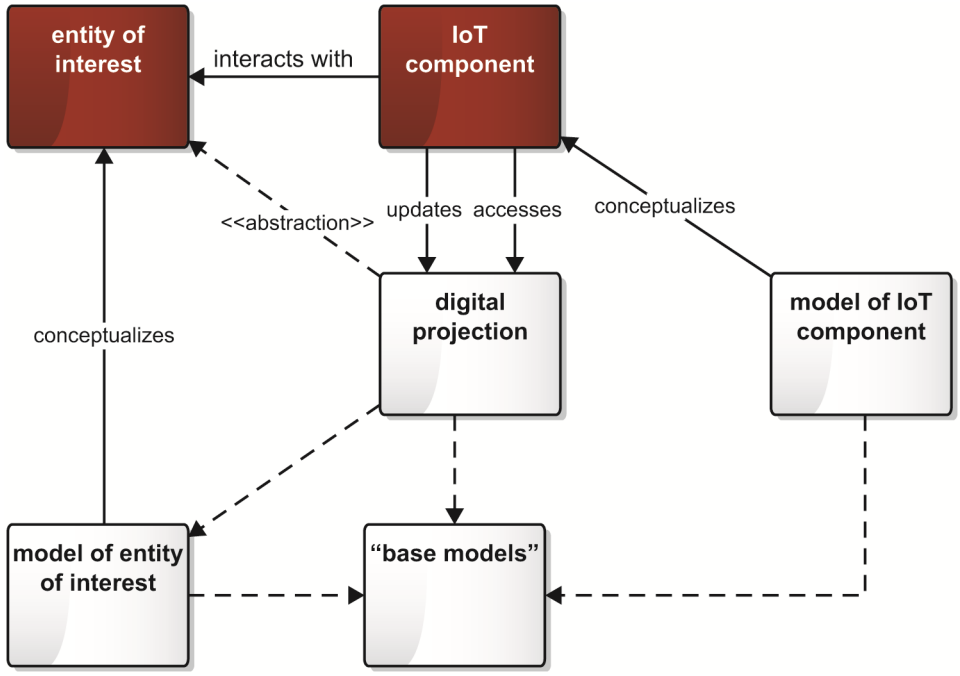


Figure 18—Class diagram of the models that conceptualize the entity of interest and the IoT device

The knowledge that supports the IoT system goal is inferred from the digital projection and the models introduced in Figure 18. There are three models in Figure 18:

— *Model of the entity of interest*: This model in not always "written down." It conceptualizes everything one needs to know about the entity of interest in order to achieve the goal(s) set out for the system (see 6.6.2.4.3).

— *"Base models"*: Contains relevant models. Examples are physical models, SI units, cultural conventions, etc.

— *Model of the IoT component*: This model is often not completely "written down." It conceptualizes everything one needs to know about the IoT device in order to achieve the goal(s) set out for the system (see 6.6.2.4.3).

Figure 19 explains how the models above and the digital projection support the production of knowledge that furthers the IoT system goal.

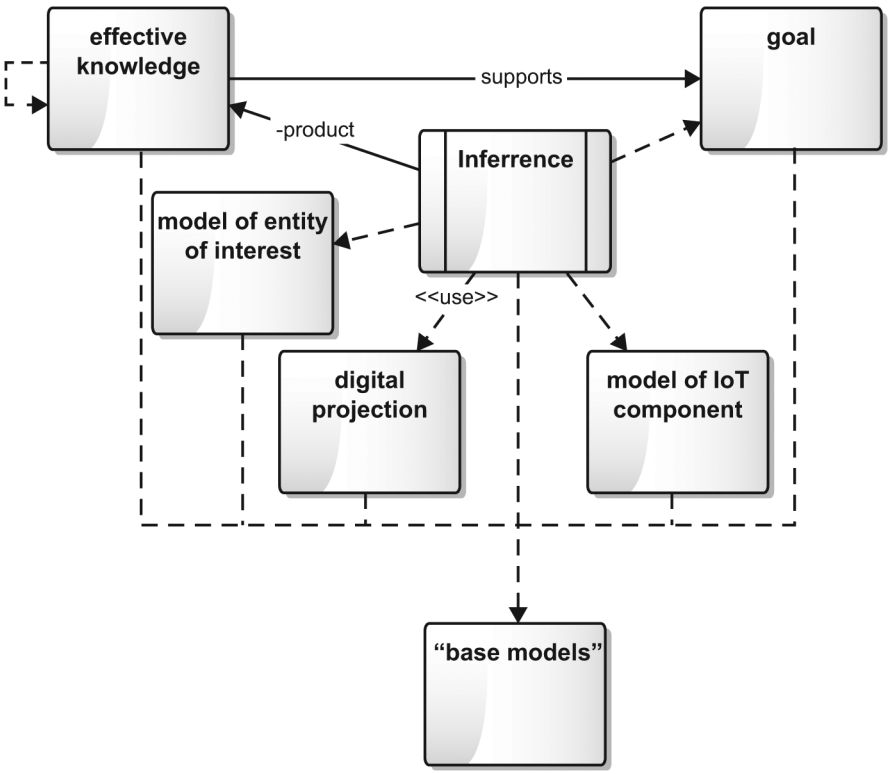


Figure 19—Class diagram of effective knowledge

6.6.2.4.7.3 Correspondence rules

Table 6 provides the correspondence rules of the conceptual viewpoint.

Table 6—Conceptual viewpoint correspondence rules

Correspondence rule #	Viewpoint	Pertinent conceptual viewpoint elements
CNC-CR-1	Compatibility	Uses concepts introduced in the system model kind.
CNC-CR-2	Threat model	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.
CNC-CR-3	Security and safety monitoring	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.
CNC-CR-4	Access control architecture	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.
CNC-CR-5	Adequate design for required security	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.
CNC-CR-6	Communication	Addresses the communication proposition in the system model kind and the communication model in the core conceptual model kind.
CNC-CR-7	Collaboration	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.
CNC-CR-8	Lifecycle	Invokes the concept of a system (system model kind); of particular importance is the boundary proposition.

The goal and the intent in the intent model kind are directly linked to business aspects of the system. The fact that the present document does not provide a business viewpoint is not to be understood that such a viewpoint should not be part of the architecture description of a concrete system of interest. An example for how real-time knowledge about entities of interest can be integrated into the business view has been discussed elsewhere in the literature (Meyer, Sperner, Magerkurth, and Pasquier [B76]).

6.6.2.4.7.4 Notes

As shown in the representation model kind, the digital projection is updated with sensor output, and the digital projection can also contain actuation instructions. This is not an abstract concept, rather the outcome of several cyber entities working together. The finer details of the interworking are illustrated by aid of two examples.

Figure 20 depicts an example of how digital projections are updated by aid of sensor output.

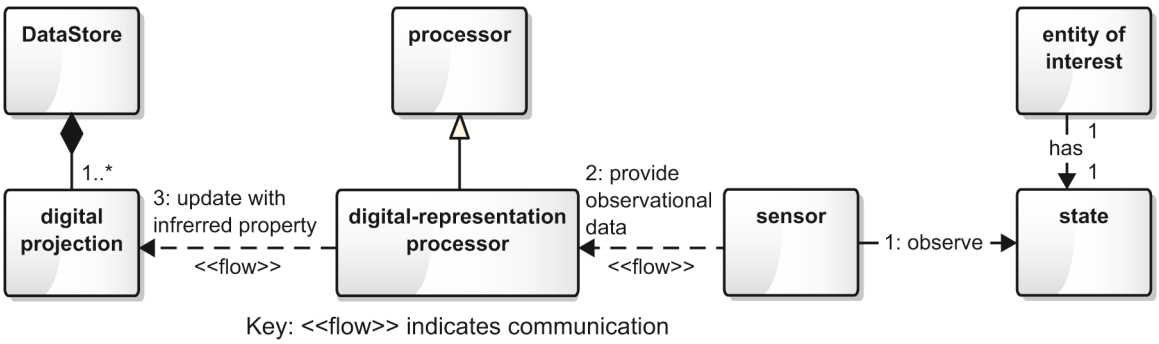


Figure 20—Communication diagram describing the relationship of the entity of interest and the digital projection in the case of sensing

The entity of interest has a state. A sensor observes (part of) this state. The gathered observational data are communicated to a digital-representation processor, which refines the input data. The inferred properties of the entity of interest are used for updating the digital projection of the entity of interest.

Figure 21 depicts an example of how the change of the state of the entity of interest is accomplished.

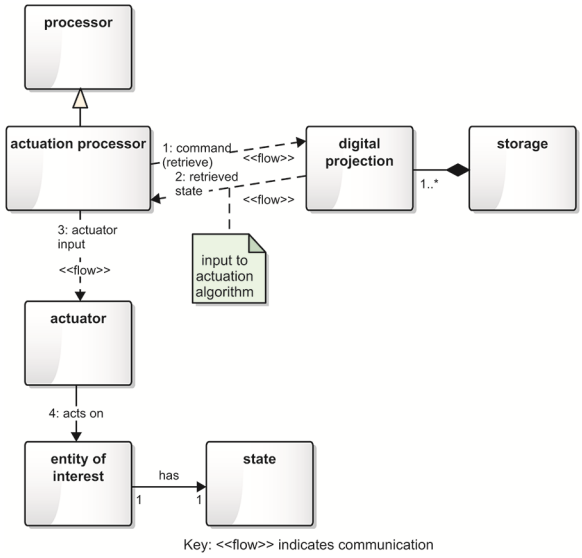


Figure 21—Communication diagram describing the relationship between the digital projection and the entity of interest in the case of actuation

The perceived state of the entity of interest is stored in the digital projection associated with the entity of interest. Based on this information, actuation input is provided to the actuator, which acts upon the entity of interest to change its state.

6.6.3 Compatibility viewpoint

6.6.3.1 General information and key features

The compatibility of systems and parts thereof is one of the driving forces behind standardization. Without any due attention, system development and deployment results in highly incompatible systems. Therefore, if compatibility is sought, cognizant engineering and deployment strategies are needed. A first step in this direction is to detail and classify the compatibility of a system. One tool for classification is the concept of compatibility levels, which form the central model kind of this viewpoint.

6.6.3.2 Stakeholders and concerns

Stakeholders for this viewpoint are as defined in 5.1.

Table 7—Compatibility viewpoint concerns

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Conceptual	How can we aid the formulation and structuring of stakeholder concerns and requirements concerning the cooperation of systems and parts thereof?
Cost	How can we lower the cost of integrating (IoT) devices into systems? How can we minimize the time and effort needed to convince stakeholders that the system meets its obligations for being safe, reliable, resilient, secure, and fulfilling privacy expectations?

6.6.3.3 Model kind: compatibility level

6.6.3.3.1 Compatibility level conventions

6.6.3.3.1.1 General

This model kind adopts the stakeholder concept promoted by ISO/IEC/IEEE 42010:2011.

6.6.3.3.1.2 Model kind languages or notations

In some domains the term *interoperability* is used synonymously to *compatibility*. In this document the term *interoperability* has a specific meaning (see 6.6.3.3.1.3).

6.6.3.3.1.3 Model kind meta-model

This model kind describes the compatibility of devices in terms of hierarchical levels. The levels depend on well-defined communication and application features (see Figure 22). Note that this method of categorization can also be applied to parts of systems and entire systems. If a device fulfills a certain compatibility level, it automatically fulfills all lower compatibility levels except the level of incompatibility. For instance, an interoperable device is also interconnectable.

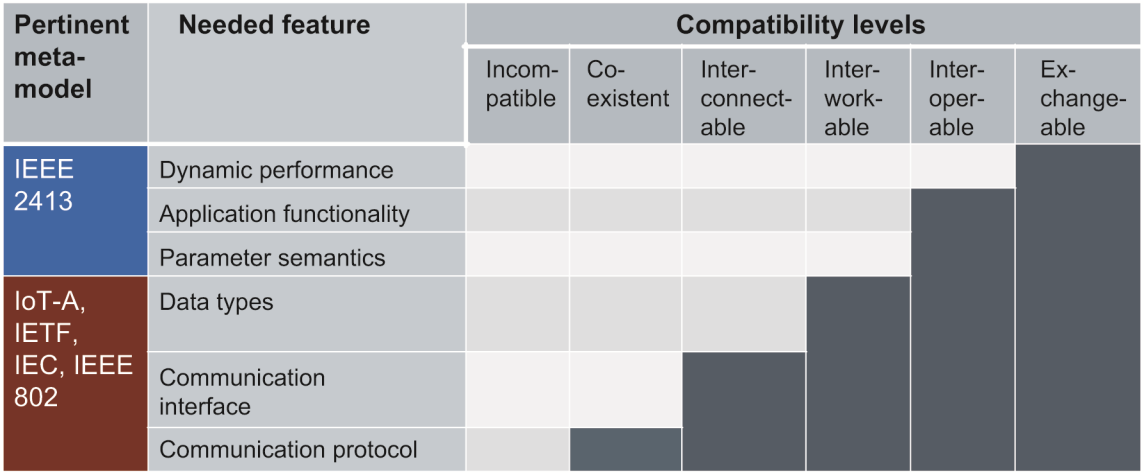


Figure 22—Levels of functional device compatibility (IEC 61804-2:2016 [B44])

Table 8 highlights the main features used for the definition of compatibility levels (see Figure 22).

Table 8—Functionality features

Functionality	Features	Description
Application features	Dynamic performance	This feature is defined by time constraints which influence the data or the general device behavior.
	Application functionality	This feature is defined by specifying the dependencies and consistency rules between the variables.
	Parameter semantics	This feature is defined by the characteristic features of the data (e.g., this can be data name, data descriptions, the data range, the substitute value of the data, the default value, the persistence of the data after power loss and deployment, etc.).
	Data types	This feature is defined by the data type of the block data input, data output, or parameter.
Communication features	Communication interface	This feature is defined by the communication service definition of the involved devices including the services and the service parameters. Additional mapping mechanisms may be necessary. The dynamic performance of the communication system is part of this feature.
	Communication protocol	This feature is defined by all relevant protocols of Layers 1 to 7 of the ISO OSI reference model (ISO/IEC 7498-1 [B62]). The relevant protocol layers depend on the functionality of the involved devices (e.g., Ethernet Switch, Internet Protocol [IP] router).

Regarding these functional features, the compatibility level names in Figure 22 are used for the classification of devices. More information is provided below.

The features defined here are not “absolute.” For example, the cell phone service for connection establishment, continuation, and termination are interoperable. However, from the perspective of a human user, cell phones are only interworkable if there is an agreement on the human language used.

NOTE—The term *interoperability* is often used as a synonym for another compatibility level, *interworkable*. For example, in the case of a network component that does not need to interpret the content of the transmitted data, the term *interoperability* is not wrong.

Compatibility

Compatibility is the ability of a device to provide the set of functions and data required by an application for a specific role in the physical process. If an application requires a set of functions and profile values “S1” at a specific role, then any device containing set “S1” or a superset of “S1” is compatible with that role in the application. The role represents both operating and communication function requirements for a device, the confidence that the IoT system can be trustworthy, and the mechanical attachments and environmental features for surviving in the intended operating location.

NOTE—Trustworthiness can only be achieved by a system. The devices can only contribute to the trustworthiness. Additional measures are necessary, for example a key handling; a single device is not aware of the system behavior.

Incompatibility

Incompatibility is the inability of a device to provide the functions and performance required by its role in a distributed application. Incompatibility can result from differences in device capability at any part of the profile in Figure 22. An incompatible device may interfere with, or prevent, proper communication or functioning (possibly even destructively) if it is connected to a distributed application network.

Coexistence

Coexistence is the ability of two or more devices to operate independently of one another in the same network respecting the common rules for sharing the same medium. Devices sharing the same communication medium or channel can only operate independently within a defined allocation of bandwidth and timing. It is not necessary to have an agreement regarding the communication services. Application- and system-specific programming in one or both devices is generally required in order for coexistent devices to work together in the same distributed application.

Interconnectability

Interconnectability is the ability of two or more devices to operate with one another using the same communications protocols and communication interface. The devices allow data exchange without agreements about the data types. A data-type conversion could become necessary. Unique application-specific programming in one or both devices is generally required for interconnectable devices to function together in the same distributed application.

Interworkability

Interworkability is the ability of two or more devices to support the transfer of data between devices with the shared knowledge of the data types of the data transmitted. If a device is replaced with a similar one of a different manufacture, it may be necessary to reprogram, reconfigure, or reparameterize the device and/or the application to accommodate different semantic uses for data from the new device. A distributed application should be designed to accommodate any unique functionality and dynamic responses of the interworkable devices used in the implementation. These devices can exchange data and parameter values, but the application may need updating to allow for different semantics, application functionality, and dynamics.

Interoperability

Interoperability is the ability of two or more devices to work together in one or more distributed applications with the shared knowledge of the data types and the semantics of the data transmitted. The data input, data output, parameters, their semantics, and the application-related functionality of each device is so defined that, should any device be replaced with a similar one of different manufacture, all distributed applications involving the replaced device will continue to operate as before the replacement, but with possible different dynamic responses. Interoperability is achieved when both a device and a system support the same combination of mandatory and optional parts of the same standard. Manufacturer-specific extensions in devices or systems from different manufacturers may prevent interoperability.

Exchangeability

Exchangeability is the ability of one device to fulfill the same role of another device in the physical operation and distributed application as required by the system design. The exchanged device shall use the device mountings or attachments of its predecessor and comply with the environmental specification of the location of operation. The configuration/setup of the replaced device is used for the new device. Following replacement and setup of an exchangeable device, any distributed applications involving the new device will continue to operate as before the replacement, including identical dynamic responses of the distributed applications.

There is a distinction between exchange of devices and the compatibility level of exchangeability. It is possible to exchange a compatible or coexistent device without influence to the application. The full exchangeability covers the entire features of the compatibility levels.

6.6.3.4 Operations on views

Figure 22 is used for an analysis of the system requirements, in other words they are used for translating stakeholder concerns into hierarchical, standardized compatibility classes. For example, the chosen classes then have implications for implementation and deployment of devices and functions.

6.6.3.5 Correspondence rules

The compatibility viewpoint influences the expression of model kinds in the other viewpoints. For example, a threat analysis for interoperable versus "only" interworkable devices is inherently different. Table 9 describes the compatibility viewpoint correspondence rules.

Table 9—Compatibility viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
CMP-CR-1	Conceptual	The entire concept of the IoT is predicated on the compatibility between and among things.
CMP-CR-2	Compatibility	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind and function viewpoint.
CMP-CR-3	Lifecycle	The communication lifecycle shall be considered throughout the lifecycle model.
CMP-CR-4	Communication	The mathematics of communication model kind and the OSI reference model kind are codependent and inform each other.
CMP-CR-5	Information	The capability to interpret the transferred data over communication channels is the basis for having information instead of data.
CMP-CR-6	Function	The function viewpoint is a prerequisite for the compatibility level of exchangeability.
CMP-CR-7	Threat model	Secure communication depends on the concepts in the threat model and need to be the same on source and destination to match the compatibility level above incompatibility level.
CMP-CR-8	Security and safety monitoring	Secure communication depends on the concepts of security and safety monitoring and needs a high level of compatibility.
CMP-CR-9	Access control architecture	Secure communication depends on the concept of access control and needs a high level of compatibility.
CMP-CR-10	Adequate design for required security	Secure communication depends on adequate design for security and needs a high level of compatibility.
CMP-CR-11	Privacy and trust	Relies on the concepts presented in the communication viewpoint and needs a high level of compatibility.
CMP-CR-12	Collaboration	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind and needs a high level of compatibility.

6.6.4 Lifecycle viewpoint

6.6.4.1 General information and key features

This viewpoint addresses the lifecycle aspect of systems, viewpoints are shaped by the overall lifecycle of a respective systems. For instance, threat vectors are usually different during bootstrapping, utilization, and support. This viewpoint introduces modeling means that help the architect with patterning the other views in the architecture according to the system lifecycle.

Usually, business goals are formulated and translated into a concept of a system that facilitates the achievement of this goal. After this, the system is developed, produced, and put into utilization. The latter can be interspersed by support and upgrade. At the end of its lifecycle, the system is retired and its

components recycled or disposed of. The part of the lifecycle a system transitions through not only impacts its runtime behavior, but it can also be reflected upon, for instance, functions (interfaces for maintenance), information structures (maintenance logs), digital projections, and so on.

6.6.4.2 Stakeholders and concerns

Stakeholders for this viewpoint are as defined in 5.1. Table 10 describes the concerns of lifecycle viewpoint.

Table 10—Lifecycle viewpoint concerns

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Assurance	How can we convince stakeholders that we have met our obligations for being safe, reliable, resilient, secure, and meeting privacy expectations?
Performance	How can the system performance be maintained, or even increased, over the lifecycle of the system?
Maintainability	In conjunction with the performance concern (see above): How can the maintenance phase be kept short, with long productive phases in between?
Evolvability	In conjunction with the performance concern (see above): How can the system lifetime be extended while keeping the down time short?

Note that the model kind provided here does not address the above concerns directly, rather it is an enabler for addressing these concerns in other viewpoints. Therefore, its main leverage is addressed in the correspondence rules.

6.6.4.3 Anti-concerns

Lifecycle costs are an anti-concern for this viewpoint.

6.6.4.4 Model kind: lifecycle sequence

6.6.4.4.1 General

Traditionally, the lifecycle of a subsystem is delineated into finite lifecycle stages (i.e., lifecycle phases). Furthermore, a sequence of lifecycle stages through which the system transitions are defined. A lifecycle consists of the following constructs:

— *Phase:* A distinct stage during the existence of a system. Phases are also referred to as processes. Examples are development phase, production phase, and utilization phase. The phases have a logical order in time, e.g., the utilization phase cannot occur before the development phase. However, a utilization phase can occur both before and after a support phase.

— *Transitions:* At the end of a phase the system transitions into another phase. This transition is assumed to be instantaneous.

— *Repeats:* As mentioned above, certain phases can occur more than once during the lifetime of a system. For instance, utlization can be followed by support and vice versa until the system is retired.

Note that lifecycle sequences do not need to be linear, i.e., sequential. Rather, the introduction of decision points in a lifecycle results in branching. An example is an if-then statement (“if utilization of system below 90% then initiate renewal phase”), which leads to a branch of the sequence. This branch can—but does not have to—join the original sequence branch. Also, parallel lifecycle phases are possible. For instance, a train can still be utilized while the software of the passenger bulletin boards is upgraded.

6.6.4.4.2 Model kind languages or notations

Lifecycle sequences have been subject to standardization (see, e.g., ISO/IEC/IEEE 15288:2015 [B66] and ISO/IEC 12207:2008 [B59]) and they are also elaborated in the scientific literature (see Saaksvuori and Immonen [B129]).

6.6.4.4.3 Lifecycle sequence correspondence rules

Not only entire systems can be modeled by use of lifecycle stages, but also parts or aspects of the system. An example for a part is a subsystem operation on views.

The following list provides aspects to be considered when operating on views:

- The logical order of lifecycle phases needs to be defined by the architect.
- Note that these phases tend to have different names in different domains. The architect needs to heed the tradition of the domain in which the system is deployed. For instance, the utilization phase is also known as operational phase in Smart Manufacturing.
- Typical or specific actions and occurrences during each lifecycle phase need to be listed by the architect.
- Repetitions and the beginning of specific phases (e.g., support) can be set *a posteriori* (for instance, as requested in regulation), and/or repetitions can be triggered by necessity (e.g., due to the unforeseen failure of a system component). Phases that are already foreseen during the conception of a system, for instance regular support phases, need to be spelled out in the lifecycle view. Every lifecycle has a last phase, i.e., the retirement phase.
- In case not all parts/aspects of a system are modeled with the same sequence of lifecycle stages, potential conflicts (e.g., support of a subsystem, while the entire system operates), and joining/synchronization points are to be identified during the concept phase and to be alleviated. For instance, if the utilization and support phase of security components are shorter than what the utilization of the system would be without the security components, this needs to be reflected in the lifecycle view. For instance, do security updates turn the system inoperable, or can the security components be updated during system utilization, resulting in intermittent, parallel lifecycle phases for the security components and the rest of the system?
- Note that the total duration of the lifecyle is the sum of the duration of all lifecycle phases if there are no parallel lifecycle phases.

6.6.4.5 Correspondence rules

The lifecycle viewpoint is used to pattern all other architecture views. For instance, it provides a pattern and timing for the business view. Usually, the development phase incurs investments only, while profit is generated during the utilization phase. Lifecycle aspects are also included in product lifecycle management (Saaksvuori and Immonen [B129]). Another example is the functional view: the phases of a lifecycle inform what functions are needed, and how they have to be deployed and configured. For instance, if the retirement of a system requires generation and archiving of logs, it has to be ensured that the pertinent functions (logging, communication, and storing) are available during this phase, while others may be terminated. This also implies that data formats for the logs are specified in the information view. Another example is a foreseen upgrade of the network addressing scheme (e.g., communication view). Such an upgrade might trigger a new threat analysis of the system (e.g., threat model view). As a rule of thumb, the lifecycle view has correspondences with all other architecture views.

6.6.5 Communication viewpoint

6.6.5.1 General information and key features

Communication is fundamental to the IoT. There exists different compatibility levels of communication systems (see 6.6.3). The communication viewpoint is intended to address the following three compatibility levels: interconnectable, interworkable, and interoperable.

6.6.5.2 Stakeholders and concerns

6.6.5.2.1 Typical stakeholders

Typical stakeholders for the communication viewpoint fit into the categories as described in Clause 5 with some specific names of their roles added to enhance the definitions for this viewpoint. While the stakeholders remain the same for the three problems, the concerns of those stakeholders as described in 6.6.5.2.2 change to fit the problem.

The stakeholders for the communication viewpoint are listed in Table 11.

Table 11—Communication viewpoint stakeholders

Stakeholders for this viewpoint	Stakeholder role	IoT common stakeholder alignment
Component developer	Conceives of, designs, builds, tests, and/or delivers the IoT device for use in practice	Developers
Network designer	Plans, designs, and lays out the network within the scope of the intended system and its use; may also acquire the network components (infrastructure) for deployment (by the designer or by others)	Acquirers, developers, builders
Infrastructure supplier	Supplies the network components (infrastructure) for use by others	Suppliers
Network/system administrator	Operates and maintains the network	Users, maintainers
Owner	Derives benefit from the use of the system (note: the owner may not actually own any of the components, systems, network, etc. as they may be provided on terms such as “as a service” or lease or others)	Users, owners, operators, acquirers
Regulator	Defines the legal, regulatory, and domain-specific norms of practice/rules that the IoT system shall fulfill with regards to safety, reliability, security, privacy, and resilience	Regulators

6.6.5.2.2 Concerns

The concerns given in Table 12 are shared by all three compatibility levels using communication—interconnectable, interworkable, and interoperable (see 6.6.3).

Table 12—Concerns shared by the three levels of communication

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Communication Monitorability Networkability Reliability	Can the stakeholder for the component count on the following to be there when needed? — Connection of the component to the communication network — Communication network itself — Transmission across the network
Communication Privacy Quality Safety Security Uncertainty	Do all components, the network, and the information on the network meet the stakeholder's requirements and risk tolerance for trustworthiness?
Communication Networkability Security	Are components, the network, and the information on the network capable of being secured in a way that achieves the stakeholder's tolerance of risk for a secure connection?
Adaptability Communication Networkability Performance Resilience Usability	Will the component connection to the communication network and the network itself be durable enough to meet the stakeholder's needs for resiliency?
Communication Complexity Deployability Maintainability Manageability Monitorability Physical Quality Reliability Usability	Can the component connection to the communication network and the network itself be monitored, diagnosed, repaired, and/or replaced in a way that addresses the stakeholders' goals for maintainability?
Communication Monitorability Networkability Reliability	Can the stakeholders for the component count on the connections along the communication path between or among components and the integrity of the communication path itself to be there when they are needed?
Communication Engineerability Functionality Performance Produceibility	Do the connections along the communication path between or among components, as well as the communication path itself, perform as intended by the designer of the communication system? Does the exchange of data between or among components perform as intended by the designer of the communication system?
Communication Maintainability Manageability Monitorability Usability	Does the communication path between or among components, as well as the complete communication path, allow the network administrator stakeholder to monitor, control, and maintain the connections of the communication system? Does the syntactic and semantic exchange of data allow the network administrator stakeholder to monitor, control, and maintain the integrity of the data?
Communication Maintainability Manageability Monitorability Usability	Can the connections at the components and along the communication path between and among components, as well as the communication path itself, be maintained in a way that addresses the network administrator mean-time-to-repair (MTTR) goals?
Adaptability Deployability Maintainability Physical Safety Security	Can the components or their connectors and the network infrastructure be replaced easily enough to achieve the stakeholder's goals?

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Cost Enterprise Functionality Measurability Operability Performance Quality Time to market Utility	Do the component connections and the performance of the network contribute to the stakeholder's key performance indicator(s)?
Communication Monitorability Networkability Reliability	Are the connections, the network, and the data there when they are required?

The concerns given in Table 13 are unique to each of the three levels of the communication viewpoint according to the compatibility viewpoint, see 6.6.3.

Table 13—Concerns unique to each communication level

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Discoverability Identity	Interconnectable level: Does the component have an identifier that is unique within the given scope universally unique identifier (UUID)?
Communication Constructivity Deployability Networkability Usability	Interconnectable level: Can a component be designed and constructed in a way that allows it to be easily joined physically and logically (syntactically) to a communication network?
Communication Complexity Deployability Networkability Usability	Interconnectable level: Is the process of bringing the component online on the communication network intuitive and easy enough for the intended stakeholders?
Communication Networkability Standardization	Interworkable level: Is the native language syntax of a component understandable either natively or through a translation function by other components?
Communication Networkability Standardization	Interworkable level: Is the language (e.g., protocol, syntax) of the device capable of being sent and received over the intended communication medium?
Communication Networkability Data semantics Standardization	Interoperable level: Is what the sender sent the same as what the receiver received?
Communication Networkability Data semantics Standardization	Interworkable level: Is the transmission from the sender to the receiver(s) understandable syntactically either natively or through a translation function by other components?
Adaptability Communication Networkability	Interoperable level: Is the communication network capable of expanding or contracting as new components are added or removed without impacting the performance of the system?
Communication Networkability Data semantics	Interoperable level: Is the communication network suitable for the type of information that will be transmitted over the network?

6.6.5.2.3 Anti-concerns

Anti-concerns for the communication viewpoint include the following:

— Anything to do with the use of data/information by components

— Methods for constructing network infrastructure

6.6.5.3 The mathematics of communication model kind

6.6.5.3.1 General

The three layers of the communication viewpoint described earlier also can be loosely correlated to the three distinct problems of communication as described by Weaver [B145]:

— PROBLEM A: How accurately can the symbols of communication be transmitted? (The technical problem)

— PROBLEM B: How precisely do the transmitted symbols convey the desired meaning? (The semantic problem)

— PROBLEM C: How effectively does the received meaning affect conduct in the desired way? (The effectiveness problem)

The three problems described by Weaver can be bridged and aligned with the data, information, knowledge, wisdom (DIKW) pyramid as shown in Figure 23.

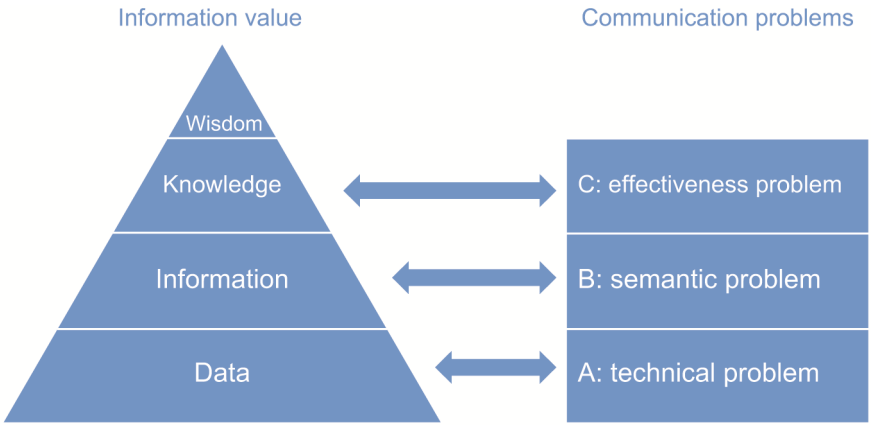


Figure 23—Alignment of DIKW and Weaver

6.6.5.3.2 The mathematics of communication model kind conventions

The mathematics of communication model kind presents an intuitive representation of the flow of communication from source to destination. While Weaver did not anticipate the complexity of the Internet, he did lay down a strong foundation for assessing the path of communication regardless of the architecture, topology, or method of communication. The linkage to the DIKW pyramid acknowledges the hierarchy of transformation: data into information, information into knowledge, knowledge into wisdom. The business perspective of the transformations recognizes the value of the communication content to the overall business goals as it moves from the fundamental technologies of connecting devices and systems together

(e.g., the bits of data) to the understanding of what the data represents (e.g., interpreting the bits as temperature values) to the awareness of how the information should be interpreted (e.g., the fluid temperature inside a vessel).

6.6.5.3.3 The mathematics of communication model kind languages or notations

This model is presented in a simple flow diagram. Other modeling languages that are capable of representing communication flow are also feasible, including natural languages.

6.6.5.3.4 The mathematics of communication model kind meta-model

Weaver envisioned what would today be considered a simplistic view of communications as shown in Figure 24, see NetSTAT [B145].

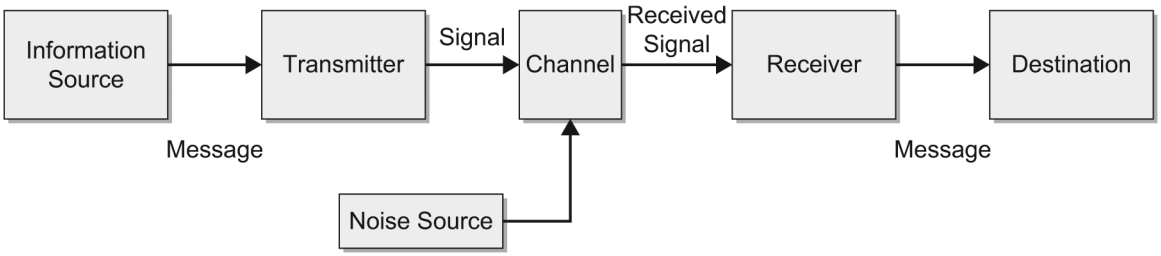


Figure 24—An illustration of communication process

Weaver introduced the concepts of the three problems described above. The messages from the information source to the transmitter and from the receiver to the destination are interpreted to be part of the semantic problem for which there is no widely accepted model. The signal and received signal between transmitter and channel, and channel and receiver are part of the technical problem. The effectiveness problem is not explicitly depicted in Figure 24, but plays an important role in the model.

6.6.5.3.5 The mathematics of communication model kind templates

There is no template for the mathematics of communication model kind.

6.6.5.3.6 The mathematics of communication model kind operations

For purposes of this model kind, it is necessary to understand the relationship of the first three layers of the DIKW pyramid to the three problems of communication.

Technical problem

The technical problem is concerned with the accuracy of transference of data from sender to receiver. With the myriad of varying applications, networks, technologies, and topologies involved in the IoT, ensuring accuracy can be challenging. Communications networks may:

— Provide continuous or store-and-forward type responsiveness

— Provide simplex or duplex transmission

— Support message confirmation or not

— Support high or low bandwidth connections

— Include other elements of communications not listed here

Multiple networks may also be available, each providing different capabilities, and the specific communications path may be selected based on the specific application requirements. Management of the technical aspects of the communication system across the IoT can be very complex.

Semantic problem

The semantic problem is concerned with the identity, or a satisfactorily close approximation, in the interpretation of meaning by the receiver, as compared with the intended meaning of the sender. Semantic interoperability (6.6.6.1.3) and the semantic model kind (6.6.6.3.3) provide more details about the semantic problem in the information viewpoint.

Effectiveness problem

The effectiveness problem is concerned with the success with which the meaning conveyed to the receiver leads to the desired action or actions by the receiver. The function model kind in the function viewpoint (6.6.7.3) addresses the role of the effectiveness problem.

6.6.5.3.7 The mathematics of communication model kind correspondence rules

The mathematics of communication model kind addresses the same topics as the OSI reference model kind in this viewpoint, but from a different perspective. The IoT is highly dependent on communications, and the mathematics of communication model kind is interdependent on all viewpoints. In particular, the three problems align well with the communication viewpoint (A: technical problem), the information viewpoint (B: semantic problem) in 6.6.6, and the function viewpoint (C: effectiveness problem) in 6.6.7.

6.6.5.4 OSI reference model kind

6.6.5.4.1 General

The open systems interconnection model (OSI model) [B62] is widely known in the fields of computing and telecommunications. The model is highly relevant to communications within and between IoT systems. The basic concepts of the OSI model are described in 6.6.5.4.

6.6.5.4.2 OSI reference model kind conventions

6.6.5.4.2.1 General

The OSI model is represented by seven layers that represent how open systems can interconnect and the interactions between and among the layers and with peer systems.

6.6.5.4.2.2 OSI reference model kind languages or notations

There are no specific languages or notations that are specific to the OSI model.

6.6.5.4.2.3 OSI reference model kind meta-model

The OSI model is depicted as a “stack” of the seven layers in Figure 25.

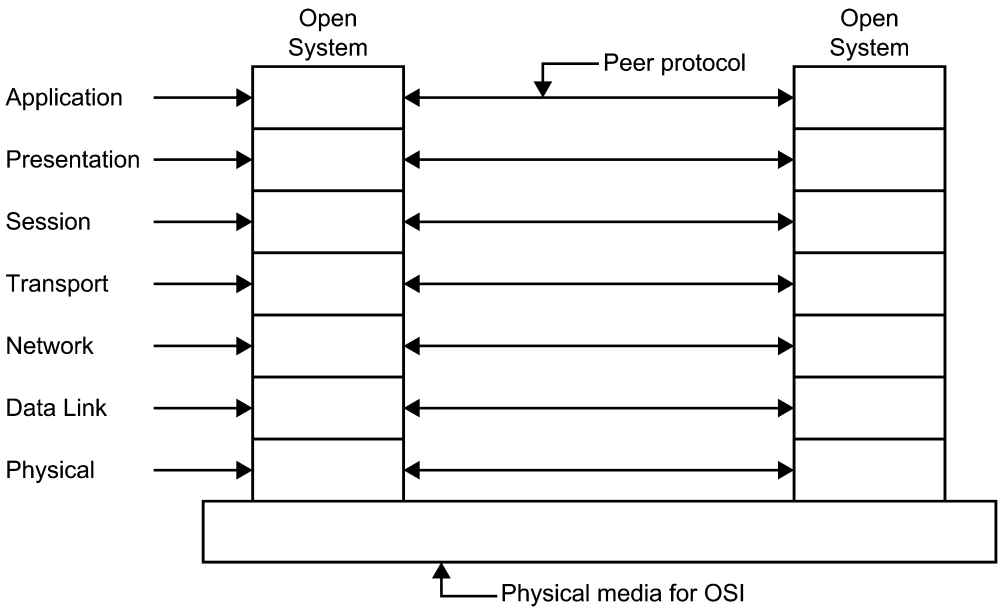


Figure 25—OSI reference model (ISO/IEC 7498-1 [B62])

The stack is useful to assist with the visualization of communications up and down the stack and between systems. The purpose of the model is to provide a framework for addressing communication needs for standards such as this one.

6.6.5.4.2.4 OSI reference model kind templates

There is no template for the OSI model kind.

6.6.5.4.3 OSI reference model kind operations

The OSI reference model is applied when designing communication systems. Each layer of the model defines a group of functions as follows:

- Layer 7, Application Layer: interacts with the target system by implementing file system operations
- Layer 6, Presentation Layer: defines the syntax of the data that is being transferred
- Layer 5, Session Layer: manages sessions between applications (opening, closing, administration)
- Layer 4, Transport Layer: provides data transport to higher layers to help ensure quality and reliability
- Layer 3, Network Layer: performs network routing
- Layer 2, Data Link Layer: defines how the bits for data transmission are organized
- Layer 1, Physical Layer: defines the electrical and physical properties for devices for communication

When strictly implemented, each layer can be changed without impacting other layers. Not all layers are used in every communication system, and in many cases multiple layers are combined to perform relevant functions. The operations and interactions between and among the layers in the OSI reference model are well defined in the model standard and are not repeated here.

6.6.5.4.4 OSI reference model kind correspondence rules

The OSI model kind, while developed at a much later date than the mathematics of communication model kind, provides clarity to how electronic communications can be realized by that model kind in this viewpoint.

6.6.5.5 Operations on views

6.6.5.5.1 General

Following the transformation as described in the DIKW model (see Figure 23), the three layers—data, information, and knowledge—inform the reader about ways to interpret important elements of the communication viewpoint. The interpretations are described in 6.6.5.5.

6.6.5.5.2 Data layer

6.6.5.5.2.1 Application-aware communication

A communication system that is cognizant of operational requirements imposed by the higher order applications is referred to as application-aware communication. Application-aware communication will manage communications based on an agreed set of parameters that the application may request during communication establishment, or on a per-flow or transaction basis depending on requirements. The communication system will use the requested parameters to manage the transfer of data between the entities. The following paragraphs describe some of the parameters that apply to application-aware communication that need to be managed by the communication system for the IoT.

Priority

Prioritization of communications allows an application to determine the relative importance of the data. The degree of importance allows the communications system to determine which data to transfer in the event of congestion caused by impairments, loss, or traffic levels greater than that supported by the system. For example, in IP-based networks, Quality of Service (QoS) mechanisms such as differentiated services codepoint (DSCP or diffserv see RFC 2474 [B55]) allow the higher order protocols to classify each packet or flow to indicate to the communication network the relative importance of the traffic type, and is widely used to prioritize voice traffic over other data types.

Accuracy

Communication networks often have an underlying error rate due to the transmission medium or caused by impairments to the transmission channel. Error detection and correction methods help maximize the performance of the channel.

When the application is able to identify the expected accuracy of the data (independent of priority), the communication system can further manage the data transfer in the event of congestion to achieve the expected data accuracy. An example of this would be an environmental temperature sensor reporting every

five minutes. Due to the relative frequency of reporting compared to the likelihood of the measured parameter changing significantly in the time period (unlikely), the loss of a single measurement (or a number of measurements in a set time period) is unlikely to cause problems with the requesting or processing application. Thus in this case the network could discard traffic if required and still achieve the required accuracy.

When the expected accuracy of the network is unpredictable or unknowable, there are techniques that can be applied to reduce uncertainty or improve the accuracy of the communication (e.g., retries, checksums, cyclic redundancy checks (CRCs), etc.).

Timing

When communicating in the IoT, the notion of time and timing has multiple uses:

- The collection of information or actuation of control may be performed at set timing intervals or at a specific time. In these cases, the communication network may be controlled to meet the timing requirements to conserve resources such as power.
- To meet performance requirements, some applications may have delivery requirements that are dependent not only on the raw performance (speed) of the communication, but also on the latency and jitter time variances experienced in the communication path(s). Techniques can be applied to manage latency and jitter to achieve the communication goals.
- Assigning time information (e.g., time and date) to data is an important concept in many domains to allow data to be analyzed and correlated with other data and events. The communication networks play an important role in the transmission of the time and data together.
- The distribution of timing reference information can also be done by the communication network to provide synchronization between the nodes, or alternatively nodes may synchronize themselves to external timing references (such as GPS).

Assurance

In the context of application-aware communication, assurance relates to the ability to not only guarantee delivery for certain data types; it may also involve the receiver sending a confirmation that data has been received to provide indication back to the application.

6.6.5.5.2.2 Network-aware communication

There are an almost endless set of communication networks applicable to the IoT. Each of these networks brings a set of capabilities to the communication system based on the functionality and operation of the technologies involved. In many cases, more than one communication network may be available to the system and often the choice of which network to use will be based on a set of parameters that are matched to the needs of the higher order application. This is referred to as network-aware communication. The following paragraphs describe some of the parameters that apply to network-aware communication that need to be managed or addressed by the communication system for the IoT.

Resource constraints

A communication system for the IoT will likely consist of a number of communication networks that are combined or selected to provide the required connectivity between application and endpoint. Depending on the specific networks and technologies available, the network will have a set of end-to-end constraints that shall be made aware to the application to help ensure that it does not request capabilities that are not available, thus causing the information delivery to fail. Alternatively, an application may need to reduce its functionality to accommodate communication needs that cannot be fulfilled by the network. An obvious

resource constraint is available network bandwidth. Other constraints include packet size, transmission delay, error rates, and others. These constraints may also not be static; they may change based on receiver location, other traffic flows, or network impairments.

Connection availability and negotiation

There are many communication network options for the IoT that do not involve a direct high-capacity wired connection and thus rely upon connections that may not always be connected or available. In some cases, more than one network option may be available and the negotiation and selection of the most appropriate option may be performed. The options may also change periodically and the communication system shall adjust accordingly. For applications in the IoT that are scheduled or periodic in nature, matching the communication system capabilities with the overall application requirements with respect to periodic or scheduled connectivity will suffice. Applications that require an autonomous, event-driven response shall have additional capabilities supported by the communication system.

The ability for a network to adapt to different domains, scale from small to extremely large, and manage its health and behaviors describes an autonomic network. Autonomic capabilities are an important consideration for communication systems. For example, collection of environmental readings for forest management can be easily met with a periodic connection where the network is available at a designated frequency (e.g., every 30 min), but is inadequate when the sensors are also detecting events such as the ignition of a fire, where the delay to the next periodic time slot (30 min) can cause significant detriment to the overall application. Each of these scenarios could be modeled as separate applications and the communications could be adjusted to meet the requirements of each application. The balance of network resource availability and awareness shall be included as part of the overall communication system.

6.6.5.5.2.3 Topologies and hierarchies for IoT

Along with the myriad of communication networks and technologies applicable to the IoT, there are a wide range of topologies that can be applied and combined to provide the end-to-end communication system. Topology choices can be made to implement specific requirements and to achieve specific objectives. For example, a mesh topology could be used to achieve specific requirements for reliability/resilience in a local context.

Unary network topologies

There are a wide range of network topologies that apply to the IoT. The choice of topology depends upon a number of parameters, including but not limited to available media, transmission paths, applicable technologies, required bandwidth, etc. Unary network topologies that can be used in the IoT include but are not limited to star, ring, bus/linear, and mesh.

Hierarchical network topologies

Hierarchical topologies are typically combinations of the unary topologies. For example, a tree topology is hierarchical and is composed of a combination of bus and star topologies that interlink various network segments together. For the IoT, it is important to consider the elements that construct the interfaces in the hierarchical topologies (e.g., switches, routers, gateways) from the perspective of network availability and performance attributes. As described above, the assurance of data delivery shall be performed in a way that meets the requirements of the application.

6.6.5.5.3 Information and knowledge layers

The information perspective is aligned with the semantic problem and the effectiveness problem informs the knowledge perspective. Both perspectives are addressed in the information viewpoint in 6.6.6.

6.6.5.6 Correspondence rules

Table 14 describes the communication viewpoint correspondence rules.

Table 14—Communication viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
COM-CR-1	Conceptual	The entire concept of the IoT is predicated on the communication between and among things.
COM-CR-2	Compatibility	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind.
COM-CR-3	Lifecycle	The communication lifecycle shall be considered throughout the lifecycle model.
COM-CR-4	Communication	The mathematics of communication model kind and the OSI reference model kind are codependent and inform each other.
COM-CR-5	Information	The transformation of data to information to knowledge is often dependent on the transfer of data over communication channels.
COM-CR-6	Function	May be dependent on concepts presented in the communication viewpoint as functions are performed across communication channels.
COM-CR-7	Threat model	Secure communication is dependent on the concepts in the threat model.
COM-CR-8	Security and safety monitoring	Secure communication is dependent on the concepts of security and safety monitoring.
COM-CR-9	Access control architecture	Secure communication is dependent on the concept of access control.
COM-CR-10	Adequate design for required security	Secure communication is dependent on adequate design for security.
COM-CR-11	Privacy and trust	Relies on the concepts presented in the communication viewpoint.
COM-CR-12	Collaboration	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind.

6.6.6 Information viewpoint

6.6.6.1 General information and key features

6.6.6.1.1 General

As mentioned in Clause 1, an IoT system can be seen as “a system of entities (including cyber-physical devices, information resources, and people) that exchange information and interact with the physical world by sensing, processing information, and actuating.”

This definition emphasizes the essential role of information in any IoT system, however information is both pervasive and multi-faceted, and as such deserves careful attention in the architecture definition and description process.

The information view resulting from this viewpoint shall describe how information is semantically defined, structured, stored, shared, manipulated, managed, and distributed across the IoT system.

Based on Murdock, Bassbouss, Kraft, and Wang [B78], this information view shall include both static information structure and dynamic information flow models in order to describe how system-level

information is structured, exchanged, and understood, but also to document key information attributes such as ownerships, relationships, quality, and persistence. The key point here is to focus on architecturally significant information items, i.e., information elements that may affect the system as a whole and not specificities that are only relevant for a subsystem or design-level aspects.

As IoT systems range from simple connected sensors to complex cross-domain systems composed by numerous and heterogeneous subsystems, the main goal for the information view should be to document system-level information, e.g., information exchanged between the various subsystems. This is a key concern for system designers as informational representation can be very heterogeneous both in terms of syntax and semantics between the different subsystems, in particular when third-party interoperability is involved.

Regarding operational data, the primary source for this type of information will be in most cases the system functional decomposition, potentially organized by functional domains, which should lead to the identification of key information items related to system monitoring, control, management, and maintenance. Then, for each functional domain, the system architect should identify architecturally significant information attributes. For instance, information quality is certainly key in both monitoring and control of an industrial system, whereas information ownerships will affect the management and security functions.

However, system information is not limited to operational data but includes commercial data (managed for example by a partner relationship management system), offer data (typically managed by online catalogs) and engineering data (both physical, e.g., building information model, and parameterization data). A key value of an IoT system is the ability to combine those sources of information to get better knowledge of the system, its state and its usage.

Another key consideration to take into account when architecting/designing an IoT system is that this system may be used in ways unthought-of at design time. This is a fundamental aspect of IoT systems, which leads to requirements in terms of openness and standard interfaces that clearly impact the information viewpoint. In that sense, system information that ought to be made available to future usage should be clearly defined and have unambiguous access control attributes.

The following subclauses provide background definitions of concepts that underpin the information viewpoint.

6.6.6.1.2 Information, knowledge, and wisdom

Information is often defined through its relationships with data and knowledge, as illustrated by the DIKW pyramid (Rowley [B126]) (see Figure 26).

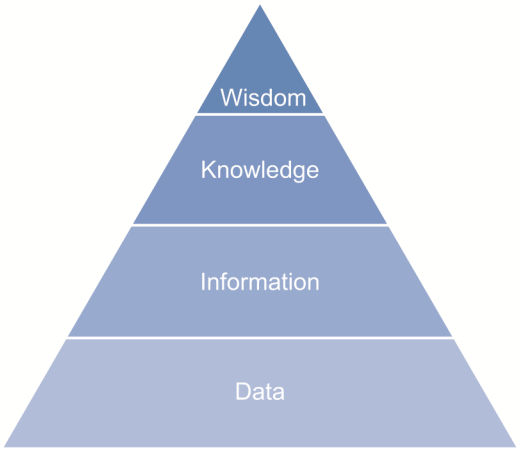


Figure 26—Data, information, knowledge, wisdom (DIKW) pyramid

In this representation:

- Data represents the **raw** bits and bytes provided by data sources, ranging from physical sensors to cloud analytics: *e.g., data = "Red, 192.234.235.245"*
- Information corresponds to **meaning** inferred from data when completed with description, including "who," "what," "where," and "when"; *e.g., information = "South facing traffic light on corner of Pitt and Georges streets has turned red"*
- Knowledge can be derived from information when put in **context**; *e.g., "The traffic light I am driving toward has turned red"*
- Wisdom can be defined as the ability to take efficient **decisions** based on knowledge: *e.g., "I better stop the car!"*

6.6.6.1.3 Semantic interoperability

Interoperability has been defined in 6.6.3. In 6.6.6.1.3 the focus is on semantic interoperability, where semantic interoperability means "enabling different agents, services, and applications to exchange information, data, and knowledge in a meaningful way, on and off the web" (Rowley [B126]).

Semantic interoperability can be achieved either by sharing (statically) a common data model/semantics or by dynamically binding with existing shared vocabularies, typically in the form of ontologies. Vocabulary can be represented at catalogue-oriented ontology (like IEC 61987 [B47] according to IEC 61360-6 [B40]) or based on semantic web technologies (rdf, rdfs, OWL, etc.). 6.6.6.1.4 will detail the concept of ontology.

Semantic interoperability can be illustrated by Figure 27 where devices using different data models have to exchange information with a cross-domain application. When such devices are communicating with other devices using the same data model (e.g., IEC 61850 [B45], OPC Unified Architecture [OPC UA] according to IEC 62541 [B50]) the semantic interoperability is ensured because the knowledge of a standardized set of information is hardcoded within the devices. However,. a cross-domain application connected to these devices shall not contain any hardcoded knowledge of the different data models and shall rather rely on a more generic approach like ontologies.

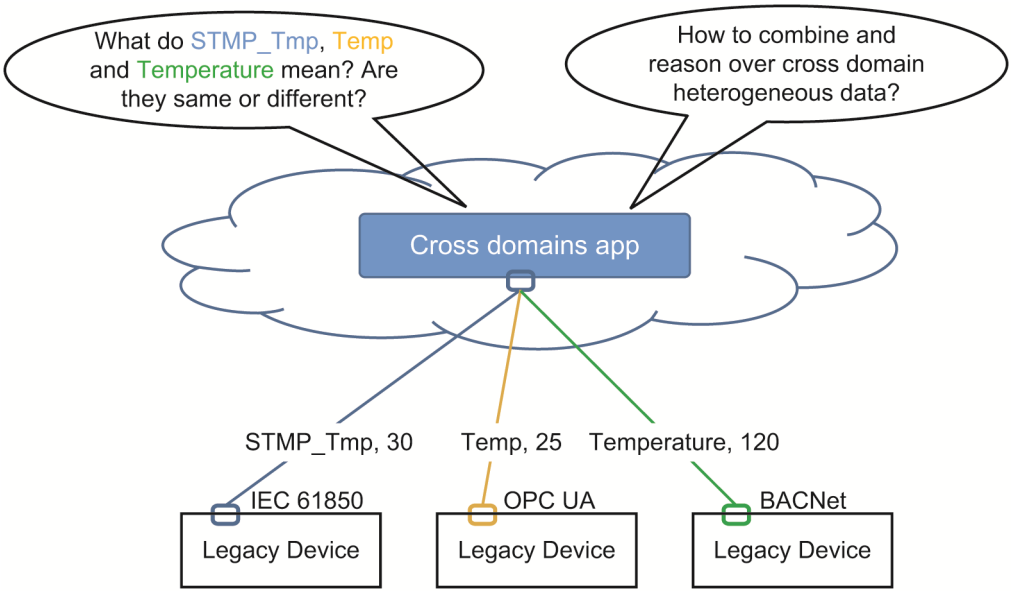


Figure 27—Semantic interoperability

6.6.6.1.4 Ontologies

Ontologies extend the concept of taxonomies to capture the concepts and their relationships in a given domain. With ontologies, richer operations like reasoning and inference can be performed on system information. As mentioned in Rowley [B126], "Ontologies build on metadata to provide a representation of knowledge about a given domain and to provide a core resource for reasoning about a domain and a context."

6.6.6.2 Stakeholders and concerns

6.6.6.2.1 Typical stakeholders

Most stakeholders are likely to be interested by system information in one form or another, especially when considering the overall information spectrum, ranging from commercial to runtime data. Here are some key stakeholders for information:

- Designers
- Developers/integrators
- Vendors/suppliers
- Installers
- Operators
- Users
- Maintainers
- Standardization bodies

6.6.6.2.2 Concerns

The concerns listed in Table 15, Table 16, and Table 17 have been identified for the information viewpoint.

Table 15—Concerns related to information ownership, conflict, and synchronization

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Identity responsibility communication	Who are the information owners and/or information producers?
Identity responsibility	Are there possibilities of having conflicts on the ownership or production of information?
Relationship between data	Will any conflict lead to incomplete/incorrect information?
Manageability	How to handle these conflicts? Are there any rules for this?
Synchronization time awareness	Is synchronization required to make information consistent and up-to-date?
Synchronization responsibility	Who will help ensure such synchronization and how it will be achieved?

Table 16—Concerns related to information purpose and usage

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Data semantics Complexity	Do we have any information which is complex or poorly understood by the entities?
Data semantics	How can we help ensure that entities have access to data together with the appropriate level of semantics and context?
Physical context Relationship between data	Do we have information from external sources?
Operations on data	Do we need to store (some or all) information? (e.g., for audit, analytics)
Operations on data	Which information is persistent and which information is volatile?
Relationship between data Constructivity	How can we combine different types of information to increase the knowledge about the system?
Adaptability Evolvability	Which information should be made available for future use, and how should it be made available?

Table 17—Concerns related to information structure and content

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Time awareness	Is the information static or periodically updated?
Data semantics Operations on data	At any stage do we need to transform the information to a “universal” format for all entities to understand and use it?
Safety Security	Is it possible to categorize information as critical and non-critical?

6.6.6.3 Model kinds

6.6.6.3.1 General

The main model kinds for any information view intended to describe an IoT system architecture include:

— **Data** model kind describing how system-level data are represented

— **Semantic** model kind describing the main ontologies used throughout the system (describes essentially the meaning of data)

— **Information structure** model kind describing how information is statically distributed over the overall system

— **Information flow** model kind describing how the main information items are dynamically exchanged throughout the system

— **Information lifecycle** model kind describing the main steps in the life of an information item, from creation to deletion

6.6.6.3.2 Data model kind

6.6.6.3.2.1 General

The data model should provide a coherent and homogeneous representation for system-wide data items.

A key aspect to consider for reaching such an objective is that data syntax and semantics should be decoupled. This will facilitate future evolutions of the data model, impacting either the syntax or the semantics of the data. This implies structuring the data model around a layered scheme, where the data are represented through the combination of an abstract **meta-model** and **data model** whereas the information semantics will be represented through a **semantic model**, as defined in 6.6.6.3.3.

It is worth noting that mapping of domain specific or proprietary data models to the system-level data model is not the responsibility of the system architect, but should be expected from the system designers and documented in the system specification document.

6.6.6.3.2.2 Data model conventions

6.6.6.3.2.2.1 Model kind languages or notations

The convention is to use UML and/or SysML as the modeling language, typically SysML block definition diagrams.

6.6.6.3.2.2.2 Model kind meta-model

A meta-model is highly recommended as the underlying model for the data model kind. This meta-model defines the basic building blocks and rules for defining a proper data model, so that different data models (potentially from different business domains) may be mapped to the same meta-model, which in turn can be mapped to different methods of information exchange. This mapping of a meta-model to a method of information exchange will allow manipulation of various data models in a consistent way, i.e., with the same application programming interface (API), thus greatly simplifying the integration effort.

6.6.6.3.2.3 Data model operations

No specific data model operations are identified.

6.6.6.3.2.4 Data model correspondence rules

The data model kind should be linkable with the semantic model kind (see 6.6.6.3.3), i.e., the corresponding meta-model should include specific placeholders for semantics tagging. For instance, a device meta-model may include attributes such as *Location* and *Usage*.

6.6.6.3.3 Semantic model kind

6.6.6.3.3.1 General

The semantic model is represented as an ontology formalizing the system concepts and their relationships. It is envisioned that one high-level semantic model may be defined for the IoT system globally in order to allow reasoning about the system, in particular for maintenance purposes.

It is worth noting that other domain-specific ontologies might be useful for reasoning about the system at runtime, although they will probably not be under the responsibility of the system architect/designer, but rather provided by domain experts.

6.6.6.3.3.2 Data model conventions

6.6.6.3.3.2.1 Model kind languages or notations

The convention is to use OWL modeling language, in particular OWL ontology definition diagrams, and alternatively UML and/or SysML, with either SysML block definition diagrams or UML class diagrams.

6.6.6.3.3.3 Data model operations

No specific data model operations are identified.

6.6.6.3.3.4 Data model correspondence rules

The data model kind should be linkable with the meta-model and data model kinds (see 6.6.6.3.2).

6.6.6.3.4 Information structure model kind

6.6.6.3.4.1 General

The information structure model should describe how system-level information is structured throughout the system, in particular the relationships between information items. As previously, the focus here is on system-level information, as opposed to subsystems specific information items.

6.6.6.3.4.2 Data model conventions

6.6.6.3.4.2.1 Model kind languages or notations

The convention is to use UML and/or SysML as the modeling language, typically SysML internal block diagrams.

6.6.6.3.4.3 Data model operations

No specific data model operations are identified.

6.6.6.3.4.4 Data model correspondence rules

No specific data model correspondence rules are identified.

6.6.6.3.5 Information flow model kind

6.6.6.3.5.1 General

The information flow model should describe how system-level information is exchanged throughout the system. Each subsystem is considered as a black box and only its external ports/interfaces should be considered, although the notion of system is recursive/fractal and a certain level of decomposition may be useful.

6.6.6.3.5.2 Data model conventions

6.6.6.3.5.2.1 Model kind languages or notations

The convention is to use UML and/or SysML as the modeling language, typically SysML internal block diagrams.

6.6.6.3.5.3 Data model operations

No specific data model operations are identified.

6.6.6.3.5.4 Data model correspondence rules

No specific data model correspondence rules are identified.

6.6.6.3.6 Information lifecycle model kind

The information lifecycle model should capture the main stages in the life of system-level information items. Main expectations from this model is to provide guidance on when an information item is created and discarded as well as the main steps in the information journey, including where and when the main transformations are applied (e.g., is the context information added at the device, at the cloud level, or at

some intermediate level such as at the edge/fog?) and important steps such as storage, publishing, filtering, and aggregation.

6.6.6.3.6.1 Data model conventions

6.6.6.3.6.1.1 Model kind languages or notations

The convention is to use UML and SysML as the modeling language, typically an UML/SysML activity diagram.

6.6.6.3.6.2 Data model operations

No specific data model operations are identified.

6.6.6.3.6.3 Data model correspondence rules

No specific data model correspondence rules are identified.

6.6.7 Function viewpoint

6.6.7.1 General information and key features

Figure 12 shows in the thing description the function as an important part of the thing. The function is defined as the intended purpose of the thing or its characteristic action.

Actuation, sensing, analysis, and control of entities of interest are the fundamental interaction patterns in IoT systems. Once a physical property has been observed or measured using a sensor and the signal converted into a digital representation, the data/information is processed to create useful knowledge. This is done by processors that combine the sensor data with data from other sensors, other data sources, and *a priori* knowledge stored in archives or other sources. In the case of complex IoT systems, this is likely a combination of information from a variety of heterogeneous sources. Processing data/information is part of the intended functionality of IoT.

Functions implement the application (sensing, actuation, control, and monitoring) by connecting their data inputs and data outputs according to the logic, rules, or policies of the function to achieve the intended results.

Part of a function can be a process. A process is a sequence of chemical, physical, or biological activities for the conversion, transport, or storage of material or energy. Processes can generally be classified as continuous, discrete parts manufacturing or batch. How a process is classified depends on whether the output from the process appears in a continuous flow (continuous), in finite quantities of parts (discrete parts manufacturing), or in finite quantities of material (batches). Batch processing aspects may apply to discrete parts manufacturing or continuous processes if, for example, the initial start, the operation, and the switch off is considered as a batch process.

The batch processes lead to the production of finite quantities of material (batches) by subjecting quantities of input materials to a defined order of processing actions using one or more pieces of equipment. The product produced by a batch process is called a *batch*. Batch processes are discontinuous processes. Batch processes are neither discrete nor continuous; however, they have characteristics of both.

6.6.7.2 Stakeholder and concerns

6.6.7.2.1 Typical stakeholders

The stakeholders for the functional viewpoint include the following:

- IoT device developer (developers)
- IoT system developer (developers)
- IoT device maintainer (maintainers)
- IoT system maintainers (maintainers)
- See also 5.1

6.6.7.2.2 Concerns

Table 18 provides the concerns for the function viewpoint.

Table 18—Concerns of function viewpoint

Concern	Concern question
Constructivity Engineerability	Can the IoT device be designed and developed using predefined functions and/or function blocks?
Constructivity Deployability Engineerability	Can an IoT system be designed and developed using predefined functions and/or function blocks?
Constructivity Engineerability	Are IoT functions and function blocks easy to develop and easily composed of existing functions?
Adaptability Cost Maintainability	Are IoT functions and function blocks easy to maintain and troubleshoot when something goes wrong with the device?
Adaptability Cost Maintainability	Are IoT functions and function blocks easy to maintain and troubleshoot when something goes wrong with the system?

6.6.7.3 Model kind: function

6.6.7.3.1 Function conventions

6.6.7.3.1.1 General

The function of an IoT device or an IoT system represents the thing. The function is the transformation of a number of input elements of an entity to a number of output elements to perform the intended functionally. The transformation can be done once, continuously, or repeated over time with discrete time periods or non-periodic.

The IoT device or an IoT system can also have properties to describe behaviors and physical properties such as mechanical, electrical, chemical, etc.

Communication interfaces enable the distribution of functions, so that an IoT system and even a system of systems can be created with an overall intended functionally. Failure states based on errors of intended functionally can be part of the function.

6.6.7.3.1.2 Model kind languages or notations

There are many ways to describe or represent a function. Some functions may be defined by a formula or algorithm that tells how to compute the output for a given input. Others are given by a picture, called the *graph* of the function. In science, functions are sometimes defined by a table that gives the outputs for selected inputs. A function could be described implicitly, for example as the inverse to another function or as a solution of a differential equation. The function can be based on a continuous process like a formula $x = a + b$, or discontinuously like a recipe for how to brew beer.

6.6.7.3.1.3 Model kind meta-model

This model kind describes the function of an IoT device or an IoT system.

Functions encapsulate variables and their processing method, algorithm, logic, or formula. The variables and processing method, algorithm, logic, or formula are those required by the design of the application.

The functions of different applications—such as system engineering, supervisory systems, and maintenance systems—can handle or interact with the specific functions of the application to create the overall function.

A processing method, algorithm, logic, or formula defined for a function in the conceptual model is not necessarily mapped one-to-one to a device; they can be mapped to a device, a proxy, a supervisory station, or the cloud if the current technology does not solve it in the device.

NOTE—In the following text the term *algorithm* comprises processing method, algorithm, logic, and formula.

Devices implement algorithms derived out of the design of the controlled process in terms of functions, often times represented by function blocks (FB). The devices are hardware and software modular, see for example, Figure 28. The components of devices are Modules, Blocks, Variables, and Algorithms. There are defined relations between the components that are specified in the UML class diagram below, see Figure 31.

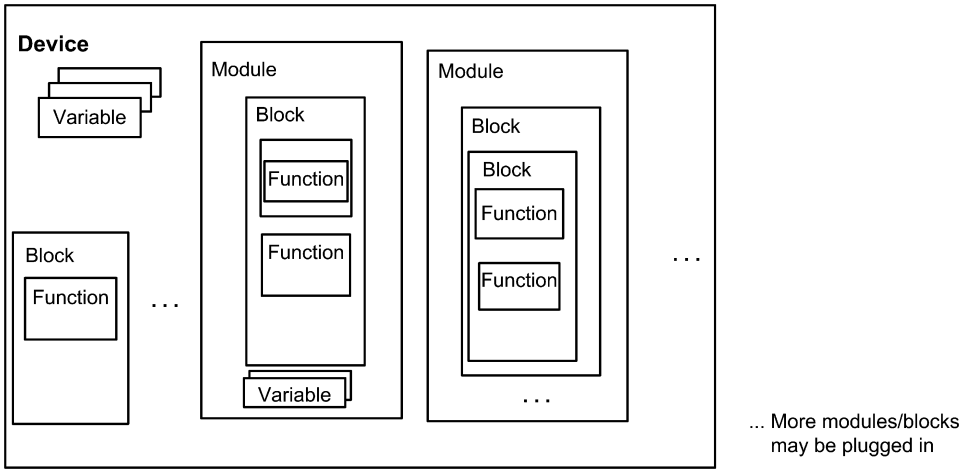


Figure 28—General components of devices

There are different block types (see Figure 29), which encapsulate the specific functionality of devices performing an application. The Technology Block represents the process attachment of a device containing the sensing or actuation principles of a device. The Technology Block is composed of acquisition or output and transformation parts. The application FB (hereafter called *FB*) contains application-related signal processing, such as scaling, alarm, or fault detection or control and calculation. Component FBs may perform mathematical and logical processing with specific additional exception handling procedures such as for example non-authorized parameter values. They shall be encapsulated within composite FBs.

The Device Block represents the resource of the device that contains information and function about the device itself, the operating system of the device, and the device hardware. The device shall have an interface to the communication system and may have system management functionalities.

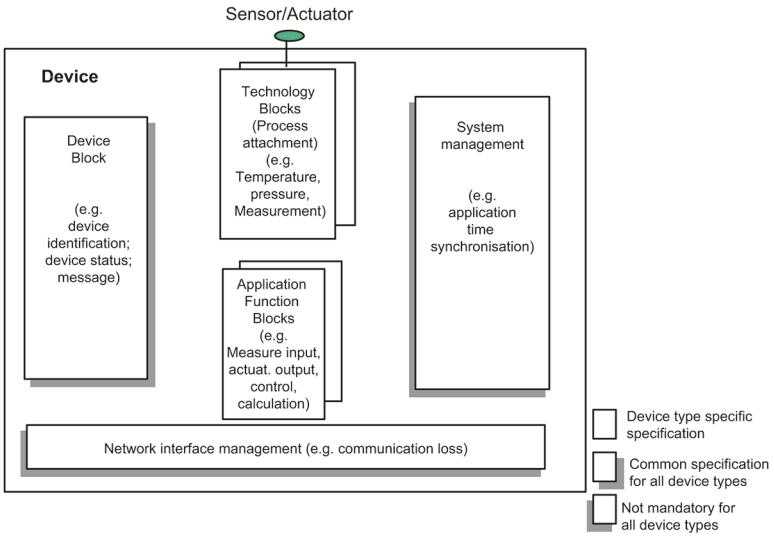


Figure 29—Block types of a device

The number and types of blocks, which are instantiated in a device, are device and manufacturer specific.

There is a data flow chain from signal detection through the Technology Block and FBs and vice versa. The signals between the parts of the chain are internal within the blocks or visible as linkages between blocks. The logical chain of technology and FB is called a *channel*. This concept is clarified in 6.6.7.3.10 and 6.6.7.3.11.

6.6.7.3.2 FB type

FBs are functional units in software that encapsulate variables and algorithms. An FB type is defined by its behavior. One FB contains one or more than one algorithm. The description of an FB is a list of algorithms, which are encapsulated in the FB together with the related data inputs and data outputs and parameters. There are algorithms that are related to the process signal flow and others that are related to other block specific algorithms. These other algorithms are called *management*. Parameters are related to process signal flow and management.

The table in Figure 30 depicts all the needed accessible data inputs, data outputs, and parameters of the FB.

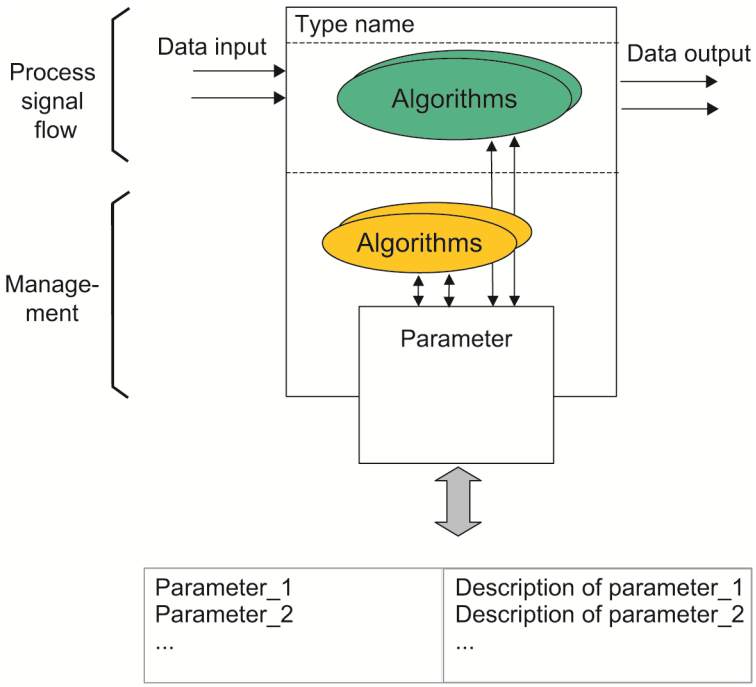


Figure 30—Block overview

NOTE—The graphical representation in Figure 30 is not normative. In other words, the data inputs and data outputs represent the intention of the process signal flow (conceptual definition), not the specific data that carry the corresponding values.

The FB is summarized by the following components:

a) Data inputs[8] which support status[9] and are related to the process signal flow only

b) Parameters[5] which are related to the process signal flow and management

c) Parameters[5] to influence functions

d) Parameters[5] to notify and make visible internal behavior

e) Parameters[5] to select functions in the signal flow

f) Internal variables with memory for support of housekeeping capabilities such as initialization

g) Mathematical/logical algorithm

FB behavior, and hence device or thing behavior, is influenced by data inputs and parameters only. The data inputs and parameters are used in the following ways:

— Data, which are used as inputs or outputs of functions (e.g., setpoint for scaling functions)

— Data, which are used as parameter of functions (e.g., limits for alarms and warnings)

— Changes of parameter data are interpreted as events which switch transitions of state automata (e.g., start, stop, resumption of operation modus of devices)

[8] The decision which data of an FB is a data input, data output, or parameter depends on specific implementations.
[9] For consistency reasons data input/data output and status are in one structure, so that both belong to each other.

— Changes of parameter data are interpreted as events, which start the transactions of sequences of algorithms (e.g., start of calibration procedures)

The data name and their description shall be checked to understand the purpose of the data.

6.6.7.3.3 FB execution

There are different execution control methods within devices. Execution control of FB algorithms is a feature of each device. There can be different execution policies within devices and in a distributed system.

NOTE—For example, combinations of the following execution control methods are possible and others can be added:

— Free running
— Device internal time schedule (time synchronization)
— Device internal event triggered
— Parameter data changes are interpreted as events (see 6.6.7.3.2)
— System-wide time synchronization (time synchronization across the communication system)
— Communication service triggered
— System-wide event triggered (e.g., IEC 61499-1 [B41])
— Distributed execution control

The FB execution control within a device is only one aspect of the overall application execution control. The overall execution control is determined by, for example:

a) Sequence order (sequential or parallel):
    1) Execution order of blocks along the signal flow
    2) Piping of data in parallel execution
    3) Handling of loss of communication between devices
b) Synchronization:
    1) Time synchronization between device
        i) Use of time in scheduling
c) Time constraints; the following elements are covered:
    1) Block execution time
        i) Communication time delay
        ii) Scan rate of measurement
        iii) Actuation time
        iv) Choice of block algorithms
        v) Time delay resulting of communication behavior
d) Block execution time:
    1) Communication time delay
    2) Scan rate of measurement
    3) Actuation time
    4) Choice of block algorithms

e) Impact of exception handling:

1) Clock error

i) Device error

ii) Communication error

The choice of technology to fulfill the device, application, or system requirements has to be based on a detailed assessment of these aspects. The choice of execution control method also depends on the technology used to build the devices. The method of FB execution control is also constrained by the communication method(s) used by the system.

6.6.7.3.4 UML specification of the device model

The device model definitions in Figure 28 and Figure 29 are general. To solve the ambiguity, the model is described as a UML class diagram (see ISO/IEC 19501-1 [B61]). The components are transformed to the UML language elements in Figure 31.

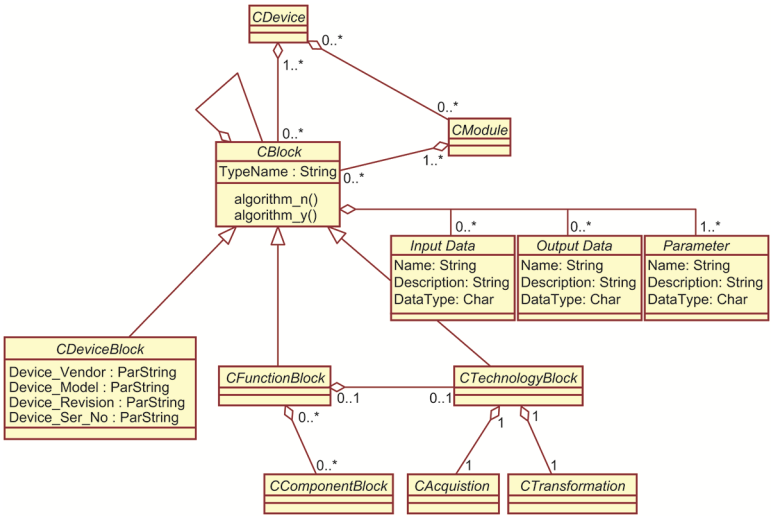


Figure 31 —UML class diagram of the device model

The following major steps are used to convert the device model into a UML class diagram:

a) Device becomes the class CDevice

b) Module becomes the class CModule

c) Device Block, FB, Component FB, and Technology Block becomes CDeviceBlock, CFunctionBlock, CComponentBlock, and CTechnologyBlock

d) The block types are of the type Block which becomes CBlock

e) A Device contains a minimum of one block

f) A Device may contain modules

g) A Module contains a minimum of one block

h) Blocks can be composed out of other blocks, i.e., may be of composite FB type

i) A Block contains a minimum of zero or more parameter

j) A Block shall have algorithms which can be internal only or visible from the outside (i.e., private or public)

k) A Device Block could contain the attributes Device_Vendor, Device_Model, Device_Revision, and Device_UUID, which are parameters

l) The FB, Component FB, and Technology Block contain the attribute TypeName

6.6.7.3.5 Classification of the algorithms

The following list provides common algorithms for use in application FBs and Device Blocks:

a) Process signal algorithms

1) Sensing acquisition:

i) Sensor connection

ii) Sensor range/calibration

iii) Analog-to-digital conversion

iv) Status estimation

2) Sensing data transformation:

i) Linearization

ii) Filtering

iii) Compensation

iv) Scaling

3) Sensing application:

i) Limit

ii) Unit

iii) Scaling

iv) Linearization

v) Simulation

4) Actuation provision:

i) Amplification

ii) Conversion

iii) Status estimation

5) Actuation acquisition:

See sensing acquisition for readback of actuator output value.

6) Actuation transformation:

i) Scaling

ii) Compensation

iii) Transition or activity limits

7) Actuation application:

i) Limit

ii) Unit

iii) Scaling

iv) Linearization

v) Simulation

b) Management

1) Estimation of device status

2) Test

3) Diagnosis

4) Operating mode

6.6.7.3.6 Algorithm description

The algorithm description is done individually for each algorithm in the appropriate language, for example plain English, Harel State Diagram, one of the IEC 61131-3 languages [B38] (for example FBD [FB diagram] or IEC 61131-3 ST [structured text]), the event driven approach according to IEC 61499 [B41], or others.

The objective of the profile description is to define a general set of rules allowing identification of a device together with classification and specification of the algorithms supported by the device.

Functions like communication network management or security management are to be described in an appropriate way (see communication and security viewpoints and requirements).

6.6.7.3.7 Input and output variables and parameter definition

For the description of the block parameters, a sample form provided by the common data dictionary of IEC according to the property description of IEC 61360 [B39] could be used. Elements to describe block parameter properties should be the following:

Parameter name:

Identifier of the variables/parameters that are accessed within the FB. The name is valid within this specification but not normative for products on the market. The decision if data are an input, output, or parameter is application dependent.

Description:

Informative text, describing the purpose of the variable/parameter.

Data type:

The following data types are conceptual ones, i.e., they identify the signal type and not the implementable data type. These will be mapped by technology profile to supported data in the following categories:

— Anumeric (e.g., float, real, long real, integer):

1) Enumerated
2) Boolean
3) String (e.g., visible string, octet string)
4) Array
5) Structure

User access read/write:

This specifies that the variable/parameter is changeable by a remote device or not.

Class m/o/c:

This specifies if the variable/parameter shall be supported within the block or not, the states are: mandatory (m), optional (o), and conditional (c).

Additional parameter attributes are:

a) Class of recovery after power fails shall have the value N or D as follows:

— N: Indicates a non-volatile parameter which shall be remembered through a power cycle, but which is not under the static update code.

— D: Indicates a dynamic parameter which is calculated by the process, the block, or read from another block.

b) Default value:

— Indicates the value is assigned to parameter in the initialization process for an unconfigured block.

6.6.7.3.8 Choice of variables and parameters

The block variables, parameters, and algorithms included in a block will be those that are significant for the algorithm and device. As a minimum, FBs will include the variables and parameters defined in the device or system specification. The names of parameters and variables are not normative.

6.6.7.3.9 Mode, status, and diagnosis

These parameters manage and indicate channel performance. They can be reported; however, the report mechanisms are technology dependent. Reported values may also include additional items such as: time stamps, priorities, indication of possible reasons, etc.

Mode describes the operation state of a channel or FB and influences the signal flow within the channel. Examples of modes are: Manual, Automatic, Local Override, and Out of Service.

Status is a characteristic aspect of a channel which may accompany information transferred within the channel i.e., FB data inputs and data outputs.

Device State describes the operational state of a device and interacts with the device technology and application blocks, it is maintained within a device by the Device Block.

Diagnosis is a report available from algorithms which assess channel or device internal performance. The results of these internal assessments may be used to construct generic measurement, control, and actuation status information.

6.6.7.3.10 Sensing channel

The technology and application FBs provide a functional chain along which the process signals flow. Together they comprise a sensing channel (see Figure 32) or an actuation channel (see Figure 33).

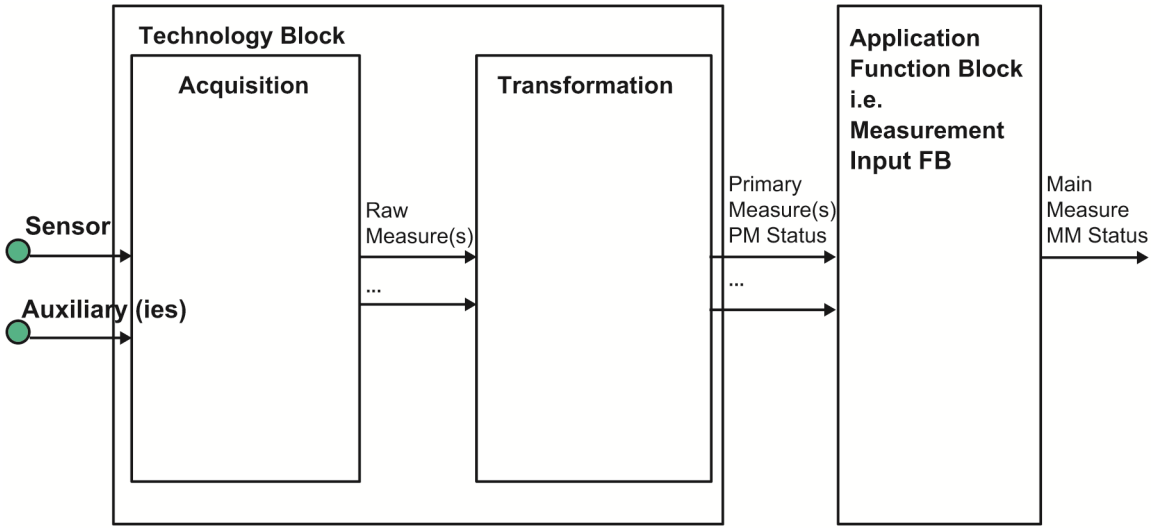


Figure 32—Sensing process signal flow

Sensing may be accompanied by optional additional auxiliary measurements, for purposes such as compensation. The technology block provides a primary measured value and its accompanying status. Additionally, the technology block may provide other outputs, for example, diagnosis or validation information.

NOTE—Additional sensor inputs can also be used and transferred by a technology block.

The application FB uses the outputs of the technology block and other internal data to generate the main measure and its accompanied status. The status is accomplished by every function in the signal flow, starting with the sensor(s) until the last function in the application FB. Information from one technology block is offered to more than one application FB. A sensing channel shall consist of at least one application FB. Channels without a technology block are possible.

6.6.7.3.11 Actuation channel

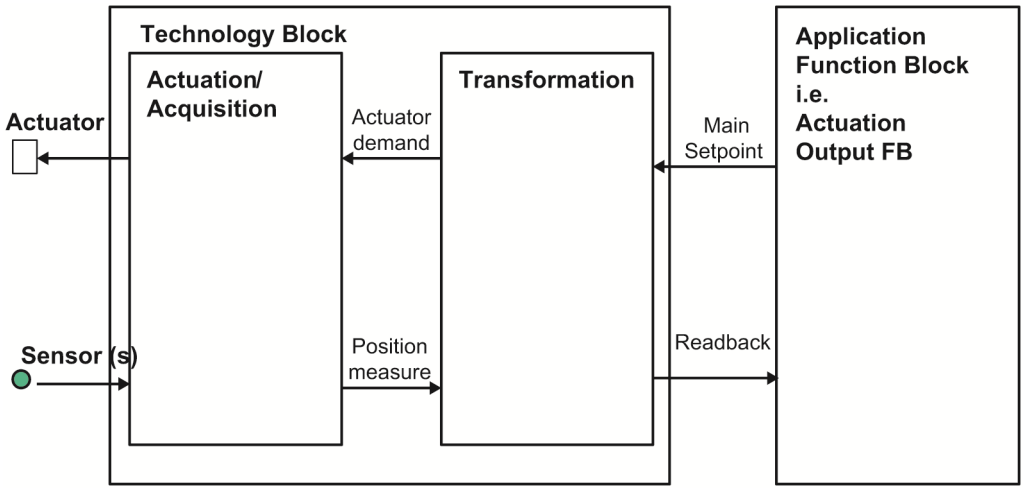


Figure 33—Actuation process signal flow

The actuation channel is performed out of the function of the actuation signal flow (Figure 33) and the additional sensing functions for the measurement of the current position of the actuator. If there is not a sensor for the position measurement, then the actuator demand will be used in the transformation to determine the readback value. Optionally, status values may accompany both signal flow directions and include information about the involved entities. The status accompanying the main setpoint carries information to give the technology block the opportunity to go in fail-safe position, if the main setpoint is not good. The status accompanying the readback carries information if the measure value is good or not. An actuation channel shall consist of at least one application FB. Channels without a technology block are possible.

6.6.7.3.12 Application

A complete application is supported by combinations of sensing and actuation channels together with control and calculation FBs (see Figure 34). The technology blocks are technology dependent and the other FBs are technology independent. There may be many different implementations of an application, depending on the technology used within the devices. The application may be performed by implementations using only sensing and actuation devices (i.e., complex devices able to perform sensing, control, and actuation) or the application may be built from sensing and actuation devices together with controller devices and other system components.

NOTE—A controller can be, for instance, integrated in the application as one calculation FB or an actuation device can take parts of programmable functions from controller devices in terms of calculation FBs.

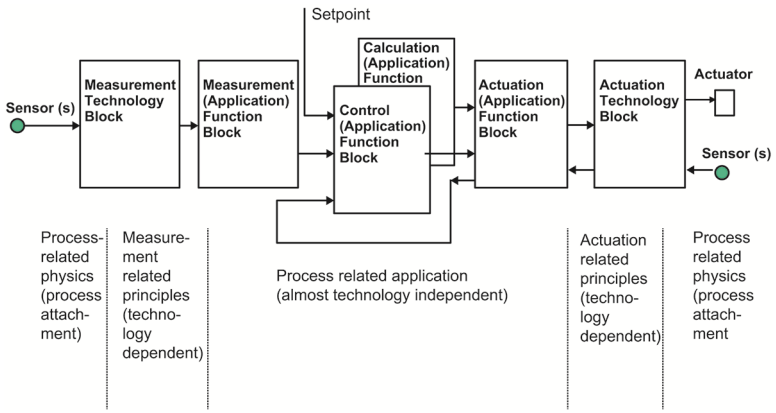


Figure 34—Application process signal flow

6.6.7.4 Operations on views

A function can represent the intended purpose of an entity or its characteristic action. The characteristic action could also be seen as an implementation function.

6.6.7.5 Correspondence rules

The function correspondence rules influence the expression of model kinds in the viewpoints lifecycle, communication, collaboration, and information. For example, parts of the communication and information exchange can be modeled as a function.

Table 19 describes the function viewpoint correspondence rules.

Table 19—Function viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
FUN-CR-1	Conceptual	The entire concept of the IoT is predicated on the function of the things.
FUN-CR-2	Compatibility	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind and function viewpoint.
FUN-CR-3	Lifecycle	The communication lifecycle shall be considered throughout the lifecycle model.
FUN-CR-4	Communication	The function model kind and the OSI reference model kind are correspondent and inform each other.
FUN-CR-5	Information	The function viewpoint produces and receives information according to the information viewpoint.
FUN-CR-6	Threat model	Secure communication is dependent on the concepts in the threat model and needs to be the same on source and destination to match the compatibility level above incompatibility level.
FUN-CR-7	Security and safety monitoring	Secure communication is dependent on the concepts of security and safety monitoring and needs a high level of compatibility.
FUN-CR-8	Access control architecture	Secure communication is dependent on the concept of access control and needs a high level of compatibility.
FUN-CR-9	Adequate design for required security	Secure communication is dependent on adequate design for security and needs a high level of compatibility.
FUN-CR-10	Privacy and trust	Relies on the concepts presented in the communication viewpoint and needs a high level of compatibility.
FUN-CR-11	Collaboration	Relies on the concepts presented in the communication viewpoint; of interest are syntax and semantics as described in the mathematics of communication model kind and needs a high level of compatibility.

6.6.7.6 Notes

The features defined here are not exhaustive. Representations of functions can vary on the demand of the stakeholder, so that it will not have the FB representation even if an insight view can be organized with FBs.

6.6.8 Threat model viewpoint

6.6.8.1 General

Subclause 6.6.8 defines a threat model viewpoint for designing secure and dependable IoT systems.

6.6.8.2 General information and key features

A threat model viewpoint is intended to identify potential threats that could exploit vulnerabilities. The threat model itself is a structured representation of these threats. A threat modeling approach looks at who would most likely want to attack the system and how they could succeed.

The less-frequently used synonym is *attack model*.

The threat model viewpoint is intended to assist architects in the design, analysis, and expression of secure IoT system architectures. Striving to achieve the utmost security, system architects and security analysts

may consider the potential adversary as a stakeholder,[10] looking for the ways of violating the vital aspects of system functioning.

In the latter case, the threat model viewpoint frames the adversary's concerns to describe how and under what circumstances a system may be severely affected by malicious actions. This description facilitates the implementation of relevant security mechanisms capable of withstanding the defined threats. This is the powerful method to provide the effective security controls, application of which is justified according to the risk of threat implementation.

It is important to determine the threats on a systematic basis to have a reliable result even under conditions of uncertainty. This is why the term *model* is used in the name of the viewpoint.

Threat models are used as a basis for an IoT system risk assessment and security policy specification. Typically, threat modeling is the first step in system security design. Security threats represent the architectural view, which may consequently be used to create the design requirements addressing the security issues in the context of system use.

A threat model can be insufficient or irrelevant in the following cases:

- When the functional or information model of the IoT system is described incorrectly
- When the assumptions about system purpose and aspects of its design, implementation, exploitation, and maintenance are set incorrectly

6.6.8.3 Stakeholders and concerns

6.6.8.3.1 Typical stakeholders

Stakeholders who have an interest in the correct and secure functioning of the system may use this viewpoint as a supporting guidance for the security assessment of the system during its lifecycle. Typically, these stakeholders have concerns opposite to the interest of adversary, and vice versa; thus, one may consider any of these two parties to identify the security-related issues requiring attention.

Consideration of adversary as a stakeholder that has an interest in a system is a straightforward way for the investigation of possible threats. Nonetheless, this viewpoint keeps the consistency with the defined set of stakeholders and their concerns and addresses the needs of the following actors:

- System owner
- System architect
- Security analyst
- Compliance analyst
- Safety engineer
- Developing engineer
- Testing engineer
- Penetration tester

[10] The standard ISO/IEC/IEEE 42010:2011 defines the stakeholder notion as an "individual, team, organization, or classes thereof, having an interest in a system." Therefore, even the malicious actor may be recognized as a stakeholder in the system security context. The malicious actor (or adversary) has concerns—goals to achieve as the result of a successful attack.

— Production engineer

— System administrator, other support staff

— Regulator

— User

The set of potential threat agents that may intentionally try to exploit the system vulnerabilities to cause a security violation and affect the system in an unforeseeable manner include:

— External agents[11]

— Users of the system[12]

— Operators of the system[12]

— Suppliers of the system[12]

— Developers of the system[12]

— Builders of the system[12]

— Maintainers of the system[12]

6.6.8.3.2 Concerns

The stakeholders pose several mostly security-related concerns from the common concerns list. Table 20 describes these concerns and their elements framed by the threat model viewpoint.

Table 20—Threat model viewpoint concerns

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Controllability	Is it possible to cause rogue system behavior, i.e., deviation from the proper functioning of the system?
Safety	Is it possible to cause a safety violation, i.e., physical injury or damage to the health of people, to property, or to the environment?
Privacy	May an adversary cause a privacy breach within the system, i.e., target some personally identifiable information (or the other types of sensitive information about the individual or group)?
Monitorability (I)	Is it possible to cause process data forgery, i.e., tamper or manipulate the data about system functioning, or its characteristics?
Physical	Is it possible to cause physical resource or equipment damage by affecting the system via informational channels?
Confidentiality	Which kind of confidential data within the system may be disclosed, and how?
Integrity	Which of stored data or software may be changed without proper authorization, and how?
Availability	Is it possible to cause a denial of service, i.e., make some functional part of a system unavailable for authorized use by affecting the system via informational channels?
Resilience	Is it possible to distribute a successful attack onto multiple systems (both IT and IoT) or to penetrate systems using a chain of threats?
Monitorability (II)	Is it possible for an adversary to remain undetected, i.e., conceal indicators of security compromise for a long time?
Identity	Is it possible for the adversary to keep anonymity, i.e., avoid revealing who is the person behind the attack?

[11] That have no initial authorization in the system, otherwise they relate to some other category.
[12] That are motivated to attack it or may cause the similar effect unintentionally.

The concerns do not equally pertain to the interests of all mentioned stakeholders. Similarly, different threat agents target the aspects related to these concerns to varying degrees. Refining the definition of concerns appropriately, one may obtain a clearer representation of the initial threat model view. Particularly, the concerns may be organized as in Table 21.

Table 21—Refining the concerns for threat model view

Stakeholders	Stakeholders' concerns	External agent	User	…	Maintainer
System owner Safety engineer Support staff Penetration tester	Controllability	Is it possible for external agent to cause rogue system behavior?	Is it possible for user to cause rogue system behavior?	…	Is it possible for maintainer to cause rogue system behavior?
System owner Safety engineer Support staff User Regulator Penetration tester	Safety	Is it possible for external agent to cause a safety violation?	Is it possible for user to cause a safety violation?	…	Is it possible for maintainer to cause a safety violation?
System architect Security analyst Developing engineer Testing engineer Penetration tester Production engineer System administrator, other support staff Penetration tester	Integrity	Which of stored data or software may the external agent change, and how?	Which of stored data or software may the user change, and how?		Which of stored data or software may the maintainer change without proper authorization, and how?
System owner System architect Security analyst Penetration tester	Identity	Is it possible for the external agent to keep anonymity?	Is it possible for the user to keep anonymity?		Is it possible for the maintainer to keep anonymity?

6.6.8.4 Model kinds

6.6.8.4.1 General

There are two approaches to threat modeling: top-down and bottom-up. If the top-down approach is used, the attack goals are identified first by the refinement of concerns. Each goal is then provided with a set of possible methods allowing the adversary to reach this goal. Then, every method is analyzed according to necessary attack prerequisites, affected system components, system vulnerabilities and exposures, adversary's skills, sequences of actions taken to perform the attack, and other conditions.

The bottom-up approach starts from the list of attacks. These attacks are joined into the groups according to the goals, which they help to implement. These goals are then used as the steps in comprising attack scenarios. Every scenario should provide a certain level of detail to define its goal, but be defined broadly to make this goal sufficiently general. Goals of scenarios are finally compared for their consistency with the specific concerns for the particular system.

These two approaches result in the structured representation of system threats referred to as an attack tree model kind.

On the interim steps of both approaches, threats may be determined according to the typical threat lists for a particular technology or particular kind of the system. These lists are constructed according to the observed attacks for this technology/kind of system, and key reasons that make them possible.

This representation of threats is referred to as typical threat list model kind.

6.6.8.4.2 Attack tree model kind

6.6.8.4.2.1 General

Attack trees refine information about threats and attacks for the IoT system by identifying the compromise of IoT system security as the root of every tree. Corresponding techniques of attack tree analysis have been known by expert practitioners for over thirty years. Attack trees allow the refinement of attacks to a level of detail chosen by the developer.

Attack trees are a way of classifying and describing the threats to a system. Every relevant concern for the system is referred to as the common attack goal by the root node of the tree. The underlying node becomes a sub-goal for its parent. The root and the following nodes are connected with their underlying nodes via logical gates (AND or OR). The type of gate determines how the goal can be reached: by reaching one of its sub-goals (if they are connected to an OR gate) or by reaching all defined sub-goals for the parent (if they are connected by AND). The ways that an adversary can cause this compromise iteratively and incrementally are represented as lower-level nodes of the tree. System description shall be detailed enough to determine the particular attack vectors, therefore, attack trees should be maintained and reused during the whole system development lifecycle.

Every attack has a scope and motivation. Threat modeling based on attack trees focuses on the identification of all possible access points to the system and the potential adversary's aims.

The model allows the root nodes of the described attack trees to be connected with the meta-root node by the OR gate, which represents a common violation of the system security.

The use of attack trees allows a comparison between technical and non-technical means of attack, supporting a more holistic analysis of threats and vulnerabilities and integrating physical, personal, and information security disciplines.

6.6.8.4.2.2 Attack tree model kind conventions

6.6.8.4.2.2.1 General

Threats are any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service (CNSSI No. 4009 [B17]).

Attacks are an attempt to gain unauthorized access to system services, resources, or information (Schneier [B132]), or an attempt to compromise system integrity, or any malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself (NIST SP 800-32 [B84]).

An IoT system typically has a set (or *forest*) of attack trees that are relevant to its operations. The root of each tree in the forest represents an event that could significantly harm the IoT system security. Each attack tree enumerates and describes in detail the methods that can be used by the adversary to cause the desired effect. Each path in the attack tree graph represents a unique attack on the system.

An attack represented by the node of an attack tree is equivalent to:

— The set of attack sub-goals connected by an AND gate, all of which shall be achieved to complete this attack

— Alternatively, the set of attack sub-goals connected by an OR gate, any of which may be achieved to complete this attack

An attack tree consists of a combination of AND- and OR-decompositions (see Figure 35 and 6.6.8.4.2.3). The following elements may be used as the leaf nodes of the attack tree:

— Actions taken by the adversary to complete the attack

— Assertions that should hold for the attack to be a success

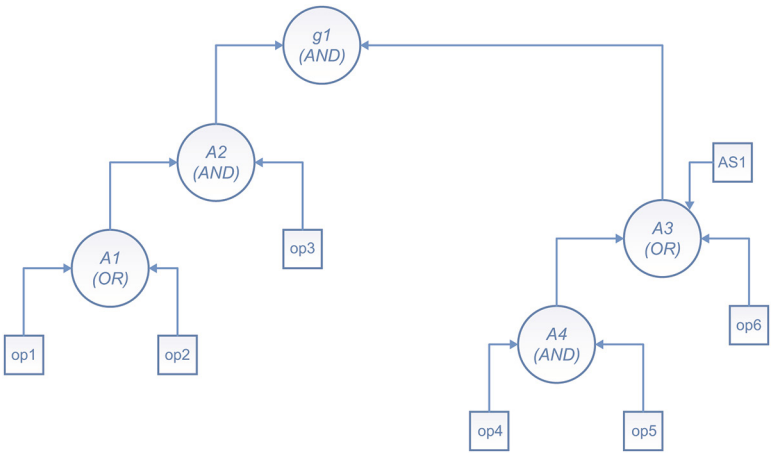


Figure 35—Example of an attack tree

An attack scenario is a set of actions that are taken by an adversary to reach the final goal. Figure 35 illustrates a possible attack scenario. The goal g1 will only be achieved when the attacks A2 and A3 are completed. The attack (resulting in sub-goal) A2 depends on the attack (sub-goal) A1 and malicious action op3. The attack A3 depends on the assertion AS1, the success of the attack A4, or the malicious action op6.

A vulnerability is a system design, implementation, or configuration weakness, which may be exploited in a special way to implement the attack.

Attack trees are the simplified representation of system security. The model accuracy depends on many factors including the time and effort spent studying the IoT system. In some cases, it is necessary to make unsubstantiated assumptions based only on the available information. The accuracy of further analysis will be constrained by the correctness of the assumptions made.

6.6.8.4.2.2.2 Model kind notations

An attack tree has a structured set of goals and sub-goals to be reached by the adversary to complete the attack. Let's define $G = \{g_i\}$ as the set of the attack goals, $Op = \{op_i\}$ as the set of the operations representing all the basic actions that can be taken by the adversary. The attack tree also may contain assertions. An assertion in this context is a condition to be validated to assume a certain branch of the attack tree is valid. We define $AS = \{as_i\}$ as the set of possible assertions.

Usually, to reach the malicious goal, the adversary takes advantage of a system vulnerability. For this reason, the attack tree may also contain descriptions of relevant vulnerabilities for specific attack vectors. We refer to $V = \{v_i\}$ as the set of known vulnerabilities. It is important that not all of the existing vulnerabilities of the system are conclusively known. Therefore, the set of possible precise actions on the lowest level of the attack tree is usually incomplete.

Formally, an attack tree AT may be defined as a tuple $AT = \{g_i, O_i, AS_i, V_i, R_i\}$, where g_i G, Op_i DO, AS_i AS, V_i V, and R_i is the set of relationships among the elements of the tree (see Figure 35). Every attack tree has only one main (or principal) goal. Relationships between assertions, operations, and vulnerabilities are established with logical gates. The only compositional constraints for the combined use of gates, assertions, operations, and vulnerabilities are:

— The output of a logical gate has to be an assertion (except in the following case)

— The output of the top logical gate has to be the goal of the attack tree

6.6.8.4.2.3 Attack tree model kind operations

Construction: Allows creation of an attack tree. Algorithm for attack tree construction may be described in the following way; see also Schneirer [B132].

All valuable system assets are potential goals of the attack. These assets and their desired properties determine the security objectives. The completeness of security objectives for an IoT system may be checked according to the list of concerns.

All malicious actions that may be taken by the adversary are candidates to be included in the attack tree. These actions are described as the attack methods.

There are two approaches to attack tree construction: top-down and bottom-up. If the top-down approach is used, the possible attack goals (opposite to security objectives) are identified first. Each goal determines the separate tree, although they might share sub-trees and nodes. The assertions and leaf nodes are described for this tree (if they exist). Then, the mediating goals are defined. This process is repeated down the tree until the tree is completed with the concrete actions and attack vectors. The lowest level of the tree may be checked for completeness using the lists of typical threats (typical attacks) for the domain area or the technologies used.

The bottom-up approach starts from the list of typical threats (typical attacks) refined for the concrete system. These attacks are joined into the groups according to the goals that they help to implement. These goals are then used as the steps in comprising the attack scenarios. Every one of these scenarios declares the necessary goals (with its local sub-trees) in the appropriate separate sub-trees. Joined sub-trees are finally compared for their consistency with the security objectives defined for the system. On the interim steps, the attack goals may be determined according to the attack methods.

Interpretation: Different attackers have different level of expertise, skills, access rights, risk tolerance, time, money, and other resources.

The attack tree may be interpreted to reveal possible attacks and scenarios for a malicious agent with various capabilities. To make this possible, the complete set of interpretation parameters is elaborated and every node of the tree is labeled with the appropriate values of these parameters. For some types of parameters, their values may also be organized into the hierarchy (e.g., access privileges). This hierarchy may be used to describe the appropriate tree layers. In this case, the values of attributes are inherent for the tree nodes.

The interpretation of the attack tree is performed for the given value of a parameter as follows:

— The top-down method is used, so one has to analyze the attack goals from the common to specific attack vector.

— Sometimes it is necessary to analyze only one goal in the tree hierarchy; sometimes all of the nodes on the path are analyzed.

— For every given goal, the value of the parameter is compared with the target value. The result of the comparison depends on the comparison predicate defined for the interpretation (comparison for the values equivalence, for the relation of domination, for the set membership, etc.).

— If all of the nodes to be analyzed meet the given criteria, the path on the tree graph is assumed to be the valid attack (or scenario) for the interpretation.

Validation: To validate an attack tree for its consistency one may:

— Ensure that there are no attacks that duplicate each other due to the possibly of inconsistent classification of attack goals on one of the tree layers.

— Ensure that at the same layer for all the described attack trees the defined attack goals have a similar semantic interpretation and can comprise the whole set types or classes of attacks that may threaten the system.

Revision: Allows the addition of nodes and relations to the attack tree. There is always the chance that some attack is missed. The system may be redesigned or improved in a way to expose new interfaces for an attack. In addition to this, security objectives or assumptions may also change. This is why the attack trees should be revised from time to time, though this revision should not affect the scenarios that are still valid. To refine the attack tree:

— If this is single attack vector or new vulnerability, or some other single condition, the appropriate leaf shall be added to all attack sub-trees that it may facilitate.

— Otherwise, if this is a new attack goal, directly withstanding the security objective, a new attack tree shall be created independently.

— Otherwise, if this is a new attack goal, a new sub-tree shall be created and then added to all attack sub-trees which it may facilitate.

Refinement: Allows specifying the new details according to updated information about the system. These details may be included in the attack tree by one of the following ways:

— If new details constrain the success factors for the particular attack, the appropriate description of attack goal, operation, or assertion is refined using the new conditions.

— If new details do not constrain the attack but specify a way it may be performed, the new node is created with a parent node, describing the attack under the broadened range of conditions.

6.6.8.4.2.4 Attack tree model kind correspondence rules

At-1: Attack tree should be consistent with the IoT system architecture (functional consistency).

At-2: All security objectives may be set against the attack goals in the attack tree (security consistency).

At-3: All relevant threats determined on an ad hoc basis should be included in the attack tree (security consistency).

At-4: All typical relevant attacks and known incidents should be covered by the threats set (security consistency).

At-5: The attack tree graph should not have unconnected nodes (design consistency).

At-6: Attack tree graph should not have closed loops (design consistency).

6.6.8.4.3 Typical threat list model kind

6.6.8.4.3.1 General

A typical threat list may be useful for identifying, interpreting, and classifying the types of malicious activities that may take place in the system.

Such lists may be constructed in two general ways:

— Typical threats for the technologies used

— Typical threats for the domain area

Not all the threats included in the list of threats for the technology will be significant for every given system using this technology. Also, not all the threats that exist in the domain area will be valid for the system due to its particular technological implementation. However, using such lists will simplify analysis and help to look at the possible security threats from different points of view.

A typical threat list for a technology comprises the possible methods of malicious exploitation of the system features implemented with this technology. The OWASP top 10 list [B120] may be considered an example of a typical threat list valid for a particular technology.

A typical threat list for a domain area comprises mostly the goals of the attacks or their possible impact. Typical threat lists valid for specific domain areas are not so well elaborated, but as the IoT expands, they will appear. Examples of such lists and areas are typical threats for automotive systems, typical threats for the public transport industry, typical threats for Smart Homes, etc.

6.6.8.4.3.2 Typical threat list model kind operations

Interpretation: A typical threat list may be used to enumerate the attacks for the system component, interface, or at the particular level of attack tree.

If a technology-specific threat list is used, for its interpretation:

— Determine all components or exposed interfaces that may employ this technology

— Identify the security objectives involving these components or interfaces

— Specify the attack vectors onto each component of interface in terms of typical threat list

— Use these attack vectors to validate, revise, and refine the attack tree (according to definition of appropriate operations)

If a domain-specific threat list is used, for its interpretation:

— Determine how high-level attack goals (opposite to security objectives) relate to the typical threats, and their direct sub-goals

— For attack goals relevant to one or more typical threats, use these threats to validate, revise, and refine appropriate attack trees and sub-trees

— For a typical threat that does not relate to any attack goal or sub-goal in attack trees, ensure that there is no actual need to address this kind of threat

6.6.8.4.3.3 Typical threat list model kind correspondence rules

Ttl-1: A typical threat list for technology should be consistent with the IoT system technologies (functional consistency).

Ttl-2: A typical threat list for domain area should be consistent with the IoT system purpose (functional consistency).

6.6.8.5 Operations on views

Requirements validation: The threat model can be used to validate the completeness and consistency of the security requirements set.

To assure the completeness of the security countermeasures one may perform the following:

— Match every security requirement to the threat that this requirement is intended to prevent

— Ensure that the threats uncovered by the requirements are covered by the set of assumptions about the system behavior and its environment and that this is refined with the appropriate security policy

6.6.8.6 Correspondence rules

Table 22 describes the threat model viewpoint correspondence rules.

Table 22—Threat model viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
THM-CR-1	Functional and Informational	The threat model viewpoint depends on functional and informational viewpoints.
THM-CR-2	Security-related viewpoints	The threat model viewpoint is a basis for all security-related viewpoints.

6.6.9 Security and safety monitoring viewpoint

6.6.9.1 General information and key features

Security and safety monitoring is the process of monitoring the events occurring in an IoT system and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of security, safety, or acceptable use policies, or standard security practices (NIST SP 800-94 [B85]). Incidents have many causes, such as malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized. Although many incidents are malicious in nature, many others are not; for example, a person might mistype the address of an IoT system and accidentally attempt to connect to a different system without authorization.

This viewpoint is intended to be used by IoT architects to describe security and safety monitoring views applicable to IoT systems. For IT systems, two types of architecture monitoring systems are well known—intrusion detection systems (IDS) and intrusion prevention systems (IPS). An IDS is a passive system that automates the intrusion detection process. An IPS is an active system that has all the capabilities of an intrusion detection system and can attempt to stop possible incidents. Intrusion detection and prevention

systems are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators.

Goals of a monitoring mechanism in an IoT system are confidentiality, integrity, availability and safety of these systems, their resources and components. The monitoring mechanism depends on the identification and authentication and authorization and auditing security mechanisms.

6.6.9.2 Stakeholders and concerns

6.6.9.2.1 Typical stakeholders

Typical stakeholders for this viewpoint are the developers, controllers, system integrators, maintainers (operators and administrators) and users of the IoT systems:

- IoT system vendors developing monitoring systems; vendors implementing the appropriate algorithms; testers/validators checking the adequacy of intrusion detection algorithms and methods and estimating the rate of analysis errors
- Certification authorities assessing the compliance of monitoring systems with requirements defined by international, national, and industry standards
- System integrators installing IoT systems that employ monitoring systems for their protection
- Maintainers (operators and administrators) maintaining the monitoring system in accordance with monitoring policy and executing the response plan in case of incident detection
- Users working with the system in accordance with monitoring policy

6.6.9.2.2 Concerns

The main concerns for a security and safety monitoring system are the detection of and response to security and safety incidents. Besides these concerns, the following additional concerns of a monitoring system may be defined:

- Identifying flaws in security policy. If an IDS system reports inappropriate behavior, it would send a signal for necessary correction of system security policy; for example, the correction of authorization security policy depending on the results of the intrusion detection system functioning could prevent unauthorized access.
- Recording the attempts of both malicious and non-intentional policy violations in the IoT system. All detected attacks would be recorded; all violators (for violations committed by a human) would be warned or punished.
- Deterring the violations of security policy through the warning about IDS-based surveillance in the system. If users know about continuous monitoring, this could cause them not to take the inappropriate actions.

Table 23—Concerns of security and safety monitoring viewpoint

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Monitorability	Does the monitoring system record all the attempts of both malicious and non-intentional policy violations in the IoT system with sufficient accuracy?
Security	Does the system security policy contain flaws?
Safety	Does the monitoring system discover security attacks that lead to safety violation?
Resilience	Could the monitoring system provide data for system recovery after attack?

6.6.9.3 Monitoring system module model kind

6.6.9.3.1 General

In Figure 36 the security and safety monitoring system module structure is shown (Denning [B22]). The model is derived from control theory and is useful for describing the IDS structure.

Figure 36 depicts that the typical monitoring system consists of following modules:

— Monitoring system sensor module (E-module)

— Analysis module (A-module)

— Storage module (S-module)

— Reaction module (R-module)

We can define various types of monitoring systems according to the types of monitoring system modules:

a) The monitoring system sensor module can be a network monitoring system sensor module or an IoT device-based monitoring system sensor; a monitoring system sensor generally determines the types of data collected and analyzed by an IDS.

b) The storage module can be local (directly on the device being monitored) or remote (on a special purpose device).

c) The analysis module can be based on the method of signature detection or the method of abnormal behavior detection. The analysis module is a core module for the monitoring system; this module makes decisions about signs of intrusion in analyzed data. The analysis module will be considered in more detail below.

d) The reaction module produces passive and/or active reactions based on decisions about the symptoms of intrusion. An improper reaction (either caused by the detection error or design error) can pose a threat to the system availability. In some situations, when availability is critical, the active reaction would be disallowed.

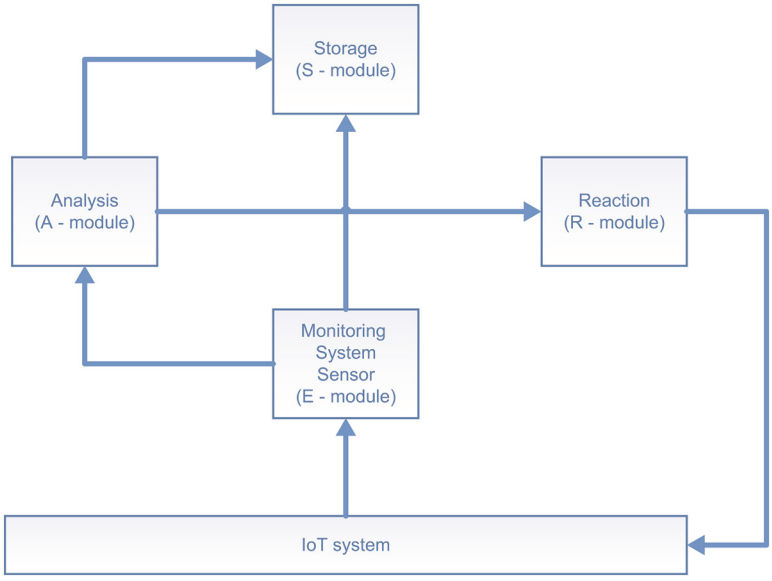


Figure 36—Monitoring system module structure

The IDS model based on control theory shows the IDS structure but reveals nothing about the methods of intrusion detection. The data types and algorithms are described:

- Input data (X): The data obtained from the IDS monitoring system sensor.
- Knowledge base (K): The description of normal and/or abnormal system behavior; can be constructed automatically by a profile-building algorithm or by a system administrator.
- Features (F): The set of data attributes intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps.
- Output data (Y): Decision of the detection algorithm.
- Expert system: Learning system with teacher and without teacher. Learning methods are used in this case for feature selection and to build the knowledge base. An expert system is a knowledge-based system that employs knowledge about its application domain and uses a reasoning procedure to solve problems that would otherwise require human competence or expertise. The power of expert systems stems primarily from the specific knowledge about a particular application domain stored in the expert system's knowledge base. The knowledge base of an expert system contains both factual and heuristic knowledge. Knowledge bases should represent notions as actions to be taken under circumstances, causality, time, dependencies, goals, and other concepts. Knowledge representation is the method used to organize the knowledge in the knowledge base. Expert systems is one most important application area in artificial intelligence.
- Representation algorithm (R): Algorithm based on a chosen set of features.
- Detection algorithm (C): Signature or anomaly detection algorithm. The detection algorithm can be implemented using artificial intelligence (for example neural network, genetic or immune algorithm, etc.).
- Profile build algorithm (P): Expert (signature base developed by vendor or administrator of monitoring system) or automatic (for example inductive learning algorithm, like C4.5) algorithm.
- Feature selection algorithm (S): Expert (vendor) algorithm of features for use in model construction. Feature selection techniques are used for three reasons: simplification of models to make them easier to interpret by researchers/users, shorter training times, and enhanced generalization by reducing.

Monitoring systems can detect security and safety incidents. Therefore, we can build one system or different systems for monitoring security and safety. There are two main types of analysis for monitoring systems—signature detection and anomaly detection.

6.6.9.3.2 Signature detection

6.6.9.3.2.1 General

The signature detection method describes every attack as a special model or signature. A signature defines attack event features. A signature can be presented as a string of symbols in a special language expression (see 6.6.9.3.2.2). The signature method is applicable in situations when attacks can be described with a finite alphabet. Monitoring system knowledge consists of the attack signatures set. The detection algorithm can be described as the search of attack signatures in input data. The monitoring system reports about attack detection in the case of signature detection.

6.6.9.3.2.2 Conventions

Chomsky hierarchy of grammars is a suitable formalism for signature representation. Chomsky hierarchy of grammars consists of regular expressions, context-free grammars, context-depended grammars, and free grammars (Hopcroft, Motwani, and Ullman [B35]).

Regular expressions are a traditional approach for description and recognition of signatures. Regular expressions describe attacks as finite automata. Advantages of regular expressions signatures detection method are:

- High performance of recognition
- Simple implementation

Disadvantages of regular expressions signatures detection method are:

- Low expressive power suitable for attack mutation detection (noisily analyzed data)
- Low expressive power for complex attack descriptions—no order and time attack attributes

Context-free grammar is a good solution for description of complex intrusions. Context-free grammars describe event relations in the attack expression as stack automata. Formally speaking, context-depended grammar differs from context-free grammar in that it can use production for non-terminal symbol in dependency from non-terminal symbol context (Hopcroft, Motwani, and Ullman [B35]). Advantages of context-free and context-depended attack signatures description are listed below:

- High expressive power
- Event order expression possibility

Disadvantages of free grammars signatures detection method are:

- Performance problems

Free grammars do not have restrictions for rules. Free grammars are equal Turing machines and should not be used for signature descriptions due to detection complexity. Advantages of free grammars detection method are:

- High expressive power

Disadvantages of free grammars signatures detection method are:

- Problem of computability

Signature detection models may be used as they are defined in the Chomsky grammar formalism (Hopcroft, Motwani, and Ullman [B35]) or they may be tailored through the use of allowed operations. Allowed operations on access rights are assignment, iteration, selection, union, and conflict resolution.

- Iteration: Allows an search signatures in input data in the following modes:
  - Detect one signature and stop signatures search
  - Process all data and stop signatures search
- Order: Include order notion in detection process
- Union: Allows the union of two signature sets for detection process

The following artificial intelligence methods can be used to improve signature detection system accuracy:

a) Expert system methods (Sebring and Whitehurst [B134]). Various expert system methods, such as logic-based methods, can be used on top of the signature system. These methods allow correlation of different intruder activities and remove detection errors.

b) Signature attack description language (Vigna and Kemmerer [B143], Vigna, Eckmann, and Kemmerer [B142]) is a good addition for a signature-based monitoring system. Signature attack language allows quality of signature description raise and sophisticated attacks detection.

The signature detection model for network traffic in terms of the model may be described in the following definitions:

— X: Input data, packet sequence {P1, P2, …}

— Y: Output data, {AttackType1, AttackType2, …, Normal}, AttackTypei—detected attack type, Normal—normal data

— F: Features <srcIP, dstIP, dstPort, payload content,…>

— K: Signature set

— S: Attack analysis and feature selection

— R: Lexical analyzer

— P: Signature set construction

— C: String detection (K) in input data (X); in the case of string recognition the monitoring system produces an alert; in other cases, the recognition monitoring system produces no alert

6.6.9.3.3 Anomaly detection

6.6.9.3.3.1 General

The anomaly detection method is based on the comparison of expected IoT system activity (normal behavior) and current IoT system activity. There is an assumption that deviation in expected and current activity means suspected activity. Suspected activity on the other hand can be a sign of security and safety IoT system violation. Usage of the anomaly detection method for monitoring system implementation is a reasonable choice, if system behavior can be predicted.

6.6.9.3.3.2 Conventions

For the explanation of anomaly detection, the monitoring system is described as a signal detection system (Axelsson [B5]).

This model assumes that in our knowledge base we have probability distributions for normal behavior (hypothesis P(X|H0)) and abnormal behavior (hypothesis P(X|H1)). The system builds these distributions during the learning stage of the monitoring system operation. The IoT system produces data X as input for the monitoring system. On the detection stage for any data X the monitoring system can estimate the probability of the IoT system behavior nature—normal or abnormal. Anomaly detection methods are often based on statistical data description and evaluation. IoT system parameters used for profile construction may be changed in dependency from system activity and context.

Models used for anomaly detection (Kruegel, Vigna, and Robertson [B72]) include:

— Operational model

— Mean and standard deviation model

— Multivariable model

— Bayesian model and its variations

— Markov chain and its variations

— Support vector machines

There are more sophisticated algorithms proposed to improve accuracy:

— Neural networks (Cannady and Mahaffey [B13])

— Genetic and immunological algorithms (Li [B73], Forrest, Hofmeyr, Somayaji, and Longstaff [B26])

— Other artificial intelligence methods

The anomaly detection model for network traffic in terms of the model can be described in the following definition:

— X: Input data, packet sequence {P1, P2, …}

— Y: Output data, {Anomaly, …, Normal}, Anomaly—detected anomaly, Normal—normal data

— F: Features <srcIP, dstIP, dstPort, payload content,…>

— K: Set of normal/abnormal behavior profiles

— S: Feature selection (i.e., principal component analysis feature selection algorithm)

— R: Lexical analyzer

— P: Knowledge base construction

— C: Anomaly detection (based on set of profiles K) in input data (X); in case of abnormal behavior, the profile recognition monitoring system produces an alert; in the case of normal behavior, the profile recognition monitoring system produces no alert

6.6.9.4 Operations on views

6.6.9.4.1 General

Operations define the methods to be applied to views and their models. Types of operations include:

— Construction

— Analysis

— Implementation

6.6.9.4.2 Construction

6.6.9.4.2.1 General

Construction of a monitoring model requires the analysis of an application domain and the related attacks. The model construction method depends on IoT system complexity and attack detection complexity.

6.6.9.4.2.2 Signature construction

The signature construction process should describe key features of an attack. A common approach for signature construction consists of the following steps:

— Find attack implementation for signature creation

— Run attack on test system; log all significant attack events

— Analyze logged attack events and identify attack signature

— Test attack signature and estimate signature accuracy and quality

6.6.9.4.2.3 Anomaly detection profile construction

The construction operation produces a behavior (normal or abnormal) profile automatically or with expert support (unsupervised learning or supervised learning). All learning mechanisms assume that normal and abnormal behavior profiles are separated. The profile construction process can be characterized with the following terms:

a) Profile construction depends on the algorithm used for profile description. A well-known profile construction unsupervised learning procedure is a clustering (Saaksvuori and Immonen [B129]).

b) Experts should be highly qualified and have good IoT system understanding for profile construction.

c) Profile quality is an important factor for the detection process.

d) A good quality profile depends on the data used for system learning.

e) Profile construction may be a time-consuming process.

f) Intruders should not have access to the system at learning stage.

6.6.9.4.3 Analysis

When comparing two models for analysis module construction, we can make the following conclusions.

a) Signature detection and behavior estimation algorithms can be built for normal (abnormal) behavior. For signature systems, signatures of secure (attack) behavior can be defined, and absence (presence) of these signatures in input data can be detected as a security or safety violation. White listing is an example of a signature method for normal behavior detection. For behavior analysis systems, a normal (abnormal) behavior profile can be defined, and absence (presence) of a corresponding profile can be detected as a violation in input data.

b) The signature detection method generally cannot detect unknown attacks. The anomaly detection method generally cannot explain attack nature.

c) High precision detection algorithms are generally time consuming and resource consuming.

d) Joint use of anomaly and signature detection in a single IoT system may be a good decision.

e) Context-aware monitoring systems are a promising solution. Context information can be obtained from IoT system sensors.

f) Monitoring of system usage in rarely changed IoT systems with well-defined behavior (for example in Industrial IoT systems) can give the best results in terms of detection accuracy.

6.6.9.4.4 Implementation

Figure 37 shows an example of a redundant network IoT system monitoring deployment.

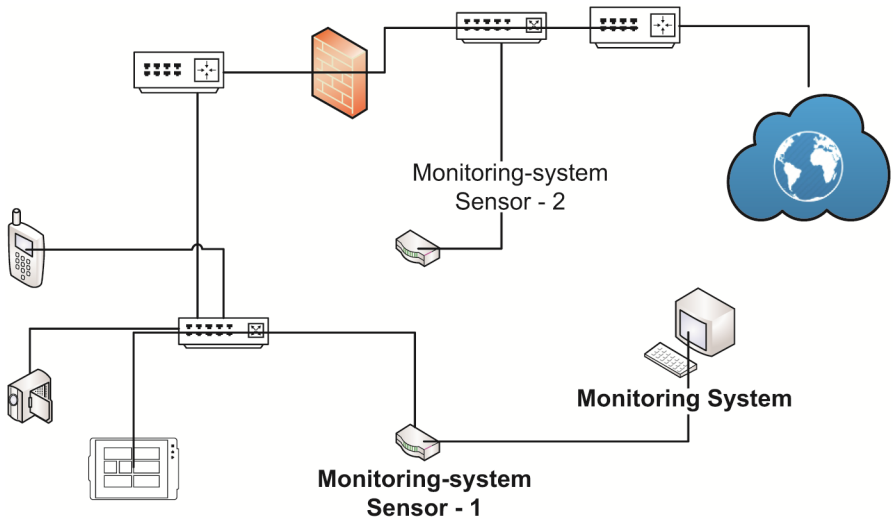


Figure 37—Example of deployment of a network IoT monitoring system

This example has the following redundant properties:

a) Monitoring system sensor 1 is responsible for IoT system data gathering. Monitoring system sensor 2 is responsible for early IoT system attack detection (can be blocked by firewall and not detected by monitoring system sensor 1).

b) The monitoring system has a dedicated control channel.

c) If we implement the monitoring system on end-point IoT devices, especially in the case of a low-resource device, we should follow these rules:

   1) The detection mechanism should be implemented using the signature detection white-listing approach.

   2) White-listing signatures should be constructed by the IoT device vendor.

   3) Signature detection should be provided for all user-visible device interfaces.

Joint use of network and device monitoring system sensors can give better quality detection results.

The following challenges for the monitoring system implementation can be defined:

— A lot of information sources (network, devices, applications) provided by various vendors; this leads to unified representation difficulties of diverse data formats and additional difficulties such as:

  — Closed proprietary data formats and protocols

  — Data encryption

  — Diverse types of analyzed data (continuous, discretion, categorical)

— Huge volumes of data contaminated with noise to analyze

— The difficulties in making proper assumptions about the difference of normal and abnormal IoT system behavior (that might indicate the security violation) (Ptacek and Newsham [B124], Timm [B137])

— Complicated multi-stage or slow attacks that may be successfully disguised from surveillance:

  - Unknown attacks (exploiting so-called "0-day vulnerabilities")
  - Slow time attacks (long time delays between the stages of attack)
  - Distributed attacks (attacks on the IoT system from different sources)
  - Denial of service attacks (Hussain, Heidemann, and Papadopoulos [B36])
  - Attacks on an IDS system itself
- Cost of analysis errors, especially cost of false positives; in the case of IPS usage, a false positive error may lead to a reaction, and, as a result of the reaction, to a violation of the availability property; this property may be very important in the IoT world.
- Intrusion detection (including the reduction of analysis errors) may require employing sophisticated techniques and algorithms; these techniques may be time consuming, and data and context constrained.
- Possible constraints on the time for analysis; incident reaction time may be time-constrained, especially in the case of critical IoT systems; therefore, the time to render a decision about a possible violation may be constrained.
- Possible constraints on the qualification of IDS supporting personnel; the functioning of an IDS may require support by users with strong experience in details of both IoT system and IDS logic; such support may significantly improve analysis efficiency while putting constraints on use of the monitoring system by inexperienced personnel.
- The necessity of context awareness in some cases—the environment is continuously changing for some types of IoT systems; this may cause detection errors when simple algorithms are applied to find out if an intrusion occurs. Considering the context and some additional factors may significantly reduce this error rate.

6.6.9.5 Correspondence rules

The monitoring architecture viewpoint depends on the functional, informational, and threat viewpoints. The following correspondence rules can be defined in Table 24. All security and safety monitoring model specifications and transformations should comply with the correspondence rules.

Table 24—Security and safety monitoring viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
SSM-CR-1	Conceptual	Threat and security policy viewpoints should govern monitoring system sensor deployment
SSM-CR-2	Conceptual	All significant security and safety events and possible attacks should be monitored by monitoring system sensors.
SSM-CR-3	Functional and informational	The functional and informational viewpoints govern endpoint IoT device interface monitoring. Signature detection should be provided for all user-visible device interfaces.
SSM-CR-4	Conceptual	The deployment model should include monitoring system components.

6.6.10 Access control viewpoint

6.6.10.1 General

The access control viewpoint determines the permitted activities of legitimate users, mediating every attempt by a user to access a resource in the IoT system. IoT (Bassi et al. [B6]) considers the "thing functionality" as a set of services. Access control is composed of three security functions:

— Identification

— Authentication

— Authorization

Access control policies assign different access rights to different stakeholders in a system. Figure 38 describes the access control viewpoint.

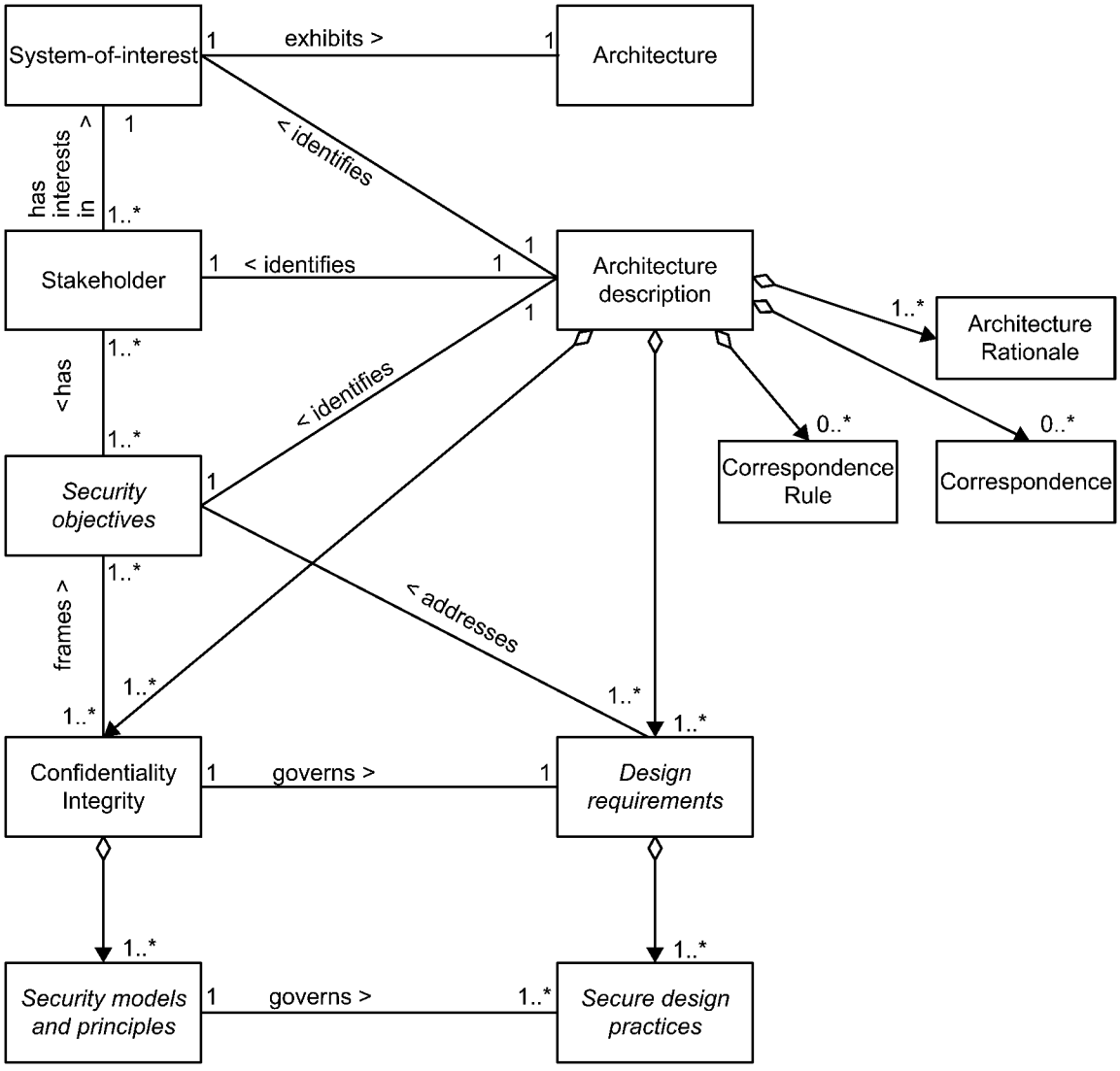


Figure 38—Access control viewpoint

6.6.10.2 Overview

This viewpoint is intended to describe access control applicable to IoT systems.

By definition (CNSSI No. 4009 [B17]) authorization is a process of granting or denying specific requests to obtain and use information and related information processing services. Authorization depends on security functions defined below (CNSSI No. 4009 [B17]):

- Identification is an act or process that presents an identifier to a system so that the system can recognize a system entity (e.g., user, process, or device) and distinguish that entity from all others.
- Authentication is a process of verifying the identity or other attributes claimed by or assumed of an entity (user, process, or device), or verifying the source and integrity of data.
- Authorization is access privileges granted (according to access control policy) to a user, program, or process or the act of granting those privileges.

The access control process in an IoT system is shown in Figure 39. Once the subject is properly identified, the IoT system attempts to determine if this subject has been given the necessary rights and privileges to carry out the requested actions. The system will look at the access control matrix or compare security labels (access control policy) to verify that this subject may indeed access the requested resource and perform the actions it is attempting. If the system determines that the subject may access the resource, it authorizes the subject.

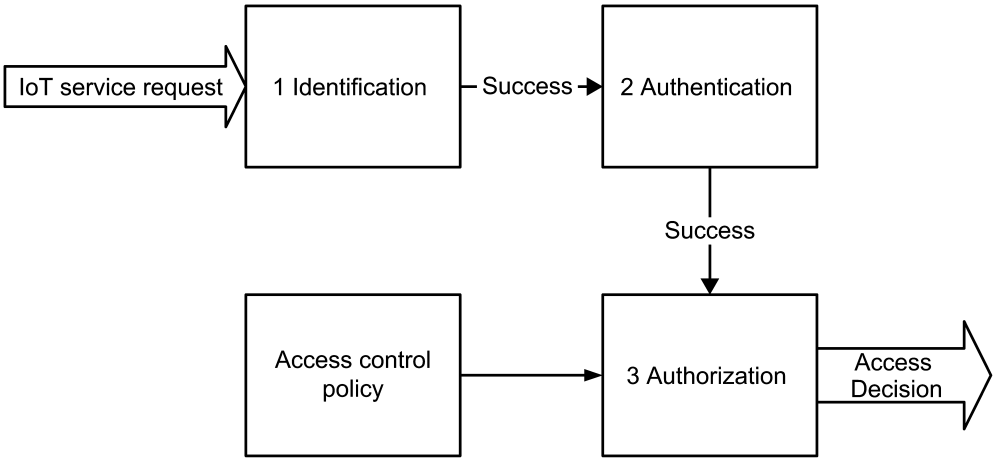


Figure 39—Access control process

The access control policy defines rules for IoT system authorization. Simple access control rules may be primitive, but effective. More sophisticated access control policies may enforce the rules oriented on the system functionality.

Authorization provides the basis for privacy, monitoring mechanisms, and network security. An authorization mechanism should help in implementation of the two security design principles:

- Principle of least privilege
- Separation of duties

Authorization determines some requirements of the appropriate security policy. For example:

- Deny access to the IoT system by an unknown or anonymous user.
- Limit the IoT functionality in the context of the usage by a superuser.

— Suspend or delay the access to the IoT for the user after a specified number of unsuccessful authentication attempts.

— Disable access to the unused functions.

— Remove redundant resource rules from accounts and group memberships.

— Remove redundant user IDs, accounts, and role-based accounts from resource access lists.

— Protect IoT system audit logs.

Authorization management can be performed in two ways:

— Centralized. With centralized authorization management only one subject (user or role) is responsible for overseeing access to all IoT system resources (the subject acts as IoT system owner). The advantages of this type of authorization management are the consistency and uniformity of the method of controlling user access rights. Examples of this approaches in IT systems include RBAC model family (ANSI [B2], Sandhu, Coyne, Feinstein, Youman [B130], Ahn and Sandhu [B1]) and DAC with root access control management (NCSC-TG-003 [B81]). Implementation examples include RADIUS (Hassell [B31]) and TACACS (RFC 1492 [B125]).

— Decentralized. This method of authorization management transfers the control of access rights to IoT system administrators that are close to the resources. When this approach is applied, the IoT system administrator assigns the access control rights to other users according to the defined security policy. The advantage of this type of authorization management is its effectiveness. Examples of this approach in IT systems include variations on the DAC model (capabilities) (NCSC-TG-003 [B81]). Implementation examples include capability-based systems and the X.509 standard (Public-Key Infrastructure [B54]).

6.6.10.3 Concerns and stakeholders

6.6.10.3.1 Typical stakeholders

Typical stakeholders for this viewpoint are developers, testers, and users of IoT systems. The following list shows examples of the typical stakeholders.

— Developers of the IoT system can implement authorization mechanisms that correspond to security models and policies.

— Testers in the development team should check adequacy of authorization mechanisms.

— Controllers in the certification team should check compliance authorization mechanisms against requirements defined in international, national, and industry standards.

— Builders of the IoT system should install an IoT system with the correct access control (AC) policy.

— Operators and maintainers of the IoT system should maintain the IoT system in accordance with AC policy.

— Users of the IoT should know the AC policy and use the IoT system correctly.

— Regulators can define the legal, regulatory, and domain-specific norms of practice/rules that the IoT system shall fulfill with regards to safety, reliability, security, privacy, and resilience.

6.6.10.3.2 Concerns

In general, access control concerns in an IoT system are confidentiality and integrity of these systems, their resources, and components. The availability property is rarely implemented by the access control

mechanism. This is because the confidentiality and integrity properties describe restrictions on access to information. The available property describes the necessity of the absence of barriers to access.

Access control concerns in IoT can be described as follows:

— Various types of users need different rights for things usage—internal users, guests, contractors, outsiders, partners, etc.

— Diverse identity data shall be kept for different types of IoT users—credentials, personal data, contact information, work-related data, digital certificates, cognitive passwords, etc.

— IoT resources can have different structures—private, public, etc. System services often should be represented as resources.

— Engineers and specialists who have a good understanding of the needs of resource protection and the nature of safety constraints often need to be involved in authorization management.

— Authorization management is often performed by users with a limited level of knowledge in security and safety areas.

— Because the cyber-environment needs are continuously changing, access control mechanisms should be context aware (e.g., the resources that need to be accessed, employee roles, actual set of employees, etc.).

— National regulations (often access control policy is a subject of government regulations, rules, etc.) need to be taken into consideration in authorization rules.

— Resources for authorization may be constrained. We may need to protect a system in a situation of low-level resource consumption.

Table 25—Concerns of access control viewpoint

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Adaptability	Does the access control policy support a context awareness property?
Maintainability, human factors	Is authorization management an easy and user-friendly process?
Security	Does the system support security properties—confidentiality, integrity, and availability? Do various types of users need different rights for things usage? What types of identity data should be kept for different types of IoT users?
Standardization	Is access control policy contradictory to national regulations/rules?

6.6.10.4 Model kinds

6.6.10.4.1 Identification and authentication

Identification and authentication describes a method of ensuring that a subject (user, thing) is the entity it claims to be. Identification can be provided with the use of an entity name or account number. The confirmation or identity validation process (authentication) is actually done by presenting some kind of proof. To be properly authenticated, the subject is usually required to provide a second piece to the credential set. This piece could be a password, passphrase, cryptographic key, personal identification number (PIN), biometric attribute, or token.

For authentication in an IoT system, we need some kind of secret held by the principal. In its simplest form the participant and the authentication authority share the same secret. The classic authentication mechanism

(e.g., login/password), “Something that you know,” may not directly work in the IoT. The biometric authentication mechanism, “Something that you are,” also is not suitable in the IoT world.

An authenticated entity becomes a subject of access control.

An authentication protocol is a type of computer communications protocol or cryptographic protocol specifically designed for transfer of authentication data between two entities. It permits authentication of the connecting entity as well as authenticating itself to the connecting entity by declaring the type of information needed for authentication as well as syntax.

After authentication, two principals should be entitled to believe that they are communicating with each other and not with intruders. More advanced concepts rely on challenge/response mechanisms, preventing the secrets from being transmitted.

In classic identity management in the internet the authentication triangle consists of three parties or components that are connected to each other:

a) The entity that has to be authenticated by the identity provider

b) The service provider or relying party requiring the authentication of an entity

c) The identity provider (IdP) which is an entity that is apart from other tasks able to authenticate another entity

An identification and authentication model can be defined as an informal model based on the entities’ interactions. In this case, identification and authentication interactions use specific rules that declare what can and what cannot happen between the principals. This approach is extremely error prone.

6.6.10.4.2 Access control model

6.6.10.4.2.1 General

The access control model is a base for access control policy. An access control model is a formalism that dictates how subjects access objects. It uses access control technologies and security mechanisms to enforce the rules and objectives of the model.

The classical access control architectural model can be described in terms of the security reference monitor (NCSC-TG-003 [B81]). The security reference monitor concept enforces an access control policy over subjects’ ability to perform operations on objects on a system. The properties of a reference monitor are listed below:

— The reference validation mechanism shall be non-bypassable so that an adversary cannot bypass the mechanism and violate the security policy.

— The reference validation mechanism shall be evaluable, i.e., amenable to analysis and tests, the completeness of which can be assured (verifiable). Without this property, the mechanism might be flawed in such a way that the security policy is not enforced.

— The reference validation mechanism shall be always invoked. Without this property, it is possible for the mechanism to not perform when intended, allowing an adversary to violate the security policy.

— The reference validation mechanism shall be tamper-proof. Without this property, an adversary can undermine the mechanism itself and violate the security policy.

A modern architectural access control model is described in the eXtensible Access Control Markup Language (XACML) specification (ASIS Committee Draft 03 [B90]). A system has the following components:

— Policy administration point (PAP): PAP creates and manages the policy and policy sets.

— Policy decision point (PDP): PDP is the system entity making access decisions by evaluating the given request against the policies.

— Policy enforcement point (PEP): The PEP component is responsible for making access control decision requests to PDP and the enforcement of the given decisions on the system.

— Policy information point (PIP): The PIP entity is the source of content values for XACML attributes.

This model is applicable for access control in the IoT world and may be considered as a meta-model. The main types of access control models are: discretionary (DAC), role-based (RBAC), and attribute-based (ABAC).

6.6.10.4.2.2 Discretionary access control

6.6.10.4.2.2.1 General

Discretionary access control (DAC)—is an access control policy that is enforced over all subjects and objects in an information system where the policy specifies that a subject that has been granted access to information can do one or more of the following:

a) Pass the information to other subjects or objects

b) Grant its access rights to other subjects

c) Change security attributes on subjects, objects, information systems, or system components

d) Choose the security attributes to be associated with newly created or revised objects

e) Change the rules governing access control

Discretionary access control can be described using the following statements:

— The control of access is based on the discretion of the owner. A system that uses DAC enables the owner of the resource to specify which subjects can access specific resources.

— DAC systems grant or deny access based on the identity of the subject. The identity can be user identity or a group identity (identity based access control), see Figure 40.

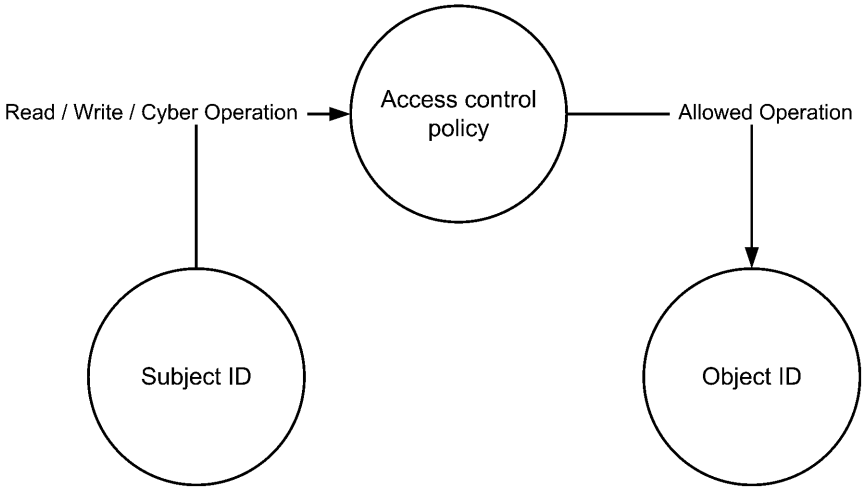


Figure 40—DAC model

Many access control models could express and implement DAC models. For example, the core RBAC model can be expressed in terms of a DAC model (Sandhu and Munawer [B131]). ABAC systems also are capable of enforcing DAC. This is possible due to the DAC model's simplicity, flexibility, and efficiency. In the IoT world, the DAC model is good enough for most solutions due to its simplicity.

6.6.10.4.2.2.2 DAC conventions

6.6.10.4.2.2.2.1 General

An access control matrix is a suitable formalism for a DAC description. An access control matrix is a table of subjects and objects indicating what actions individual subjects can take upon individual objects.

The size of an access control matrix is often reduced with subject grouping, object grouping, and default access rights usage. A default deny principle states that if access is not defined in the access control matrix, then it is implicitly prohibited.

6.6.10.4.2.2.2.2 DAC model kind languages or notations

A DAC model can be defined as an informal model based on the subject-to-object rights. In this case, access control uses specific rules that declare what can and what cannot happen between the subject and object. This approach works quite well for relatively small systems.

XACML is a dialect of XML used to specify and enforce the authorization policies. The XACML Technical Committee defines a core XML schema for representing authorization and entitlement policies (ASIS Committee Draft 03 [B90]). The standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate access requests according to the rules defined in policies. XACML is primarily an attribute-based access control system (ABAC), where attributes (bits of data) associated with a user, action, or resource are inputs into the decision whether a given user may access a given resource in a particular way. DAC can be easily expressed in the scope of the ABAC model, so DAC can conveniently be expressed with XACML language.

6.6.10.4.2.2.2.3 DAC model kind meta-model

The DAC meta-model is depicted in Figure 41.

The description of the DAC model is based on the following notions defined in a meta-model:

— Entity: Every named and referenced system element.

— Object: A passive entity, used in a model for information storage. All operations involve an object as a parameter. Object examples include resources, sensors, actuators, network ports, files, etc.

— Subject: An active entity, users, or non-person entity (NPE); initiates operations in a system. All operations involve a subject as a parameter. Subjects can be organized into groups. Objects may also be put in groups.

— Operation: An action initiated by a subject and performed under an object. Typically, operations perform information flow transfers (read or write object, information actions) or with action execution (cyber actions) (NIST SP 800-162 [B87]).

Subjects, objects, and operations should have the proper granularity and be defined on the proper abstraction level.

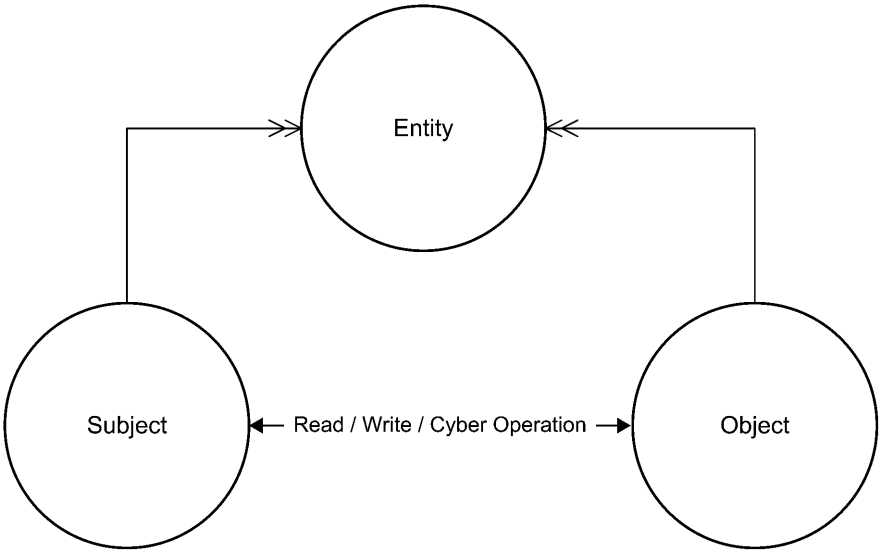


Figure 41 —DAC meta-model

6.6.10.4.2.2.3 DAC operations

DAC access rights may be used exactly as they are defined in the DAC meta-model formalism or they may be tailored through the use of permitted operations. Allowed operations on access rights are assignment, iteration, selection, union, and conflict resolution.

— Iteration: Allows an access control matrix to be used more than once with varying operations.

— Assignment: Allows the specification of an access control matrix.

— Selection: Allows the specification of one or more items from an access control matrix.

— Union: Allows the union of two or more access control matrices.

— Conflict resolution: Procedure for an access control decision in the case of a conflict between two or more access control matrices.

6.6.10.4.2.3 Role-based access control

6.6.10.4.2.3.1 General

Role-based access control (RBAC) is based on user roles or IoT functionality and uses a centrally managed set of controls to determine how subjects and objects may interact in the IoT world.

While RBAC marks a great advance in access control, the administrative issues of large systems still exist, albeit in a markedly more manageable form. In large systems, memberships, role inheritance, and the need for finer-grained customized privileges make administration potentially unwieldy.

Due to their adaptability, role-based access control models are quite popular in different application-level domains, such as Web and database application access control. There are many extensions and modifications of these models, which can be easily adopted to usage in the IoT domain.

6.6.10.4.2.3.2 RBAC conventions

In RBAC for IoT, the roles that active entities have in an IoT system underpin the access decisions. The role name designates the specific set of access rights and restricts the use of resources for the active entities authorized to use the role. The RBAC model grants the operating entities membership in roles according to their responsibilities and associated functions. To use the role, the entity shall activate it. The operations that an entity is allowed to perform depend on the roles that are currently active for the entity.

RBAC conventions also contribute the procedures of access control management. The membership in the new role can be established if it is required by appropriate assignments of responsibilities to the entity. Also the membership in a role for the entity can be revoked. In the IoT world, the thing functionality can be represented as the set of services. In this case, the RBAC model can regulate the access to these services according to the roles assigned to other things.

The basic RBAC model is shown in Figure 42. This core model embodies the essential aspects of RBAC. At first, it assigns active items to roles. Items acquire permissions according to this direct assignment. Item-role and rights-role assignment is a many-to-many relation. Items can use the different role permissions simultaneously. The model includes the concept of user sessions. A session is a mapping between a user and an activated subset of assigned roles. The role activation for the session is not restricted for any roles assigned to the user.

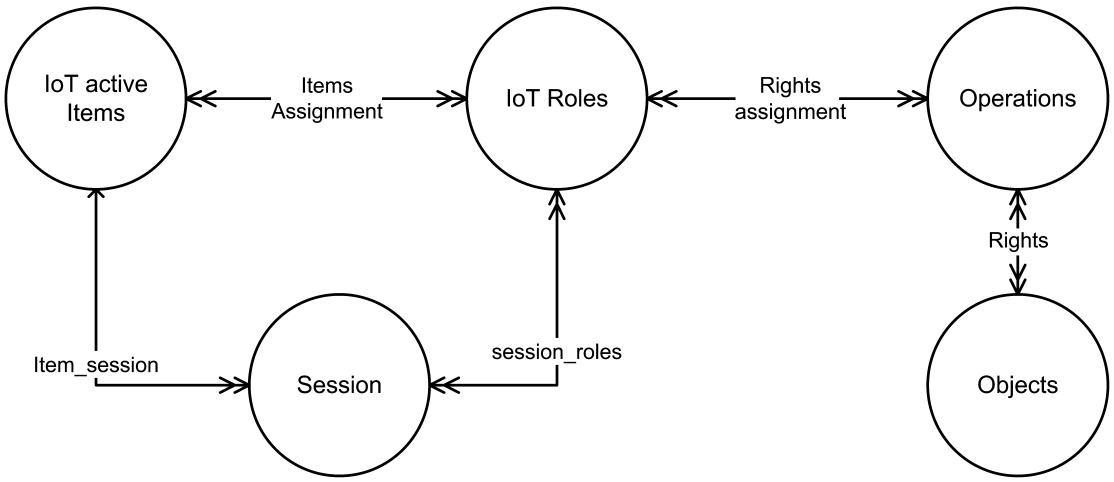


Figure 42—RBAC core model

The basic RBAC model can be extended with access control management extensions.

A role hierarchy (see Figure 43) defines roles that have unique attributes and that may contain other roles. One role may implicitly include the operations that are associated with another role. The hierarchical model adds the concept of privilege inheritance. It is useful in many widely-used scenarios. Items can obtain permissions according to the direct assignment or they can inherit permissions. This concept is useful for specifying access functions for versatile items. The corresponding RBAC model with hierarchy extension is shown in Figure 43.

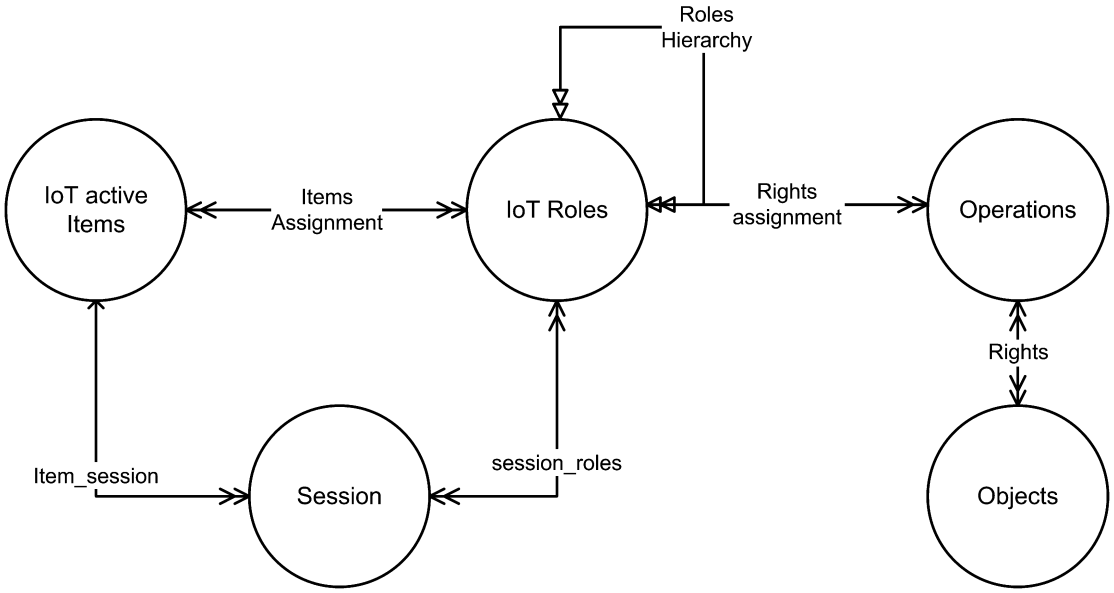


Figure 43—RBAC model with roles and hierarchy

An RBAC model with roles hierarchy and constraints (see Figure 44) covers static and dynamic constraints. Temporal constraints are formal statements of access policies that involve time-based restrictions on access to resources; sometimes required in IoT scenarios. In some applications, temporal constraints may be required to limit resource use. In other types of applications, they may be required for controlling time-sensitive activities. These time-based constraints should be evaluated for generating dynamic authorizations

during execution time. Popular access control policies related to temporal constraints are the history-based access control policies. History-based access control is defined in terms of subjects and events where the events of the system are specified as the object access operations associated with activity at a particular security level. This assures that the security policy is defined in terms of the sequence of events over time, and that the security policy decides which events of the system are permitted to ensure that information does not "flow" in an unauthorized manner. Static and dynamic separation of duties and various constraint rules are shown in Figure 44.

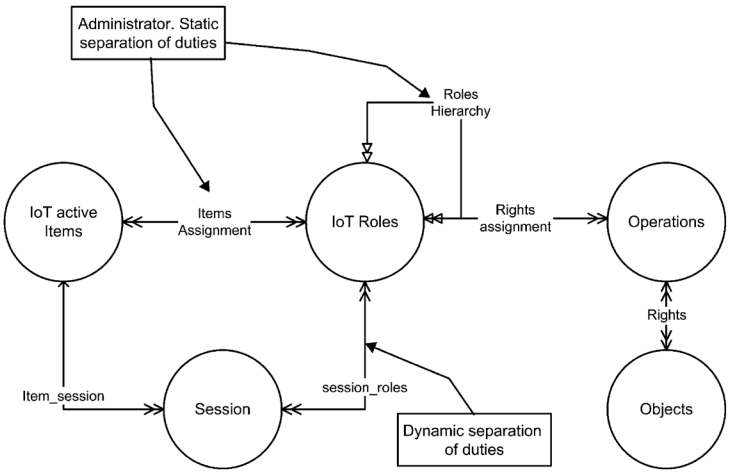


Figure 44—Model with roles hierarchy and constraints

Figure 45 shows the family of standardized role-based models. As shown, the core model can be refined with role hierarchy and static or dynamic constraints on operations. The comprehensive model includes all the extensions.

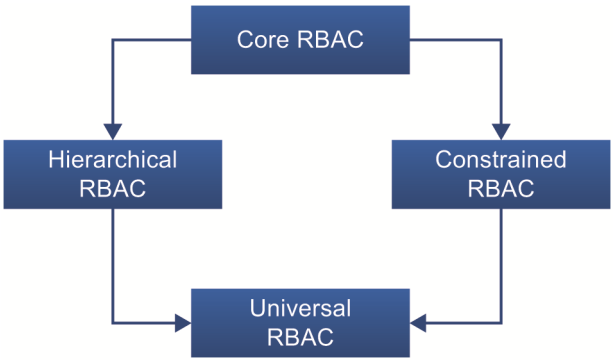


Figure 45—RBAC models family

6.6.10.4.2.3.3 RBAC model kind languages or notations

The RBAC model for IoT can be defined as an informal model. In this case, access control uses specific rules that indicate what can and cannot happen between a role, subject, and object. This approach works well for small, simple systems.

The XACML Technical Committee defines a core XML schema for representing authorization and entitlement policies (ASIS Committee Draft 03 [B90]). The standard defines a declarative access control

policy language implemented in XML and a processing model describing how to evaluate access requests according to the rules defined in policies. Roles are expressed as XACML subject attributes (one or more). The standard implements core and hierarchical RBAC components.

The UML programming language is a native approach for RBAC model description in the IoT world. Annotation-based approaches could be used for model access specification. The first annotation-based approach combines the UML and OCL languages (Basin and Doser [B9], Lodderstedt, Basin, and Doser [B74]), and the second uses the UML stereotypes (Jan [B69]).

OCL language exposes the following constructions for access specification: invariants, operations, pre- (and post-) conditions, and state transition restrictions. The UML extension proposes the following constructions for access specification: stereotypes (specializes model element using <<label>>), tagged values (attaches {tag = value} pair to stereotyped element), and constraints (refines semantics of stereotyped element).

6.6.10.4.2.3.4 RBAC operations

RBAC roles and access rights may be used exactly as they are defined in the RBAC model formalism, or they may be tailored using the allowed operations. Allowed operations on access rights are assignment, iteration, selection, union, and conflict resolution.

- Iteration: Allows access rights to be used more than once with varying operations.
- Assignment: Allows the specification of an access control matrix; assignment can be direct or indirect (inheritance for hierarchy entities).
- Selection: Allows the specification of one or more rights to a role.
- Union: Allows union of rights from two or more active roles.
- Conflict resolution: Procedure for access control decision in case of conflict of two or more access rights from different roles.

6.6.10.4.2.4 Attribute-based access control

6.6.10.4.2.4.1 ABAC conventions

6.6.10.4.2.4.1.1 General

The attribute-based access control model enforces the rules based on the attributes of entities.

There are characteristics or attributes of a subject such as a name, date of birth, home address, training record, and job function that may, either individually or when combined, comprise a unique identity that distinguishes that person from all others. These characteristics are called *subject attributes*. Like subjects, each object has a set of attributes that help describe and identify it. These traits are called *object attributes* and are sometimes referred to as *resource attributes*. Other object attributes are object content, object informational state, etc. Environment conditions are dynamic factors, independent of subject and object, that may be used as attributes at decision time to influence an access decision. Examples of environment conditions include time, location, threat level, and temperature.

Therefore, ABAC is a model that is based on subject, object, and environment attributes (see Figure 46).

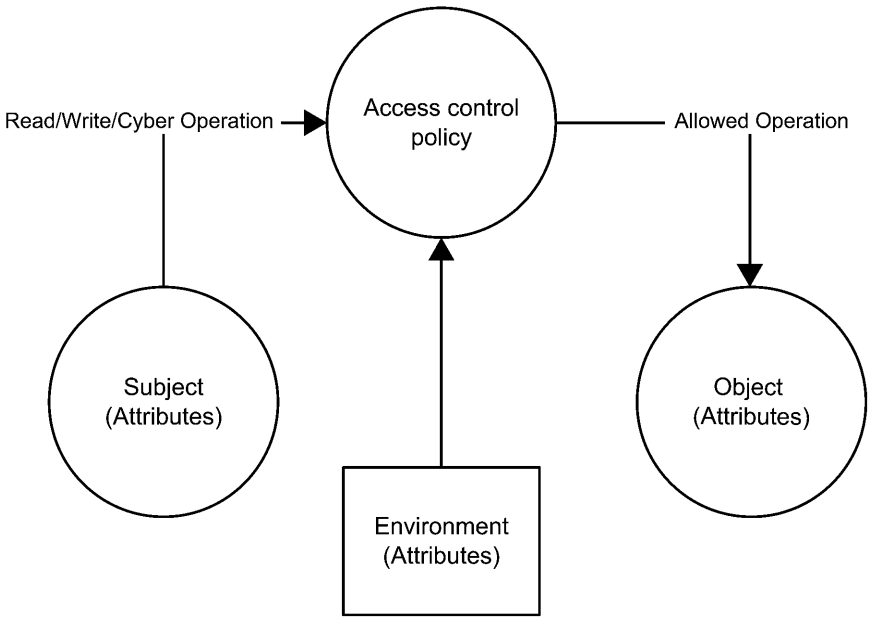


Figure 46—ABAC model

The ABAC model is simpler to implement than RBAC. Also ABAC can accommodate real-time environment states as access control parameters.

6.6.10.4.2.4.1.2 ABAC model kind languages or notations

ABAC is consistent with XACML. ABAC systems are capable of enforcing both discretionary and role-based access control models.

6.6.10.4.2.4.1.3 ABAC model kind meta-model

The ABAC meta-model is presented in Figure 47.

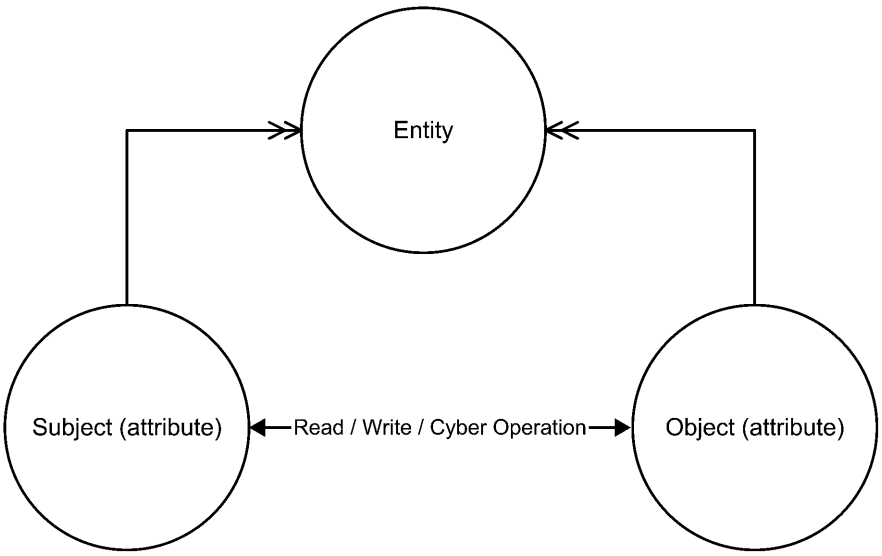


Figure 47—ABAC meta-model

Description of the ABAC model based on the following notions defined in a meta-model:

- Entity: Every named and referenced IoT system element.
- Object: A non-operating entity, used in a model for information storage. All operations involve an object as a parameter. Object examples are resources, sensors, actuators, network ports, files, etc.
- Subject: An operating entity, such as a user or non-person entity (NPE), initiates operations in a system. All operations involve a subject as a parameter. Subjects can be organized into groups or roles. Objects can also comprise groups.
- Environment conditions: Dynamic attributes independent of subject and object that may affect access decision. Examples of environmental conditions are time, location, threat level, and temperature.
- Attribute: A well-defined and measurable characteristic of subject, object, or environment.
- Operation: An action initiated by a subject in an authorized role and performed under an object. Typically, operations perform information flow transfers (read or write object, information actions) or with actuator execution.

Subjects, objects, roles, and operations should have the proper granularity, be uniform and adequately abstract.

Information about policy, such as author, policy effective date, conflict resolution methods, etc. are sometimes called *meta-policy*. Information about attributes such as attribute authority, attribute creation date, etc. are sometimes called *meta-attribute*s (RFC 1492 [B125]). Meta-policy and meta-attributes depend on the application domain and are used in view construction.

6.6.10.4.2.4.2 ABAC operations

ABAC attributes may be used exactly as they are defined in the ABAC model formalism or they may be tailored through the use of permitted operations. Allowed operations on attributes are assignment, iteration, selection, union, and conflict resolution.

- Iteration: Allows attributes to be used more than once with varying operations.

— Assignment: Allows the specification of attributes. Assignment can be direct or indirect (inheritance for hierarchy attributes).

— Selection: Allows the specification of one or more attributes.

— Union: Allows union of attributes.

— Conflict resolution: Procedure for access control decision in the case of a conflict between two or more attributes.

6.6.10.5 Operations on views

6.6.10.5.1 General

Operations define the methods to be applied to views and their models. Types of operations include:

— Construction

— Analysis

— Implementation

6.6.10.5.2 Construction

6.6.10.5.2.1 General

Construction of an access control model requires the analysis of application domain security policies and requirements. The model construction method depends on IoT system complexity and the chosen access control model.

6.6.10.5.2.2 DAC model construction

A DAC model should list all system subjects, objects, and access operations. A system that uses DAC enables the owner of the resource to specify which subjects can access specific resources. Therefore, the system should allow the specification of access control rules by the owner (users or administrators). By default, secure access control configuration should be done by system developers or system integrators. Correspondence rules CR1 to CR5 should be respected under DAC model construction.

6.6.10.5.2.3 RBAC model construction

The construction of an RBAC model (role engineering) for an IoT system is based on the specification process shown in Figure 48.

Scenario-driven role engineering utilizes scenarios of system use to derive the appropriate user permissions (Baumgrass, Strembeck, Rinderle-Ma [B8]). In general, the scenario describes possible or actual action and event sequence. Thus, to implement a scenario, the subject needs to be provided with the exact set of permissions that are necessary to complete each step of this scenario. After describing the permissions, the scenarios are grouped to form the tasks and work profiles. In general, the scenarios and process models can be defined in a wide variety of (modeling) languages, such as UML activity and interaction models, event-driven process chains (EPCs) (Hommes [B34]), Business Process Model and Notation (BPMN) models (Business Process Model and Notation [B93]), or via the Business Process Execution Language (BPEL) (WS-BPEL 2.0 [B92]).

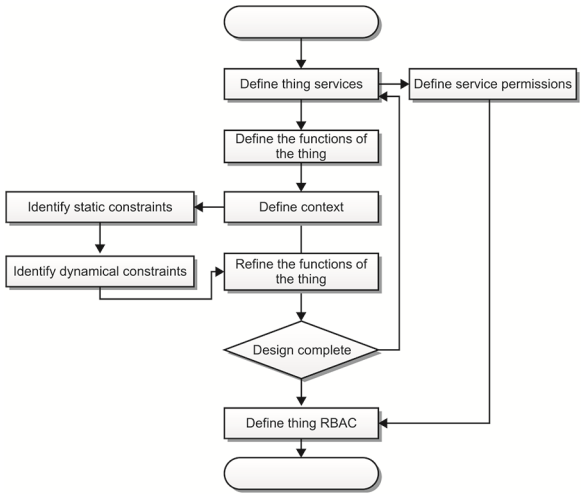


Figure 48—RBAC role definition for IoT

6.6.10.5.2.4 ABAC model construction

System developers or system integrators can construct an ABAC model for IoT similar to an RBAC model. The exception is that after role engineering one should map the roles to attributes, define the attribute-based policies for role-permission assignment, and perform the user-to-role assignment. Attributes should also be defined for environment specification. Attributes should be defined, described, and managed for a potentially large number of entities. Furthermore, attributes have no sense until they are associated with a user, object, or relation. It does not matter which users have access to a given permission and what permissions have been granted to a given user without the information about appropriate attributes (Coyne and Weil [B18]).

6.6.10.5.3 Analysis

To help ensure the proper implementation and enforcement of access control rules, auditing of the actions (which are under control) or related events is applied. These actions and events should be accountable.

For DAC model analysis we need to analyze not only informal security estimation but also verify the right propagation in the model. The access control right propagation method should be formally estimated to check the possibility of access rights leaking. Well-known models that propose such analysis methods are the Harrison-Ruzzo-Uhlman model, Take-Grant model, and the Typical Access Matrix model (Bishop [B7]).

Access rights propagation is prohibited in an RBAC model, so analysis is required only for the consistency checking of the model developed for the IoT system with this system. The analysis of an ABAC model is a more complicated task. As a rule, RBAC requires significant effort to configure, but allows easy management and user permission review, while ABAC makes the reverse trade-off; it is easy to set it up, but analyzing or changing user permissions can be problematic (Kuhn, Coyne, and Weil [B71]).

Accountability is the method of tracking and logging the subject's actions on the objects. Auditing is an activity where the user/subject actions on the objects are monitored in order to verify that the sensitivity policies are enforced and can be used as an investigation tool.

Access control model construction should comply with following rules:

a) All subject-to-object access should be implicitly or explicitly defined in an access control matrix.

b) All subject-to-object access should be controlled by access control mechanisms (including non-bypassable control enforcement).

c) The model should define the procedure for conflict resolution (or substantiate the impossibility of conflicts).

d) The consequence of operations necessary for implementing access along with all related checks is a transaction. The transaction is considered successful only if all corresponding operations are successfully executed.

e) A basic RBAC model can be transformed into a DAC model. All correspondence rules defined for the DAC model are also set for the RBAC model.

f) DAC and RBAC are in some ways special cases of ABAC in terms of the attributes used. DAC works on the attribute of *identity*. RBAC works on the attribute of *role*.

6.6.10.5.4 Implementation

6.6.10.5.4.1 Identification and authentication implementation: authentication protocols

The way to store and present identification and authentication information may be substantially different for IoT devices. Note that in typical enterprise networks, the endpoints may be identified by a human credential (e.g., username and password, token, or biometrics). The IoT devices should be presented in an identification and authentication process by identifiers without human interactions. Such identifiers include radio frequency identification (RFID), shared secret, X.509 certificates, the MAC address of the devices, or some type of immutable hardware-based root of trust.

Secure Sockets Layer (SSL) provides a secure method of communication for TCP connections, especially for HTTP connections. The IP Authentication header provides strong authentication and integrity for IP datagrams. Depending on the signing algorithm used, it may also provide non-repudiation, excluding those fields that are changed during transmit, like hop count or time to live. The authentication is transport-protocol independent, so there may be data from more than one different protocol, for instance TCP and UPD. The authentication data are calculated with a message digest algorithm.

Kerberos authentication was developed at the Massachusetts Institute of Technology (MIT). There are two main components: a ticket, which is used for user authentication and securing data, and an authenticator that is used to verify that the user is the same user to whom the ticket was initially granted. The major issue with Kerberos is its scalability. The Kerberos server shall store secret keys for each of the users and each of the ticket-granting service (TGS). Kerberos can get very complex in IoT implementations where trust relationships need to be in place between multiple IoT systems.

Establishing authentication through X.509 certificates provides a strong system. However, in the IoT domain, many devices may not have enough memory to store a certificate or may not even have the required CPU power to execute the cryptographic operations of validating the X.509 certificates (or any type of public key operation).

Modern identification and authentication methods such as IEEE 802.1AR and authentication protocols as defined by IEEE 802.1X can be leveraged for low resource devices that can manage both the CPU load and memory to store strong credentials. However, the challenges of the new form factors, as well as new modalities, create the opportunity for further research in defining smaller footprint credential types and less compute-intensive cryptographic constructs and authentication protocols.

6.6.10.5.4.2 Access control implementation

6.6.10.5.4.2.1 DAC implementation

Different implementation technologies are available to support the different access control models.

For a DAC model the access control matrix could be successfully implemented at the low level of the IoT system. In this case, low level means the operating system of a local system or networking transport model level for a distributed IoT system.

The implementation of DAC may be based on the access control lists (ACLs) which are configured by the system owners and enforced by the operating system. Access control lists map values from the access control matrix to the object, describing the subjects that can or cannot access this object. The particular sequence of ACL elements may be inconsistent, so a conflict resolving procedure is required. Revoking privileges from the ACL can be difficult.

The access rights that are assigned to individual subjects are called *capabilities*. Each row in an access matrix is a subject capability. This technique uses a capability table to specify the capabilities of a subject pertaining to specific objects. A capability can be in the form of a token, ticket, or key. Kerberos uses a capability-based system where every user is provided with a ticket from within the appropriate table. Capabilities can be hard to revoke. This approach is suitable for low-level resources in the system.

6.6.10.5.4.2.2 RBAC and ABAC implementation

6.6.10.5.4.2.2.1 General

For RBAC and ABAC models, the application level of implementation is a common solution. Currently a common approach to the implementation of these access control models does not exist.

In Microsoft active directory, RBAC is implemented using groups to allow users to perform specified functions based on those group roles. In a typical Microsoft environment, users are assigned to one or more groups to control their access to systems and functionality based on their job functions.

Security Enhanced Red Hat Linux (SELinux) has features that enable RBAC implementation. The user's role is an attribute of RBAC. In SELinux, users are authorized for roles, and roles are authorized for domains. For SELinux users, a role determines the relationship between domains and users. Access to a domain is controlled by the user's role. Ultimately, the object types a user can access are based on the user's role.

Regarding IoT, we can distinguish two common situations—decision with central appliance and without central appliance.

6.6.10.5.4.2.2.2 Access control with central appliance

PDP is implemented at the security gateway, which governs access to all end-point devices (Figure 49).

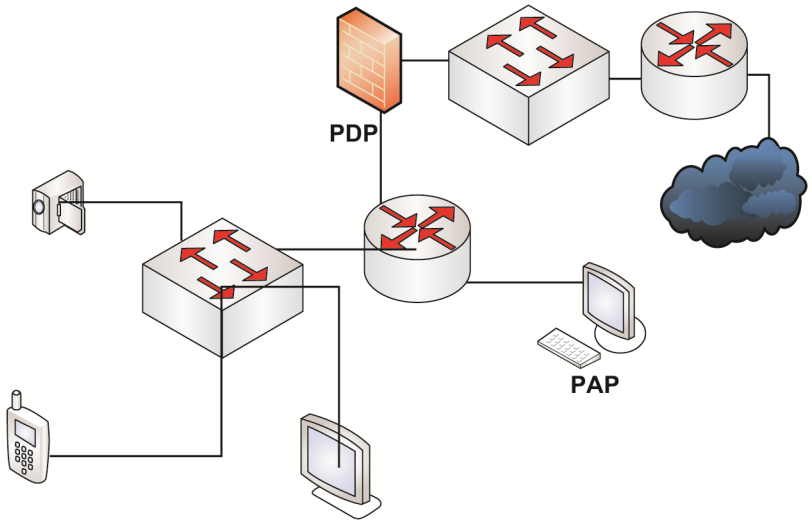


Figure 49—IoT access control with central appliance

This approach has the following advantages:

— Constrained resources on the end-devices do not affect the overall system efficiency

— The possibility of complex access control models implementation

— No need to modify end-devices to implement access control

— The normalized form of requests to end-devices potentially reduces the attack surface for the IoT system

This approach has the following disadvantage:

— The central appliance is a single point of failure for the overall IoT system.

6.6.10.5.4.2.2.3 Access control without central appliance

PDP is implemented at end-point devices (Figure 50).

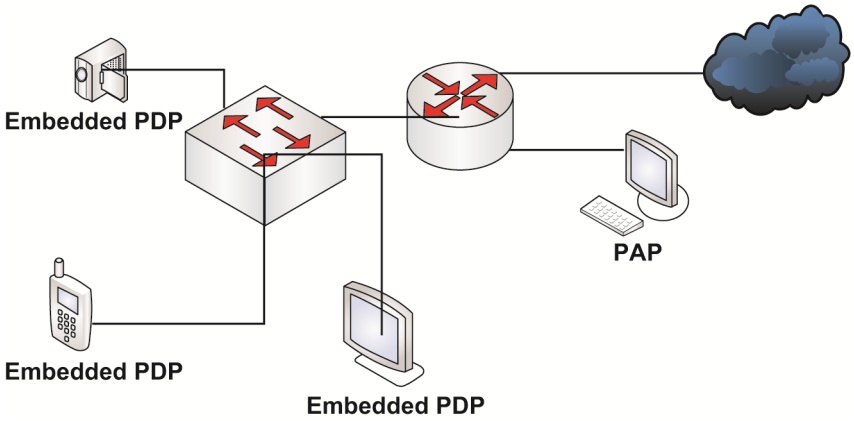


Figure 50—IoT access control without central appliance

This approach has the following advantage:

— There is no single point of failure.

This approach has following disadvantages:

— End-devices resource constraints may cause issues with the access control implementation.

— The implementation of complex access control models in a distributed system may be challenging.

— Non-uniform requests to the end-devices potentially increases the attack surface.

6.6.10.6 Correspondence rules

The access control viewpoint depends on functional, informational, and threat viewpoints. All access control specifications and transformations should comply with the correspondence rules in Table 26.

Table 26—Access control viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
ACA-CR-1	Functional	All IoT operations defined in the functional viewpoint should be subject of access control and the access control architecture viewpoint.
ACA-CR-2	Threat model and functional	Policy decision point implementation for the access control viewpoint depends on the threat model and functional viewpoint.
ACA-CR-3	Security and safety monitoring	The access control viewpoint is a basis for security and safety monitoring viewpoint.

6.6.11 Adequate design for required security viewpoint

6.6.11.1 General

Some IoT systems are operated and maintained over a long time period. The security measures of IoT systems will have to continually deal with environmental changes adequately throughout the lifecycle of the system. The adequate design for required security viewpoint is intended to be used by designers and maintainers for the target IoT system to describe the adequate design for required security view.

This viewpoint is intended to be used by IoT designers and maintainers to design adequate security measures of an IoT system by using the Hardening—Adaptivity, Responsivity, Cooperativity (H-ARC) model (IEC White Paper 2015 [B52], Nakano et al. [B80], IEC: White Paper 2020 [B53]).

This document shows that each security measure defined by the H-ARC model can adapt well to the changes in the security environment during the period when target IoT systems are operated.

It also shows how to use the H-ARC model with a usage example in a system design phase and an operational phase.

6.6.11.2 Stakeholders and concerns

6.6.11.2.1 Typical stakeholders

The typical stakeholders for the adequate design for required security viewpoint are listed in Table 27.

Table 27—Adequate design for required security viewpoint stakeholders

Stakeholders for this viewpoint	Stakeholder role	Common stakeholder alignment
Designers of a system	Formulate adequate security policies for the target system and design it in the system planning/system design phase	Acquirers, assessors, builders, developers, owners, maintainers, operators, production engineers, system administrators, testers, users
Maintainers of a system	Take adequate security measures through a system lifecycle	Maintainers, operators, owners, production engineers, support staff, system administrators, testers, users

6.6.11.2.2 Concerns

Table 28 shows main concerns of adequate design for required security.

Table 28—Concerns of adequate design for required security viewpoint

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Measurability	In the initial system design phase for security of the target system: Can we take adequate security measures for the target system? Can we recognize the sufficiency of the result of the security measures?
Evolvability	In the operation phase for security of the target system: Can we continue to take adequate and timely security measures against environmental changes (technologies, threats, system configurations, operations, etc.) after the initial system design?

6.6.11.2.3 Anti-concerns

Anti-concerns for adequate design for required security are as follows:

— It does not provide concrete security measures to be equipped in a target system.

— It does not provide objective evaluation of adequacy for security measures.

6.6.11.3 Model kinds

This viewpoint uses the H-ARC model. The model helps stakeholders take adequate security measures for the target system using four security points of view. Stakeholders using this model can take adequate security measures during both the system design phase and the operating phase through a system lifecycle.

The H-ARC model depends on a basic idea of the symbiotic-autonomous decentralized system (S-ADS) model (IEC White Paper 2015 [B52], IEC White Paper 2020 [B53]). Each element forming security measures are improved and strengthened autonomously according to security environmental change.

6.6.11.4 H-ARC

6.6.11.4.1 General

H-ARC helps stakeholders make clear security measures for IoT systems adequately from four points of view:

— Ensuring robustness (hardening) with respect to likely threats based on the target system's configuration

— Giving security measures the ability to adapt as needed to continually changing threats and system configurations (adaptivity)

— Providing responses that will minimize the impact on the IoT system if a security threat does materialize (responsivity)

— Having different organizations work together to help ensure the early identification of security threats (cooperativity)

6.6.11.4.2 H-ARC conventions

Figure 51 shows a structure of the H-ARC model. The H-ARC model comprises the following:

a) IoT Security policy: It consists of four points of view, which are hardening elements (b), adaptivity elements (c), responsivity elements (d), and cooperativity elements (e).

b) Hardening elements: These are fundamental security elements to help ensure the robustness with respect to likely threats. Technology examples are traffic control, authentication, and encryption technologies. Popular devices that apply these technologies are firewalls, active network monitoring, and control devices capable of disconnecting an unauthorized device forcibly, or unidirectional gateways.

c) Adaptivity elements: These are security elements to measure the ability to adapt as needed to counter continually changing threats and system configurations. For example, they consist of configuration changes or programs that update according to a change of security policy.

d) Responsivity elements: These consist of security elements to minimize the damage and facilitate a speedy recovery if a security threat does materialize. For example, a technology that disconnects a device infected with a virus forcibly minimizes the damage on the overall IoT system.

e) Cooperativity elements: To protect IoT systems against security threats, it is important for a number of organizations to work together cooperatively. These elements, for example, consist of cooperation methods between organizations, and methods to transmit security information.

f) Security environmental change (events/incidents, technologies, products, business, time, organizations, etc.) (f) forces each element—(b) (c) (d) (e)—of IoT Security policy (a) to improve their security measures. They also wake up IoT Security policy (a) and system designers/maintainers (g) to improve.

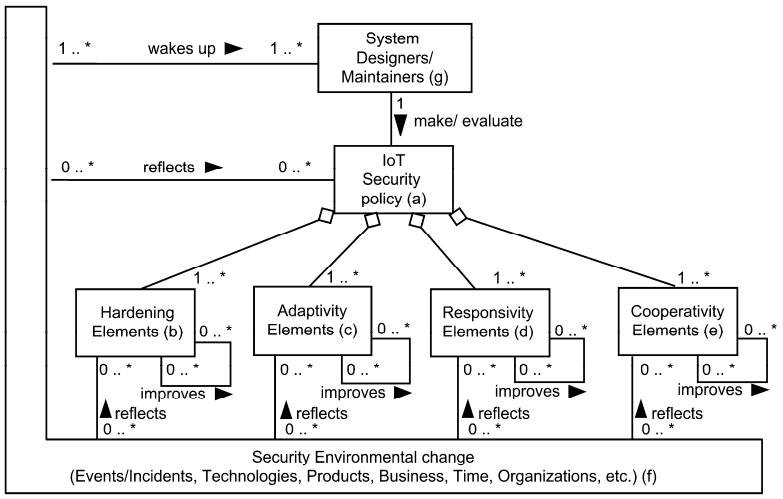


Figure 51—Structure of H-ARC model

6.6.11.4.3 H-ARC correspondence rules

This viewpoint defines the correspondence rules among hardening (H), adaptivity (A), responsivity (R), and cooperativity (C) on the H-ARC model in Table 29.

Table 29—Adequate design for required security viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
ADS-CR-1	Security-related viewpoints	Security measures on the system are included in any of "H," "A," "R," and "C."
ADS-CR-2	Security-related viewpoints	Each security measure defined by "H," "A," "R," and "C" elements may not be independent but may depend on each other element.
ADS-CR-3	Threat model	Some security measures of "H," "A," "R," and "C" elements may be imported from the result of other security viewpoints such as the threat model viewpoint.

6.6.11.5 Operations on views

6.6.11.5.1 General

Operations define the methods to be applied to views and their models. Types of operations include:

— Construction methods

— Analysis methods

— Implementation methods

6.6.11.5.2 Construction methods

The target IoT system is analyzed and evaluated from the four views (hardening—adaptivity, responsivity, cooperativity) of the H-ARC model.

The hardening view evaluates the strength of security measures in accordance with the security assurance levels (SAL) criteria defined by IEC 62443-3-3 [B49]. For example, IoT systems applied to social infrastructure systems require level 3 or 4 security measures.

In the same way, the adaptivity, responsivity, and cooperativity views also evaluate the SAL and take security measures.

It is important to evaluate the SAL and to take security measures across two different lifecycle phases (system design phase and operational phase).

Operations are as follows:

a) From a hardening view:

— Evaluate the latest security technologies and products that can be applied to the target system that needs to be protected.

— Formulate best practices.

b) From an adaptivity view:

— Assess how to deal with threats and risks.

— Formulate the practices in terms of cyberspace, physical space, and operational management, and implementation plans.

— Formulate the methods to update the system against new threats.

— Formulate the methods of maintaining operation of retrofitted systems when adding new systems and components.

c) From a responsivity view:

— Formulate the methods of incident monitoring and incident detection.

— Establish/formulate an operational organization having the responsibility for the incident response and procedures.

— Formulate the methods of system operation continuity and quick recovery after an incident.

d) From a cooperativity view:

— Formulate the inter-organizational communication methods.

  For example, to formulate a mechanism to exchange information between organizations (standardize the meaning of terminology used to indicate the security situation at each organization, protocol, and unified information management).

— Formulate the methods for confirming the reliability of other organizations and for exchanging information from other organizations.

6.6.11.5.3 Analysis methods

It is necessary to apply the latest security technologies and products to IoT systems in accordance with the security policy. Continued timely security measures using the plan, do, check, act (PDCA) cycle or observe, orient, decide, act (OODA) loop should be taken. It is also necessary to confirm the establishment of appropriate relationships among the concerned organizations.

a) Adaptivity through the PDCA cycle:

Identify new threats, establish improvement practices (plan), formulate implementation plans (do), implement security measures (check), and apply improvement (act).

b) Responsivity through the OODA loop (Mimura [B77]):

To minimize the impact of security threats on a system, it is important to identify the signs of a threat quickly and to take action against it.

To make an effective approach, change detection (observe), damage assessment (orient), establishment of countermeasures (decide), and issue countermeasure instructions (act).

6.6.11.5.4 Implementation methods

It is important to implement security measures from two points of view, namely the system design phase and the operational phase.

a) From the point of view of the system design phase:

To build a system utilizing the security concepts (Nakano et al. [B79]) advocated in IEC 62443-2-1:2010 [B48], IEC 62443-3-3:2013 [B49], and IEC 62443-3-2 (draft).

1) Partitioning of the system into zones in which the same security policy applies based on a risk analysis of the system
2) Identification of conduits (interconnections) between zones
3) Formulation of security measures

For example:

— Installation of an active network monitoring and control device (IIC: Industrial Internet of Things Volume G4 [B56]) that disconnects an unauthorized device forcibly on the network within a zone

— Installation of a unidirectional gateway (IIC: Industrial Internet of Things Volume G4 [B56]) that permits data transmission from only one side of the zone

Figure 52 shows an implementation example.

b) From a point of view of the operational phase:

1) Implement measures for responding rapidly to security incidents
2) Establish formal security management systems

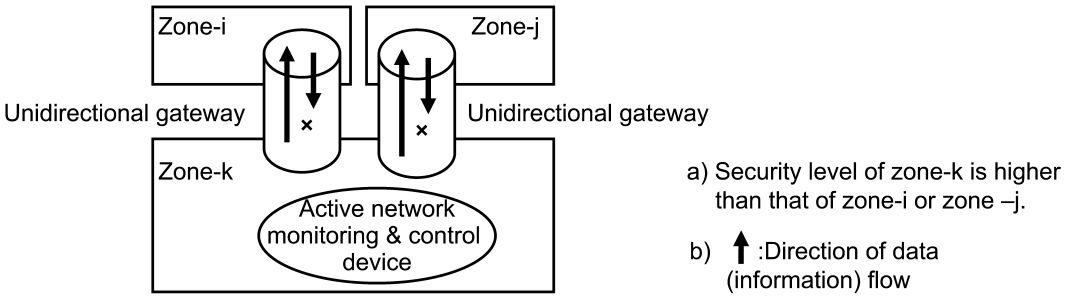


Figure 52—Implementation example

6.6.11.6 Correspondence rules

This viewpoint defines the following correspondence rules regarding hardening (H), adaptivity (A), responsivity (R), and cooperativity (C) from the H-ARC model in Table 30.

Table 30—Adequate design for security viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
ADS-CR-4	Security-related viewpoints	Each security measure defined by H, A, R, and C elements are well adapted to the security environmental changes during the period in which the target IoT systems are operated.
ADS-CR-5	Security-related viewpoints	This viewpoint's security measures and other viewpoint's security measures are mutually related.
ADS-CR-6	Security-related viewpoints	Each security measure defined by H, A, R, and C, elements can import the security measures of the other viewpoint. For example, H sometimes imports the result of access control viewpoint (6.6.10) as one of the security measures.
ADS-CR-7	Collaboration	This viewpoint is applied not only within a particular domain but also among domains defined in collaboration viewpoint (6.6.13).

6.6.12 Privacy and trust viewpoint

6.6.12.1 General information and key features

6.6.12.1.1 General

The privacy and trust architecture viewpoint describes privacy and trust aspects of IoT architectures. Privacy and trust is a wide field. There are many concepts, technologies, and implementations, as well as regulations and laws that are dealing with the privacy of individuals or groups and trust in systems or organizations. Most of the approaches clearly distinguish between personal data that can be linked to a certain person and other arbitrary data.

Privacy and trust becomes crucial in the IoT because even arbitrary data, like a temperature, might be related to a user when it is combined with other data like location or it is profiled over a certain time period. An example of a privacy issue is the ability to determine what kind of television program a user is watching just from measuring the energy consumption with very frequent samples (Greveler, Justus, and Loehr [B29]).

Privacy is perceived very differently in various regions, countries, and cultures. This viewpoint intends to address various concerns and provide basic models. Architects and system designers can choose aspects that are appropriate to their privacy and trust requirements.

Trust is fundament for all business relationships. People will only use systems in the long run when they are trustworthy.

Different definitions of privacy and trust have been proposed. Both terms have social aspects and cannot only be explained in a technical way. This subclause introduces design strategies, guidelines, and models, as well as examples of technical concepts to address the privacy and trust concerns of different stakeholders.

Privacy

The UNESCO views privacy as an important and fundamental right and explains it in the following way:

Privacy is a fundamental right, even though it is difficult to define exactly what that right entails. Privacy can be regarded as having a dual aspect—it is concerned with what information or side of our lives we can keep private; and also with the ways in which third parties deal with the information that they hold—whether it is safeguarded, shared, who has access and under what conditions (Mendel et al. [B75]).

Trust

Trust is a social construct and expresses the belief that a person or a system behaves in an expected way within a certain context.

This document describes two notions of trust. One notion is trust in the behavior of a person or system. The second notion of trust is the belief in the authenticity of a person or a system. This is in fact a precondition of the first notion.

6.6.12.1.2 Privacy by design

Ann Cavoukian, Information and Privacy Commissioner of Ontario, Canada, wrote Privacy by Design—7 Foundational Principles (Cavoukian [B15]), a paper that described the privacy-related design principles that should be taken into account when building an IT system. These principles shall also be applied to IoT systems:

a) Proactive not reactive

   A system should be designed in a privacy-preserving way from the beginning and not just being adapted when privacy infractions occur.

b) Privacy by default

   When a system is delivered or installed, no further action should be necessary to have the most restrictive privacy settings. Even if a user or an administrator does no initial configuration at all, the system should be in privacy-preventing status.

c) Privacy is embedded

   Privacy should always be a part of the core system, not an extra package or optional add-on.

d) Full functionality

   Privacy should not diminish the functionality of the system.

e) End-to-end security—full lifecycle protection

   The system should see the data across the complete lifecycle. So data should not only be securely retained but also destroyed in a secure way when it is not needed anymore. End-to-end security refers to the principle of taking all components into consideration starting from clients, server, and up to the data layer.

f) Visibility and transparency

   The system design should be transparent and designed according to stated objectives and design goals. The design should be subject to independent verification.

g) Respect for user privacy

   The system shall be designed in a user-centric way.

6.6.12.1.3 Privacy legislation

Privacy is subject to regulation in different legislations. Data protection laws are different depending on region and country. The European General Data Protection Regulation (GDPR) was ratified in early 2016 and is an example for a transnational privacy regulation (EU General Data Protection Regulation [B24]).

NOTE—The final enforcement of GDPR was in May 2018, at which time those organizations in non-compliance face heavy fines if they are doing business in the European Union.

The GDPR has some definitions having direct impact to a privacy-preserving architecture in terms of how to handle storage, access, and user control of personal data. The GDPR defines personal data:

'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.

As a consequence, GDPR sees IP addresses, location data, pseudonyms, etc. as personal data and as such these data have to be handled and protected.

Definition of user consent:

any freely given, specific, informed and unambiguous indication of his or her wishes by which the data subject, either by a statement or by a clear affirmative action, signifies agreement to personal data relating to them being processed ...

It states clearly that the collection of personal data needs a clear purpose and it should be reasonable when and how the consent was given. Furthermore there should always be a way for the user to withdraw the given consent.

"The right to be forgotten" is defined:

…The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay…

Basically, the GDPR forces service providers to erase data on demand or if they are no longer needed for fulfilling the service.

6.6.12.2 Stakeholders and concerns

6.6.12.2.1 Typical stakeholders

The stakeholders for the privacy and trust viewpoint are listed in Table 31.

Table 31—Privacy and trust viewpoint stakeholders

Stakeholders for this viewpoint	Stakeholder role	IoT common stakeholder alignment
Acquirers	Oversee the privacy aspects of a system during the procurement process	Acquirers
Developers	Implementing an IoT system in a privacy ensured way	Developers
Data controller	Determines purposes and means of processing of personal data	Operators, support staff, system administrator
Data processors	Process privacy effecting data in order to provide service to the user of the IoT system	Operators, support staff, system administrator
Assessors	Oversee and/or prove the conformance of the IoT system with privacy regulations	Users, maintainers
Owner	Owners of an IoT system have legal privacy obligations and/or benefit from protecting privacy of their users	Owners, operators, acquirers
System users	Users of an IoT system being aware and/actively managing their privacy aspects and configurations	Users

6.6.12.2.2 Concerns

Subclause 6.6.12.2.2 describes privacy concerns of stakeholders of an IoT system. Not all concerns are relevant or appropriate for every IoT system. The following list of concerns intends to cover a broad range of privacy and trust. There might be systems where, for example, complete transparency is adequate, as well as systems where even the fact that there is some kind of communication or transaction should be obscured.

Table 32—Concerns of the privacy and trust viewpoint

Concern from the common concern list	Elements of the concerns framed by the architecture framework
Privacy Policy Regulatory Relationship between data	Is an IoT system or component only collecting, transmitting, and processing the minimal set off data that is really necessary to fulfill a certain function?
Policy Quality	Does an IoT system provide an overview of data transactions, data access attempts, or data process steps conducted by different stakeholders while operating?
Privacy Regulatory Responsibility	Does an IoT system provide stakeholders with means to agree or disagree whether certain (potentially sensitive) data are collected, transmitted, or processed?
Privacy Regulatory Responsibility	Does an IoT system provide means to revoke granted access rights or given consent?
Privacy Regulatory Responsibility	Does an IoT system provide means to interrupt or stop collecting, transmitting, and processing any data at any time?
Privacy Policy Confidentiality Regulatory Relationship between data	Does an IoT system provide the possibility to aggregate traces over a certain time or certain sources before transmission or processing, and thus to hide detailed information?
Privacy Policy Regulatory Confidentiality Relationship between data	Does an IoT system provide means to transmit or process data without revealing the identity of the receiver or/and the recipient?
Privacy Policy Discoverability Relationship between data	Is an IoT system able to use and manage identifiers other than real names?
Privacy Confidentiality Relationship between data	Does an IoT system provide means to disguise to observing parties the transmission of data?
Privacy Confidentiality Relationship between data	Does an IoT System provide means to disguise that two or more entities or messages belonging together?
Privacy Confidentiality Policy Regulatory Relationship between data	Does an IoT system provide stakeholders with the ability to delete all data belonging to a certain device, user, or service on demand or automatically as soon as they are not needed anymore?
Security Confidentiality Privacy Policy Regulatory Integrity Relationship between data	Does an IoT system disguise information using cryptographic tools for data storage or transmission?
Confidentiality Integrity Availability Policy	Can a user trust in a system to act in an expected way within a certain context?
Confidentiality Integrity Policy	Can an IoT system prove that an identity was authenticated according to a certain level of confidence?
Confidentiality Integrity Availability Policy	Can an IoT system adapt the level of confidence in an identity according to a required level?

6.6.12.2.3 Anti-concerns

The identity and privacy viewpoint is not appropriate to address general security concerns.

6.6.12.3 Privacy design strategy model

The European Network and Information Security Agency (ENISA) introduced eight privacy design strategies (Danezis et al. [B21]). They are data oriented strategies and process oriented strategies. Subclause 6.6.12.3 adapts the more general strategies to make them applicable under the conditions of the IoT.

a) Minimize—is the most basic design strategy and states that only a minimum amount of data should be processed, transmitted, or even collected. An IoT system designer should only take data into account that are absolutely necessary for the intended use cases.

b) Hide—states that sensitive data and their interrelationships never should be shown in plain view. On the one hand this aims at inside attackers but this strategy also focuses at unlinkability and unobservability in order to hide information from observers during transport and processing. This means that, for example, two interacting entities or messages of one communication cannot be related.

   Related design patterns are the use of encrypted transport and the storage of encrypted data.

   Furthermore, attribute-based credentials (ABC4Trust [B4]) allow for authentication by minimally revealed information required by the application, without giving away full identity. Also the use of pseudonyms or anonymity techniques falls into the hide strategy.

c) Separate—states that sensitive data should be processed and stored in a distributed manner so it is harder to build complete profiles over time.

   Data from separate sources should be stored in separate databases that are not linked. Data should be processed locally whenever possible and stored locally if feasible. Database tables should be split when possible. In data tables, identifiers should be removed and exchanged for pseudonyms.

   Remark: ENISA states in its paper (Danezis et al. [B21]): “These days, with an emphasis on centralized web based services this strategy is often disregarded.”

   Separate is contrary to the common idea of a central IoT Platform within one company or group.

d) Aggregate—sensitive data should be aggregated to the state that is highest and still most useful level. Data can be aggregated over time or over groups of devices, gateways, or users.

   Aggregation is supported by design patterns like “aggregation over time” (for example used in energy metering), dynamic location granularity, k-anonymity (Sweeney [B135]), and differential privacy (Dwork [B23]).

e) Inform, control—are process-regarded privacy strategies. They aim at transparency toward the stakeholder. Stakeholders should be informed about what kinds of data are processed and they should be given a technical control point where they can interrupt processing or transmitting of data that potentially harms their privacy.

f) Enforce, demonstrate—are related to privacy policies that should be in place and in compliance with legal regulations. As a further step, an operator of a system should be able to show that a privacy policy is in place and is being enforced. In some legislation this is explicitly required (for example by EU privacy regulation (Proposal for a Regulation [B122]) later superseded by GDPR). Logging and auditing mechanisms and processes are regarded design patterns.

6.6.12.4 Basic privacy terms, concepts, and models

6.6.12.4.1 General

In order to explain basic terms and concepts, two interaction patterns are described: communication and data access. In a communication process, sender A sends a message to recipient B. Sender and recipient might be users, devices, or services. In a data access process, requestor A wants to access a resource of owner B.

6.6.12.4.2 Basic terminology, concepts, and models for privacy in communication scenarios

Subclause 6.6.12.4.2 describes important terms used in privacy design strategies. Descriptions are taken from a terminology paper published by Technical University Dresden and ULD Schleswig-Holstein in Germany (Pfitzmann and Hansen [B123]).

Figure 53 depicts a basic setting of a communication or data transmission. There is a set of senders sending messages through a communication system to a set of recipients.

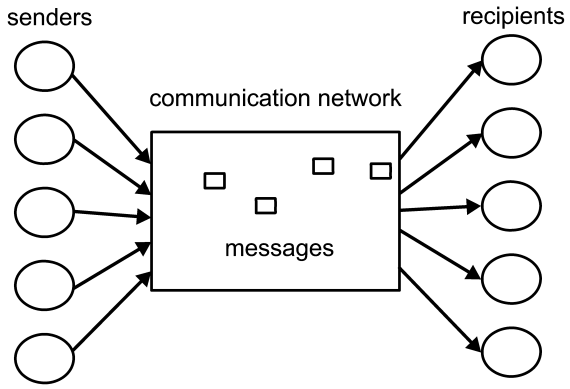


Figure 53—Setting of a communication system

To enable anonymity of a subject, there always has to be an appropriate set of subjects with potentially the same attributes. This leads the first set of definitions:

- "Anonymity of a subject means that the subject is not identifiable within a set of subjects, the so-called anonymity set. The anonymity set is the set of all possible subjects" (Pfitzmann and Hansen [B123]).
- "Pseudonymity is the use of pseudonyms as identifiers. A pseudonym is an identifier of a subject other than one of the subject's real names" (Pfitzmann and Hansen [B123]).
- "Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not" (Pfitzmann and Hansen [B123]).
- "Unobservability of an item of interest (IOI) means undetectability of the IOI against all subjects uninvolved in it and anonymity of the subject(s) involved in the IOI even against the other subject(s) involved in that IOI" (Pfitzmann and Hansen [B123]).

There are technical concepts to achieve privacy-ensured communication including anonymity, pseudonymity, unlinkability, and unobservability. David Chaum introduced the concept of mix cascades [B16] (Figure 54).

A mix cascade is comparable to a cascade of proxy servers. Instead of sending a data packet direct from A to B, it is sent via a number of intermediaries. Classic examples are so-called "anonymity proxy services." This kind of service enables anonymous web surfing. At every hop in the mix cascade the information is decoded and newly encoded, and along with the new IP address of the sender, transmitted to a next mix. A visited web site cannot see from what source IP address the request comes because it was changed several times along the way. The visitor remains anonymous. The visitor does not see the actual IP address of the web server. The visitor just knows the IP address of the first proxy. This way, the server also remains anonymous.

Another way of implementing mix cascades are so-called "onion ring" networks (e.g., Tor project [B138]). Here every user has its own mix proxy where its own traffic and traffic of other users is handled.

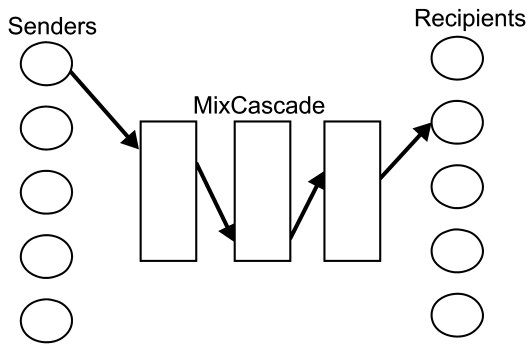


Figure 54—Mix cascades for implementing anonymous communication

The cascade is only secure if first and last proxies in the cascade are not under control of an attacker. It is common to place the controlled proxies of a cascade in the same organizations, countries, or legislations.

Other potential improvements: Even when the address is changed, an attacker could derive knowledge from observing incoming and outgoing traffic at one or more cascades. An observer could see at what time a package is received by a proxy and when a new one is sent out immediately afterwards. That is why mixes might wait a random time before they forward a data package. An additional way is to add dummy traffic. This way not only a connection between a sender and receiver could be disguised, but also the fact that there is any kind of intended communication might be covered.

6.6.12.4.3 Basic terminology, concepts, and models for privacy in resource access scenarios

In more sophisticated service scenarios, users or services need access to resources located in a cloud or at a device. Data or resource access scenarios imply various trust and privacy issues.

When a user device or service A wants to access the resources owned by user B, it needs an authorization in combination with a set of entitlements.

Figure 55 describes the basic setup of a resource access scenario. A requesting party wants to gain access to a resource. This resource is protected by a control point. This control point is managed by the resource owner.

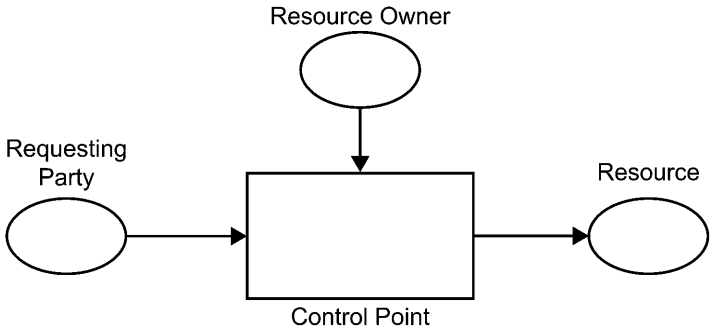


Figure 55—Setting of the resource access system

At the control point different privacy-relevant processes can be conducted. The resource owner is able to impose policies under what conditions and prerequisites a resource access is allowed. A resource might be a resource server in the cloud, or a device like a sensor or actuator.

Data minimization—Smart software algorithms might allow or block access to resources according to access policies.

Transparency—The control point is an instance where all access attempts might be recorded. At the control point, metadata like identity or the requesting party, time, requested resource, etc. can be tracked. It is possible to provide a transparent view on who had access to what kind of data.

CAUTION

While on the one hand this setup leads to more transparency, it might harm the privacy of the requesting party depending on detailed usage scenarios.

Consent—The control point is able to trigger and impose user consent. This consent can be requested ad hoc if the user is online or it can be stored. So a requesting party can get access to a resource with the consent of the owner even when the owner is offline.

Revocation—The control point is also an instance to manage access rights. While access rights can be granted, they might be revoked because of an event or a changed policy or through the resource owner.

Intervene—A control point can be used to stop any kind of access to a resource at any time.

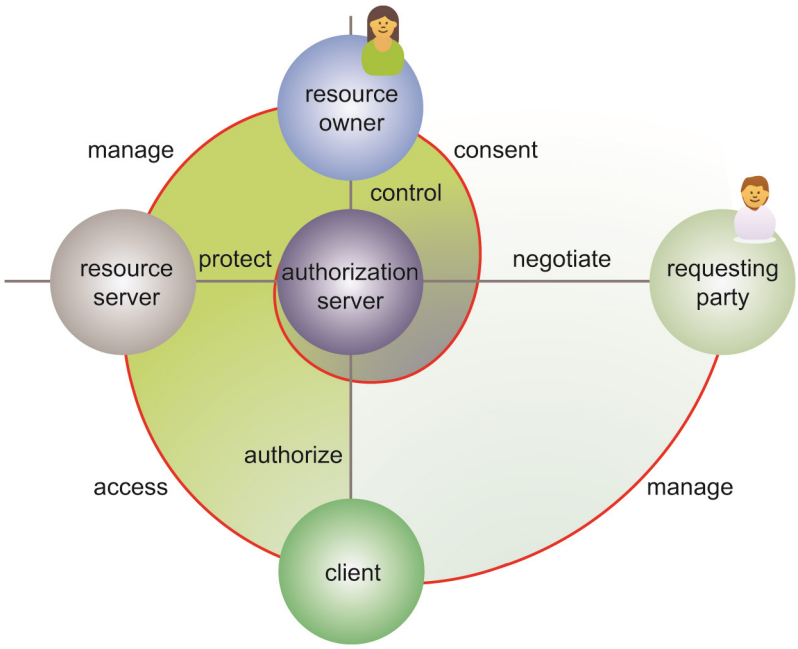
Aggregation—The control point is a designated place to implement privacy mechanisms that ensure aggregation over time. It is possible to allow subsequent access attempts only after a certain time period. Smart software algorithms can be imposed to ensure only summarized data.

The control point might be also cascaded or it is possible to use one control point for different sources. This way an aggregation over different sources can also be implemented.

A control point can be co-located with an authorization server. User-managed access (UMA) [B141] is an example of a standard that implements both access control to resources and advanced privacy mechanisms.

UMA is a profile of the OAuth 2.0 authorization protocol. UMA's architectural approach is to put the authorization server under control of the resource owner (Figure 56). In the UMA flow, an application requests a so-called "protected resource." The requesting application has to present a requesting party token. This can be obtained from an authorization server. The requesting party has to provide to the authorization server any identity claims needed in order to associate sufficient authorization data with the requesting party. The identity claims might be, for example, certificates. An online presence of a user is not necessary.

The authorization server could be equipped with a sophisticated policy engine where the owner or administrator of a resource can define what other applications may have access to certain resources and what kind of identity claims they have to provide.



Source: User-Managed Access (UMA) Profile of OAuth 2.0 draft http://tools.ietf.org/html/draft-hardjono-oauth-umacore-07

Figure 56—Mix UMA architectural approach—authorization server as control point

The UMA model has the following roles:

Resource owner—This role manages online resources on the resource server.

Resource server—This role hosts the online resources of a resource owner. It is managed by the resource owner and protected by the authorization server. It can be accessed by the client of the requester in order provide the requested resources.

Authorization server—This role is the central control point in this model. It is controlled by the resource owner and protects the resources by authorizing clients of a requesting party to access the resources. The authorization depends on the consent of the resource owner and possible claims the requesting party has to provide in a negotiation step.

Client—This role is the software that is used by the requesting party to access online resources of the resource owner.

Requesting party—This role is the user, device, or service that wants to access a certain resource of the resource owner.

The resource owner controlled authorization server (AS) can also be implemented as a fundamental privacy control point. It is an element where the following privacy concerns of stakeholders can be implemented:

Authorization—The AS protects sensitive resources or data of a system and authorizes the access to it.

Revocation—The AS is able to revoke an authorization.

Access policies—Fine-grained policies can be implemented at the AS in order to allow different stakeholders access to different data.

Transparency—At the AS, actual access and access attempts can be logged.

Consent—At the AS, user consent can be implemented and stored for auditing reasons or for later use.

Intervene—At the AS, a user can interrupt access, processing, and transmission of sensitive data.

6.6.12.5 Trust models

6.6.12.5.1 General

There are many different views and models regarding trust. Subclause 6.6.12.5 provides examples of models to establish trust between users, devices, and services of an IoT environment. Although trust is a broader concept, most models emphasis authentication since it is the precondition for other trust characteristics.

6.6.12.5.2 Direct trust model

Direct trust exists when users, instances, or devices share, for example, the same infrastructure, security domain, processes, etc. and/or have contractual agreements, common operational rules, and technical standards.

When two users or devices are known to each other they can share technically a direct trust relationship. Direct trust in the authenticity of an identity means one party A validates the credentials of another party B (e.g., the public key) without any third party (Figure 57). Direct trust is typical in situations where instances or devices belonging to one system or organization when all components are known and under control of one stakeholder (user, administrator, etc.).

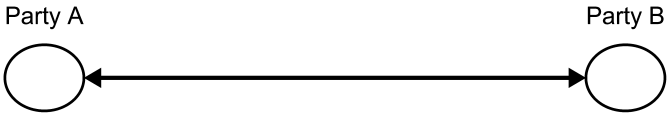


Figure 57—Direct trust between party A and party B

6.6.12.5.3 Indirect trust model

6.6.12.5.3.1 General

Indirect trust exists when users, instances, or devices have a trust relationship with a trusted third-party instance. This trusted third party then has relationships to other users or devices. This way both party A and party B can establish a trust relationship (Figure 58). The trusted third party is a trust anchor shared by both parties. The authenticity of an identity can be proven by using the key of the trusted third party. An interesting characteristic is that trust is transitive and can be transferred or delegated.

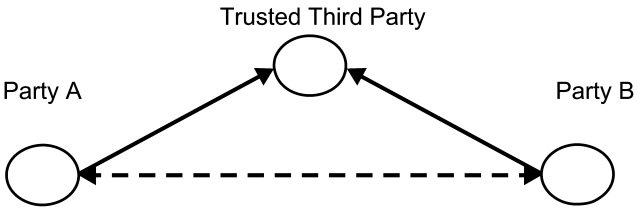


Figure 58—Indirect party A trusts party B through a trusted third party

6.6.12.5.3.2 Hierarchical trust models

A hierarchical trust model can be compared with a tree containing a root instance (Figure 59). This root is the source for all trust relationships as a trust anchor. The root can delegate trust to child instances which can also delegate trust. The hierarchical trust model uses transitive trust and delegates it from the trust anchor all the way down from node to node. A classic implementation of the hierarchical trust model is the PKI X.509 infrastructure. The ITU X.509 standard [B68] describes a public key infrastructure for the creation of digital certificates. At the top, there are companies or organizations that have the root certifying authority (CA). This root CA can sign certificates of users, servers, or even other CAs. This way a complete hierarchical trust tree can be designed. Two users can trust each other if they check whether the certificate is signed by a trusted CA.

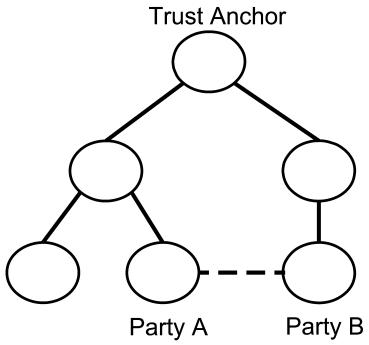


Figure 59—Hierarchical trust

6.6.12.5.3.3 Web of trust

Web of trust is a distributed trust model (Figure 60). The basic idea is that a so-called “introducer” C introduces user B to another user A. When both user B and user A have a trust relationship with C they can now trust each other. Sometime this concept is also referred to as a “friend-of-a-friend” relationship. In the web of trust, every instance can be an introducer or a so-called “trust anchor” for a trust relationship. Technically in the web of trust an introducer C signs the key of party A (C trusts A). Party B and introducer C have established a trust relationship before and they know each other (B trusts C). Now party B can use the key of introducer C to validate the credentials of A (A can trust B through C).

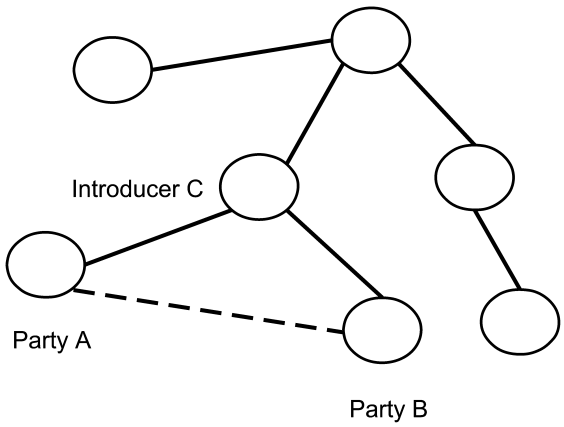


Figure 60—Web of trust

In "Trust In Cyberspace" [B133] authentication is defined with an additional aspect of authentication levels as "the process of confirming a system entity's asserted identity with a specified, or understood, level of confidence." This definition introduces the concept of *level of confidence* which applies to every authentication process. In fact, there is no 100 percent secure way for a system or a user to be authenticated and to be safe against any kind of identity theft attack. It is only possible to apply different kinds of authentication methods that have been declared as sufficient to pass the authentication process. There are many different authentication methods. On the classic web they are from three different categories, "Something that you know," "Something that you have," and "Something that you are." The first category is mostly interpreted as a shared secret, the second one is a token or certificate, and the last category refers to biometric authentication methods.

The level of confidence in the authentication process of an identity can be strengthened by combining different authentication methods or taking additional factors or context information into account. Additional information, for example, could be taken from the network layer, from geographical information, or from other use case specific factors.

6.6.12.5.4 Trust elevation

Trust elevation is the process of increasing the level of confidence in the authentication of an identity. There may be situations where a certain level of authentication is sufficient (e.g., username/password is sufficient for checking a bank account). But for certain transactions it needs a higher level of authentication. In this case an additional authentication method is applied in order to increase the *level of confidence* (e.g., for an online money transfer a bank might request username/password and an additional one-time password taken from an out-of-band channel like SMS).

6.6.12.5.5 Trust frameworks

Trust frameworks (e.g., Kantara Initiative Identity Assurance Framework [B70] and Open Identity Exchange OIX [B118]) are operational approaches to establish trust between different participants of an identity ecosystem (Figure 61).

A trust framework is a common set of proceedings, protocols, and agreements. Policymakers can choose a trust framework provider like, for example, OIX or Kantara Initiative, or they can write their own set of rules. An assessor checks an identity service provider according to given framework requirements. A trust framework is more or less a certificate that an organization handles its identity services according to certain rules. Service provider—also called *relying parties*—and their users can trust certified identity service

providers. The principle of trust frameworks is very scalable and works for small organizations or worldwide ecosystems.

Figure 61—Entities of a trust framework

6.6.13 Collaboration viewpoint

6.6.13.1 General information and key features

This viewpoint addresses the collaboration of systems that belong to different application domains. Interactions across system borders may range from a simple transmission of data to one requiring very sophisticated adjustment, enablement, and discovery of resources shared by several systems like devices, calculation resources, energy, and so on.

This viewpoint addresses system collaboration based on symbiotic ecosystems (IEC White Paper 2015 [B52]). *Symbiotic* is a biological term that describes multiple types of organisms living together in a mutually reciprocal relationship, in which the organisms do not harm each other, but rather live close together while providing each other with various benefits (IEC White Paper 2015 [B52]).

6.6.13.2 Stakeholders and concerns

6.6.13.2.1 General

Figure 62 shows the classification of the collaboration of systems. The collaboration of systems is classified as the direct collaboration model (Figure 62 [part a]) and the indirect collaboration model (Figure 62 [part b]).

In the direct collaboration model, systems exchange their own information directly with each other like pure peer-to-peer applications. The advantages are the sustainability and flexibility of the Collaborative Platform. The disadvantages are the difficulty in the security and resource management and the high demand on the processing performance in each system.

In the indirect collaboration model, systems exchange their own information via the Collaborative Platform. The advantages are that it is easy to realize the security and resource management by central control and the implementation of each system is simple. The disadvantage is the high demand for the reliability and scalability in the Collaborative Platform.

Typical stakeholders are systems and the Collaborative Platform.

— System #1 … #*n*: Autonomous system. Stakeholders belonging to application domain #1 … #*n* ($n \geq 2$);

— Collaborative Platform: Collaboration stakeholder, for instance a provider of Collaborative Platforms. This stakeholder exists in the indirect collaboration model (Figure 62 [part b]). Systems and the Collaborative Platform are connected by a collaboration interface.

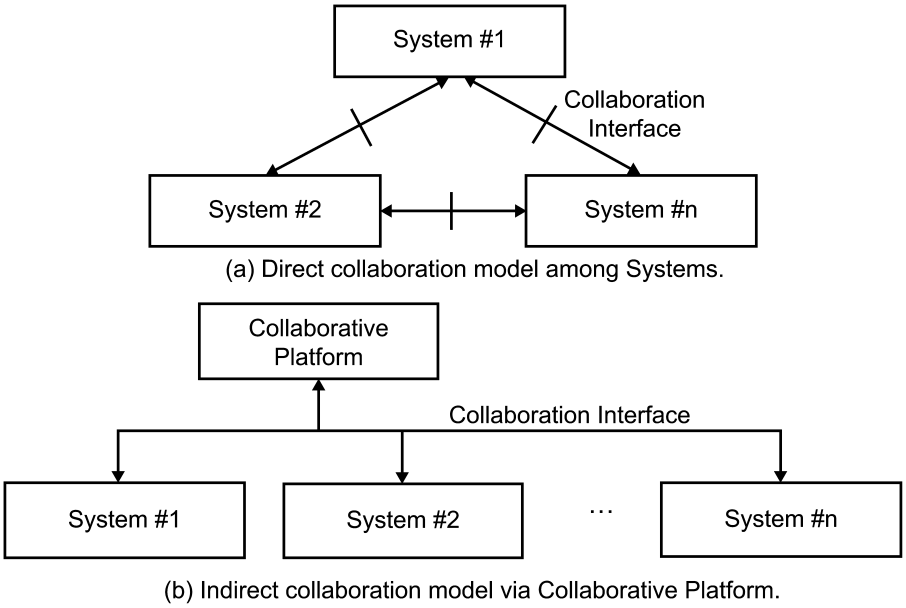


Figure 62—Classification of collaboration viewpoint

6.6.13.2.2 Concerns

Table 33 shows concerns of the collaboration viewpoint. Concerns are selected from IoT concerns in Table 2.

Table 33—Concerns of collaboration viewpoint

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Cost	How can each system reduce cost by sharing knowledge and resources between systems?
Utility	How can each system enhance operation efficiency by sharing knowledge and resources between systems?
Optimization	How can systems have a mechanism and an algorithm for sharing resources optimally—like devices, calculation resources, energy, and human resources—among several systems?
Privacy	How can each system protect its local information from other systems?
Security	How can each system guard against improper modification of system, and include ensuring non-repudiation and authenticity?
Data Semantics	How can systems share the meaning of each system's data?
Relevance	Can systems recognize the meaning of data collected from each system and interpret one system's semantics to other system's semantics?
Responsibility	Can systems and the Collaborative Platform identify systems to control and manage the operation of each system?
Identity	Can systems and the Collaborative Platform recognize systems and data?
Networkability	Can systems and the Collaborative Platform connect to other systems easily and reliably?
Collaboration	Can systems and the Collaborative Platform have a mechanism or a common interface for data exchange among several systems?
Autonomy	Can systems connect to other systems even after entering each domain operation into service?
Evolvability	How can an autonomous system evolve or be functional with newer technology to collaborate with other autonomous system?

6.6.13.3 Model kind: Collaboration

6.6.13.3.1 Conventions

6.6.13.3.1.1 General

Figure 63 shows collaboration processes in the direct collaboration model and in the indirect collaboration model. Function blocks and interfaces in Figure 63 are classified based on four levels: network level, device monitoring level, data level, and application level. The collaboration viewpoint considers various kinds of information like operational logs and sensor data, business data, and progress data. The performance for system collaboration depends on the requirements of the IoT system.

Figure 63 part (a) shows the direct collaboration model. Each system collects the resource data from devices in its own system (Step 1). Then, each system stores and converts the collected data for sharing between systems, sends the collected data to other systems, and confirms and receives data sent from other systems (Step 2). Each system inputs the collected data into an analysis application, and the analysis application analyzes the data and creates a plan to improve productivity or performance in each system (Step 3). For example, in the Smart Grid domain, the analysis application analyzes the information of energy demand and supply and creates a new energy management plan for reducing the energy loss and ensuring energy stability. Also, each system inputs the data received from other systems into an analysis application (Step 4). Each system analyzes data, and creates the collaboration plan (Step 5). Then, each system controls the devices based on the received feedback (Step 6). Each system shares the feedback information (Step 7).

Figure 63 (b) shows the process in the indirect collaboration model. Each system collects the resource data from devices in its own system (Step 1). Then, each system stores and converts the collected data for sharing between systems, and sends the collected data to the Collaborative Platform (Step 2). The

Collaborative Platform confirms the validation of the connecting system (Step 3) and the received data, analyzes the data, and creates the collaboration plan (Step 4). Then, the Collaborative Platform sends feedback to each system (Step 5), and each system controls the devices based on the received feedback (Step 6).

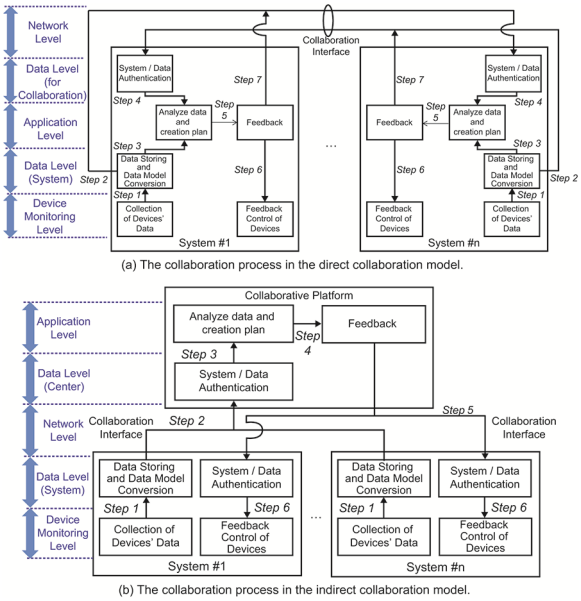


Figure 63—Collaboration process

Table 34 shows the relationship between each process/interface and a concern to be solved.

Table 34—The relationship between each process/interface and each concern

Level	Process/interface	Role of process/interface	Concern to be solved
Network level	Collaboration interface	To provide a common interface to connect to other systems or the Collaborative Platform	Networkability Collaboration
		To provide a common interface for creating a collaboration system even after entering each system operation into service	Autonomy Evolvability
Data level	Data storing and data Modeling	To store devices' data and feedback information To form devices' data for sharing among systems	Data semantics Relevance
		To protect private data by data abstraction	Privacy Security confidentiality
	System/data Authentication	To identify system and data	Responsibility Identity
Application level	Analyze data and create plan	To create a plan to optimize cost, resource utilization by data analysis	Cost Utility Optimization
	Feedback	To send feedback information to optimize cost, resource utilization	Cost Utility Optimization

6.6.13.3.2 Collaboration operations

As outlined in 6.6.13.3.1.1, the collaboration processes can be organized in sequences.

6.6.13.3.3 Collaboration correspondence rules

Entire systems as well as aspects of the system can be modeled by use of the collaboration of systems. An example of an aspect is subsystems. In the event that all aspects of a system are not modeled with the same sequence of the collaboration process, potential conflicts (e.g., support of a subsystem, while the entire system operates) are to be identified during the concept phase and to be alleviated.

6.6.13.4 Correspondence rules

Table 35 describes the collaboration viewpoint correspondence rules.

Table 35—Collaboration viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
COL-CR-1	Conceptual	According to the requirements of the IoT system, the direct collaboration model or the indirect collaboration model can be selected.
COL-CR-2	Adequate design for required security	The design of security policy depends on the cooperativity view in the adequate design for required security viewpoint.
COL-CR-3	Other related viewpoints	The detailed functions and specifications of each function shown in Figure 63 are designed based on other viewpoints and the requirements of the IoT system.

6.6.14 Computing resource viewpoint

6.6.14.1 General information and key features

Examples of computing resources are:

- Microcontrollers embedded in sensors and actuators
- Controllers like PLCs, DCS, etc.
- Data centers
- Gateways
- Others

Computing resources are connected through communication networks.

Since IoT components can have a many-to-many relationship, it is possible to connect any component to another component.

The IoT application needs computing resources according to the requirements of the use case. Choosing the computing resource is based both on the capabilities of the computing resource and the capabilities of the network between the computing resource and the application needs.

Dynamic assignment of resources and sharing of resources should be considered.

6.6.14.2 Stakeholders and concerns

6.6.14.2.1 Typical stakeholders

Stakeholders to consider include:

- Builders
- Communicators
- Developers
- Maintainers
- Operators
- Owners
- Production engineers
- Suppliers
- System administrators
- Users

6.6.14.2.2 Concerns

Table 36 shows concerns of the computing resource viewpoint.

Table 36—Concerns of computing resource viewpoint

Concern from common concerns list	Elements of the concerns framed by the architecture framework
Autonomy	How can functionality be optimized under independent operating conditions in a changing environment?
Collaboration	Can of two or more systems form an ecosystem that guarantees value delivery?
Communication	How can information be exchanged between peers located in one system or in different systems?
Complexity	How can we understanding the behavior of the system due to the richness and heterogeneity of interactions among its components, such as existence of legacy components and the variety of interfaces?
Cost	How can cost be reduced (especially the communication costs over public wide area networks)?
Data semantics	Can agreed and shared meaning(s) of data held within, generated by, and transiting a system be achieved?
Data velocity	Can the speed with which data operations are executed meet requirements?
Data volume	Can the volume or quantity associated with a CPS's operation meet requirements?
Evolvability	Can the system evolve and be functional with newer technology (backward compatibility)?
Functionality	Can the function that a CPS provides meet requirements?
Identity	Can entities (people, machines, and data) be accurately recognized when interacting with or being leveraged by a CPS?
Networkability	Can the system easily and reliably be incorporated within a network of other systems?
Operations on data	Which abilities are needed to create, read, update, process, and delete the data?
Optimization	Can systems or resources perform optimally?
Performance	How can required operational targets be met?
Privacy	How can unauthorized entities be prevented from gaining access to the data?
Relationship between data	How and why can sets of data be associated with each other and what is the value or harm that can be derived from those associations?
Reliability	How does the system deliver stable and predictable performance in expected conditions?
Resilience	How does the system withstand instability and unexpected conditions?
Security	Can confidentiality, integrity, and availability be achieved?
Standardization	Can standards including credibility and trust, autonomy, east-west communication, machine learning algorithms, components functionality definition, semantic interoperability, fault detection, embedded system containerization, and carrier mode selection be met?
Time interval and latency	How can requirements for time intervals between pairs of events be met?

6.6.14.3 Computing resource model kinds

6.6.14.3.1 General

All three of the following models can be combined in use. Components within a centralized or decentralized system are typically generic upon their use.

6.6.14.3.2 Computing components type model

Computing components have computing resources while non-computing components do not have computing resources. Four types of components are listed:

— Type A: Highly centralized computing components, for instance, data centers and cloud servers

— Type B: Computing components (other than Type A and Type C) connected to the network, for instance, gateways, PLCs, edge-cloud servers, and PCs

— Type C: Computing resources embedded in sensors and actuators

— Type D: Non-computing resources

6.6.14.3.3 Centralized computing resources model

Figure 64 shows the centralized computing resources model in which a central component connects other components.

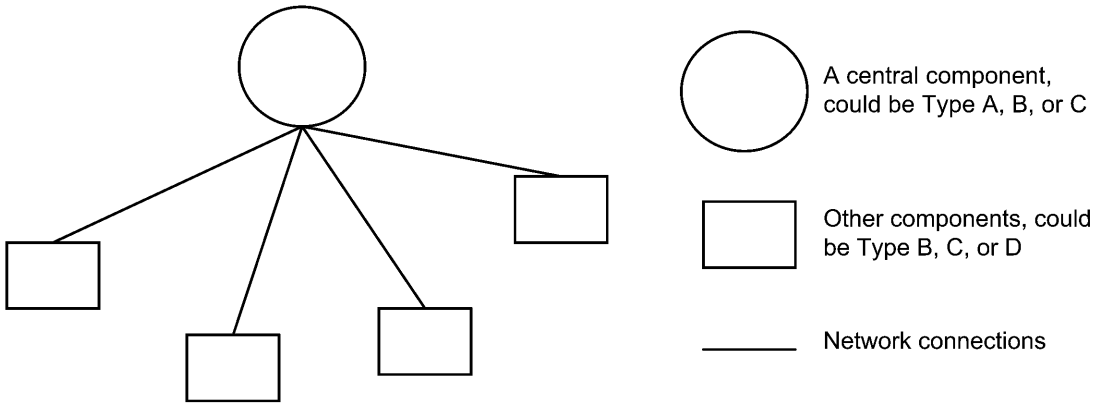


Figure 64—Centralized computing resources model

In this model, when the central component is a Type A component, other components could be Type B, C, or D. When the central component is a Type B component, other components could be Type C or D. When the central component is a Type C component, other components should be Type D. Network connections between components are not required to be continuously activated.

6.6.14.3.4 Distributed computing resources model

Figure 65 shows the distributed computing resources model in which distributed components have connections.

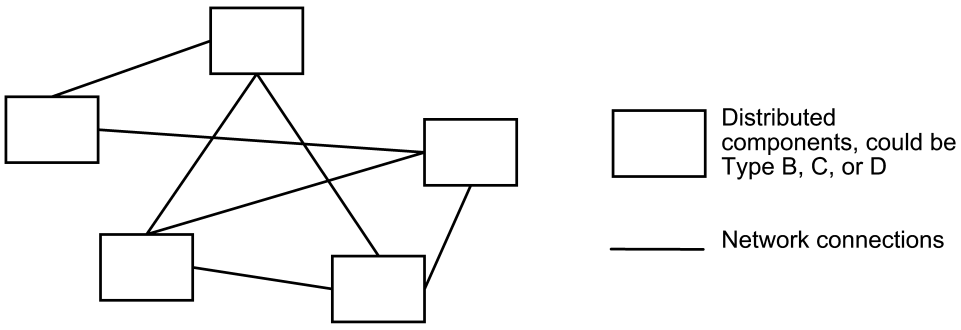


Figure 65—Distributed computing resources model

In this model, the components should not all be a Type D component. A definite every-to-every connection is not required. Network connections between components are not required to be continuously activated.

Distributed computing resources shall provide status information that the component is in use by whom or be available to get computing tasks assigned.

6.6.14.4 Operations on views

The centralized computing resources model and the distributed computing resources model can be combined to fit various computing resource conditions. Figure 66 shows an example of combining these two computing resources models.

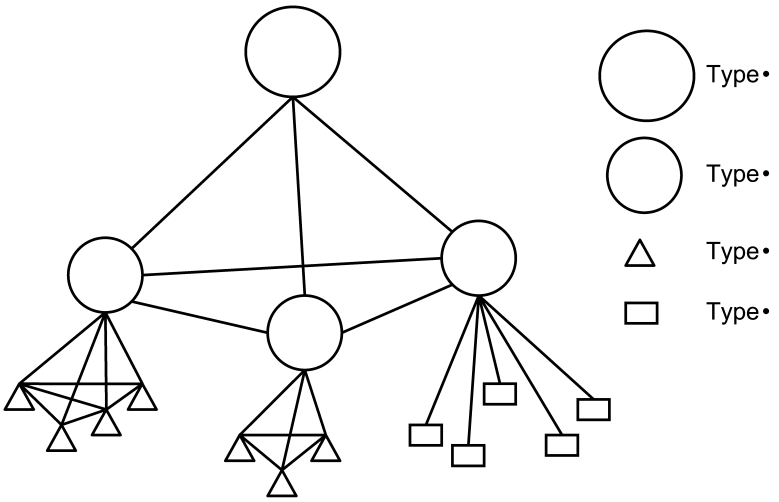


Figure 66—Combination of the two models

Dynamic assignment of resources and sharing of resources should be considered based on both the capabilities of the computing resource and the capabilities of the network between the computing resource and the application needs. For example, when a component α has a sudden rise of the computing resource requirement that exceeds its current assigned computing resource capability, the computing resources could be dynamically reassigned. By sharing the computing resources of components connecting to component α, the computing resource requirement of component α could be satisfied again.

6.6.14.5 Correspondence rules

Table 37 lists the computing resources viewpoint correspondence rules.

Table 37—Collaboration viewpoint correspondence rules

Correspondence rule	Viewpoint	Pertinent conceptual viewpoint elements
CPR-CR-1	Communication	Components in the computing resource viewpoint that communicate and interoperate with each other according to the communication viewpoint.
CPR-CR-2	Functional	The computing resource viewpoint depends on functional viewpoints.

6.6.14.6 Examples

The edge computing architecture is an example of using the computing resource viewpoint, see 7.4. Batch processing is an example of using the computing resource viewpoint, see 7.7.

7. Architecture examples

7.1 General

The architecture of a system constitutes what is essential about that system considered in relation to its environment. Architecture descriptions are used to express architectures for systems of interest. The architecture and the architecture descriptions are provided in the example architecture of a system.

The architecture and the architecture descriptions are created based on viewpoints listed in the viewpoint catalogue in 6.6. The architects select appropriate viewpoints to solve concerns for a target system, and they do not have to use all viewpoints.

7.2 Example architecture of system A

7.2.1 General

Stakeholders of a system have concerns with respect to the system-of-interest considered in relation to its environment. A concern could be held by one or more stakeholders. Concerns arise throughout the lifecycle from system needs and requirements, from design choices, and from implementation and operating considerations. A concern could be manifest in many forms, such as in relation to one or more stakeholder needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks, or other issues pertaining to the system.

Framed concerns versus common concerns are specific for the system of interest.

7.2.2 Framed concerns and their stakeholders

There are two aspects to a viewpoint: the concerns it frames for stakeholders and the conventions it establishes on views, see IEEE/ISO/IEC 42010:2011, 4.2.4.

7.2.3 Architecture view

IEEE/ISO/IEC 42010:2011, 5.5 requires:

- An architecture description shall include exactly one architecture view for each architecture viewpoint used.
- Each architecture view shall adhere to the conventions of its governing architecture viewpoint.

7.3 Example architecture of adequate design for required security

7.3.1 General

IoT systems are operated and maintained over a long time period. The security measures of IoT systems will have to continually deal with environmental changes adequately throughout the lifecycle of the system.

This adequate design for required security architecture example is intended to be used by IoT system designers and maintainers to design or maintain/implement adequate security measures of a target IoT system through a system lifecycle.

This example based on a hardening—adaptivity, responsivity, cooperativity (H-ARC) help them to design/maintain a target system adequately and easily.

7.3.2 Framed concerns and their stakeholders

7.3.2.1 Framed concerns

Table 38 shows main framed concerns of adequate design for required security.

Table 38—Concerns of adequate design for required security

Framed concerns list	Elements of the concerns framed by the architecture framework
Measurability	In the initial system design phase for security of the target system: Can we take adequate security measures for the target system? Can we recognize the sufficiency of the result of the security measures?
Evolvability	In the operation phase for security of the target system: Can we continue to take adequate and timely security measures against environmental changes (technologies, threats, system configurations, operations, etc.) after the initial system design?

7.3.2.2 Stakeholders

Typical stakeholders are designers of a system and maintainers of a system.

— Designers of a system: Formulate adequate security policies for the target system and design it in the system planning/system design phase.

— Maintainers of a system: Take adequate security measures through a system lifecycle.

7.3.3 Architecture view

7.3.3.1 General

This adequate design for required security (ADRS) architecture example is intended to be used by IoT system designers and maintainers to design or maintain/implement adequate security measures of a target IoT system through a system lifecycle.

This example based on an ADRS viewpoints help them to design/maintain a target IoT system adequately and easily.

Figure 67 shows the relation of ADRS examples between typical IoT architectures and ADRS architecture.

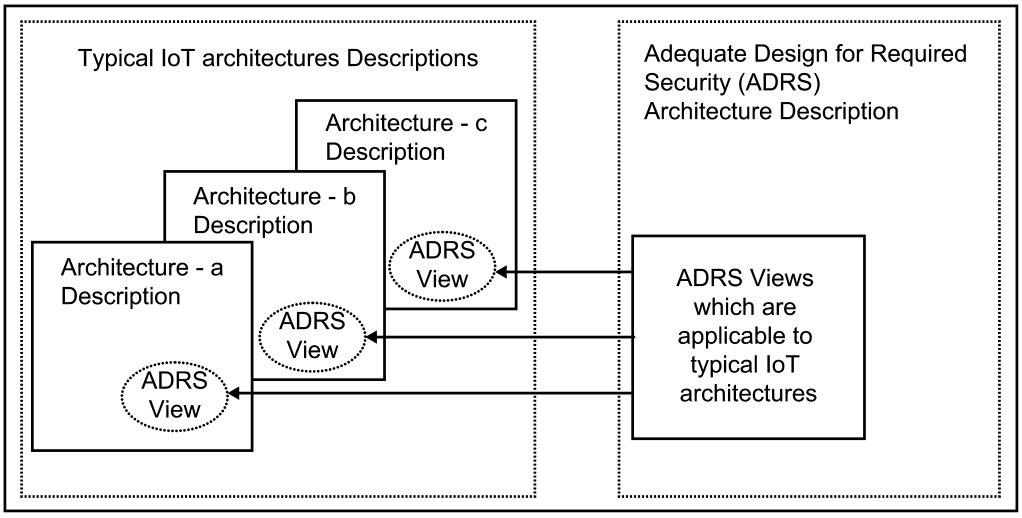


Figure 67—Relation of ADRS views between typical IoT architectures and an ADRS architecture

The architecture example of ADRS is applicable to typical IoT systems. Typical IoT systems are necessary to continue to keep sufficient security strength through their system lifecycle. From this point of view, it is desirable to include an ADRS example within each of typical IoT architecture.

Figure 68 shows an adequate design for required security architecture example. This architecture example consists of three major layers, which are an H-ARC perspective layer, an H-ARC operation layer, and a function layer.

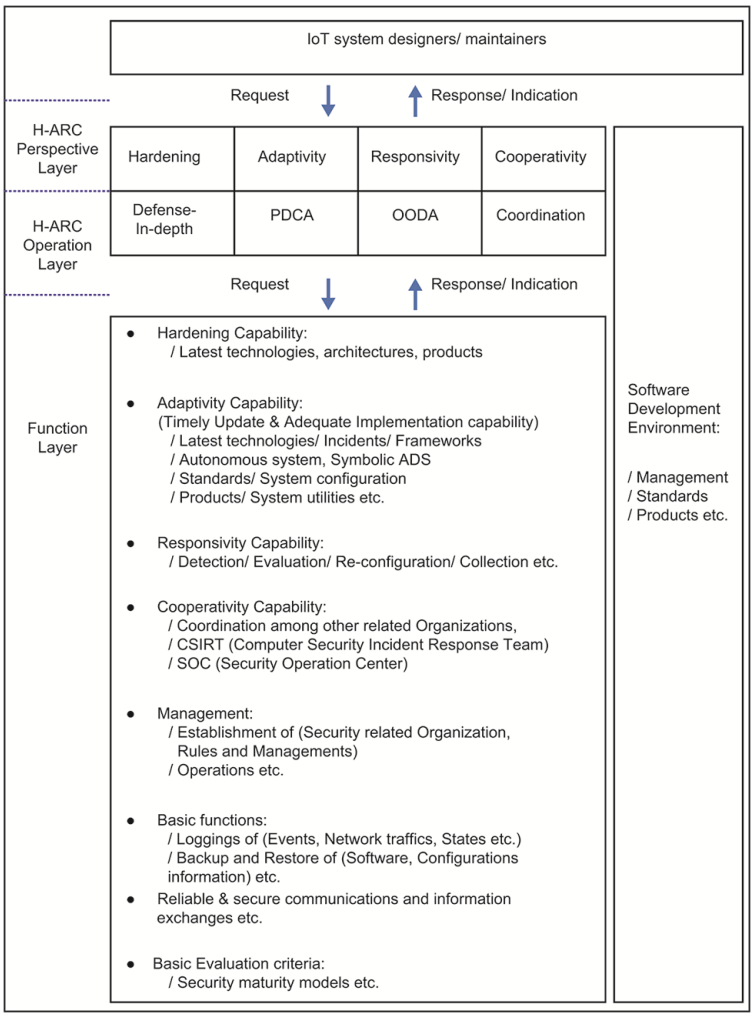


Figure 68—Adequate design for required security architecture example

IoT system designers and maintainers are always necessary to make full use of this example through a system lifecycle. The ideas of an H-ARC perspective layer and the actual procedures of an H-ARC operation layer will be useful for them to get adequate security measures.

It is important to take security measures by using the H-ARC perspective and the corresponding process of the H-ARC operation layer, when planning the system, designing the system, when changes occur in the environment surrounding the system, and during periodic reviews of the system.

Here, items of a function layer are typical examples. Request/response/indication are logical procedures to explain the idea in Figure 68.

7.3.3.2 Considerations for H-ARC operation layer

Subclause 7.3.3.2 provides some important information (Nakano et al. [B80]) to help IoT system designers and maintainers understand H-ARC perspective and H-ARC operations.

a) Achieving adaptivity and the PDCA process (see Figure 69)

The plan, do, check, act (PDCA) cycle plays an important part in adapting quickly to changes in threats technologies.

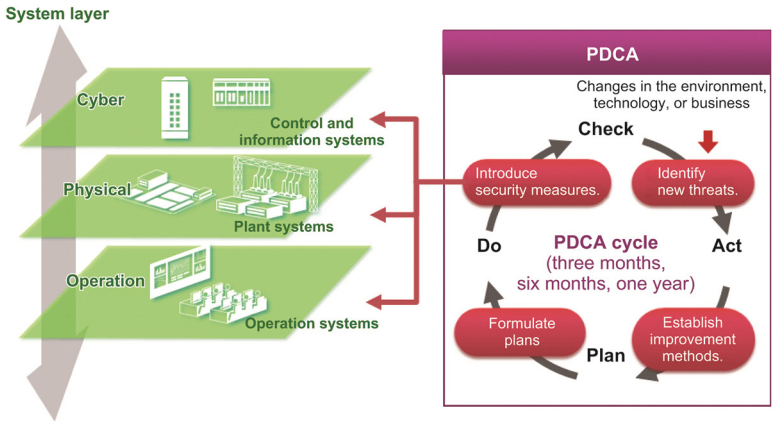


Figure 69—Achieving adaptivity and the PDCA process

b) Achieving responsivity and the OODA process (see Figure 70)

The observe, orient, decide, act (OODA) process plays an important part in responsivity to identify the signs of threat quickly and to take action against it. This process minimizes the impact of security threats on a system.

c) Achieving cooperativity and the coordination process (see Figure 71)

This process is important to establish mechanisms for sharing timely information about security threats among related systems or organizations.

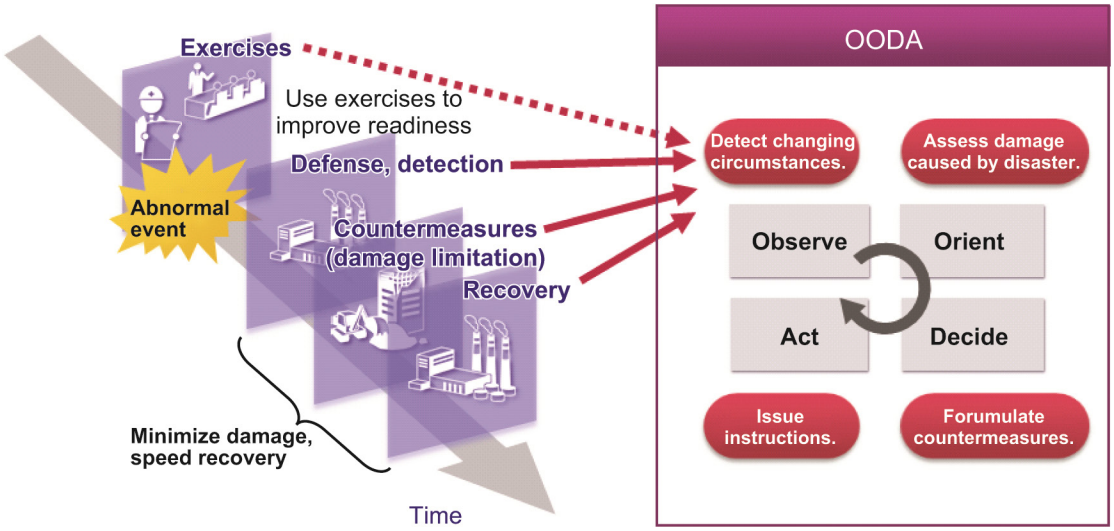


Figure 70—Achieving responsivity and the OODA process

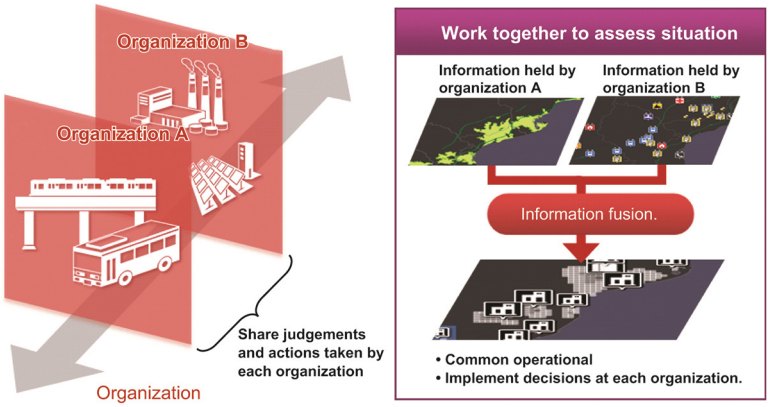


Figure 71 —Achieving cooperativity and the coordination process

7.4 Example architecture of edge computing

7.4.1 General

This example architecture of edge computing is intended to be used by IoT system designers and operators to design or operate a target IoT system.

This example architecture of edge computing is based on the conceptual viewpoint, function viewpoint, and computing resource viewpoint to help them to design or operate a target system adequately and easily.

Computing is fundamental to IoT and many IoT components have computing resources. Since IoT components can be connected with a many-to-many relationship by networking, it could be beneficial to dynamically assign the computing resource based on both the use case requirements and the capabilities of the network between the computing resources. As many IoT systems today are not cloud centric, edge computing is extending data processing to the edge of a network in addition to computing in a cloud or a central data center. Cloud computing can focus on non-real-time and long-period Big Data analytics, and supports periodic maintenance and service decision making. In contrast, edge computing emphasizes real-time and short-period data analysis, and supports real-time smart processing and execution of local services. Edge computing focuses on seamless, effortless, and high-quality experiences between the edge and the cloud, resulting in computation at the right place, right time, on the right form factor to drive the right outcome.

Edge computing is performed on an open platform at the network edge near things or data sources, integrating network, computing, storage, and application core capabilities and providing edge intelligent services.

As the example in Figure 72 shows, the centric cloud connects edge computing nodes (ECNs). An ECN could have connections with other ECNs, and ECNs connect the things, e.g., meters, actuators, cars, and sensors. This example shows a combination of the centralized computing resources model and the distributed computing resources model as defined in the computing resource viewpoint.

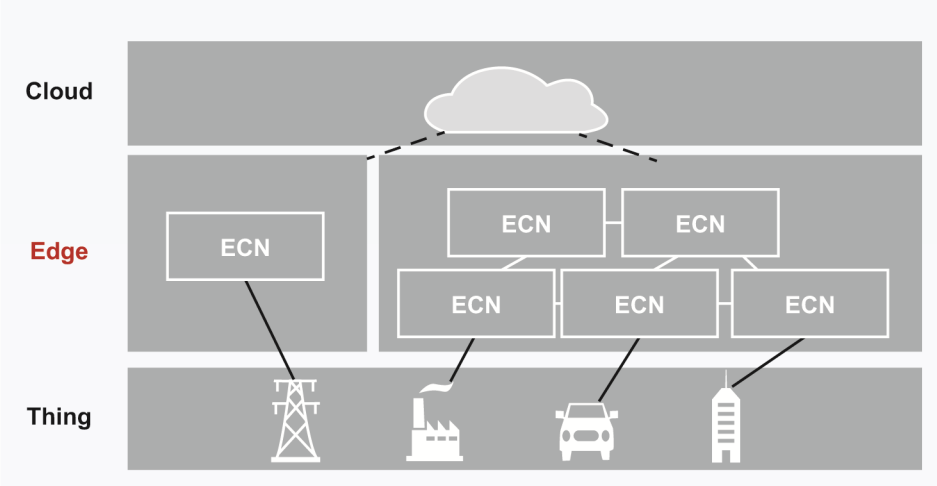


Figure 72—An edge computing example

By introducing intelligence at the edge computing nodes, systems can make decisions:

a) More quickly, by bring data processing and decision-making closer to the data source.

b) To lower communication cost and by reducing communication over public-wide area network, using caching or local algorithms to pre-process the data so that decisions or alarms only can be forwarded to the cloud servers, rather than raw data. This feature can support autonomous operation as well, especially in intermittent connectivity conditions.

c) According to local identity management and access control policies, specific to the running applications, securing the data close to its source, following the local regulations.

d) To load-balance the user, application, or network requests based on changes in the edge or core infrastructure, adapting to temporary failures or maintenance procedures.

e) Based on the alarms or pre-processed information exchange between the edge devices, that is, east-west communication between two peers on the edge.

7.4.2 Framed concerns and their stakeholders

The compromise between required data volume and available bandwidth, the need for intermittent connectivity, and the requirement for immediate responses are three main framed concerns that edge computing may address.

— Data volume versus available bandwidth: Devices and sensors can produce more data than is economically feasible to transmit to the cloud. IoT solutions are often cost sensitive, and communication costs specifically represent a significant portion of ongoing expenses.

— Immediate response: Decisions based on sensor data often shall be made in real-time if there is no time for a roundtrip to the core. Network latency likely would create severe safety issues.

— Intermittent connectivity: When devices and sensors are in locations with only intermittent connectivity, they need local data processing and decision-making in order to keep operating.

Table 39 shows the framed concerns of the edge computing (without prioritization or weighting).

Table 39—Concerns of edge computing

Concerns list	Elements of the concerns framed by the architecture framework
Autonomy	How to optimal functionality under independent operating conditions in a changing environment?
Communication	How do information exchange between peers that locate in one system or in different systems?
Cost	How to reduce cost (especially the communication costs)?
Networkability	Concerns related to the ease and reliability with which the edge computing system can be incorporated within a network of other systems.
Operations on data	Which abilities are needed to create, read, update, process, and delete the data?
Optimization	How does the system perform optimally?
Performance	How to meet required operational targets?
Privacy	How to prevent unauthorized entities from gaining access to the data?
Reliability	How does the system deliver stable and predictable performance in expected conditions?
Resilience	How does the system withstand instability and unexpected conditions?
Security	Concerns related to confidentiality, integrity, and availability.
Standardization	Needed standards include credibility and trust, autonomy, east-west communication, machine learning algorithms, ECN functionality definition, semantic interoperability, fault detection, embedded system containerization, and carrier mode selection.
Time interval and latency	How to meet the requirements for time intervals between pairs of events?

Stakeholders to consider include:

— Developer and builders: Construct and deploy the edge computing system.

— Communicators: Explain edge computing to other stakeholders via its documentation.

— Suppliers and production engineers: Design, supply, and deploy the hardware and software on which edge computing will run.

— Operators and maintainers: Run the edge computing system once it has been deployed and manage the evolution.

— Owners: Derive the benefits of edge computing when in use.

— Users: Define the edge computing functionality and make use of it.

— Regulators: Define the legal, regulatory, and domain-specific norms of practice/rules for safety, reliability, security, privacy, and resilience.

7.4.3 Architecture view

7.4.3.1 Conceptual model: IoT component capability model

An example architecture of edge computing (Figure 73) contains application domain, data domain, network domain, and device domain, according to IoT component capability model kind. The network, computing, storage, and application security are four key capabilities.

1) Application domain:

Based on open interfaces provided by the device, network, and data functional domains, the application domain enables applications to operate at the network edge and provides full-lifecycle management of

applications. This domain also supports highly efficient operation and visualized management of edge services.

2) Data domain:

This domain provides full-lifecycle data optimization services such as extraction, aggregation, interoperation, semanticization, data analysis, and data presentation, as well as ensuring data security and privacy.

The mainstream architectures of data aggregation include OPC UA and data distribution service (DDS). To realize cross-vendor data interoperation and analysis, unified semantic meanings are required. It has been a consensus across the industry that building an unified information model architecture helps realize compatibility of multiple information models. The data domain adapts to data analysis models, performs real-time data cleansing and analysis, and triggers pre-defined service response policies based on data analysis results. This domain also provides data computing results for the application domain, and supports flexible and unified data presentation modes.

3) Network domain:

This domain provides services for system interconnection, data aggregation, and data transmission, covering the following functions:

Automatic operation and maintenance (O&M) and compatible heterogeneous connections. Software-defined networking (SDN) is becoming a mainstream technology that separates the control plane from the forwarding plane to make the network programmable. Applying SDN to edge computing enables millions of devices to access the network and supports flexible scalability. SDN provides highly efficient and low-cost automatic O&M, and realizes policy collaboration and convergence of network and security.

Real-time connection: Network connections shall guarantee time accuracy and data integrity. The Institute of Electrical and Electronics Engineers (IEEE) formulated time-sensitive networking (TSN) to unify technical standards for key services such as real-time priority and clocks. These standards indicate the future development of industrial Ethernet connections.

4) Device domain:

This domain includes discrete or embedded on-site nodes, such as meters, robots, and other devices, to support real-time smart interconnections and applications. Operating systems can support two different scenarios. One scenario is characterized by lightweight devices with low power consumption that supports zero touch provisioning (ZTP), self-networking, and cross-platform capabilities. The other scenario is for real-time computing that supports multi-task and priority-based scheduling capabilities to enable event response and task processing within given real-time requirements.

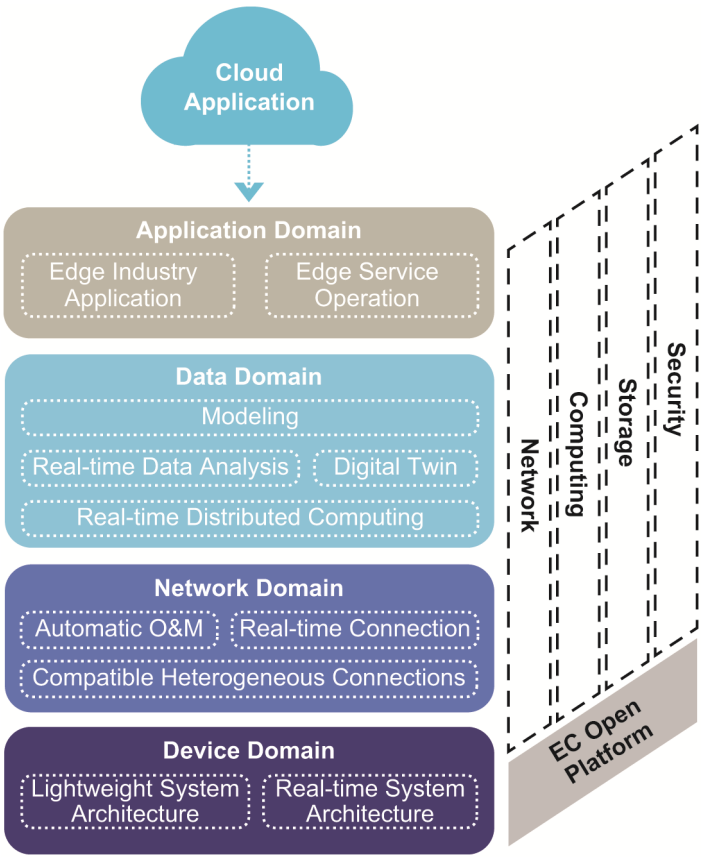


Figure 73—Conceptual architecture of edge computing

7.4.3.2 Function model

From the horizontal perspective, the example edge computing architecture (Figure 74) based on function model has the following characteristics:

a) Smart services are based on development service framework and deployment and operation service framework. Through these frameworks, intelligent coordination between service development and deployment is achieved. These frameworks enable consistent software development interfaces and automatic deployment and operations as well.

b) Smart service orchestration defines the E2E service flow through the service fabric (SF) to realize service agility.

   A SF provides the following functions: workflow and workload definition, visualized display, semantic check and policy conflict detection, version management of fabric and service models.

c) Use of a connectivity and computing fabric (CCF) enables a simplified architecture and simplifies the distributed edge intelligence architecture for services. The CCF also enables automatic and visualized deployment and operations of the OT and ICT infrastructure, supporting coordination between edge computing resource services and the service needs of industries.

   A CCF provides the following functions: resource awareness, edge virtualization function awareness, workload scheduling, data collaboration, multi-view display, open service interfaces.

d) Intelligent edge computing nodes (ECNs) are compatible with a variety of heterogeneous connections, support real-time processing and response, and deliver integrated hardware and software security.

e) Management service, full lifecycle data service, and security service are considered as common function elements through and across all these horizontal characteristics.

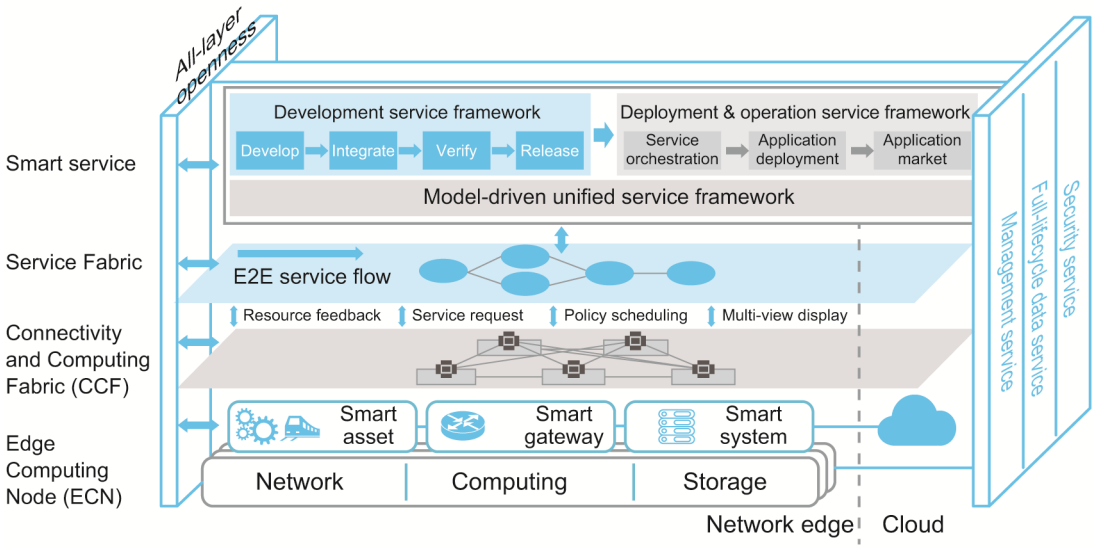


Figure 74—Function architecture of edge computing

7.4.3.3 Computing resource model

Edge computing nodes (ECNs) are Type A or Type B computing resource components as defined by computing components type model. An ECN has not only computing capability as a computing resource, but also network and storage capabilities. As a result, ECN could use both centralized computing resources model and distributed computing resources model to build solutions to fit various computing resource conditions.

7.5 Example architecture of IoT platform for Smart Cities

7.5.1 General information and key features

7.5.1.1 General relation to viewpoints according to 6.6

This example architecture of IoT Platform for Smart Cities is intended to be used by IoT system architects, designers, operators, and users to have the whole picture of the IoT system for Smart Cities and to help them design or implement function blocks of it. This example architecture describes the components of the IoT system for Smart Cities. And the example architecture is based on conceptual, compatibility, functional, collaboration, and lifecycle viewpoints.

In the conceptual viewpoint, the abstract IoT component is defined. Specific components are described in the example architecture.

The compatibility viewpoint—defined coexistance, interworkability, and exchangeability—can apply to different components of IoT systems so operators and owners can exchange different components of IoT systems easily.

In the functional viewpoint, three important components of IoT systems have been described: device, communication, and application in sensing channel (see 6.6.7.3.10) and actuation channel (see 6.6.7.3.11). In the architecture, IoT Platform components are introduced in the IoT system.

Collaboration model and collaboration process defined in the collaboration viewpoint apply to interaction and data sharing among different industries or departments of Smart Cities.

7.5.1.2 IoT Platform for a vertical industry

Supporting rich and diverse applications is one of the most important goals of building an IoT network. Based on requirements of governments, enterprises, and consumers, a variety of IoT applications will be derived to create great social value. Normally IoT application is a software run on an IoT Platform to solve a specific, or a group of, problems of end users (governments, enterprises, and consumers). For example:

- Analysis, prediction, and control of city traffic
- State monitoring and analysis of city assets
- Monitoring of city environment (air quality, water quality, soil quality)
- Analysis and early warning (such as wind, rainfall, landslide)
- Health monitoring and medical recommendations, etc.

Figure 75 describes the introduction of IoT Platform for a particular vertical industry. On the left side is the legacy M2M systems that have only three components: devices, communication networks, and applications. On the right side, the IoT Platform is introduced for a particular vertical industry. But it is also reasonable for some industries which keep three components (Device, Communication, and Application) without the new IoT Platform, e.g., parking in Figure 75.

An IoT Platform is a variety of industry applications integrated unified and open platform which performs device management and connection management of diverse industrial devices, through unified data management and standard open interfaces to provide service support to above different vertical applications. From a functional perspective, an IoT Platform is a technical system that provides a range of enabling services to support the delivery of IoT applications.

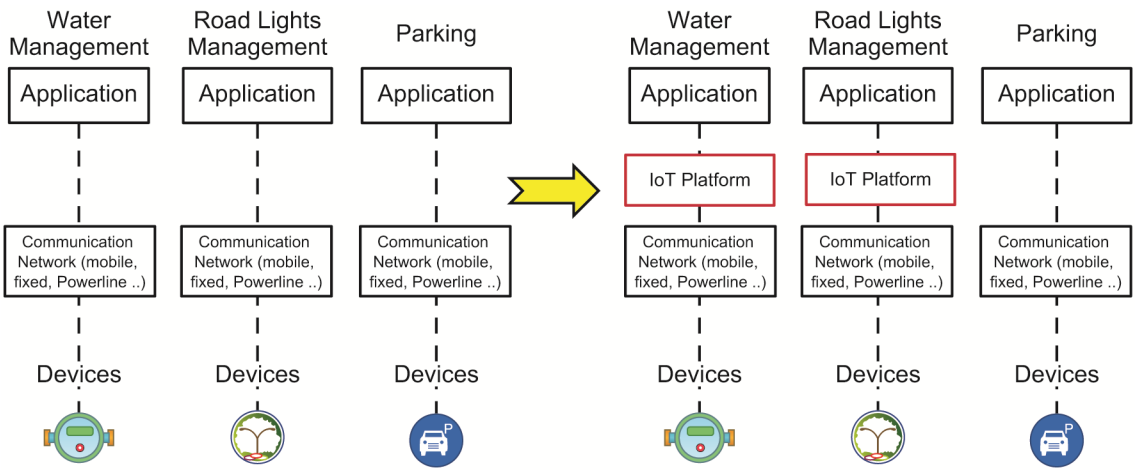


Figure 75—IoT Platform for a vertical industry

IoT Platform provides common functionality like data, device, user, and connectivity management, data storage, sharing, and preprocessing. The communication network just conveys information, while the IoT Platform decouples the intricacies and the specificity of the network to the applications. The application developers do not need to understand and consider the detailed of the communication access technologies when they develop the vertical IoT applications.

The solution developer can focus on the application development while the details are hidden behind APIs. This is an advantage independent of the size of the IoT solution. There is a bigger advantage for small IoT solutions as limited development resources do not have to be spent for the generic IoT functionality. IoT Platforms also allow to start with small deployments and grow in the future.

Figure 76 describes the internal function blocks and external interfaces of IoT. Currently there are nearly 450 IoT Platforms all over the world, it is clear that an IoT Platform needs two mandatory function blocks which all IoT Platforms should have:

— To downlink: the device and connection management to collect the data and control the devices

— To uplink: the application enablement to support vertical industrial applications

There are important but optional function blocks with data storage, data sharing, data analytics, Big Data and even artificial intelligence in IoT Platform. Different IoT Platforms can have different features in these function blocks. For a simple IoT Platform, it can just connect to internal modules (e.g., Big Data) to let the IoT system have the related functions.

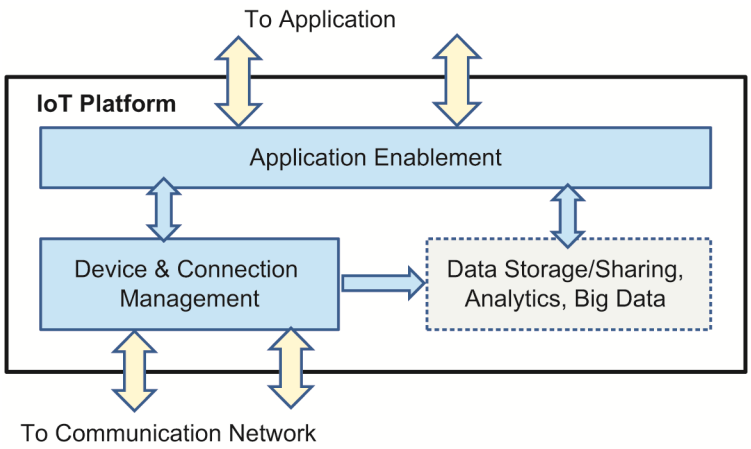


Figure 76—Function blocks of IoT Platform

7.5.1.3 IoT Platform for cross-industries

One of challenges of IoT systems for Smart Cities is to facilitate object and data reuse across vertical application domains/silos (e.g., water management, road lights management, waste management, parking, and environment monitoring). It is needed to consider the fragmentation of the standards and protocols since they are targeting specific vertical industries. For many IoT applications the aim is providing general standards, protocols, and solutions for as many vertical industry types as possible with the implication of developing limited adaptations to the applications that they need to support.

The IoT Platform is a topic of particular relevance; in fact, it is the platform that will induce the value of IoT solutions. Small IoT solutions have limited needs on software platforms, but the more we move toward larger systems, the more we need an IoT Platform that is capable of supporting many functionalities and offering a high level of programmability.

One horizontal platform to support multiple vertical industries application is more resource effective and increasingly popular in the IoT industrial world. There are hundreds of platforms developed and deployed all over the world now and the number is increasing. Most of the mainstream IoT-related standardization organizations and fora (e.g., ITU-T, oneM2M, AIoTI, and ETSI) are doing IoT Platform-related research and standardization.

Programming the world means that smart objects can be controlled by a highly distributed platform by means of APIs or protocols. The more objects to control, the more functions and data will be available. This yields interoperability and standardization problems as well as a management problem. It will become impossible to use traditional ways of management when the entity to be managed will number in the billions. So the platform should also be able to support autonomics and self-organization capabilities. Virtualization will also cause the replication of functionalities to be managed in non-traditional ways.

Figure 77 shows the cross-industry IoT Platform. The government considers that the Smart City cross-industry IoT Platform should have the device and connection management functions to share the investment and information among many different government departments, and the application enabler functions to support different vertical applications. Some vertical IoT Platforms (for example, road lights management in Figure 77) may disappear as all these functions of these vertical IoT Platforms can be provided by the cross-industry IoT platform.

Water management IoT Platform and cross-industries IoT Platform use the direct collaboration model process to share their data and information. Road lights management and parking use the indirect collaboration model and process to share their data and information, see 6.6.13 and Figure 63.

But real scenarios are much more complex than shown in Figure 77. Some IoT Platforms (for example water management in Figure 77) keep running and connect to Smart City cross-industries IoT Platform with west-east interface to share their data and information with a cross-domain IoT Platform.

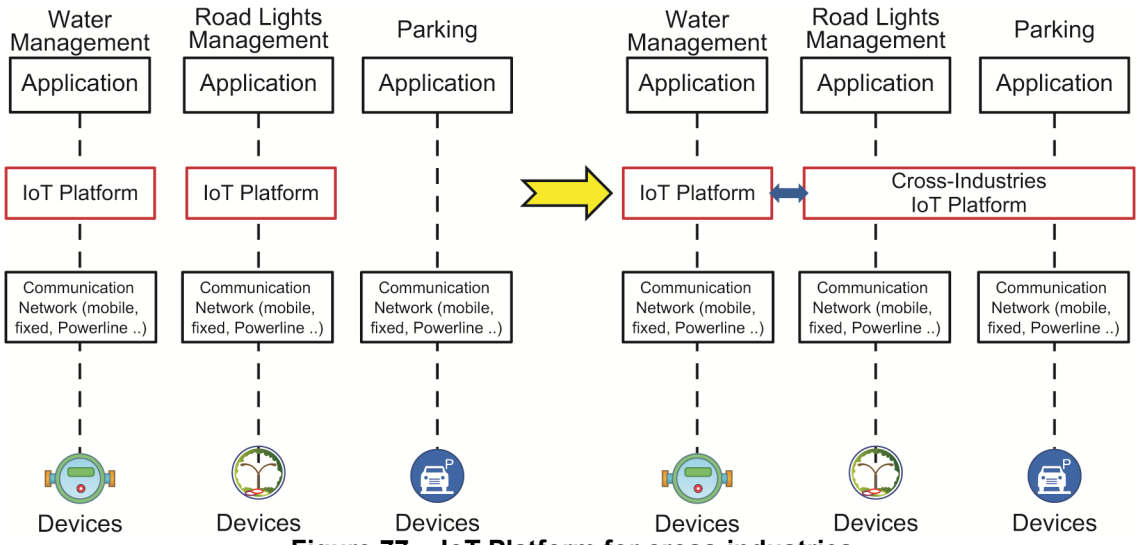


Figure 77—IoT Platform for cross-industries

In addition to the 5V characteristics (volume, variety, velocity, value, and veracity) of Big Data, the data sets in Smart City cross-industries scenarios have their own features, such as higher correlation, sensitivity to time order, and historical context. Big Data in different industries are processed and analyzed on the Cross-Industries IoT Platform for various application scenarios and purposes such as industrial automation, system health monitoring, predictive maintenance, and remote operation, etc. In this situation, to support these application scenarios, diverse Big Data analytics functions are performed, including but not limited to:

— Complex aggregation analysis: to profile information of different time periods or locations

— Multi-dimensional query and analysis: to examine and deep-mine the machine data from different perspectives

— Log data analysis: to monitor system and operational health

— Time-window-based stream data analysis: to identify temporal features and trends

— Complex event processing: to detect patterns and anomalies

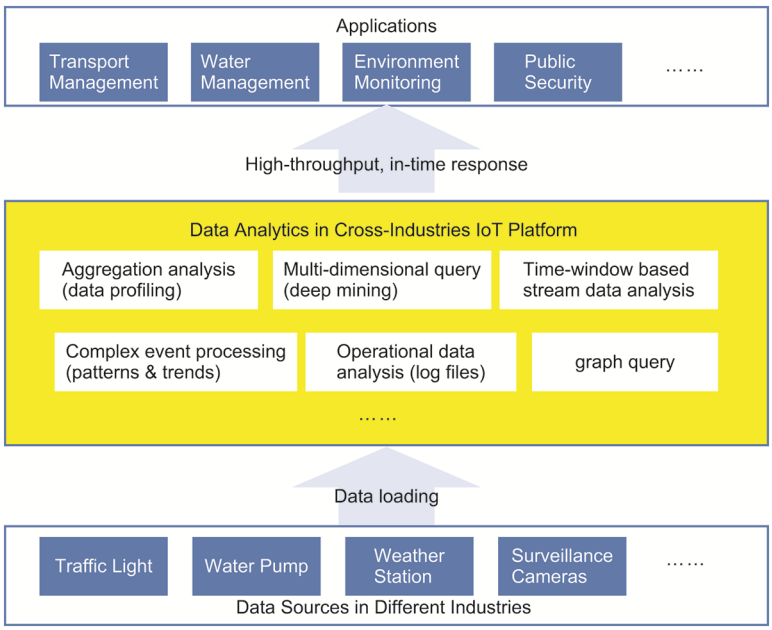


Figure 78—Data analytics in Cross-Industries IoT Platform

As shown in Figure 78, the Cross-Industries IoT Platform needs to address and support the processing of multi-typed input data from different data sources, such as traffic light, surveillance cameras, etc. And the applications may need to analyze these data sets simultaneously. To thoroughly analyze and mine the data (either real-time or historical) for value, diverse types of queries and analyses need to be applied. For in-time condition detection and decision making, these Big Data analyses need to be completed under throughput and latency requirements.

Analytics functions in Cross-Industries IoT Platform usually face stringent requirements such as:

— High performance in data loading

— Query and analysis

— A single copy of input data for different types of analytics

— In-time response to concurrent queries and commands

7.5.1.4 Smart City platform

In Smart Cities scenarios, the government needs a platform on top of different IoT Platforms to perform the interaction among different IoT Platforms. The main focus of the Smart City Platform is this interaction with the platforms of the domains and not the direct interaction with sensors and actors. The applications are just above this Smart City Platform, and normally Smart City Platform has no device management and connection management directly to devices (see Figure 79), but in some particular scenarios Smart City Platform can have device management and connection management directly to devices (see Figure 80).

Especially for Smart City projects interaction between several IoT Platforms is very important. Water, waste, traffic, energy, and pollution management all could have their own platform to manage and control their specific industries. The Smart City Platform is on top of that and coordinates and orchestrates all these industries. The Smart City Platform shall communicate with the IoT Platforms (vertical IoT Platforms and cross-industries IoT Platform), and operate with aggregate data from these platforms. It shall also initiate actions at the IoT Platforms (vertical IoT Platforms and cross-industries IoT Platform) and IoT Platforms

(vertical IoT Platforms and cross-industries IoT Platform) will initiate the actions in their industries accordingly.

For example, if the pollution management IoT Platform reports an increased air CO_2 or NO_2 pollution in a specific city area, the Smart City Platform can direct the traffic management IoT Platform to reroute car traffic. This is an interaction at platform level and not at the end devices.

There are hundreds of IoT Platforms now in the world, and they have different API interfaces to applications, it means that application developers need to take more time to modify their applications to adapt to different IoT platforms when they meet new IoT Platforms in new business cases or opportunities. Another important advantage for the Smart City Application Enabler Platform is that it provides a unified interface for application developers.

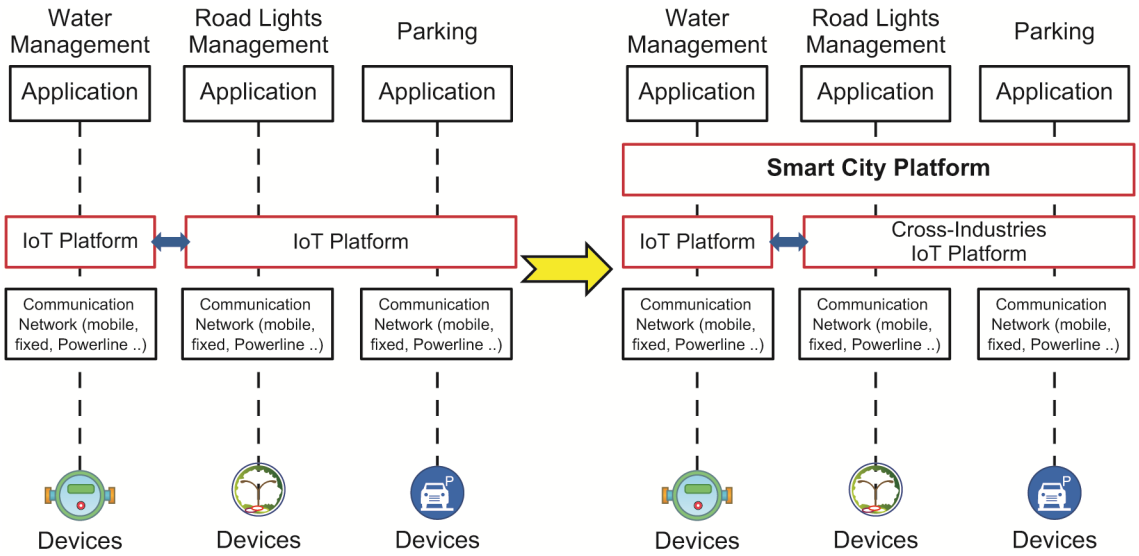


Figure 79—Smart City Platform

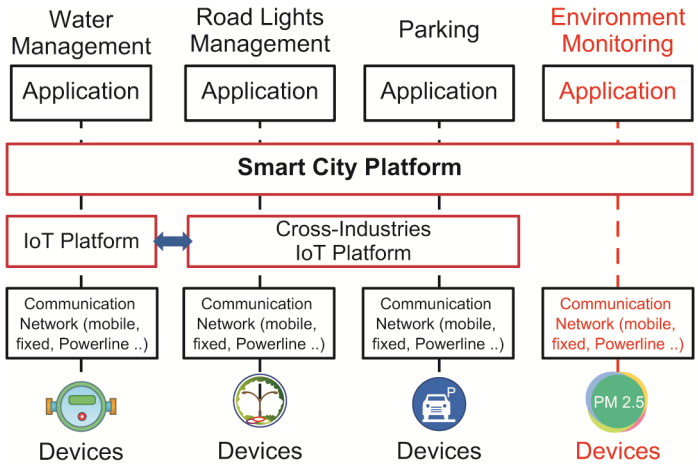


Figure 80—Smart City Platform with device and connection management to devices

7.5.2 Framed concerns and their stakeholders

7.5.2.1 Typical stakeholders

Some key stakeholders for architecture of IoT system for Smart Cities are:

- Planners/architects/designers/developers/testers/integrators
- Vendors/suppliers/installers
- Operators/maintainers/users
- Standardization and industrial organizations

7.5.2.2 Concerns

Key requirements of IoT are sensors and actuators, ubiquitous networks, massive data storage, sharing, and value creation.

Sensors and actuators are the basic devices that enable things to be connected and communicate with one another, while ubiquitous networks are the architecture that connects things. Much of the world is not covered by networks, and the vast majority of things remain unconnected.

Different industries have different sensor interfaces and incompatible sensors, making interconnections and interoperations impossible. It is necessary to standardize sensor platforms and make them intelligent, so things can connect and communicate to create more value.

Different connection scenarios have different requirements. For example, long battery life is crucial for smart meters, whereas low latency is a precondition for video surveillance and unmanned driving.

The target is to build ubiquitous networks that accommodate all IoT scenarios. There are a diverse range of access technologies for applications like unmanned driving, Smart Production, and intelligent meter reading.

When leveraging data to create value, problems with data ownership, security, privacy, and sharing need to be solved.

As the device, communication, and application are described in related viewpoints, Table 40, Table 41, Table 42, Table 43, and Table 44 only list concerns for the edge computing, cloud computing, horizontal IoT Platform, Smart City Platform, and security.

Table 40—Concerns of edge computing

Concern from common concerns list	Concern from edge computing	Elements of the concerns framed by the architecture framework
Time interval and latency	Time delay	Concerns related to time delay, 10 ms (e.g., industry automation, remote control of unmanned aerial vehicle); the response to device can be processed locally or by the edge node (e.g., gateway or cellular base station) Concerns related to real-time service analysis and intelligent decision making
Communication Cost Network Data velocity Data volume	Bandwidth consumption	Concerns related to the backhaul bandwidth consumption which is very costly and can be saved or avoid if it can be processed locally or in the edge nodes (e.g., video surveillance)
Collaboration	Data sharing	Concerns related to east-west communication between peers, breaking down the data silos
Safety Security	Security and privacy	Concerns related to local identity management and access control policies, specific to the running applications, securing the data close to its source, following the local regulations

Table 41—Concerns of cloud computing

Concern from common concerns list	Concern of cloud computing	Elements of the concerns framed by the architecture framework
Reliability	Reliability/distribution	Concerns related to stable ability to serve the IoT applications under the circumstance that the ICT hardwares are distributed
Resilience Evolvability	Scalability/elasticity	Concerns related to the capability of extension when the IoT devices number, or the IoT applications number, increase Concerns related to more computing power for cutting-edge applications
Cost	Low complexity of implementation	Concerns related to lower investment and O&M of the ICT infrastructure with planning, design, and implementation Concerns related to responsive service for more users at less cost
Security	Storage	Concerns related to safer, more secure storage, backup, and recovery

Table 42—Concerns of IoT Platform

Concern from common concerns list	Concerns of IoT Platform	Elements of the concerns framed by the architecture framework
Manageability	Management and configuration of devices	Concerns related to the management and configuration of gateways/devices including resource constrained devices
Maintainability	Firmware update of device	Concerns related to the capability for software management of devices
Communication	Connection management	Concerns related to the connections using SIM and non-SIM cards and connection management
Communication Networkability	Access agnostic	Concerns related to enable fixed and wireless (2G/3G/4G/NB-IoT) access across a large number of devices, independent of devices and networks and support for multi-protocol adaptation
Engineerability	Powerful exposure and integration capabilities	Concerns related to the networking, security, and data APIs allowing integrators and developers to ensure a secure connection, obtain data on demand, and deliver personalized user experiences
Data semantics Data Volume Relationship between data	Big Data analysis and real-time intelligence	Concerns related to hierarchical intelligence and control over cloud-based platforms, border gateways, and intelligent devices; the IoT Platform also provides intelligent analysis tools such as rule engine
Relationship between data	Process and analysis data set in different types	Concerns related to identification of the relationships among data sets originated from different resources
Collaboration	Process and analysis data set from different sources	Concerns related to support process and analysis data from different sources in a single copy
Standardization	Standardized query interface	Concerns related to support the standardized query interface such as SQL, etc.
Standardization	Supporting vertical applications	Concerns related to capability to integrate the diverse ICT technologies into one platform and support different industries and innovative IoT solutions Concerns related to the horizontal, unified, open, and cloud-based platform serving to different operators, enterprises, and verticals
Standardization	Open APIs for different applications	Concerns related to open APIs and agents to integrate with various IoT applications and provide access to sensors, devices, and gateways Concerns related to help operators, enterprises, and industry partners implement fast integration and rapid device access, as well as secure and reliable full connection management, propelling industry innovation and the fast generation of an IoT environment
Standardization	Support for mainstream IoT standards	Concerns related to support mainstream IoT standards and function implementation, for example, OPC UA, oneM2M, and ETSI standards In the Smart Home field, the platform complies with standards such as Z-Wave, ZigBee, BlueTooth, and Allseen In the IoV field, the platform complies with the JT/T 808 standard
Operability	Operation and charging	Concerns related to SIM card management and charging

Table 43—Concerns of Smart City Platform

Concern from common concerns list	Concerns of IoT application enabler platform	Elements of the concerns framed by the architecture framework
Security Relationship between data	Data sharing	Concerns related to sharing data among multiple devices/gateways within an application service, or among different application services
Communication	Interworking	Concerns related to machine socialization functionalities (such as existence discovery, correlated task discovery, message interface discovery, and process optimization for multiple machines with same tasks)
Data semantics	Ontology	Concerns related to re-using common ontologies (e.g., location, time ontologies, etc.) which are commonly used in different IoT applications
Standardization	APIs	Concerns related to decoupling applications from devices, freeing customers from adapting to private protocols and enabling them to deploy devices in phases Open ecosystem can accelerate industry application innovation

Table 44—Concerns of IoT Security

Concern from common concerns list	Concerns of IoT Security	Elements of the concerns framed by the architecture framework
Security	Endpoint security: covers the security of gateways and sensors	Concerns related to local data security which encrypt local data for storage Concerns related to local authentication which implements authentication and rights control for local logins Concerns related to basic system security which adopts trusted platform module (TPM) or trusted execution environment (TEE) based hardware security mechanisms to achieve secure boot and upgrade; performs hardening and antivirus protection on the local system; manages resources in the local system to prevent DoS attacks to local access
Security	Connection security: covers security of local and remote connections	Concerns related to data transmission security which encrypts data during transmission, and performs scrambling and signature on data Concerns related to two-way identity authentication on both parties of a connection. For example, the platform authenticates the identities of the gateway and sensor, the gateway authenticates the identities of the platform and sensor, and the sensor authenticates the identities of the gateway and platform. In addition, the identities of access devices are authenticated at the network layer to prevent unauthorized access. Concerns related to basic network security which implements IoT protocol filtering and IoT network isolation
Security	Platform and application security	Concerns related to cloud data security which encrypts data for storage in the cloud and protects privacy in a multi-user environment Concerns related to cloud authentication which includes user identity authentication and rights control Concerns related to basic system security which implements antivirus protection, security patch, sandbox isolation for code execution, and web application firewall (WAF) for web access, and builds the entire system on the basis of hardware security technologies such as trusted platform module (TPM) and trusted execution environment (TEE) Basic system security also covers data center network security, including DDoS prevention and firewalls
Security	Overall security management and control	Concerns related to security policy which develops the security policy for the entire system Concerns related to security monitoring and audit which monitors, records, and audits security events of the entire system Concerns related to response which handles security anomalous events Concerns related to key certificate management which manages security keys and certificates Concerns related to threat defense based on Big Data which analyzes system behavior by using the Big Data analysis technology to detect potential intrusions

7.5.3 Architecture view

7.5.3.1 Model kinds

7.5.3.1.1 Function model

According to function model, the IoT system includes several components to provide a variety of services, such as the Sensor and Actuator, Communication Network, IoT Platform, and Application. Normally the IoT architecture also includes security all over the IoT system (see Figure 81).

The IoT Sensor and Actuator component is composed of various sensors and controllers, including temperature and humidity sensors, two-dimensional code labels, RFID tags and readers, cameras, GPS, light switch, electronic lock, air-conditioning controller, and other sensors and actuators. The Sensor and Actuator component is the source of the collected information of the devices; also it can actuate the terminal devices. There is local control functionality using PLCs and industrial PLCs. So the Sensor and Actuator could be a motor, but also a drilling and milling machine, a robot, a production line, or even a whole factory.

The IoT communication network is responsible for communication of the information from the Sensor and Controller to the IoT Platform or the control command from the platform to the Sensor and Controller. The communication network can use a variety of transmission technologies, including the wireless and fixed networks. Communication between direct device to device (D2D) shall also be supported in the IoT system.

The IoT Platform enables enterprises to monitor and control IoT devices, build applications to meet digital business requirements, and will be an essential element in the development of a digital single market. In the new digital economy, the IoT Platform ecosystems are the foundation for new value creation and the driver for developing new IoT applications. Normally, the IoT Platform includes device management, connectivity and management, data sharing, information exchange, and Big Data analysis. IoT Platform is optional for small and legacy IoT network. It is very useful and necessary in the future IoT network.

The IoT application (see definition in 3.1) is responsible for delivering application-specific services to the user. It defines various applications in which the IoTcan be deployed, for example, Smart Homes, Smart Cities, and Smart Energy.

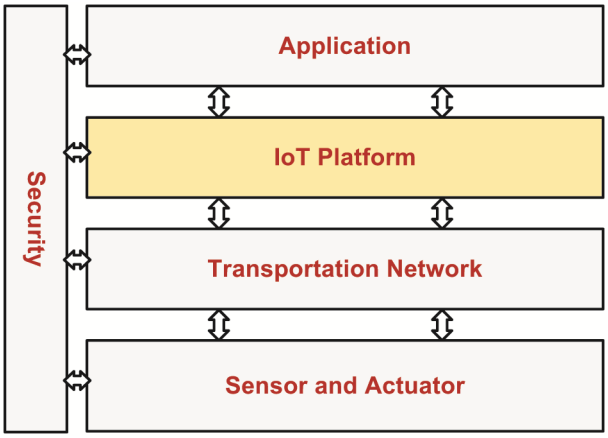


Figure 81—IoT components of IoT system

7.5.3.1.2 Data access, storage, and sharing model

For specific implementation case, the IoT system can have edge computing, cloud computing, and Application Enabler Platform components (see Figure 82).

The IoT edge computing provides the distributed computing for reduction of time delay and bandwidth consumption. IoT edge computing normally includes the device communication module and the edge computing IoT gateway. IoT edge computing can be done on sophisticated end devices or dedicated computing devices also. Edge computing is optional for IoT systems.

The IoT cloud computing is the cloud data center which has cloud computing and cloud storage. The IoT cloud computing provides more power, safer data, and easier access to the information and tools needed for success in any industries or organizations. The IoT cloud computing is corresponding to infrastructure as a service (IaaS) of the cloud computing. Cloud computing is optional for IoT systems.

Most IoT Platforms start with a connectivity management function block. It has the function of bringing different protocols and different data formats into one unified interface. This is necessary in order to help ensure all devices can be interacted with and data are read correctly. Having all device data in one place and in one format is the basic necessity to monitor, manage, and analyze IoT devices. Connectivity management should have common understanding of data semantics.

The connectivity management function block also includes the component of SIM management, at which operators are professional. The SIM management provides the functions of lifecycle management, status monitoring, and fault diagnosis for SIM cards. It can be implemented independently from, or together with, other functional entities of the connectivity management function block. Independent deployment of the SIM management is a major IoT Platform design pattern in the past few years. Operators can better serve IoT customers, usually the enterprises, and earn the value of the “pipe” in the IoT value chain through the SIM management.

Another function block of an IoT Platform performs the device management, which help ensure the connected objects are working properly and its software and applications are updated to the required versions and running properly. Tasks performed in this function block include device provisioning, remote configuration, firmware/software updates, and troubleshooting. As thousands or even millions of different devices become part of an IoT enabled solution, bulk-actions and automation are essential to control the costs and reduce the manual labor. Device management involves IoT device access, data collection, and device status monitoring and maintenance.

To interwork with massive and heterogeneous IoT devices regarding the functions of device management and connectivity management, the IoT Platform normally requires an IoT agent to perform protocol translation and access control in order to provide easy and quick access to the IoT devices.

Some IoT Platforms may provide a data management function block. This function block include data storage, data sharing, data analysis (Big Data), data monitoring, and data visualization functions. Other IoT Platforms may implement only part of those functions while connect to external entities for the rest of the functions.

An IoT Platform may also support the application enablement function block which helps IoT application developers to quickly develop and deploy IoT applications. Built-in application programming interfaces (API), software development kits (SDK), and gateways are the key enablers to the integration of third-party systems and applications. To meet the requirements, the application enablement function block should provide further components like rule engine, third-party capability integration (such as GIS and email), industrial enablement suite, and application market.

In some scenarios, the IoT Platform does not support applications directly and it can connect to Smart City Platform which can support many more applications. Smart City Platform has the Northbound APIs to

support diverse vertical applications development and southbound APIs to connect different IoT Platforms. In this scenario, the IoT Platform developers do not need to consider different interfaces from IoT Platforms, so the Smart City Platform is very useful for building IoT application ecosystem.

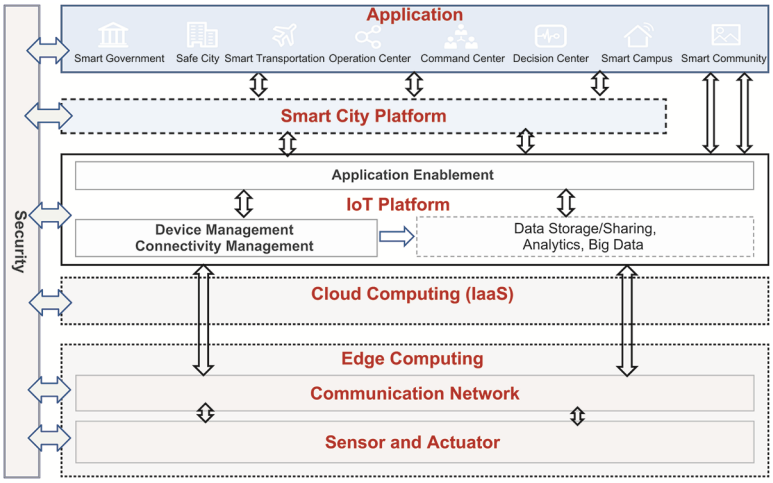


Figure 82—IoT components of an IoT system with optional edge computing, cloud computing, and Application Enabler Platform

7.5.3.2 Correspondence rules

This architecture view has correspondence with other viewpoints. Different components normally have different functions.

Conceptual viewpoint: Components are defined in conceptual viewpoint.

Compatibility viewpoint: The components of an IoT system can be exchangeable so the architects can easily replace each component from different providers.

Communication viewpoint: The communication network is based on the communication viewpoint.

Collaboration viewpoint: The different industries of Smart Cities share their data and information, and interact with each other using the collaboration model and collaboration process.

Functional viewpoint: Different function blocks defined in the architecture shall follow the rules in function viewpoint.

Lifecycle viewpoint: Along the whole lifecycle of an IoT system, the components can be removed, added, updated accordingly, and the different IoT Platforms can be deployed based on the concerns from related stakeholders.

Threat model viewpoint, security and safety monitoring viewpoint, adequate design for required security viewpoint, and private and privacy viewpoint are related to security components.

7.6 Example architecture of OPC Unified Architecture (UA)

7.6.1 General Information and key features

7.6.1.1 Introduction

This example architecture of OPC Unified Architecture (UA) is intended to be used by IoT system architects, system designers, and developers to architect, design, and construct a target IoT system. It is also intended for maintainers including production engineers and system administrators who typically run the IoT system once it has been deployed.

OPC Unified Architecture (UA) defines information integration interoperability with high levels of security, reliability, and availability. An IoT solution with OPC UA consists of one or more OPC UA applications. Any OPC UA application may be a supplier and/or requestor of information. The solution may be architected such that an OPC UA application functions as a consolidator of other OPC UA applications.

7.6.1.2 Relationship to viewpoint catalogue

This example architecture is based on the following viewpoints: conceptual, compatibility, communication, information, function, and computing resource viewpoints.

In the conceptual viewpoint, the focus of the IoT component capability model kind is not on the internal implementation but rather on the component's capabilities. This focus on capabilities is possible if an IoT component uses standardized interfaces to expose capabilities. Specific capabilities that can be exposed by OPC UA are described in the example architecture.

IoT solutions based on OPC UA demonstrate high degrees of compatibility as defined by the compatibility viewpoint. This example architecture describes how interconnectability, interworkability, interoperability, and exchangeability can be achieved in an IoT solution based on OPC UA.

The communication viewpoint includes the open systems interconnection model (OSI model) as a model kind to define communications within and between IoT systems. The OPC UA example architecture demonstrates the alignment of OPC UA with the OSI reference model kind.

Semantic interoperability as described in information and function viewpoints aligns with the information modeling capabilities of OPC UA. An example is shown in Figure 84.

The OPC UA security architecture aligns with access control viewpoint as well as with the privacy and trust viewpoint.

OPC UA is scalable for deployment on a variety of computing resources including those listed in computing resource viewpoint. Depending on the use case, the IoT application is the OPC UA application or the OPC UA capabilities are a subset of an IoT application total set. In either case, OPC UA can be deployed in a variety of computing resource distribution models including those defined in computing resource viewpoint: centralized and distributed computing resources model.

7.6.2 Framed concerns and their stakeholders

Among the principles that drove the OPC UA Architecture are (1) securely connect to more with less effort, and (2) easily integrate more types of data.

— Connectivity from sensor to enterprise requires platform independence as well as the ability to securely traverse internet and firewalls

— Highly performant yet robust and fault tolerant

— Interoperability and scalable via common information model that is also extensible to incorporate other standard data models. Solutions deployed at the sensor level likely require a far simpler model than those deployed at the enterprise level.

Table 45 shows the framed concerns of OPC Unified Architecture (UA) (without prioritization or weighting).

Table 45—Concerns of OPC Unified Architecture (UA)

Concerns list	Elements of the concerns framed by the architecture framework
Communication	How does information exchange between peers that locate in one system or in different systems?
Confidentiality	How to help ensure that information is not made available or disclosed to unauthorized individuals, entities, or processes?
Networkability	How to maintain the ease and reliability with which the OPC UA applications can be incorporated within a network of other systems?
Operations on data	Which abilities are needed to create, read, update, process, and delete the data?
Optimization	How does the system perform optimally?
Performance	How to meet required operational targets?
Privacy	How to prevent unauthorized entities from gaining access to the data?
Reliability	How does the system deliver stable and predictable performance in expected conditions?
Resilience	How does the system withstand instability and unexpected conditions?
Security	How to assure concerns related to confidentiality, integrity, and availability?
Standardization	How do OPC UA applications from multiple suppliers interoperate?

Stakeholders to consider include:

— Developer and builders: Construct and deploy the OPC UA applications.

— Communicators: Explain OPC UA applications to other stakeholders via its documentation.

— Suppliers and production engineers: Design, supply, and deploy the hardware and software on which OPC UA applications will run.

— Operators and maintainers: Run the OPC UA applications once it has been deployed and manage the evolution.

— Owners: Derive the benefits of OPC UA applications when in use.

— Users: Define the connectivity requirements of components from single or multiple suppliers.

— System administors: Define the security infrastructure within OPC UA applications.

7.6.3 Architecture view

7.6.3.1 Model kinds

7.6.3.1.1 Conceptual model: component capability model

The OPC UA architecture consists of data modeling and transport mechanisms driven by a comprehensive set of secured connectivity requirements—reliable communication that adheres to industry mandated levels

of robustness and fault tolerance, platform independence, scalability, performant, interoperable, and adherence to industry standard-based security and access control. These mechanisms can be conceptually modeled using the component capability model defined in the conceptual viewpoint.

An IoT component may consist of one or more applications with OPC UA.

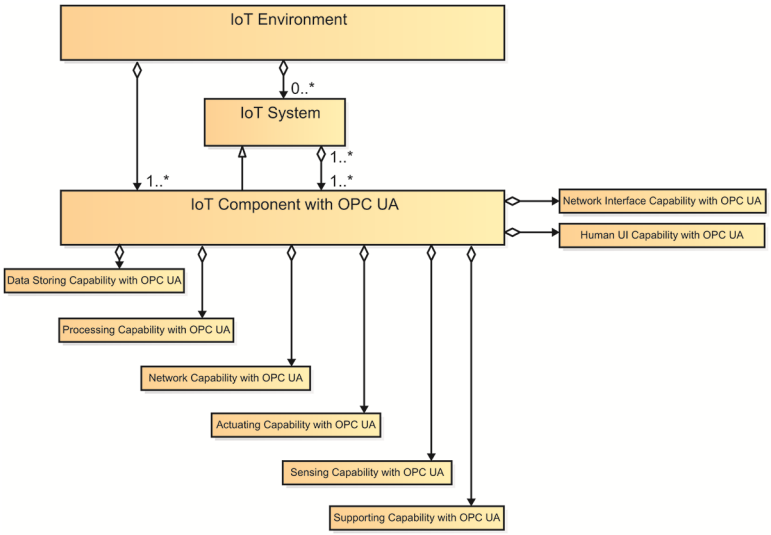


Figure 83—Class diagram of the capabilities of an IOT component with OPC UA

The capability classes in Figure 16 are described as follows:

Sensing/actuation capabilities with OPC UA: Given a suitable platform, OPC UA can be deployed with a lightweight profile suitable for sensors or actuators. Such profiles shall consider proper resource utilization within the constraints applicable to a targeted embedded platform.

Processing capability with OPC UA: Information modeling capability with OPC UA provides the ability to transform data based on a defined algorithm. Base information models defined by OPC Foundation enable derivation and specialization by other industry domains to define processing algorithms.

Data-storing capability with OPC UA: In addition to the historical access information model, OPC UA is a strong enabler of data-storing capabilities through its support of base types such as files.

Network capability with OPC UA: Real-time data, historical data, alarms, and conditions are three examples of ability with OPC UA to move data from one physical location to another. These services can be used in a client-server, peer-to-peer, or pub-sub model.

Supporting capability with OPC UA: OPC UA addresses security at all levels of the architecture. This includes encryption, authentication, auditing, two-way trust between endpoints, etc. Its design is based on the latest security standards and protocols. Organizations such as NIST, Industrie 4.0, and China 2025 recognize OPC UA for its comprehensive set of security features. OPC Foundation has an aggressive security stance with respect to security—the job is never done when it comes to protecting data. So even though security is baked in, it is fairly extensible in its ability to support new mechanisms.

Network interface capability with OPC UA: Refer to Figure 84 to understand the network interface capabilities with OPC UA.

Human UI capability with OPC UA: OPC UA consists of enablers for UI capabilities. Typical client applications addressed by OPC UA architecture include process or system displays, trends, alarm management, logging, configuration, and discovery.

7.6.3.1.2 Compatibility model

Refer to the Table 46 to understand how OPC UA supports the compatibility model introduced in 6.6.3.

Table 46—Compatibility levels

Level	Definition	OPC UA
Interoperability	Ability of two or more devices to work together in one or more distributed applications with the shared knowledge of the data types and the semantics of the data transmitted	A minimum level of interoperability can be achieved via the base information model defined by OPC Foundation. Higher levels of semantic interoperability can be achieved via companion specifications.
Exchangeability	Ability of one device to fulfill the same role of another device in the physical operation and distributed application as required by the system design	OPC UA can participate in a solution that offers exchangeability. It does not preclude this level of compatibility, but should be considered with platform infrastructure as well as hardware design.

7.6.3.1.3 Communication model: OSI reference model kind meta-model

Figure 84 shows the relationship of OPC UA transport mechanisms, or services, to the overall communication stack. OPC UA transport is located just above the session/communication layer. OPC UA transport consists of two types of mechanisms: data encoding and transport protocols.

Data encoding is the serialization of the service messages including its input and output parameters to a network format.

Transport protocols are used for establishing a connection between client and server.

Figure 84 also shows how security is inherent to the architecture. Security mechanisms are built on industry standards yet are designed to adapt to the changing technology.

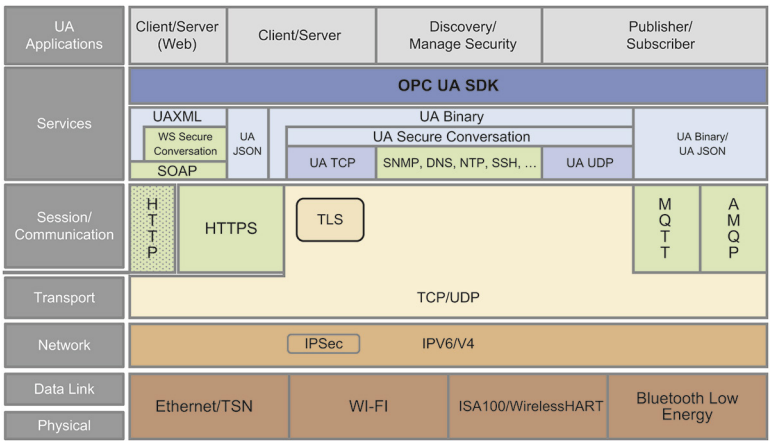


Figure 84—OPC UA communication relationship to OSI reference model

7.6.3.1.4 Information and function models: semantic interoperability

OPC UA data modeling mechanisms are based on a common model for all data. Real-time data, historical data, alarms, and conditions, etc. are layered upon the same object-oriented common model. An extensible type system enables the exposure of metadata. This same common model can be used as the base for other standard data models, also known as *companion specifications*.

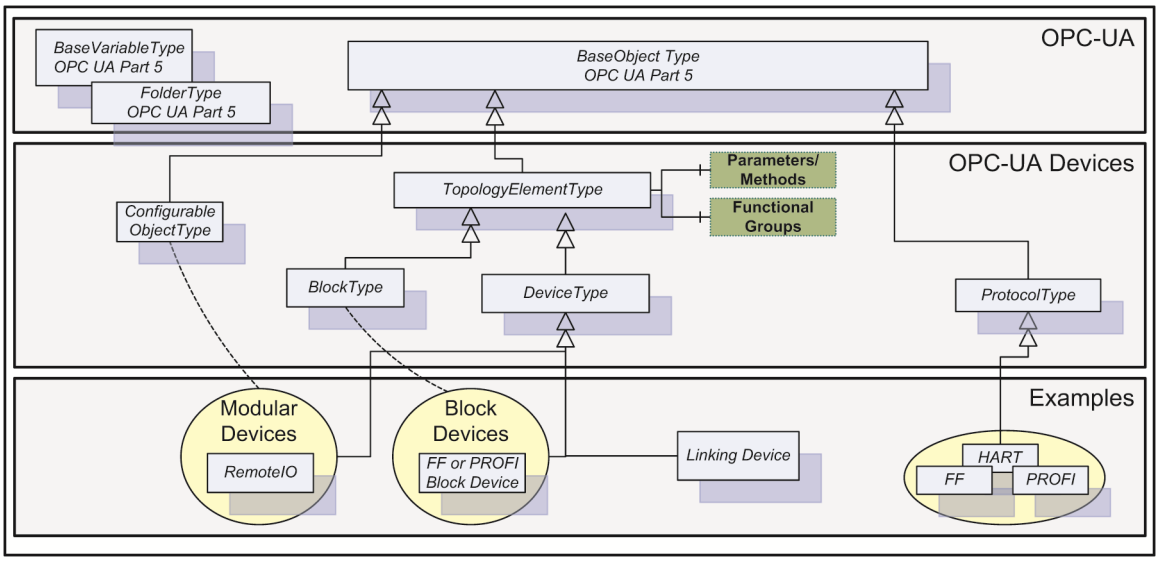


Figure 85—UA data model example

Figure 85 is a partial example of the data model for device integration. The OPC UA box (upper box) shows core OPC UA types. The middle box, OPC UA devices, introduces device specific types which are derived from core OPC UA types. Finally, the examples box (lower box) shows device and protocol examples that are derived from types defined in OPC UA devices. The inclusion of the metadata in the information model allows for automated discoverability of data and automatic adaptability of systems to the data.

7.6.3.1.5 Access control and privacy and trust model

The OPC Foundation considered security as a fundamental requirement for OPC UA. Mechanisms supported within the OPC UA model are based on extensive threat analysis. As illustrated in Figure 86, security is integrated into its architecture at the three primary levels: user, application, and transport. This security model as shown assumes client and server support the richest set of security capabilities. OPC UA also includes certificate management capabilities. However, the OPC UA security model is intended to integrate with an existing security infrastructure to enable certificate issue, as well as definition of users and roles, etc.

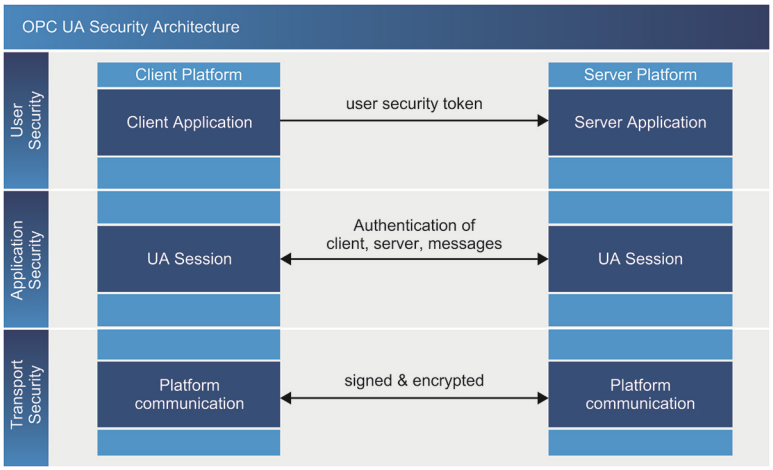


Figure 86—OPC UA security model

Mechanisms are provided at user, application, and transport levels. User level security grants access for a specific user with a given role. At the application level, digitally signed X.509 certificates can be exchanged between the client and server to ensure authentication of messages in both directions. A mechanism to sign messages provides integrity and authentication of the message at the transport level. Encryption of messages at the transport level prevents eavesdropping.

7.6.3.1.6 Computing resource model

The computing resource viewpoint states that “the IoT application needs computing resources according to the requirements of the use case. Choosing the computing resource is based both on the capabilities of the computing resource and the capabilities of the network between the computing resource and the application needs.”

Referring to Figure 84, the OPC UA application layer represents the breadth of OPC UA—each UA application can be tailored to use only the required data model and transport mechanisms it needs. This vertical slice of the layered OPC UA architecture is referred to as a profile. An OPC UA profile is a named collection of capabilities including capacity characteristics. It is the OPC UA profile that aligns with the set of use cases required on a computing resource. A very thin vertical slice or profile would be deployed on an embedded device such as a sensor or actuator. Controllers, such as PLCs, require well managed use of resources and as such may deploy a profile slightly richer than that of an embedded device. Data centers and gateways are typically capable of supporting a very rich set of OPC UA profiles.

7.7 Example architecture of a distributed computing architecture for a chemical process in a Smart Manufacturing Ecosystem using BATCH control

7.7.1 General

The system of interest is a distributed computing architecture for a chemical process in a Smart Manufacturing ecosystem using BATCH control according to IEC 61512 (IEC 61360-1:2009 [B39]). The architecture description in 7.7 expresses the architecture that exhibits the system of interest and identifies the framed concerns listed in 7.7.2. The stakeholders listed in 7.7.2 have interest in the system of interest.

Each architecture view in 7.7.3 of the architecture description in 7.7 adheres to the conventions of its governing architecture viewpoint.

The architecture description in 7.7 includes exactly one architecture view in 7.7.3 for each architecture viewpoint used.

The structure of the architecture description consists of several process stages which shall be organized as an ordered set, which can be serial, parallel, or both. A process stage is a part of a process that operates independently from other process stages. It results in a planned sequence of chemical or physical changes in the material being processed. The architecture is described in IEC 61512. The example given in 7.7 is based on the process stages in the polyvinyl chloride (PVC) process. The process stages are described in IEC 61512-1:1997, 4.1.3.1 [B43]:

— Polymerize: polymerize vinyl chloride monomer (VCM) into polyvinyl chloride
— Recover: recover residual vinyl chloride monomer
— Dry: dry polyvinyl chloride

7.7.2 Framed concerns and their stakeholders

The stakeholders listed in 5.1 have interest in the distributed computing architecture for a chemical process in a Smart Manufacturing ecosystem using BATCH control, which is the system of interest in 7.7.

Each viewpoint may provide unique stakeholder names that interpret and can be cross-referenced to the stakeholders in 5.1. The cross-reference is provided within each viewpoint description.

The framed concerns are shown in Table 2 (without prioritization or weighting).

7.7.3 Architecture views

7.7.3.1 Conceptual view

The conceptual viewpoint is intended to assist architects in the design, analysis, and expression of system architectures employing a shared language. It frames a number of architecture concerns related to the concepts within architecture description and the use of such descriptions. The architecture view to the conceptual viewpoint is described in IEC 61512-1:1997, 5.3.3.2 [B43].

The following concerns of Table 2 are addressed:

— Conceptual

— Autonomy

— Data semantics

7.7.3.2 Compatibility view

The processes, batches, and batch processes (see IEC 61512-1:1997, 7.2.4.2 [B43]) require a high level of compatibility. For example, the lower four levels of the model in Figure 2 of IEC 61512-1:1997 [B43] refer to specific equipment types. An equipment type is a collection of physical processing and control equipment grouped together for a specific purpose. This grouping needs at least the compatibility level of interoperability. Otherwise bridges or converters are needed and will violate concerns of the stakeholder.

The following concerns of Table 2 are addressed:

— Adaptability

— Behavioral

— Collaboration

— Communication

— Constructivity

— Cost

— Data velocity

— Data volume

— Deployability

— Disposability

— Engineerability

— Enterprise

— Evolvability

— Functionality

— Identity

— Logical time

— Maintainability

— Manageability

— Monitorability

— Networkability

— Operability

— Operations on data

— Optimization

— Performance

— Physical

— Physical context

- Privacy
- Procureability
- Producibility
- Quality
- Relationship between data
- Relevance
- Safety
- Security
- Sensing
- Standardization
- States
- Synchronization
- Time awareness
- Time interval and latency
- Uncertainty
- Usability
- Utility

7.7.3.3 Lifecycle view

Process and control engineering (see IEC 61512-1:1997, 6.1.3 [B43]) describes the architecture view to the lifecycle viewpoint. The batch process needs a dynamic application of the lifecycle so that a batch could be seen as a sub-element of the lifecycle and the batch can be seen as a superset of a lifecycle.

The stakeholders listed in 6.6.4 apply.

7.7.3.4 Communication view

The general recipe provides a means for communicating processing requirements to multiple manufacturing locations. The details are given in IEC 61512-1:1997, 4.3.2 [B43]. Also security and network requirements are specified in 4.3.2, Clause 5, and 6.1.2.2 of IEC 61512-1:1997 [B43].

The stakeholders listed in 6.6.5 apply.

7.7.3.5 Information view

IEC 61512-1:1997, Clause 5 [B43] specifies the batch control concepts that exchange information and interact with the physical world by sensing, processing information, and actuating according to 6.6.6.

A structure for batch control can have three types of control needed for batch manufacturing. When these control types are applied to equipment, the resulting equipment entities provide process functionality, control capability, and information exchange according to 6.6.6. The concept of recipes in IEC 61512-1:1997, Clause 5 includes the four types of recipes and the contents of these recipes in terms of

the information categories used to describe a recipe. A relationship is established between the procedure in a recipe and the control associated with specific equipment entities (equipment control). The concept of collapsibility of the recipe procedure and of equipment control is discussed. Recipe transportability criteria are introduced for the four types of recipes. The procedural control may be entirely defined as part of equipment control or it may be based on procedural information passed on to the equipment entity from the recipe according to 6.6.6. Recipes contain the following categories of information: header, formula, equipment requirements, procedure, and other information. IEC 61512-1:1997, 7.2.3.1.5.2 [B43] provide details regarding these categories. Any significant changes from one recipe type to another are noted.

Stakeholders and concerns apply according to 6.6.6.

7.7.3.6 Function view

Both IEC 61512-1:1997, 4.1.3 [B43] and Figure 87 provide an illustration of a specification of the batch process stages.

Operations, see 7.7.4.1, and actions, see 7.7.4.2, shall use the function viewpoint during the design, implementation, and operation phases.

The functions of an operation see also 7.7.4.6.2.3.4 shall be composed by using instances of function block types, see 6.6.7.3.2 and there operations using FB execution according to 6.6.7.3.3.

7.7.3.7 Computing resource view

IEC 61512-1:1997, 5.6 [B43] specifies the batch allocation and arbitration of computing resources according to computing resource viewpoint in 6.6.14.

A distributed computing resource shall provide status information that it is in use by whom or available to get computing tasks assigned. Subclause 7.7.4.6.7 addresses mechanisms for allocating resources to a batch or unit and for arbitrating the use of common resources when more than one requester needs to use a common resource at the same time.

Stakeholders and concerns apply according to 6.6.14.

7.7.4 Architecture and models with their descriptions

7.7.4.1 Process stages

The process shall consist of several process stages which shall be organized as an ordered set, which can be serial, parallel, or both. A process stage is a part of a process that operates independently from other process stages (see also 6.6.4.4.1). It results in a planned sequence of chemical or physical changes in the material being processed. The process stages in the polyvinyl chloride process are described in 7.7.1.

7.7.4.2 Process operations

Each process stage shall consist of an ordered set of one or more process operations. Process operations represent major processing activities. A process operation shall result in a chemical and physical change in the material being processed. The process operations for the polymerization of vinyl chloride monomer into polyvinyl chloride process stage shall be:

— Prepare reactor: evacuate the reactor to remove oxygen

— Charge: add demineralized water and surfactants

— React: add vinyl chloride monomer and catalyst, heat to 55 °C to 60 °C and hold at this temperature until the reactor pressure decreases

The process operations shall be performed according to 6.6.7.

Figure 87 shows the process model.

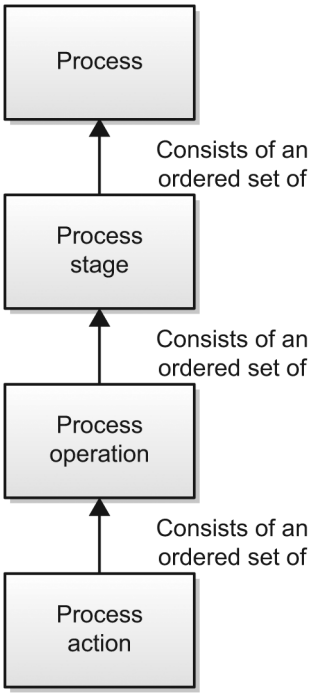


Figure 87—Process model

7.7.4.3 Process actions

Each process operation shall be subdivided into an ordered set of process actions that carry out the processing required by the process operation. Process actions describe minor processing activities that are combined to make up a process operation. Process actions for the react process operation shall be:

— Add: Add the required amount of catalyst to the reactor.

— Add: Add the required amount of vinyl chloride monomer to the reactor.

— Heat: Heat the reactor contents to 55 °C to 60 °C.

— Hold: Hold the reactor contents at 55 °C to 60 °C until the reactor pressure decreases.

The process actions shall be performed according to 6.6.7.

7.7.4.4 Physical model

7.7.4.4.1 General

The physical model that describes the physical assets of an enterprise in terms of enterprises, sites, areas, process cells, units, equipment modules, and control modules.

Figure 88 shows the physical model.

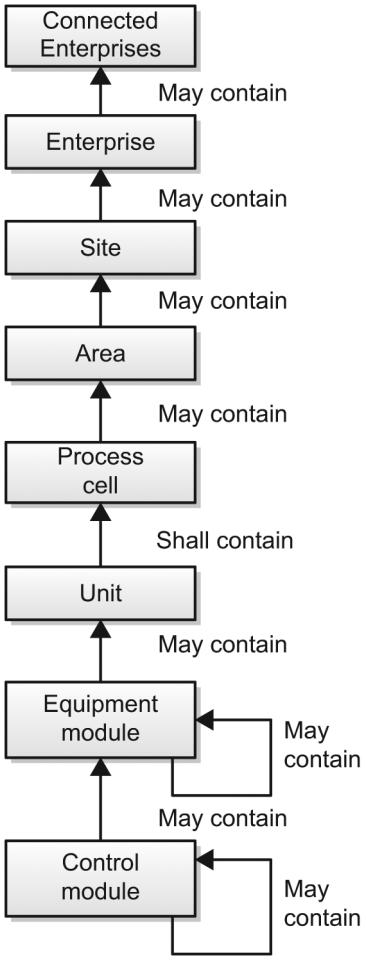


Figure 88—Physical model

The lower four levels of the model in Figure 88 refer to specific equipment types. An equipment type in Figure 88 is a collection of physical processing and control equipment grouped together for a specific purpose and represents a thing. The lower levels in the model are specific to technically defined and bounded groupings of equipment. The four lower equipment levels (process cells, units, equipment modules, and control modules) are defined by engineering activities. During these engineering activities, the equipment at lower levels is grouped together to form a new higher level equipment grouping that represents again a thing. This is done to simplify operation of that equipment by treating it as a single larger

piece of equipment. Once created, the equipment cannot be split up except by re-engineering the equipment in that level.

7.7.4.4.2 Connected enterprise level

A connected enterprise is a collection of two or more differently owned enterprises or contracted enterprises that are connected to each other according to 6.6.13. The connected enterprises are responsible for determining what products will be manufactured, at which enterprises they will be manufactured, and in general how they will be manufactured.

7.7.4.4.3 Enterprise level

It may contain sites, areas, process cells, units, equipment modules, and control modules. The enterprise is responsible for determining what products will be manufactured, at which sites they will be manufactured, and in general how they will be manufactured.

7.7.4.4.4 Site level

A site is a physical, geographical, or logical grouping determined by the enterprise. It may contain areas, process cells, units, equipment modules, and control modules. The boundaries of a site are usually based on organizational or business criteria as opposed to technical criteria.

7.7.4.4.5 Area level

An area is a physical, geographical, or logical grouping determined by the site. It may contain process cells, units, equipment modules, and control modules. The boundaries of an area are usually based on organizational or business criteria as opposed to technical criteria.

7.7.4.4.6 Process cell level

A process cell contains all of the units, equipment modules, and control modules required to make one or more batches. Process control activities respond to a combination of control requirements using a variety of methods and techniques. Requirements that cause physical control actions may include responses to process conditions or to comply with administrative requirements. A frequently recognized subdivision of a process cell is the train. A train is composed of all units and other equipment that may be utilized by a specific batch. A batch does not always use all the equipment in a train. Furthermore, more than one batch and more than one product may use a train simultaneously. The order of equipment actually used or expected to be used by a batch is called the *path*. Although a process cell may contain more than one train, no train may contain equipment outside the boundaries of the process cell.

A process cell is a logical grouping of equipment that includes the equipment required for production of one or more batches. It defines the span of logical control of one set of process equipment within an area. The existence of the process cell allows for production scheduling on a process cell basis, and also allows for process cell-wide control strategies to be designed. These process cell-wide control strategies might be particularly useful in emergency situations.

7.7.4.4.7 Unit level

A unit is made up of equipment modules and control modules. The modules that make up the unit may be configured as part of the unit or may be acquired temporarily to carry out specific tasks. One or more major processing activities—such as react, crystallize, and make a solution—can be conducted in a unit. It combines all necessary physical processing and control equipment required to perform those activities as an independent equipment grouping. It is usually centered on a major piece of processing equipment, such as a mixing tank or reactor. Physically, it includes or can acquire the services of all logically related equipment necessary to complete the major processing task(s) required of it. Units operate relatively independently of each other.

A unit frequently contains or operates on a complete batch of material at some point in the processing sequence of that batch. However, in other circumstances, it may contain or operate on only a portion of a batch. This standard presumes that the unit does not operate on more than one batch at the same time.

7.7.4.4.8 Equipment module level

Physically, the equipment module may be made up of control modules and subordinate equipment modules. An equipment module may be part of a unit or a stand-alone equipment grouping within a process cell. If engineered as a stand-alone equipment grouping, it can be an exclusive-use resource or a shared-use resource. An equipment module can carry out a finite number of specific minor processing activities such as dosing and weighing. It combines all necessary physical processing and control equipment required to perform those activities. It is usually centered on a piece of processing equipment, such as a filter. Functionally, the scope of the equipment module is defined by the finite tasks it is designed to carry out.

7.7.4.4.9 Control module level

A control module is typically a collection of sensors, actuators, other control modules, and associated processing equipment that, from the point of view of control, is operated as a single entity. A control module can also be made up of other control modules. For example, a reader control module could be defined as a combination of several on/off automatic block valve control modules.

Control modules contain the following computing resources for control modules:

- A regulating device consisting of a transmitter, a controller, and a control valve that is operated via the set point of the device
- A state-oriented device that consists of an on/off automatic block valve with position feedback switches that is operated via the set point of the device
- A header that contains several on/off automatic block valves and that coordinates the valves to direct flow to one or several destinations based upon the set point directed to the header control module

7.7.4.5 Process cell classification

7.7.4.5.1 General

Subclause 7.7.4.5 addresses the classification of process cells by the number of different products manufactured in the process cell and by the physical structure of the equipment used in the manufacturing using distributed computing.

7.7.4.5.2 Classification by number of products

A process cell is classified as single product or multi product based on the number of products planned for production in that process cell. A single-product process cell produces the same product in each batch. Variations in procedures and parameters are possible. For example, variations may occur in order to compensate for differences in equipment, to compensate for substitute raw materials, to compensate for changes in environmental conditions or to optimize the process. A multi-product process cell produces different products utilizing different methods of production or control. There are two possibilities:

— All products are produced with the same procedure using different formula values (varying materials and/or process parameters)

— The products are produced using different procedures

7.7.4.5.3 Classification by physical structure

The basic types of physical structures discussed here are single path, multiple path, and network. A single-path structure is a group of units through which a batch passes sequentially (see Figure 89). A single-path structure could be a single unit, such as a reactor, or several units in sequence. Multiple input materials are typically used; multiple finished materials may be generated. Several batches may be in progress at the same time.

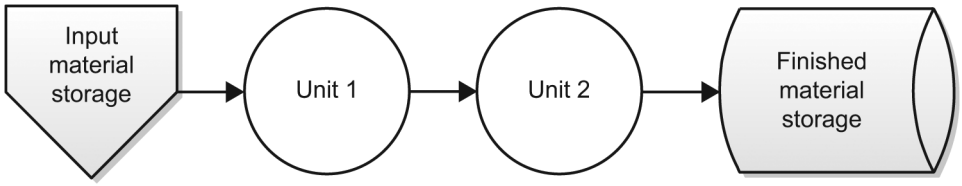


Figure 89—Single-path structure

A multiple-path structure is shown in Figure 90. It consists of multiple single-path structures in parallel with no product transfer between them. The units may share raw material sources and product storage. Several batches may be in progress at the same time. Although units within a multi-path structure may be physically similar, it is possible to have paths and units within a multi-path structure that are of radically different physical design.

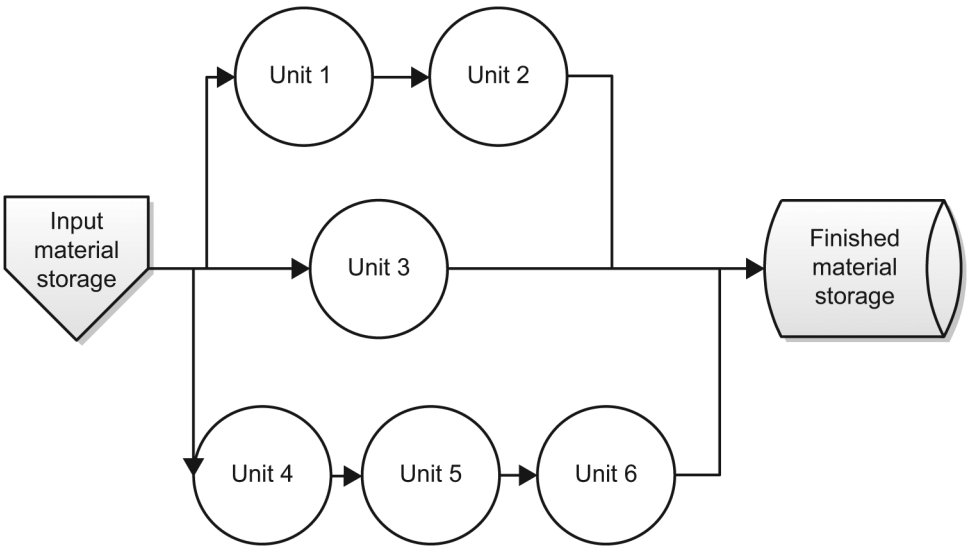


Figure 90—Multiple part structure

A network structure is shown in Figure 91. The paths may be either fixed or variable. When the paths are fixed, the same units are used in the same sequence. When the path is variable, the sequence may be determined at the beginning of the batch or it may be determined as the batch is being produced. The path could also be totally flexible. For example, a batch could start with any unit and take multiple paths through the process cell. The units themselves may be portable within the process cell. In this case, verification of the process connections may be an important part of the procedures. Note that several batches may be in production at the same time. The units may share raw material sources and product storage.

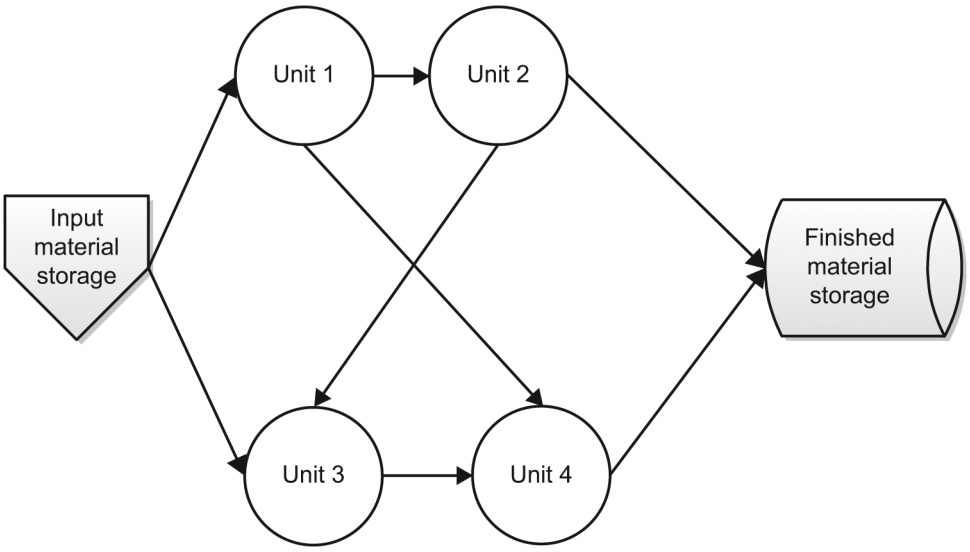


Figure 91—Network structure

7.7.4.6 Batch control concept

7.7.4.6.1 General

The batch control concept addresses the batch processing as part of the Smart Manufacturing needs and to define a consistent way of operating a batch manufacturing plant. When the control types are applied to equipment, the resulting equipment entities provide process functionality and control capability.

The concept of recipes, including the four types of recipes described in this document and the contents of these recipes in terms of the information categories used to describe a recipe. A relationship shall be established between the procedure in a recipe and the control associated with specific equipment entities (equipment control). The concept of collapsibility of the recipe procedure and of equipment control is discussed. Recipe transportability criteria are introduced for the four types of recipes also representing a mechanism of distributed computing.

Production plans and schedules, reference information, production information, allocation and arbitration, modes and states, and exception handling are other needed batch control concepts.

The introduced models and terminology are intended to establish the necessary batch control so that the control functions that are needed to address the diverse control requirements of batch manufacturing can be achieved.

7.7.4.6.2 Structure for batch control

7.7.4.6.2.1 General

A physical model shall be defined for the hierarchy of equipment in the batch manufacturing environment. Three types of control are needed in batch manufacturing:

— Basic control

— Procedural control

— Coordination control

7.7.4.6.2.2 Basic control

Basic control comprises the control dedicated to establishing and maintaining a specific state of equipment and process. Basic control:

— Includes regulatory control, interlocking, monitoring, exception handling, and repetitive discrete or sequential control

— May respond to process conditions that could influence the control outputs or trigger corrective actions

— May be activated, deactivated, or modified by operator commands or by procedural or coordination control

Basic control in a batch environment is, in principle, no different from the control of continuous processes. However, in the batch environment, there may be higher requirements on the ability for basic control to receive commands and to modify its behavior based on these commands.

7.7.4.6.2.3 Procedural control

7.7.4.6.2.3.1 General

Procedural control directs equipment-oriented actions to take place in an ordered sequence in order to carry out a process-oriented task. Procedural control is a characteristic of batch processes. It is the control that enables equipment to perform a batch process.

Procedural control is made up of procedural elements that are combined in a hierarchical manner to accomplish the task of a complete process as defined by the process model. The hierarchy of identified and named procedural elements is illustrated in Figure 92 and consists of procedures, unit procedures, operations, and phases.

7.7.4.6.2.3.2 Procedure

The procedure is the highest level in the hierarchy and defines the strategy for carrying out a major processing action such as making a batch. It is defined in terms of an ordered set of unit procedures. An example of a procedure is “make PVC.”

7.7.4.6.2.3.3 Unit procedure

A unit procedure consists of an ordered set of operations that cause a contiguous production sequence to take place within a unit. Only one operation is presumed to be active in a unit at any time. An operation is carried to completion in a single unit. However, multiple unit procedures of one procedure may run concurrently, each in different units. Examples of unit procedures include:

— Polymerize vinyle chloride monomer (VCM)

— Recover residual VCM

— Dry PVC

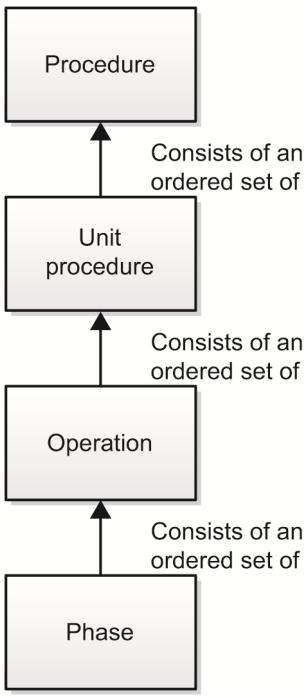


Figure 92—Procedural control model

7.7.4.6.2.3.4 Operation

An operation is an ordered set of phases that defines a major processing sequence that takes the material being processed from one state to another, usually involving a chemical or physical change. It is often desirable to locate operation boundaries at points in the procedure where normal processing can safely be suspended.

Operations include:

- Preparation: Pull a vacuum on the reactor and coat the walls with antifoulant.
- Charge: Add demineralized water and surfactants.
- React: Add VCM and catalyst, heat, and wait for the reactor pressure to drop.

7.7.4.6.2.3.5 Phase

A phase is the smallest element of procedural control that can accomplish a process-oriented task. A phase may be subdivided into smaller parts. The steps and transitions as described in IEC 60848 document one method of defining subdivisions of a phase. A phase can issue one or more commands or cause one or more actions, such as:

- Enabling and disabling regulating and state-oriented types of basic control and specifying their set points and initial output values
- Setting, clearing, and changing alarm and other limits
- Setting and changing controller constants, controller modes, and types of algorithms
- Reading process variables, such as the gas density, gas temperature, and volumetric flow rate from a flow meter, and calculating the mass flow rate through the flow meter
- Conducting operator authorization checks

The execution of a phase may result in:

— Commands to basic control

— Commands to other phases (either in the same or another equipment entity)

— The collection of data

A phase is intended to cause or define a process-oriented action, while the logic or set of steps that make up a phase are equipment specific. Examples of phases include:

— Add VCM

— Add catalyst

— Heat

7.7.4.6.2.4 Coordination control

Coordination control directs, initiates, and/or modifies the execution of procedural control and the utilization of equipment entities. It is time-varying in nature, like procedural control, but it is not structured along a specific process-oriented task.

Examples of coordination control are algorithms for:

— Supervising availability or capacity of equipment

— Allocating equipment to batches

— Arbitrating requests for allocation

— Coordinating common resource equipment

— Selecting procedural elements to be executed

— Propagating modes

The control functions that are needed to implement coordination control are discussed in more detail in Clause 6.

7.7.4.6.3 Equipment entities

7.7.4.6.3.1 General

Suclause 7.7.4.6.2.3 addresses equipment entities that are formed from the combination of equipment control and physical equipment. This combination results in four equipment entities: process cells, units, equipment modules, and control modules. Guidelines for structuring these equipment entities are also discussed.

When the terms *process cell, unit, equipment module,* and *control module* are used, they generally refer to the equipment and its associated equipment control. Whether equipment control in an equipment entity is implemented manually or by way of automation, it is only through the exercise of equipment control that the equipment can produce a batch.

The concept of equipment control being part of an equipment entity is not a statement of the physical implementation of equipment control, but an understanding on a logical basis. However, it is essential that equipment control is identified for a particular equipment entity. This interaction of equipment control and physical equipment is described purposely without any reference to language or implementation. A

framework is defined within which equipment control and physical equipment may be defined and discussed.

7.7.4.6.3.2 Procedural control model/physical model/process model relationship

The general relationship between the procedural control model, the physical model, and the process model is illustrated in Figure 93. This mapping of procedural control with individual equipment provides processing functionality described in the process model.

The concept of equipment capabilities and usage of these capabilities to accomplish processing tasks is a major point of this standard. The procedural control capability of equipment entities is the mechanism that enables this. The procedural control may be entirely defined as part of equipment control, or it may be based on procedural information passed on to the equipment entity from the recipe.

7.7.4.6.3.3 Equipment control in equipment entities

7.7.4.6.3.3.1 General

The control capability possible in the different equipment entities are important characteristics and a main basis for classification of equipment entities. Subclause 7.7.4.6.3.3 addresses the equipment control for the individual equipment entities.

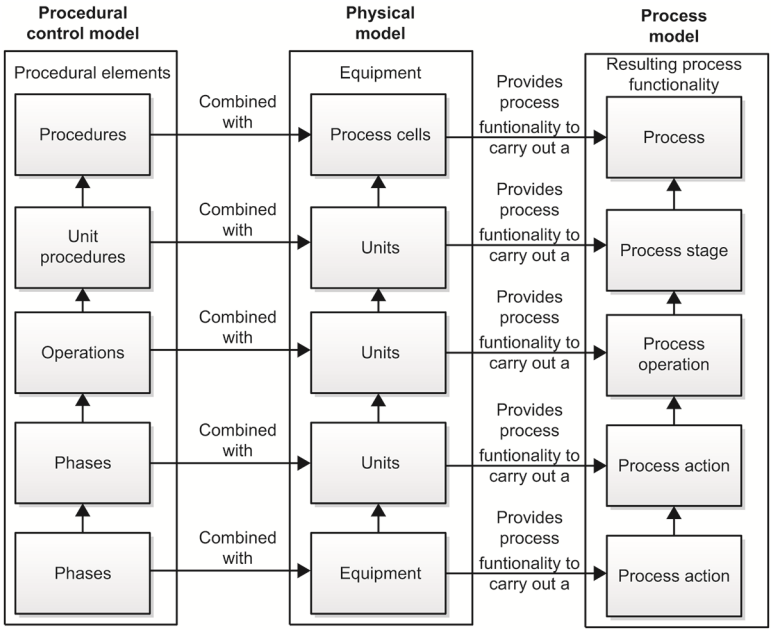


Figure 93—Procedural control/equipment mapping to achieve process functionality

7.7.4.6.3.3.2 Process cell

7.7.4.6.3.3.2.1 General

The process cell is capable of orchestrating all processing activities for one or more batches. It receives recipes containing procedure, parameter, and other information and a schedule containing operational requirements for each batch. It may also need to prepare and monitor equipment or resources not currently involved in batch processing, such as which units are available, what units and piping are going through a clean-in-place (CIP) routine, and what the current inventories of raw materials are.

The complexity of control within the process cell will depend on the equipment available within the process cell, the interconnectivity among this equipment, the degree of freedom of movement of batches through this equipment, and the arbitration of the use of this equipment so that the equipment can be used most effectively.

Equipment control in the process cell may be distributed in the same manner as the physical equipment is subdivided. For example, if the process cell is subdivided into trains, equipment control within the process cell may be distributed among the various trains. Equipment modules and control modules may exist as separate entities under direct control of the process cell.

7.7.4.6.3.3.2.2 Basic control in process cells

The process cell may include basic control that spans several units. For example, an interlock that shuts one unit down may need to be propagated to the upstream units that are feeding this particular unit.

7.7.4.6.3.3.2.3 Procedural control in process cells

The execution of a procedure and initiation of the individual unit procedures is a process cell responsibility. The execution may or may not be integral to the coordination control involved with the movement of batches as described in 7.7.4.6.3.3.2.4.

7.7.4.6.3.3.2.4 Coordination control in process cells

More coordination control is needed in process cells than in the lower-level equipment entities because:

- The process cell may contain multiple units and process multiple batches at the same time. This involves coordinating the execution of a number of different procedures.
- The control of the movement of batches may involve a number of choices between alternate paths. Although these choices may be made via links between units, actual routing may be determined by the process cell.
- Arbitration may be needed at the process cell level to optimize the use of resources, such as shared resources and resources that should be reserved well in advance of the time actually needed.

Examples of coordination control in a process cell include algorithms that:

- Manage the initialization and movement of the batches being processed within the process cell.
- Initiate and/or associate unit procedures, parameters, and other information in individual units in the proper order to cause them to process the product described by the unique combination of schedules and recipes.

7.7.4.6.3.3.3 Unit

7.7.4.6.3.3.3.1 General

Units coordinate the functions of the lower-level entities such as equipment modules and control modules. The primary purpose of equipment control in a unit is control of the processing of the batch that is currently associated with the unit.

7.7.4.6.3.3.3.2 Basic control in units

Basic control in a unit is generally performed by regulatory control and discrete control in equipment modules and control modules within the unit.

7.7.4.6.3.3.3.3 Procedural control in units

Units may include and execute equipment phases, equipment operations, and equipment unit procedures, or they may execute recipe operations and recipe unit procedures passed on to it.

7.7.4.6.3.3.3.4 Coordination control in units

Equipment control in a unit will include a substantially higher level of coordination control than any of the lower-level equipment entities. This may include, for example, algorithms that manage unit and acquired resources; arbitrate requests for services from other units or from the process cell; acquire the services of resources from outside the unit; and communicate with other equipment entities outside unit boundaries.

7.7.4.6.3.3.4 Equipment module

7.7.4.6.3.3.4.1 General

The primary purpose of equipment control in an equipment module is coordination of the functions of other equipment modules and lower-level control modules. An equipment module may be commanded by a process cell, a unit, an operator, or, in some cases, another equipment module.

7.7.4.6.3.3.4.2 Basic control in equipment modules

Basic control in an equipment module is generally performed by regulatory control and discrete control in control modules within the equipment module.

7.7.4.6.3.3.4.3 Procedural control in equipment modules

Equipment modules may execute equipment phases, but they do not have the capability of executing higher-level procedural elements.

7.7.4.6.3.3.4.4 Coordination control in equipment modules

Coordination control in an equipment module includes coordination of its component parts and may include algorithms for propagating modes and for arbitrating requests from units.

7.7.4.6.3.3.5 Control module

7.7.4.6.3.3.5.1 General

Equipment control normally found at this level directly manipulates actuators and other control modules. A control module can direct commands to other control modules and to actuators if they have been configured as part of the control module. Control of the process is effected through the equipment-specific manipulation of control modules and actuators.

Examples of equipment control in control modules include:

— Opening or closing a valve, with confirmation failure alarms

— Regulating the position of a control valve based on a sensor reading and PID control algorithm

— Setting and maintaining the state of several valves in a material header

7.7.4.6.3.3.5.2 Basic control in control modules

Control modules contain basic control. Although this control is normally either regulatory or state oriented, in some cases it is both. It may also include conditional logic. For example, open the valve if the temperature is within limits and the downstream valve is open.

Regulatory control is dedicated to maintaining a process variable or variables at or near some desired value. Complex control strategies such as multivariable control, model-based control, and artificial intelligence techniques may also fit into this category of regulatory control.

State-oriented control refers to setting the state of a piece of equipment as opposed to the state of a process variable or variables. A state-oriented device has a finite number of states. It defines a product-independent processing sequence.

Control modules may contain exception handling.

7.7.4.6.3.3.5.3 Procedural control in control modules

Control modules do not perform procedural control.

7.7.4.6.3.3.5.4 Coordination control in control modules

Coordination control in a control module may include, for example, algorithms for propagating modes and for arbitrating requests from units for usage.

7.7.4.6.3.4 Structuring of equipment entities

7.7.4.6.3.4.1 General

Subclause 7.7.4.6.3.4 addresses the general principles involved in segmenting a process cell into equipment entities that can carry out specified processing activities or equipment specific actions. Total explanation of process segmentation principles is beyond the scope of this standard.

It is important to note that the physical process cell design can greatly influence the implementation of batch control. Minor differences in the physical system can dramatically affect the organization of equipment entities and procedural elements.

All control-related clauses of this standard assume that the process cell in question (both physical equipment and related control activities) has been subdivided into well-defined equipment entities such as units, equipment modules, and control modules. Effective subdivision of the process cell into well-defined equipment entities is a complex activity, highly dependent on the individual requirements of the specific environment in which the batch process exists. Inconsistent or inappropriate equipment subdivisions can compromise the effectiveness of the modular approach to recipes suggested by this standard.

Subdivision of the process cell requires a clear understanding of the purpose of the process cell's equipment. Such understanding allows the identification of equipment entities that work together to serve an identifiable processing purpose.

7.7.4.6.3.4.2 Structuring of process cells

The subdivision of a process cell usually follows the principles listed below:

— The function that any equipment entity serves in product processing should be clear and unambiguous.

— The function performed by the equipment entity should be consistent in terms of processing task, and should be usable for that task no matter what product is being manufactured at a given time.

— Subordinate equipment entities should be able to execute their task(s) independently and asynchronously, allowing the highest level equipment entity to orchestrate the activities of its subordinates.

— Interactions between equipment entities should be minimized. While planned interaction is periodically necessary, each equipment entity should perform its functions while influencing the functioning of other equipment entities as little as possible.

— Equipment entities should have clear boundaries.

— A consistent basis is essential for the definition of equipment entities. An operator subsequently interacting with similar equipment entities should be able to do so naturally and without confusion.

— Necessary interaction between equipment entities is, in so far as possible, coordinated by equipment entities at the same level or at the next higher level.

7.7.4.6.3.4.3 Structuring of units

The definition of a unit requires knowledge of the major processing activities, as well as the equipment capabilities. The following guidelines apply:

— One or more major processing activities, such as reaction or crystallization, may take place in a unit.

— Units should be defined such that they operate relatively independently of each other.

— A unit is presumed to operate on only one batch at a time.

7.7.4.6.3.4.4 Structuring of equipment modules

The definition of an equipment module requires knowledge of specific minor processing activities and equipment capabilities. Equipment modules can carry out a finite number of minor processing activities, such as dosing and weighing, and are typically centered around a set of process equipment. Collections of control modules can be defined as equipment modules or as control modules. If the collection executes one or more equipment phases, then it is an equipment module.

7.7.4.6.4 Recipes

7.7.4.6.4.1 General

Subclause 7.7.4.6.4 addresses the four types of recipes covered in this standard, the five categories of information contained in a recipe, and how this information changes for the different recipe types, as well as the relationship of the control recipe procedure to the equipment procedure. Some guidelines for recipe transportability are also presented.

7.7.4.6.4.2 Recipe types

7.7.4.6.4.2.1 General

Subclause 7.7.4.6.4.2 addresses four types of recipes typically found in an enterprise. A recipe is an entity that contains the minimum set of information that uniquely defines the manufacturing requirements for a specific product. Recipes provide a way to describe products and how those products are produced. Depending on the specific requirements of an enterprise, other recipe types may exist. However, this standard discusses only the general recipe, site recipe, master recipe, and control recipe.

Fundamental to the practical application of recipes is the concept that different parts of an enterprise may need information about the manufacture of a product in varying degrees of specificity, because different recipients of the information use it for different purposes. Therefore, more than one type of a recipe is needed in an enterprise.

It should be noted that whether a particular recipe type actually exists, who generates it, and where it is generated will vary from case to case and from enterprise to enterprise. For example, an enterprise may choose not to implement one or more of the recipe types.

A product may be made in many different arrangements of equipment at many different sites. Recipes that are appropriate for one site or set of equipment may not be appropriate for another site or set of equipment. This can result in multiple recipes for a single product. There should be sufficient structure in the definition of recipes to allow tracing of the genealogy of any given recipe.

The recipe contains neither scheduling nor equipment control. The recipe contains process-related information for a specific product. This permits batch processing equipment to make many different products without having to redefine equipment control for each product. There is a substantial difference between general/site recipes and master/control recipes. The general and site recipes describe the technique, that is, how to do it in principle. Master and control recipes describe the task, that is, how to do it with actual resources.

7.7.4.6.4.2.2 General recipe

The general recipe is an enterprise-level recipe that serves as the basis for lower-level recipes. The general recipe is created without specific knowledge of the process cell equipment that will be used to manufacture the product. It identifies raw materials, their relative quantities, and required processing, but without specific regard to a particular site or the equipment available at that site. It is created by people with knowledge of both the chemistry and processing requirements peculiar to the product in question, and reflects their interests and concerns.

While the general recipe is not specific to equipment or to a particular site, the technology for manufacturing a product will usually have evolved sufficiently beyond the laboratory so that equipment requirements can be described in enough detail to define the type of equipment needed at a particular site or in a particular set of batch plant equipment. The general recipe provides a means for communicating processing requirements to multiple manufacturing locations.

Quantities may be expressed as fixed or normalized values. Equipment requirements are expressed in terms of the attributes needed by the equipment, such as pressure requirements and materials of construction.

The general recipe may be used as a basis for enterprise-wide planning and investment decisions. It may be part of, or referenced by, production specifications and, as such, used for production planning and for information to customers and authorities.

7.7.4.6.4.2.3 Site recipe

The site recipe is specific to a particular site. It is the combination of site-specific information and a general recipe. It is usually derived from a general recipe to meet the conditions found at a particular manufacturing location and provides the level of detail necessary for site-level, long-term production scheduling. However, it may also be created directly without the existence of a general recipe. Such things as the language in which it is written or local raw material differences are accommodated as site-specific variances. It is still not specific to a particular set of process cell equipment. Typically, the site recipe is the output of a local "site-focused" process development function. There may be multiple site recipes derived from a general recipe, each covering a part of the general recipe that may be implemented at a specific site.

7.7.4.6.4.2.4 Master recipe

The master recipe is that level of recipe that is targeted to a process cell or a subset of the process cell equipment. A master recipe can be derived from a general recipe or a site recipe. It can also be created as a stand-alone entity if the recipe creator has the necessary process and product knowledge.

Some characteristics of master recipes include:

- There may be multiple master recipes derived from a site recipe, each covering a part of the site recipe that may be implemented in a process cell.
- The master recipe is sufficiently adapted to the properties of the process cell equipment to help ensure the correct processing of the batch. This is done by combining the functionality of the specific set of process cell equipment with the information from the master recipe.
- In a master recipe, the formula data may be specified as normalized values, calculated values, or fixed values.
- The master recipe may contain product-specific information required for detailed scheduling, such as process input information or equipment requirements.

— The master recipe level is a required recipe level because without it no control recipes can be created and, therefore, no batches can be produced.

— Whether the batch manufacturing equipment is operated manually or fully automatically, the master recipe exists either as an identifiable set of written instructions or as an electronic entity.

7.7.4.6.4.2.5 Control recipe

The control recipe starts as a copy of a specific version of a master recipe and is then modified as necessary with scheduling and operational information to be specific to a single batch. It contains product-specific process information necessary to manufacture a particular batch of product. It provides the level of detail necessary to initiate and monitor equipment procedural entities in a process cell. It may have been modified to account for actual raw material qualities and actual equipment to be utilized. The selection of units and appropriate sizing can be done any time before that information is needed.

Since modifications of a control recipe can be made over a period of time based on scheduling, equipment, and operator information, a control recipe may go through several modifications during the batch processing. Examples include:

— Defining the equipment that will actually be used for the control recipe at the initiation of the batch or when it becomes known

— Adding or adjusting parameters based on an “as-charged” raw material quality or midbatch analysis

— Changing the procedure based on some unexpected event

7.7.4.6.4.3 Recipe contents

7.7.4.6.4.3.1 General

Recipes contain the following categories of information: header, formula, equipment requirements, procedure, and other information. Subclause 7.7.4.6.4.3 provides details regarding these categories. Any significant changes from one recipe type to another are noted.

7.7.4.6.4.3.2 Header

The administrative information in the recipe is referred to as the header. Typical header information may include the recipe and product identification, the version number, the originator, the issue date, approvals, status, and other administrative information. For example, a site recipe may contain the name and version of the general recipe from which it was created.

7.7.4.6.4.3.3 Formula

The formula is a category of recipe information that includes process inputs, process parameters, and process outputs. A process input is the identification and quantity of a raw material or other resource required to make the product. In addition to raw materials which are consumed in the batch process in the manufacture of a product, process inputs may also include energy and other resources such as human labor. Process inputs consist of both the name of the resource and the amount required to make a specific quantity of finished product. Quantities may be specified as absolute values or as equations based upon other formula parameters or the batch or equipment size. Process inputs may specify allowable substitutions, expressed in the same basic form.

A process parameter details information such as temperature, pressure, or time that is pertinent to the product but does not fall into the classification of input or output. Process parameters may be used as set points, comparison values, or in conditional logic.

A process output is the identification and quantity of a material and/or energy expected to result from one execution of the recipe. This data may detail environmental impact and may also contain other information such as specification of the intended outputs in terms of quantity, labelling, and yield.

The types of formula data are distinguished to provide information to different parts of an enterprise and need to be available without the clutter of processing details. For example, the list of process inputs may be presented as a condensed list of ingredients for the recipe or as a set of individual ingredients for each appropriate procedural element in a recipe.

7.7.4.6.4.3.4 Equipment requirements

Equipment requirements constrain the choice of the equipment that will eventually be used to implement a specific part of the procedure.

In the general and site recipes, the equipment requirements are typically described in general terms, such as allowable materials and required processing characteristics. It is the guidance from and constraints imposed by equipment requirements that will allow the general or site recipe to eventually be used to create a master recipe which targets appropriate equipment. At the master recipe level, the equipment requirements may be expressed in any manner that specifies allowable equipment in process cells. If trains have been defined, then it is possible for the master recipe (and the resulting control recipe) to be based on the equipment of the train rather than the full range of equipment in the process cell. At the control recipe level, the equipment requirements are the same as, or a subset of, the allowable equipment in the master recipe. The control recipe may be used to include specific allocations of process cell equipment, such as reactor R-501, when this becomes known.

7.7.4.6.4.3.5 Recipe procedure

7.7.4.6.4.3.5.1 General

The recipe procedure defines the strategy for carrying out a process. The general and site recipe procedures are structured using the levels described in the process model since these levels allow the process to be described in non-equipment specific terms. The master and control recipe procedures are structured using the procedural elements of the procedural control model, since these procedural elements have a relationship to equipment.

The recipe creator is limited to the use of procedural elements that have been, or will be, configured and made available for use in creating a procedure. He or she may use any combination of these procedural elements to define a procedure. Determination as to which of these procedural elements may be part of the procedure is an application specific design decision based on many factors, including the capabilities of the controls and the degrees of freedom appropriate for the recipe creator in a given application.

7.7.4.6.4.3.5.2 General recipe procedure

The procedure information in the general recipe is expressed in three levels of breakdown: process stages, process operations, and process actions. The functionality of these levels corresponds to the functionality of the analogous levels in the process model (see 7.7.4.6.2).

The process stage, process operation, and process action are not constrained by unit boundaries in any real plant. They describe processing activities that others may choose to execute in one or in many different units as the general and site recipe is transformed to run in one or more real plants.

7.7.4.6.4.3.5.3 Site recipe procedure

The procedure information in a site recipe consists of process stages, process operations, and process actions that relate directly to those defined by the general recipe. In general, there is a one-to-one correspondence between the process stages in a general recipe and the process stages in a site recipe, between the process operations in a general recipe and the process operations in a site recipe, and between the process actions in a general recipe and the process actions in a site recipe. As with the other site recipe information, the process stages, process operations, and process actions may be modified to make the recipe site specific.

7.7.4.6.4.3.5.4 Master recipe procedure

The recipe procedure portion of the master recipe may contain recipe unit procedures, recipe operations, and recipe phases.

The creation of a procedure in a master recipe from a procedure in a site recipe may be quite complex. It is essential for the master recipe to contain sufficiently detailed equipment requirements information so that resources may be determined and allocated to create and initiate a control recipe. It is at this recipe level that the set of recipe phases necessary to carry out the intended process actions, process operations, and process stages can be determined.

There may be a 1:1, 1:n, or n:1 relationship between process actions in the general or site recipe and recipe phases in the master recipe, between process operations in the general or site recipe and recipe operations in the master recipe, and between process stages in the general or site recipe and recipe unit procedures in the master recipe (see Figure 94). The actual relationship may depend on the equipment being used.

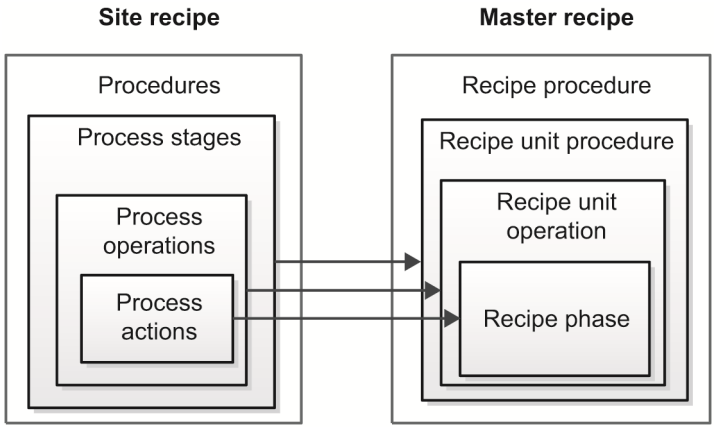


Figure 94—Procedural element relationships in the site recipe and master recipe

Although there is a general similarity between the processing intent of process actions and the processing function defined by recipe phases, there is not necessarily a one-to-one correspondence between the two. One process action may correspond to several recipe phases, and several process actions may correspond to a single recipe phase.

There is a similar relationship between process operations and operations. There are significant differences also. Operations are carried to completion in a single unit in the target equipment while process operations are not constrained to units in any specific facility. A single-process operation might require one or more operations to carry out the processing intent described.

There is a similar relationship between process stages and unit procedures as there is between process operations and operations. Unit procedures are also carried to completion in a single unit in the target equipment while process stages are not constrained by equipment boundaries in any specific facility. A single process stage might require one or more unit procedures to carry out the processing intent described.

7.7.4.6.4.3.5.5 Control recipe procedure

The procedure of a control recipe consists of recipe unit procedures, recipe operations, and recipe phases that relate directly to those defined by the master recipe. At the control recipe creation time, there is a one-to-one correspondence between recipe unit procedures in the master recipe and recipe unit procedures in the control recipe, between recipe operations in the master recipe and recipe operations in the control recipe, and between recipe phases in the master recipe and recipe phases in the control recipe. Changes in the control recipe procedure during the execution may cause it to differ from the master recipe procedure. In a control recipe, as in a master recipe, the procedure is divided along unit procedure boundaries to provide the process cell with the processing requirements of the recipe on a unit-by-unit basis.

7.7.4.6.4.3.6 Other information

Other information is a category of recipe information that may contain batch processing support information not contained in other parts of the recipe. Examples include regulatory compliance information, materials and process safety information, process flow diagrams, and packaging/labelling information.

7.7.4.6.4.4 Control recipe procedure/equipment control relationship

7.7.4.6.4.4.1 General

It is essential to link the control recipe to the equipment control that causes the equipment to operate and make batches, as the control recipe does not contain enough information to operate the process cell. Equipment control is not considered to be part of the recipe. Subclause 7.7.4.6.4.4 addresses the separation between the control recipe procedure and equipment control, the procedural elements that are used in the control recipe procedure and in equipment control, and the mechanism that is used to link the control recipe procedure with equipment control.

7.7.4.6.4.4.2 Control recipe procedure/equipment control separation

It is essential that the control recipe procedure contains at least one procedural element, which is the recipe procedure. It is essential for equipment control to contain at least one procedural element that provides the linkage needed to operate the physical equipment.

In order for control recipes to be executed, recipe procedural elements are linked (by reference) to equipment procedural elements in equipment control. In those cases where the control recipe procedure does not include all levels of procedural elements (recipe unit procedures, recipe operations, or recipe phases), linkage occurs between the lowest level recipe procedural element and an equipment procedural element at the corresponding level of the procedural control hierarchy. For example, whenever recipe operations are the lowest level used in the control recipe procedure, they are linked to equipment operations.

When recipe unit procedures, recipe operations, and recipe phases are not used as part of the control recipe procedure, it may still be helpful to use lower-level equipment procedural elements (some or all) as part of equipment control to provide a modular structure to the equipment control.

7.7.4.6.4.4.3 Control recipe/equipment procedural elements

With recipe procedural elements are typically associated:

— A description of the functionality required

— Formula and other parameter information specific to the procedural element

— Equipment requirements specific to the procedural element

7.7.4.6.4.5 Recipe transportability

Recipe transportability helps ensure that recipe information movement is possible between batch control implementations at the same recipe level. It is essential that the recipe information be understood by each implementation.

The general recipe is transportable from where it was created to any site. The site recipe is transportable, but not to the same extent as the general recipe. It is intended to be used within a specific site and is transportable within that site.

The master recipe is transportable to another process cell, recognizing that the master recipe has been customized for a particular set of process cell equipment. When the master recipe is transported to another process cell, some process engineering analysis may be necessary to either:

— Determine that the new set of process cell equipment is configured similarly so that the master recipe can be used unchanged

— Make the necessary changes so that the modified master recipe will run on the target set of process cell equipment

The control recipe is not transportable.

7.7.4.6.5 Production plans and schedules

Production plans and schedules state the production requirements for the enterprise, sites, areas, and process cells. Since these levels of the physical model operate on different time horizons, a number of different types of plans and schedules are typically needed within an enterprise. A detailed discussion of the various types of plans and schedules is outside the scope of this standard. Only the batch schedule, which satisfies the scheduling needs at the process cell level, is specified.

The batch schedule typically contains more detailed information than production plans and schedules aimed at higher levels in the enterprise. It contains information such as which products to produce, how much of each product to produce, and when they are needed for a specific process cell. It identifies which batches to make, their order, and the equipment to use. This schedule also deals with issues such as personnel requirements, raw material options, and packaging requirements.

Time horizons for the batch schedule are dependent on the speed of the processes and might be measured in minutes, hours, shifts, or days. The batch schedule is based on the specific resources and requirements of the process cell. The possible paths and equipment options are determined at this point. For the batch schedule to be totally meaningful, this schedule would need to be redone any time there is significant variance from the time projections, resource assumptions, or other anticipated factors on which the

schedule was based. For example, the schedule may need updating if an activity is not completed close to the scheduled time. Whether that activity is delayed or whether it is completed ahead of time, the primary concern is whether that activity can affect other schedules in this process cell or other associated process cells.

In a batch schedule shall contain the following information:

- Product name
- Master recipe name
- Quantity (with engineering units) of product
- Equipment and materials permitted to be used, such as path and raw material
- Projected mode of operation
- Order of initiation and priority
- Lot ID (if preassigned)
- Batch ID (if preassigned)
- Projected start time and end time
- Disposition of the finished batch
- Specific customer requirements

A key to efficient batch manufacturing is a comprehensive method that links the various plans and schedules with batch data collection. Batch data collection is the source of timely information that provides feedback so that these plans and schedules can be fine-tuned. During the actual manufacturing of a batch, information is needed in real time so that schedules can be updated within a short time horizon. This update information also allows the user to be kept apprised of the status of lots and/or batches in the schedule.

7.7.4.6.6 Production information

7.7.4.6.6.1 General

Subclause 7.7.4.6.6 addresses information that is generated in the course of production. Information needs to be collected and made available to various levels of the enterprise. The type of information needed varies between different parts of the enterprise. At the enterprise level, for example, summary information may be all that is needed. Examples include the amount of production of a particular product that was achieved at a specific site or at all sites, or how much product is available in inventory. Process development may need detailed processing information on the individual batches in order to perform statistics and comparisons. At the process cell level where the batches are actually produced, there is a need for more detailed information in order to monitor the day-to-day production, to perform adjustments to the schedule, or to adjust processing from batch to batch.

Production information may be batch-specific or it may be common to several or all of the batches produced.

7.7.4.6.6.2 Batch-specific information

The batch-specific information may include:

- A copy of the control recipe that was used to make the batch. This may not be identical to the original recipe because of operator changes, equipment problems, etc. It may be desirable to record both the original recipe and the actual recipe.
- Recipe data: this is actual process data that correspond exactly to the recipe formula such as the amount and type of material charged. This can then be compared to the original recipe.
- Recipe-specified data: this is data whose collection is specified by the recipe. An example is process control information to be trended.
- Summary batch data: this is data such as utilities consumption, equipment run times, and temperatures for the entire batch.
- Operator comments
- Continuous data: this is process data that is collected independently of specific events within the batch with the purpose of giving an accurate history of that measurement.
- Event data: this is data from predictable and unpredictable events, such as recording start and stop times of procedural elements, or unpredictable process or equipment events.
- Operator data: this includes any operator intervention that may affect the processing of the batch (includes operator's ID).
- Analysis data: this is data that is related to off-line measurements or analyses such as measured variables, operator ID, lab technician ID, time of entry of results, and time of sample.

7.7.4.6.6.3 Common (non-batch-specific) batch information

Examples of common (non-batch-specific) batch information include:

- Quality control information: this is information related to monitoring raw material qualities and processing quality.
- Utility systems information: this is process information for equipment such as process heating and cooling that do not produce batches themselves but support equipment that does produce batches.
- Equipment history: this is historical information, such as equipment utilization, calibration, and maintenance.
- Operational documentation: this includes documentation such as production volumes, material consumption summaries, and inventory statistics.
- Materials information: this is typically information such as quality information and packaging and labelling information of input and output materials.

7.7.4.6.6.4 Batch history

All recorded information pertaining to a batch is referred to as the batch history. The batch history will typically include the batch-specific information. Common (non-batch-specific) batch information may be included in the batch history. Since information of this nature typically applies to all or several batches being processed in a process cell, it may be included in the individual batch histories by reference.

In many regulated industries, the record of the batch history is as important as the product itself. Without reliable and accurate batch record keeping, product quality and traceability cannot be ensured. Complete batch record keeping also provides information that is invaluable in process analysis and continuous improvement efforts.

It is essential that batch history be stored in a way that makes it possible to associate the data with that batch (or batches) to which it relates and the processing that has taken place. This means that, in addition to the specific batch identity, it is essential that the data be associated with the actual execution of the appropriate procedural elements, where relevant. The structure of the executed procedure may differ from what is specified in the original recipe because of operator intervention, exception handling, or even planned diversity in the procedure such as changes caused by varying resource limitations.

7.7.4.6.6.5 Batch reports

The extraction of data related to one or more batches is called a *batch report*. The extraction and ordering of the data in a report may vary based on the intended recipient of the batch report. Some of the typical recipients of batch reports and the types of information typically included in their reports are:

- Production management: these batch reports typically provide key economic information on the processing result and resource utilization from multiple batches.
- Product development: these batch reports typically include detailed process information for an individual batch or compare similar data between a group of batches.
- Plant operations: these batch reports typically include the data collected to the current point of processing.
- Quality management: these batch reports typically contain information for documenting batch quality, which may be useful in quality statistics.
- Authorities: these batch reports are typically provided as documentation of production complying with regulations.
- Customers: these batch reports usually relate to documentation concerning product quality and process uniformity.

7.7.4.6.7 Allocation and arbitration

7.7.4.6.7.1 General

Subclause 7.7.4.6.7 addresses mechanisms for allocating resources to a batch or unit and for arbitrating the use of common resources when more than one requester needs to use a common resource at the same time.

Resources such as equipment are assigned to a batch or a unit as they are needed to complete or to continue required processing. Allocation is a form of coordination control that makes these assignments. When more than one candidate for allocation exists, a selection algorithm such as “select lowest duty time” might be used as a basis for choosing the resource. When more than one request for a single resource is made, arbitration is needed to determine which requester will be granted the resource. An algorithm such as “first come/first served” might be used as a basis for arbitration.

In 7.7.4.6.7, allocation and arbitration are discussed in terms of equipment. The concepts apply equally well to other resources, such as operators.

7.7.4.6.7.2 Allocation

The very nature of batch processing requires that many asynchronous activities take place in relative isolation from each other with periodic points of synchronization. Many factors, both expected and unexpected, can affect the time required by one or more of the asynchronous activities from one point of synchronization to the next. For these reasons, and because of the inherent variation in any manufacturing

process, the exact equipment which will be available at the time it is needed is very difficult to predict over a significant period of time. Even though a schedule may have been planned to totally optimize the processing sequence from the standpoint of equipment utilization, it is often desirable to allow alternate equipment to be used if the units planned for a batch are not available when planned. In this case the allocation of units to the batch—the routing or path of the batch—is a decision which is made every time there is more than one path the batch can take through the available equipment.

If more than one unit can acquire or request the services of a single resource, the resource is designated as a common resource. Common resources are often present with complex batch processes. Common resources are often implemented as either equipment modules or control modules. A common resource may be either exclusive-use or shared-use.

If the resource is designated for exclusive use, only one unit may use the resource at a time. A shared weigh tank in a batch plant might be an example of an exclusive-use resource. It can be used by only one reactor at a time. It is essential that the schedule or some other basis for allocation takes this exclusive-use resource into consideration. If a reactor waits for the use of the weigh tank while another is using it, the waiting reactor is idle and nonproductive, which has a negative effect on equipment utilization.

If the common resource is designated as shared, several units may use the resource at the same time. Some shared-use resources in a batch plant might be a process heater serving multiple units at the same time, or a raw material distribution system which is capable of delivering material to more than one unit at a time. If the capabilities of a shared-use resource are limited, then it is possible that the requests for service might exceed the capacity of the resource. In this case, some of the same concerns about allocation which apply to exclusive-use resources also apply to shared-use resources. Care should also be taken so that one unit does not improperly shut off or deactivate a resource while other units are using it.

7.7.4.6.7.3 Arbitration

If there are multiple requesters for a resource, arbitration is essential so that proper allocations can be made. Arbitration resolves contention for a resource according to some predetermined algorithm and provides definitive routing or allocation direction. The algorithm may take various forms such as a predetermined schedule with reservations, a batch priority scheme, or it might rely upon operator judgement. Arbitration may bring with it two distinct issues which affect complexity: resource reservation and preemption.

Reservation allows a claim to be placed on a resource prior to actual allocation. Reservation allows arbitration to be based on future needs rather than allowing the first request for allocation of an idle resource to take precedence regardless of priority.

Preemption occurs when a higher priority batch is allowed to cancel or interrupt the use of a resource assigned to a lower priority batch. When allowed, it is most often associated with allocation of exclusive-use common resource but can apply to allocation of any resource.

7.7.4.6.8 Modes and states

7.7.4.6.8.1 General

Subclause 7.7.4.6.8 discusses the modes and states of equipment entities and of procedural elements. In the preceding subclauses, models describing equipment entities and procedural elements have been defined. In these models, transitions for procedural elements and for equipment entities occur within each hierarchical level. The status of equipment entities and of procedural elements may be described by their modes and states. Modes specify the manner in which these transitions take place; states specify their current status. Other resources, such as materials, may also have states.

7.7.4.6.8.2 Modes

Equipment entities and procedural elements may have modes. Example modes are described in this standard in relation to batch control. The mode of an equipment entity may be based on procedural elements or equipment entities utilizing basic control functions, depending on the main control characteristic of the entity.

This standard uses as examples three modes (AUTOMATIC, SEMI-AUTOMATIC, and MANUAL) for procedural elements, and two modes (AUTOMATIC and MANUAL) for equipment entities. Control modules contain basic control functions and will have AUTOMATIC and MANUAL modes, whereas a unit running procedural control would also have a SEMI-AUTOMATIC mode.

This standard does not preclude additional modes or require the use of the modes defined here. The functionality of the modes presented is felt to be generally useful in most batch applications. By naming the modes and including them in this standard, a defined set of terms is documented that can be used when communicating on batch control issues.

A mode determines how equipment entities and procedural elements respond to commands and how they operate. In the case of procedural elements, the mode determines the way the procedure will progress and who can affect that progression. In the case of a control module, such as an automatic block valve that contains basic control functions, the mode determines the mechanism used to drive the valve position and who/what, such as another device or an operator, may manipulate it to change its state.

For procedural elements, the mode determines the way the transitions are treated. In the AUTOMATIC mode, the transitions take place without interruption when the transition conditions are fulfilled. In the SEMI-AUTOMATIC mode, the procedure requires manual approval to proceed after the transition conditions are fulfilled. Skipping or re-executing one or more procedural elements, without changing their order, is usually allowed. In the MANUAL mode, the procedural elements and their order of execution are specified manually.

For equipment entities containing basic control functions, the mode determines how their states may be manipulated. In AUTOMATIC mode equipment entities are manipulated by their control algorithms, and in MANUAL mode the equipment entities are manipulated by an operator.

Table 47 lists possible behaviors and commands associated with the example modes.

Table 47—Possible implementations of example modes

Mode	Behavior	Command
AUTOMATIC (procedural)	The transitions within a procedure are carried out without interruption as appropriate conditions are met.	Operators may pause the progression, but may not force transitions.
AUTOMATIC (basic control)	Equipment entities are manipulated by their control algorithm.	The equipment cannot be manipulated directly by the operator.
SEMI-AUTOMATIC (procedural only)	Transitions within a procedure are carried out on manual commands as appropriate conditions are fulfilled.	Operators may pause the progression or re-direct the execution to an appropriate point. Transient may not be forced.
MANUAL (procedural)	The procedural elements within a procedure are executed in the order specified by an operator.	Operators may pause the progression or force transitions.
MANUAL (basic control)	Equipment entities are not manipulated by their control algorithm.	Equipment entities may be manipulated directly by the operator.

Equipment entities or procedural elements may change mode. This change can occur if the conditional logic requirements for the change are met by internal logic or by an external command such as one

generated by another procedural element or by an operator. A mode change takes place only when the conditions for the change request are met.

A change of mode in one equipment entity type or procedural element type may cause corresponding changes in other types. For example, putting a unit procedure to the SEMIAUTOMATIC mode may cause all lower-level procedural elements in that unit to go to the SEMIAUTOMATIC mode, or a safety interlock trip may cause several control modules to go to the MANUAL mode with their outputs at minimum value. The propagation can be in either direction, from a higher-level entity to a lower-level entity, or conversely. This standard does not specify propagation rules.

7.7.4.6.8.3 States

7.7.4.6.8.3.1 General

Equipment entities and procedural elements may have states. Example states are described in this standard in relation to batch control. The state completely specifies the current condition of equipment entities or procedural elements. In the case of a valve, the state may be PERCENT OPEN, and in the case of a procedural element, it may be RUNNING or HOLDING.

This standard uses as an example a self-consistent set of procedural states and commands. The number of possible states and commands and their names vary for equipment entities and for procedural elements.

Examples of states for procedural elements include RUNNING, HOLDING, PAUSED, STOPPED, ABORTED, and COMPLETE. Examples of states for equipment entities include ON, OFF, CLOSED, OPEN, FAILED, TRAVELLING, TRIPPED, 35% OPEN, and AVAILABLE. Examples of commands applicable to procedural elements are START, HOLD, PAUSE, STOP, and ABORT.

This standard does not require these states or preclude additional states. The functionality of the states and commands presented is felt to be generally useful in most batch applications. By naming the states and commands and including them in this standard, a defined set of terms is documented that can be used when communicating on batch control issues.

Equipment entities or procedural elements may change state. This change can occur if the conditional logic requirements for the change are met by internal logic or by an external command such as one generated by another procedural element or by an operator.

A change of state in one equipment entity type or procedural element type may cause corresponding changes in other types. For example, putting a unit procedure to the HELD state may cause all procedural elements in that unit to go to the HELD state, or, a safety interlock trip may cause all procedural elements in that unit to go to the ABORTING state. The propagation can be in either direction, from a higher-level entity to a lower-level entity, or conversely. This standard does not specify propagation rules.

A set of procedural states and commands is provided below as a representative example to illustrate one way to define these procedural states and commands. Figure 95 shows the state transition diagram for the three initial states (IDLE, RUNNING, and COMPLETE).

7.7.4.6.8.3.2 Procedural states

Figure 95 provides a list of valid procedural states:

— IDLE: the procedural element is waiting for a START command that will cause a transition to the RUNNING state.

— RUNNING: normal operation.

— COMPLETE: normal operation has run to completion. The procedural element is now waiting for a RESET command that will cause a transition to IDLE.

— PAUSING: the procedural element has received a PAUSE command. This will cause the procedural element to stop at the next defined safe or stable stop location in its normal RUNNING logic. Once stopped, the state automatically transitions to PAUSED.

— PAUSED: once the procedural element has paused at the defined stop location, the state changes to PAUSED. This state is usually used for short-term stops. A RESUME command causes transition to the RUNNING state, resuming normal operation immediately following the defined stop location.

— HOLDING: the procedural element has received a HOLD command and is executing its HOLDING logic to put the procedural element into a known state. If no sequencing is required, then the procedural element transitions immediately to the HELD state.

— HELD: the procedural element has completed its HOLDING logic and has been brought to a known or planned state. This state is usually used for a long-term stop. The procedural element is waiting for a further command to proceed.

— RESTARTING: the procedural element has received a RESTART command while in the HELD state. It is executing its restart logic in order to return to the RUNNING state. If sequencing is not needed, then the procedural element transitions immediately to the RUNNING state.

— STOPPING: the procedural element has received a STOP command and is executing its STOPPING logic, which facilitates a controlled normal stop. If sequencing is not needed, then the procedural element transitions immediately to the STOPPED state.

— STOPPED: the procedural element has completed its STOPPING logic. The procedural element is waiting for a RESET command to transition to IDLE.

— ABORTING: the procedural element has received an ABORT command and is executing its ABORT logic, which is the logic that facilitates a quicker, but not necessarily controlled, abnormal stop. If sequencing is not needed, then the procedural element transitions immediately to the ABORTED state.

— ABORTED: the procedural element has completed its ABORTING logic. The procedural element is waiting for a RESET command to transition to IDLE.

7.7.4.6.8.3.3 Commands

Figure 95 provides a list of valid commands consists of the following:

— START: this command orders the procedural element to begin executing the normal RUNNING logic. This command is only valid when the procedural element is in the IDLE state.

— STOP: this command orders the procedural element to execute the STOPPING logic. This command is valid when the procedural element is in the RUNNING, PAUSING, PAUSED, HOLDING, HELD, or RESTARTING state.

— HOLD: this command orders the procedural element to execute the HOLDING logic. This command is valid when the procedural element is in the RUNNING, PAUSING, PAUSED, or RESTARTING state.

— RESTART: this command orders the procedural element to execute the RESTARTING logic to safely return to the RUNNING state. This command is only valid when the procedural element is in the HELD state.

— ABORT: this command orders the procedural element to execute the ABORTING logic. The command is valid in every state except for IDLE, COMPLETE, ABORTING, and ABORTED.

— RESET: this command causes a transition to the IDLE state. It is valid from the COMPLETE, ABORTED, and STOPPED states.

— PAUSE: this command orders the procedural element to pause at the next programmed pause transition within its sequencing logic and await a RESUME command before proceeding. This command is only valid in the RUNNING state.

— RESUME: this command orders a procedural element that has PAUSED at a programmed transition as the result of a PAUSE command to resume execution. This command is only valid when the procedural element is in the PAUSED state.

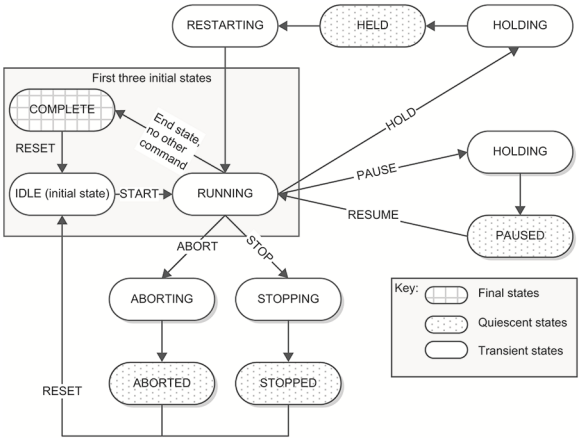


Figure 95—State transition diagram for example states for procedural elements

7.8 Example architecture of oneM2M

7.8.1 General information and key features

7.8.1.1 Introduction

This example architecture of oneM2M is intended to be used by IoT system architects, system designers, and developers to architect, design, and construct a target IoT system. It is also intended for maintainers including production engineers and system administrators who typically run the IoT system once it has been deployed.

oneM2M [B96] is a global standards initiative for machine-to-machine communications and the IoT. It has delivered an umbrella of specifications as open standards which are developed by the members from its global partner organizations. Two major releases have been published (oneM2M published specifications [B97]) and the third release is at the finalization stage (oneM2M latest drafts [B98]).

oneM2M is positioned as the standards for IoT horizontal middleware, e.g., the Common Service Entity (CSE), that can be used across different IoT entities like devices, gateways, and platforms. Common Service Functions (CSF) are provided by the CSEs via RESTful APIs, covering the typical IoT service capabilities like communication management, device management, application management, data management, location, group, security, and semantics. Multiple protocol bindings (e.g., HTTP, CoAP, MQTT, WebSocket) are specified for the same set of API primitives.

7.8.1.2 Relationship to viewpoint catalogue

This example architecture is based on the following viewpoints: conceptual, compatibility, communication, information, access control, collaboration, and computing resource viewpoints:

- The conceptual viewpoint focuses on mapping of the IoT component capability model kind to the oneM2M Common Service Functions which are exposed as RESTful resources.
- The compatibility viewpoint focuses on the model kind of different compatibility levels that oneM2M can provide by different standardized technologies.
- The communication viewpoint shows the mapping of the OSI reference model kind to the oneM2M protocol stack.
- The information viewpoint shows the example mapping of the data model kind and semantic model kind to the oneM2M Home Appliance Information Model (HAIM) and Base Ontology respectively.
- The access control viewpoint shows the mapping of different access control model kinds as well as the identification and authentication model kind to corresponding oneM2M functionalities.
- The collaboration viewpoint shows the mapping of the direct/indirect collaboration model kind to oneM2M architectural deployment and interworking framework.
- The computing viewpoint shows the mapping of the computing resource model kind to the CSE deployment on different oneM2M nodes.

7.8.2 Framed concerns and their stakeholders

oneM2M in general addresses most of the concerns identified in 5.2. The analysis is given in Table 48 in a non-exhaustive manner.

Table 48—IoT Concerns of oneM2M

Concern	Elements of the concerns framed by the architecture framework
Actuation	How to deliver control commands to devices as part of the data delivery service?
Adaptability	How to provide remote upgrade and configuration of a CPS to meet new conditions, needs, or objectives?
Autonomy	How to trigger actions according to pre-defined conditions?
Behavioral	How to provide Common Service Functions in a unified way for multiple application areas?
Collaboration	How to interwork with various external ecosystems?
Communication	How to provide standardized interfaces for the exchange of information between oneM2M entities?
Complexity	How to support interworking with various legacy systems and the variety of interfaces via interworking proxy?
Conceptual	How to develop requirements and architectural elements from use cases?
Confidentiality	How to protect data over oneM2M interfaces?
Controllability	How to affect the property of a physical thing by updating its digitalized resource representations?
Data semantics	How to define high-level common ontology and support semantic annotation, discovery, query, validation, mashup, etc.?
Data volume	How to specify and enforce data storage quota policy?
Deployability	How to provide developer guides for different implementation scenarios with different feature flavors?
Discoverability	How to discover resources and data semantics within the system?
Disposability	How to support offline data buffering for device sleep or out-of-service?
Functionality	How to specify viable Common Service Functions for various applications?
Identity	How to uniquely identify resources, applications, nodes, and CSEs within the system?
Integrity	How to help ensure the message and data integrity in the communications?
Logical time	How to manage time series data?
Manageability	How to provide device management functionality for devices?
Monitorability	How to provide monitoring/logging function for the device, and accounting/charging function for resource usage?
Networkability	How to run oneM2M over IP in a network independent manner, and how to optimize the network usage if the underlying network expose network services (e.g., 3GPP)?
Operations on data	How to provide simple create/read/update/delete operations on data?
Optimization	How to optimize the procedures and resource design to support constrained devices?
Privacy	How to support privacy preference and profile?
Procureability	How to define product profiles for procurement and certification?
Producibility	How to define product profiles and functional features for manufacturing, testing, and certification?
Relationship between data	How to describe the relationships between different data models, information models, and ontologies?
Security	How to provide a complete security support from key negotiation to secure transport association, data encryption, etc.?
Standardization	How to specify the open standards of the common service layer of IoT in a global collaborative way?
States	How to design the oneM2M system in a stateless (RESTful) way?
Synchronization	How to support data synchronization between local and remote entities?
Time awareness	How to associate proper timestamp to messages and resources, and manage the lifecycles based on time?

Stakeholders to consider include:

— Acquirers: use oneM2M specifications as the reference for procurement.

— Assessors: oversee the IoT system’s conformance to oneM2M certification requirement.

— Support staff: provide support to users for the oneM2M product or system when it is running.

— Testers: test the oneM2M system to help ensure that it is suitable for use and conform to the conformance and/or interoperability test specifications.

— Suppliers, developer, builders, production engineers: design, construct, and deploy the oneM2M system.

— Communicators: explain oneM2M system to other stakeholders via its documentation.

— Operators, system administrators, and maintainers: run the oneM2M system once it has been deployed and manage the evolution.

— Owners: derive the benefits of oneM2M when in use.

— Users: define the oneM2M functionality and make use of it.

7.8.3 Architecture view

7.8.3.1 Model kinds

7.8.3.1.1 Conceptual viewpoint: component capability model

Each of the oneM2M system component is denoted as an entity, which can be categorized as Common Service Entities (CSE), Application Entities (AE), and Network Service Entity (NSE). CSEs provides component capabilities to AEs and other CSEs (via Mca, Mcc, and Mcc reference points), and can also invoke the component capabilities provided by the NSEs (via Mcn reference points). CSEs and AEs can be deployed on different types of nodes that stands for various physical equipment (e.g., IoT devices, gateways, platforms). Figure 96 illustrates non-exhaustively the major oneM2M components concepts and relationships at the architecture level. Details are specified in (oneM2M TS-0001 [B99]).

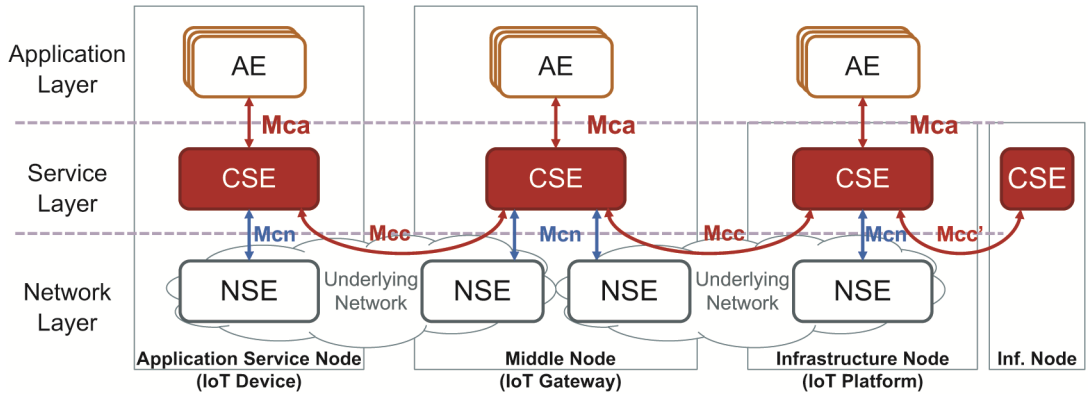


Figure 96—oneM2M architecture overview

oneM2M specifies the capabilities provided by CSEs as Common Service Functions (CSF) shown in Figure 97. Each of the CSF is a collection of service capabilities represent by one or more resource types. Note oneM2M doesn't specify the capabilities of NSE, but leverage them from the underlying network e.g., 3GPP.

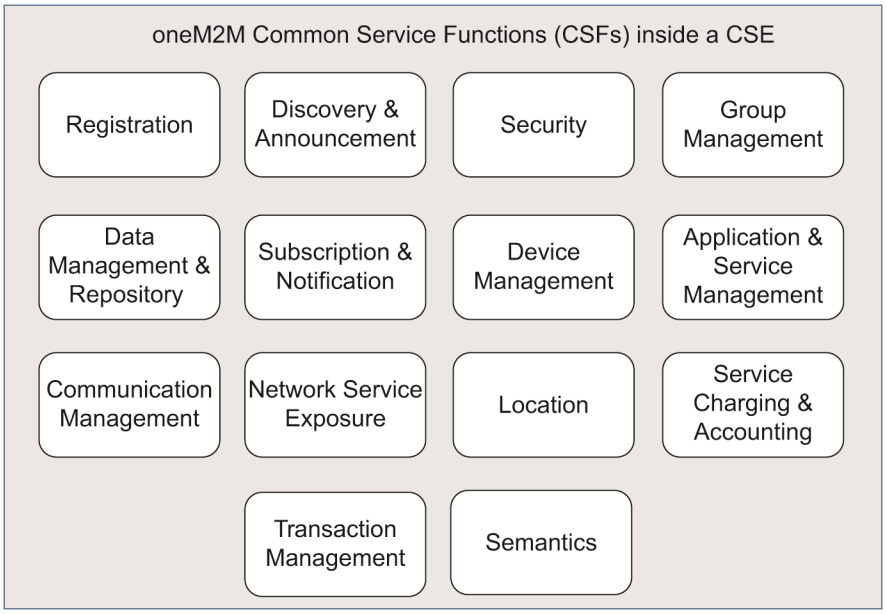


Figure 97—Common Services Functions supported by a oneM2M CSE

The mapping between oneM2M CSFs and the component capability meta-model (defined in 6.6.2.4.6) is given below:

— *Sensing/actuation/data-storing capabilities:* oneM2M provides an abstract Home Appliance Information Model (HAIM) which specifies the common sensing and actuation capabilities of various devices in a modular fashion. All those capabilities are instantiated as data resources (e.g., <container>, <flexContainer>) in oneM2M and managed by the Data Management & Repository CSF. The data resources can store both temporal and historical data with necessary associated quota and volatility policies.

— *Processing capability:* oneM2M specifies abstract device information models (e.g., HAIM) for both device management and service enablement. It also specifies a Base Ontology (BO) with semantic features enabled by the Semantics CSF (see more in oneM2M TS-0034 [B115]). All these features can provide the capability of information/semantic-level processing like discovery, mashup, and interworking.

— *Network capability:* oneM2M provides data transportation between components (i.e., entities) via standardized interfaces (e.g., Mca, Mcc). Data Management CSF and Communication Management/Delivery Handling CSF are mostly involved and can provide synchronized or synchronized, blocking or non-block means to communicate.

— *Supporting capability:* oneM2M specifies the dedicated Security CSF that provides comprehensive features including security provisioning, authentication, authorization, privacy protection, data encryption, and security environment abstraction (oneM2M TS-0003 [B100]). Other CSFs like Device Management CSFs, Service Charging and Accounting CSF, and Registration CSF also provide supporting capabilities.

— *Network interface capability:* oneM2M is designed as independent from underlying networks. It can interface with any IP network. In particular, the Mcn reference point is used to interact with network specific service capabilities (e.g., charging, location, device management, triggering), which can be further exposed to the AEs by the Network Service Exposure CSF.

— *Human UI capability:* oneM2M does not provide the human UI capability.

7.8.3.1.2 Compatibility viewpoint: compatibility-level model

oneM2M provides a comprehensive set of features that can realize different levels of compatibility subject to the deployment choices. The feature mapping analysis is given in Table 49.

Table 49—Compatibility levels supported by oneM2M functionality features

Compatibility level (features)	oneM2M functionality features
Exchangeable (dynamic performance)	oneM2M provides device abstraction and semantic features (e.g., annotation, validation, discovery, query, mashup) to decouple the data and services from physical devices that may be exchangeable.
Interoperable (application functionality, parameter semantics)	oneM2M provides abstract information models (e.g., HAIM) and ontology (e.g., BO) to represent the functionality and semantics of applications, devices, and data so as to make them interoperable across different entities. Based on these, external systems can also interoperate with oneM2M via Interworking Proxy Entities (IPE).
Interworkable (data types)	oneM2M defines specific data types for resources and attributes that are exchanged over the standardized interfaces.
Inter-connectable communication interface	oneM2M specifies a common set of service layer interfaces (e.g., Mca, Mcc) to interconnect different entities.
Co-existent (communication protocol)	oneM2M supports various standardized transport protocols like HTTP, MQTT, CoAP, Websocket and uses IP as its network layer protocol. At the application/data layer it uses XML, JSON, SPARQL, and other standardized protocols.

7.8.3.1.3 Communication model: OSI reference model

Figure 98 shows overall protocol stack of oneM2M mapped to the OSI reference model kind (see 6.6.5.4).

Since oneM2M is positioned as the service middleware standard for IoT, it specifies mainly the Representation Layer (Layer 6) and part of the Application Layer (Layer 7) and the Session Layer (Layer 5), but does not touch the underlying layers (Layer 1 through Layer 4).

At the Representation Layer, oneM2M specifies the resource representations, service layer primitives, device information models, and ontologies using existing languages like XML and JSON.

Based on that Application Layer functionalities are partially realized as the CSFs provided by CSEs.

Furthermore, between the Representation Layer and the Transport Layer, oneM2M specifies different bindings and adaptors so as to be able to transmit the service layer primitives and resources over different protocols (e.g., HTTP oneM2M TS-0009 [B105], CoAP oneM2M TS-0008 [B104], MQTT oneM2M TS-0010 [B106], WebSocket oneM2M TS-0020 [B109]) properly and also leverage existing device management protocols (e.g., OMA DM/LwM2M oneM2M TS-0005 [B102], BBF-TR069 oneM2M TS-0006 [B103]).

Note that although oneM2M is designed with the basic assumption of IP-based Network Layer, it can also work with the non-IP network in the case of using the 3GPP NB-IoT technology (see more in oneM2M TS-0026 [B112]).

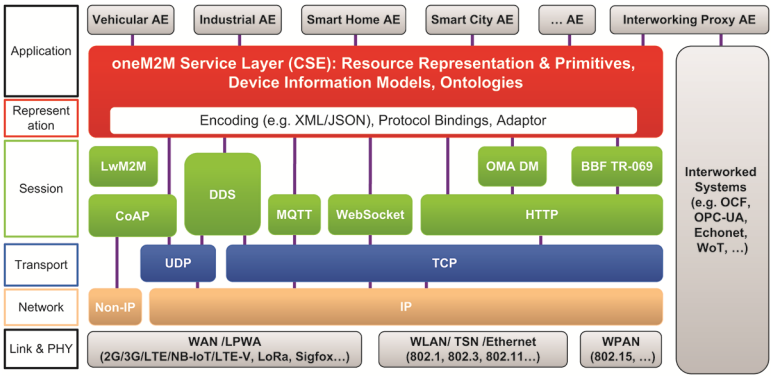


Figure 98 —oneM2M communication stack mapped to OSI reference model

7.8.3.1.4 Information ciewpoint: data and semantic model

oneM2M uses the Smart Device Template (SDT) as the modular meta-model to build concrete Home Appliance Information Models (HAIM) (oneM2M TS-0023 [B110]). There are currently near 50 home device models defined in oneM2M with over 80 reusable module classes. Figure 99 illustrates the SDT 3.0 high-level meta-model and a resource mapping example in oneM2M. The SDT can also abstract heterogeneous device models outside oneM2M (e.g., OCF, OMA GotAPI) so that a native oneM2M application can interact with those devices without dealing with the technology-specific protocols and data models.

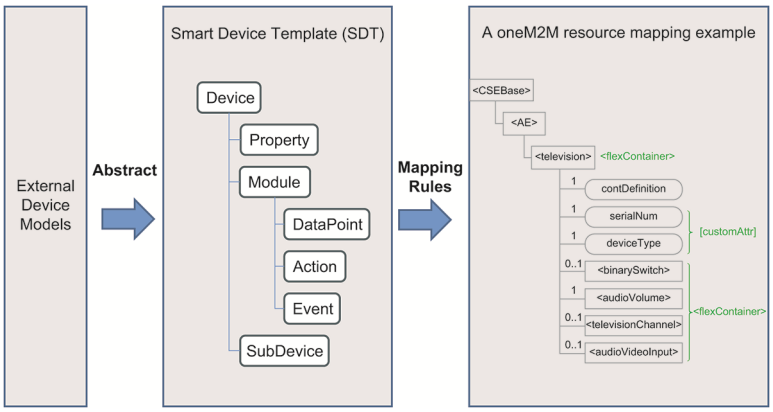


Figure 99 —Smart Device Template and oneM2M resource mapping example

In addition to the SDT-based device modeling, oneM2M also provides semantic-level modeling such as the base ontology. With proper semantic annotation and ontology mapping, heterogeneous external systems and devices can interwork with oneM2M at the semantic level as long as the ontology of the external systems can be mapped with oneM2M base ontology. Examples can be found in (oneM2M TS-0030 [B113]).

7.8.3.1.5 Access control viewpoint: access control model

oneM2M entities (i.e., components, such as AEs and CSEs) and resources (i.e., service capabilities) are all identified by unique identifiers (e.g., AE-ID, CSE-ID, Resource-ID), and access to the oneM2M system and shall be authenticated based on those identifiers and the corresponding credentials.

After authentication, access to any oneM2M resources is subject to the associated access control policy, which defines the access privileges and is also represented as resources and defines. Figure 100 shows the relation between the target resource instances and the associated access control policies.

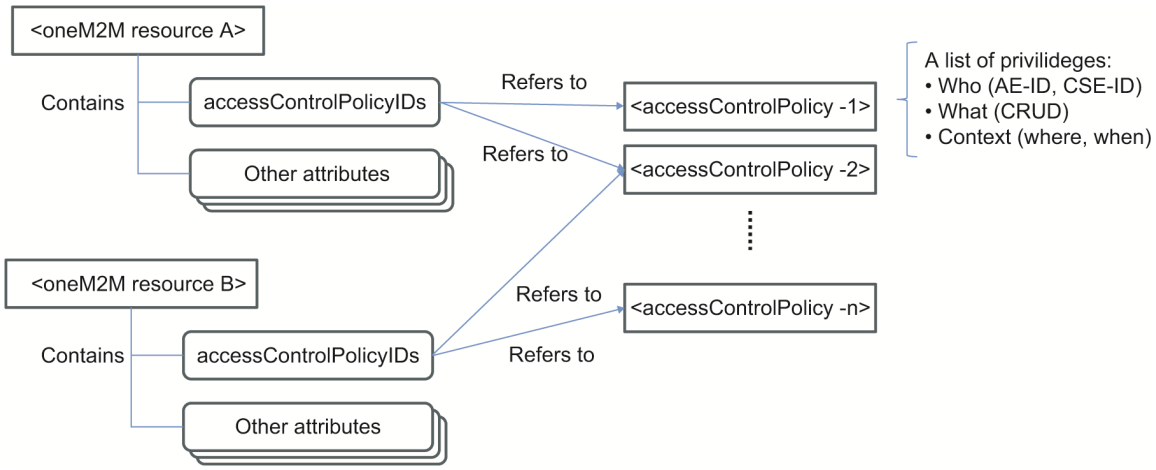


Figure 100—Relation between oneM2M resource instances and access control policies

oneM2M supports not only static access control policy, but also dynamic authorization (using token), role-based authorization and distributed authorization. Details can be found in (oneM2M TS-0003 [B100]).

7.8.3.1.6 Collaboration viewpoint: collaboration model

oneM2M supports both the direct and the indirect collaboration models within its internal system as well as with external systems.

Internally, oneM2M architecture configuration is flexible as shown in Figure 101. It supports both direct and indirect collaboration models between different nodes that may provide different service capabilities and run different applications. A node in oneM2M is a logical representation of a subsystem, of which the physical representation can be a device, a gateway or a platform. Node types defined in oneM2M are:

- Infrastructure node (IN): usually be implemented as a platform
- Middle node (MN): usually implemented as a gateway
- Application service node (ASN): usually implemented as a smart device
- Application dedicated node (ADN): usually implemented as a constrained device
- Non-oneM2M node (NoDN): for an external non-oneM2M entity (device/gateway/platform)

Externally, oneM2M can collaborate with heterogeneous non-oneM2M systems via the interworking proxy entity (IPE). The IPE is a specialized application (running on a oneM2M node) that can talk on both oneM2M and non-oneM2M sides, and is responsible for translating the communications in between. Such collaboration (i.e., interworking in oneM2M terminology) can be realized at different compatibility levels (see 7.8.3.1.2) depending on implementation. oneM2M has specified a set of interworking solutions for external technologies such as LwM2M [B108], OIC/OCF [B111], and OSGi [B116] based on a common interworking framework (oneM2M TS-0033 [B114]).

The collaboration models that are supported by different oneM2M communication relations are described in Table 50.

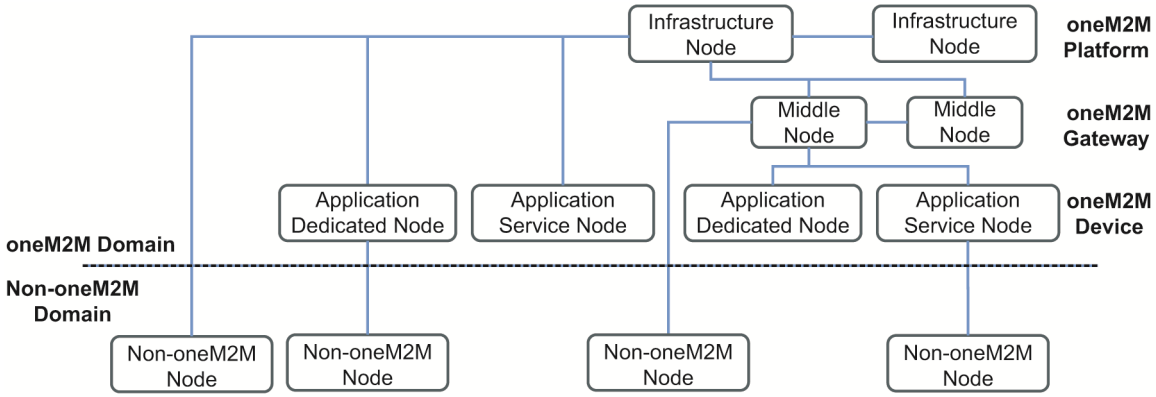


Figure 101—oneM2M architecture configurations

Table 50—Collaboration models supported by oneM2M

Collaboration model	Applicable relations
Direct model	ASN/ADN—MN
	MN—MN
	IN—IN
	MN—IN
	NoDN—IPE
	IPE—ASN/MN/IN
Indirect model	ADN/ASN/MN—MN/IN—ADN/ASN/MN
	NoDN—IPE—ASN/MN/IN
	NoDN A—IPE A—ASN/MN/IN—IPE B—NoDN B

7.8.3.1.7 Computing resource viewpoint: computing resource model

As described in 7.8.3.1.6 and Figure 101, oneM2M architecture configuration can be very flexible depending on the deployment choices, which usually take the computing resource of different nodes into consideration. Table 51 shows an exemplary mapping between the computing component types defined in 6.6.14.3.2 and the oneM2M node types defined in (oneM2M TS-0001 [B99]).

Accordingly, oneM2M supports both the centralized computing resource model (via IN) and partially the distributed computing resource model (between MN-MN, MN-ASN/ADN).

Note that oneM2M currently does not support the distributed computing resource model in a pure peer-to-peer manner, because a node can only communicate with another that has a direct registration relationship with it.

Table 51—Exemplary mapping between the computing component types and oneM2M node types

Computing component type	oneM2M node type
Type A: highly centralized computing components, e.g., data centers, cloud servers, etc.	IN (Infrastructure Node)
Type B: computing components (other that Type A and Type C) connected to the network, e.g., gateways, PLCs, edge-cloud servers, PCs, etc.	MN (Middle Node)
Type C: computing resources embedded in sensors and actuators	ASN (Application Service Node) or ADN (Application Dedicated Node)
Type D: non-computing resources	A sub-set of NoDN (non-oneM2M node) which does not provide any computing resources

7.9 Example architecture of Industrial Value Chain Initiative—reference architecture (IVI-RA)

7.9.1 General information and key features

The Industrial Value Chain Initiative (IVI) is a forum of Smart Manufacturing for connected industries based in Japan and to design a new society by combining manufacturing and information technologies, and for all enterprises to take an initiative collaboratively.

Actively discussing how human-centric manufacturing will change with IoT, IVI aims at building mutually connected system architecture, based on collaboration areas between companies. This means, IVI does not start from the area where an enterprise has its own competitive advantage (which should be kept), but investigates scenarios where companies naturally collaborate, and step by step gathers a broader understanding of more general connection models (reference models), without an urgency to build *the* one general model out of it. This is why they employ the term *loosely defined standard*, as it means an adaptable model instead of a rigid system. A rigid new system would face many challenges in manufacturing environments, which are complex and typically heterogeneous, with a mixture of “old” and “new” elements. A pragmatic, reality-based approach, starting from state-of-the-art today, seems therefore the most suitable to develop the next level of manufacturing. So, using the *loosely defined standard*-based connectivity, IVI works to increase the value for each enterprise by cyber-physical production systems.

7.9.1.1 Relationship to viewpoint catalogue

7.9.1.1.1 Level of relevance of viewpoints

From IVI’s perspective, 13 viewpoints raised in this document and listed in Table 52 are all valid and adaptable to IVI’s scenarios. However, there is a difference in the level of relevance for IVI.

Table 52—List of 13 viewpoints and level of relevance to IRI

Abbreviation	Viewpoint	Relevance to IRI
CNC	Conceptual viewpoint	IVI main
CMP	Compatibility viewpoint	IVI main
LFC	Lifecycle management viewpoint	IVI main
COM	Communication viewpoint	Platform
INF	Information viewpoint	IVI main
FUN	Function viewpoint	IVI main
THM	Threat model viewpoint	Platform
SSM	Security and safe monitoring viewpoint	Platform
ACA	Access control viewpoint	Platform
ADS	Adequate design for required security viewpoint	Platform
PTR	Privacy and trust viewpoint	Platform
COL	Collaboration viewpoint	IVI main
CPR	Computing resources viewpoint	Platform

Most of IVI's activities (IVI main: as shown above) are concentrated in conceptual and information viewpoints, and to some degree compatibility and collaboration viewpoints, because IVI believes these are the center of focus for IVI's Smart Manufacturing development. For other viewpoints, IVI has adopted "Platform" system (Platform: as shown above), where they let the platformers, people supplying "Platform" for IVI take care of their own systems.

From this point of discussion, showing the relationship to each of the viewpoint catalogue of this standard, consisting of 13 viewpoints, which are framed concerns, or requirements of the stakeholders are useful in characterizing the activities of certain project.

7.9.1.1.2 Summary of IVI's activities related to viewpoint catalogue

As a summary of the description given above, the IVI's activities in relationship to viewpoint catalogue may be summarized as follows:

— Conceptual viewpoint model and IVI's AS-IS and TO-BE models

— Lifecycle viewpoint and IVI's exploration, recognition, orchestration, realization (EROR) cycle

— Compatibility viewpoint and IVI's platform reference model

— Collaboration viewpoint and IVI's portable loading unit (PLU)

In 7.9.2 and 7.9.3, the first two points have been described in detail, together with related IVI's activities.

7.9.2 Conceptual viewpoint model and IVI's AS-IS and TO-BE models

7.9.2.1 General

Although there may be some difference arising from the level of abstraction, conceptual viewpoint fits well with main activities of IVI projects, especially the scenario-forming part. It seems intent model kind (6.6.2.4) coincides with what we call TO-BE model in IVI.

7.9.2.2 Models of conceptual viewpoint

This model is reproduced as below in Figure 102. The modeling of the "intent" coincides with IVI's AS-IS and TO-BE models.

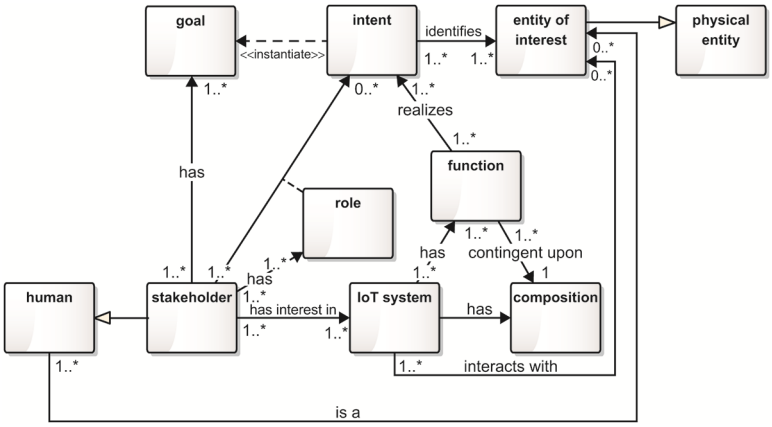


Figure 102—Class diagram of the intent model kind

7.9.2.3 AS-IS and TO-BE models in IVI

In IVI, they have developed a modelling tool for inputting these AS-IS and TO-BE models, and a repository to store these models. This is to assist input of a scenario, and also help standardizing these models, as well as help referring scenarios for future reference.

AS-IS model and TO-BE model

In the activity layer of the IVRA, contents of an SMU are described in the level of activities conducted by individual actors. Two types of models—AS-IS model and TO-BE model—are used for describing concrete activities in SMUs.

AS-IS model

For both the cyber world and the physical world, existing "things" and real "occurrences" in the activity layer can be picked up and described according to a particular problem recognition. A model described in such a way is referred to as an *AS-IS model*. An activity scenario described as an AS-IS model is referred to as an *AS-IS scenario*. An AS-IS scenario is what expresses the current way of working. The real situation is expressed as it is by intention in order to enable discussions on its good/bad points and where problems lie.

The purpose of an AS-IS model is to clarify the *problem to be concerned*. A *problem to be concerned* means a state the situation in reality is different from a desired state. For this purpose, the real situation should be visualized and shared among stakeholders. In general, a problem is described in the form of "XX is YY."

TO-BE model

Likewise, things that should exist and occurrences that should happen in reality are described as a desired state of the future. This model is called a *TO-BE model*. An activity scenario described as a TO-BE model is referred to as a *TO-BE scenario*. TO-BE is what expresses how a situation should be. It does not mean an ideal state. A TO-BE scenario describes a status which is expected to be realized by utilizing digital technologies such as IT and IoT.

The target of a TO-BE model is to clarify the problem to be solved. A *problem to be solved* means actions that should be taken to bring the reality to a desired state. For that, how the situation should be is described and means to achieve that are clarified. Generally it is written in the form of "XX will be made YY".

7.9.2.4 Method of activity scenario modeling in IVI

Here, activity scenarios are cut out from operation flows as meaningful bundles and used as contents to write models. A scenario has the same meaning as that for a play or a novel. A scenario always has a writer who describes it based on his/her intention, but the contents are composed of elements existing (or likely to exist) in reality.

Activity scenarios are easy to understand and memorable for listeners and readers as stories unfold from a perspective of an actor who appears there. Expressing manufacturing activity models in the activity layer as scenarios is effective for building consensus and communicating accurately, since it helps stakeholders understand the contents.

Subclause 7.9.2.4 introduces elements composing activity scenarios. First, there are four elements in the physical world which corresponds to production sites.

Actor

An actor conducts activities related to production by autonomously judging. Not only workers in charge but also automated machines can be regarded as actors. A person in charge of a work in each scene is defined by his/her role. Actors are named in a manner the role (function) is obvious instead of using proper names.

Activity

An activity is a unit of work conducted by an actor. It is defined as a unit of work whose output can be taken over by another actor. In other words, if an activity is stopped in the middle and the remaining part can be continued by a substitute actor, it is regarded as a single set of activity.

Thing

A thing is a visible object existing physically. It exists at a point in the physical space at a certain point of time. All products, parts, equipment, and jigs in a manufacturing site can be things, but here it is an object of a concrete operation in an activity.

Information

Information is what expresses contents needed by actors to make some decision. In reality, physical things such as paper, boards, and display devices become media. Here information means contents shown on them.

On the other hand, the four elements in the cyber world are defined in Figure 103, and as follows:

Service

When an activity—which used to be conducted by an actor—is done in the cyber world, a part of it is executed by an alternative existence. This is a service. It is embodied by a software moving in a platform by using digital technologies.

Process

A process is a unit of function constituting a service which corresponds to an activity in the physical world. The input and output of a process are data. The function is defined by the states before and after.

Data

Data are units existing in the cyber world which express contents describing things and information in the physical world. In practice, a unit of data is composed of multiple attributes which have values. It becomes an input or an output of a process, and in some cases data can be correspondent to things or information.

Condition

In the cyber world, a category of a condition is decided depending on values that data attributes have. Thus it changes from time to time. Categories of conditions are utilized in several ways such as logics inside various processes, execution of processes, and a trigger to activate an alert to the physical world.

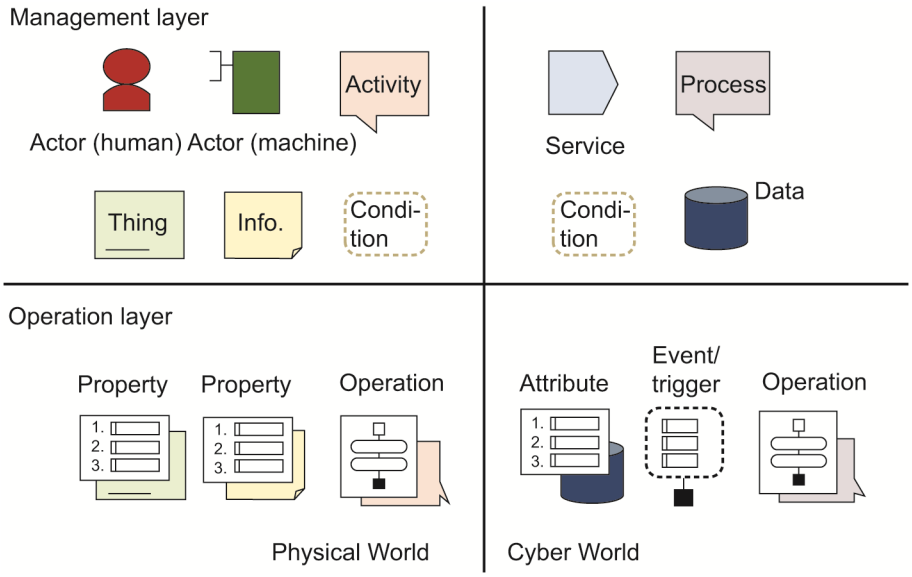


Figure 103—Icons of scenario defining elements

Most of elements constituting activity scenarios belong to the activity layer of the IVRA, but a part of them also cover the operation layer. The following are elements in the operation layer in Figure 104.

Operation

An operation is a unit expressing contents of an activity of an actor in the physical world or a process in a service defined in the cyber world. It is possible to differentiate the term by referring to it as an *activity operation* in case of the physical world and as *process operation* in the cyber world.

Property

A property is a unit that expresses a concrete content of a thing or information in the physical world. In some cases, it is specified as a thing property or an information property. Contents of properties are modified by activity operations. For example, when a thing is lifted, a value of the property of location (altitude) is changed.

Attribute

An attribute is a constituent element to express contents of data in the cyber world. It is also called *data attribute*. For a unit of data one or more attributes are defined. The data are concretized by values of attributes. A value of an attribute is set or changed by a process operation. A condition category is changed by a combination of attribute values.

Event/trigger

An event is an occurrence defined beforehand. It is specified in a condition or a process. For example, it includes a phenomenon that temperature exceeded a certain threshold defined in the cyber world. Among events, ones which start certain processes are referred to as triggers.

Connection of the cyber world and the physical world

A cyber-physical system is a system in which the physical world, such as production sites, and the digitalized cyber world are harmoniously combined. In fact, digital technologies and network technologies have been used for years on factory floors. In addition to digitalization by data input conventionally done by workers, IoT has made it possible to create data directly from things, as well as send data directly from network to things. Figure 104 shows a cross-section of an activity scenario in which the physical world—consisting of actors, activities, things, and information—and the cyber world—with data and service—are interconnected.

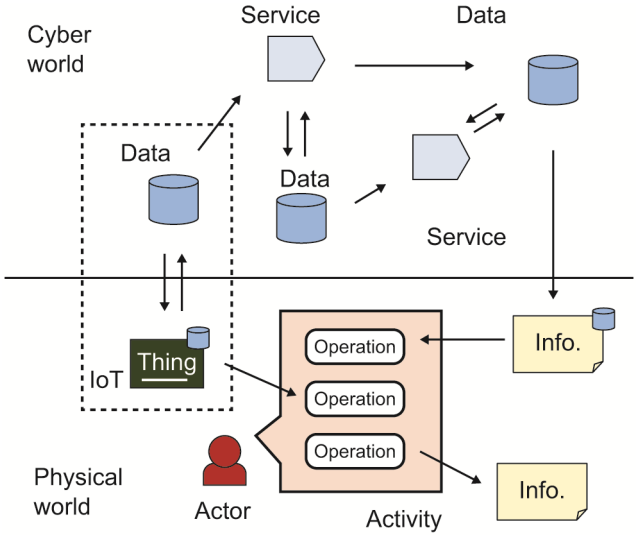


Figure 104—Cyber and physical connection

Things and information in the physical world that are connected to data in the cyber world are distinguished by attaching digital marks (small icons indicating data). Information with a digital mark is equal to a device such as a display device. On the other hand, a thing with a digital mark is a so-called "IoT device" characterized by sending data directly from the thing on the factory floor to the cyber world.

7.9.3 Lifecycle viewpoint and IVI's exploration, recognition, orchestration, realization (EROR) cycle

7.9.3.1 General

Here, the lifecycle viewpoint is shown more or less as a one-way path toward concept to retirement, but from IVI's perspective, it is shown in more enhanced way.

Plan, do, correct, act (PDCA) is believed to be the essence of *kaizen* activities, and they see the "gaining experience in one project and using that experience to the same project, or to the next project" is the important factor.

7.9.3.2 Model kind: lifecycle sequence

Lifecycle sequence displayed in Figure 105.

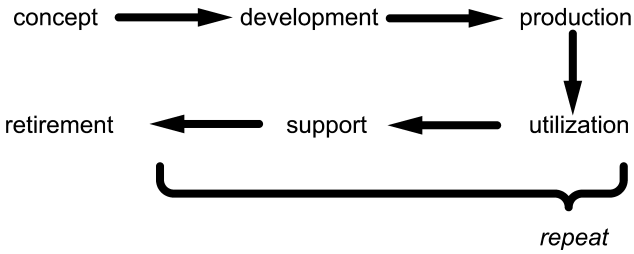


Figure 105—Example of lifecycle sequence as per ISO/IEC/IEEE 15288:2015 [B66]

7.9.3.3 IVI's EROR cycle

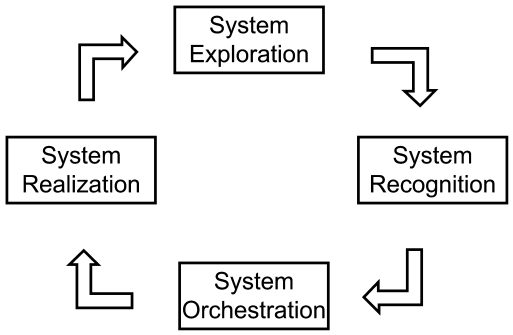


Figure 106—EROR cycle for evolution

Four stages for autonomous evolution

A Smart Manufacturing Unit (SMU) is a unit for Smart Manufacturing that thinks by itself and autonomously changes. It defines its own current condition, problems, and tasks as well as how it should become. Based on that, it evolves by modifying its structure or composition elements. A PDCA cycle in the activity view of SMU is a *kaizen* cycle that is originated in a production site. This cycle is suitable for solving relatively small problems. In case of a larger change, such as a change in organization structure or a workflow, an EROR cycle is executed.

An EROR cycle to realize Smart Manufacturing starts from the current system at all times and are evolutionary. In other words, a system existing at a point in time is referred as the origin when a new system for connected manufacturing is designed and embodied. This cycle is executed repeatedly.

An EROR cycle is composed of four steps: exploration, recognition, orchestration, and realization. A cycle completes usually in one to three months and at most one year.

Stage 1: system exploration

In the stage of system exploration, fundamental issues such as where a problem lies and what the problem is are discussed. First, the range or scope of a targeted problem is set.

Then through conducting as many hearings as possible from actual persons in charge and persons concerned, the real situation—such as who has a problem where it is grasped concretely—and the contents

are arranged. A problem structure revealed from that and the nature of a problem, which can be detected from such a structure, are discussed. For example, brainstorming and the KJ method (affinity diagram) are effective means.

Stage 2: system recognition

In the stage of system recognition, persons concerned need to mutually understand the situation of the reality correctly. Whereas a concern is an interpretation of the real situation, the actual activities in the situation where the concern arises are expressed as an activity scenario here, in order to share an understanding of the matter. An activity scenario shows a picture in which multiple actors conducting operations in a manufacturing site perform various activities through exchanges of things and information.

Stage 3: system orchestration

In system orchestration, a problem to be solved is clarified by expressing how a manufacturing organization or system should be as an activity scenario based on the problem in the real situation. In comparison to an activity scenario for the current situation (AS-IS scenario), useless parts are eliminated from information flows of a scenario for a desired state (TO-BE scenario) and added value is increased by automation and adoption of cyber technology. Means to achieve such targets or goals are also discussed in the stage.

Stage 4: system realization

Current flows of operations or ways of working have to be changed in order to realize an activity scenario of a desired state (TO-BE scenario). At the same time, uselessness is eliminated and efficiency is enhanced by digitalization of traditional activities and taking advantage of platforms that effectively utilize the cyber world. By gradually expanding the world connected with digital technology, new operation flows—which used to be impossible to realize—will be created and the added value will be drastically enhanced.

Annex A

(informative)

Examples of system collaboration based on collaboration viewpoint

A.1 Introduction

This annex introduces examples of systems based on a collaboration viewpoint that addresses the interaction of systems that belong to different application areas. As a framework of the system including the collaboration viewpoint, a model kind named s*ymbiotic-autonomous decentralized system (S-ADS)* is introduced. All other system collaborations can be looked upon as having simple symbiotic viewpoints.

A.2 Symbiotic ADS

A.2.1 Concept

Symbiotic ADS is a model kind for system collaboration based on a symbiotic ecosystem (IEC White Paper 2015 [B52]). In the symbiotic ADS, several ADSs residing in, for instance, Smart Manufacturing domain, Smart Grid domain, and Intelligent Transport Systems domain can readily be connected. The symbiotic ecosystem also provides an environment for mutually accommodating the use of limited resources between or among multiple ADSs, each of which has to determine autonomously whether or not it can provide accommodations without significantly harming its availability to reach its own objectives (IEC White Paper 2015 [B52]).

A.2.2 Functional model

Subclause A.2 introduces the symbiotic ADS architecture and the functional model of symbiotic ADS as one example of a system of systems. Figure A.1 shows the overview of the symbiotic ADS architecture. As explained in 6.6.13.3.1.1 and Figure 63, the collaboration viewpoint has two collaboration models which are the direct collaboration model and the indirect collaboration model. The symbiotic ADS architecture adopts the indirection collaboration model. The symbiotic ADS architecture consists of systems and a Collaborative Platform.

In the symbiotic ADS architecture, each system is defined as the ADS Platform with operational technology (OT) systems, information technology (IT) systems, and an ADS function. OT systems are industrial control systems for manufacturing, transportation, and utilities such as supervisory control and data acquisition (SCADA) in the Smart Manufacturing domain. IT systems are management systems for manufacturing processes, resources, and business operations such as manufacturing execution software (MES) and enterprise resource planning (ERP) software in the Smart Manufacturing domain.

The ADS function manages the resources of the system and exchanges information with the Collaborative Platform. The ADS function collects the data of OT systems like operational logs and sensor data of machines, and the data of IT system like business data and progress data. The function also sends the resource information that is shared with other systems to the Collaborative Platform.

The Collaborative Platform has a data repository for system collaboration to store the resource information from systems. The Collaborative Platform provides an analytics function and action plan creation function for resource optimization among systems. The Collaborative Platform sends feedback information to the ADS function in each system. The ADS function controls OT systems and IT systems to apply the action plan created by the Collaborative Platform.

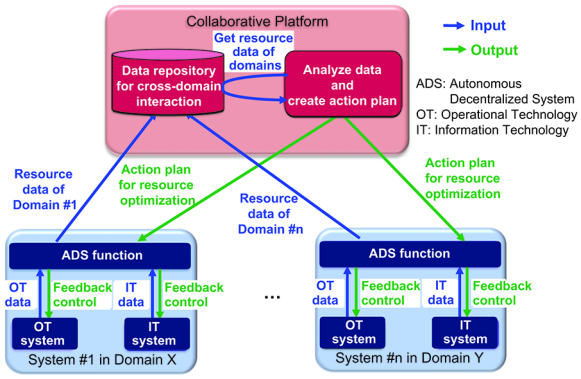


Figure A.1—Overview of symbiotic ADS

Figure A.2 shows the functional model of symbiotic ADS. In each system, OT systems and IT systems are connected to an ADS function. The ADS function collects data from OT systems and IT systems, and manages the resources and operations of devices and facilities in the system. The ADS function also sends the resource information to the Collaborative Platform.

The Collaborative Platform executes the resource management between systems based on the shared resource information.

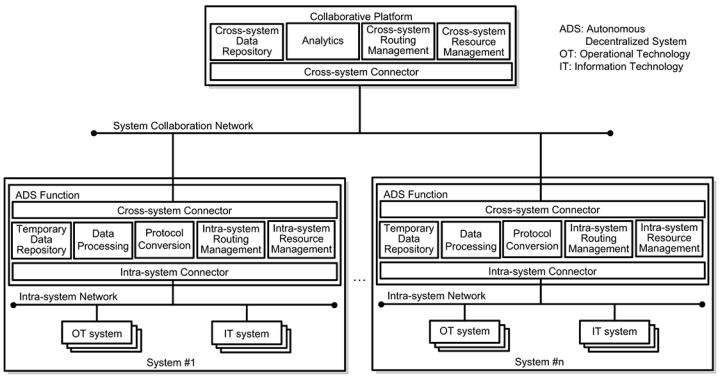


Figure A.2—Functional model of symbiotic ADS

Table A.1 shows the functions of the ADS at each system. Table A.1 indicates a function name, collaboration viewpoint's concerns (see Table 33) solved by this function, and technical requirements for the function.

Table A.1—Function list of the ADS at each system

#	Function name	Collaboration viewpoint's concern to be solved	Technical requirement
1	Intra-system connector	Networkability: Can systems and the Collaborative Platform connect other systems easily and reliably?	Interface for collecting data from an intra-system network like wireless protocols and industrial network protocols
2	Cross-system connector	Collaboration: Can systems and the Collaborative Platform have a mechanism or a common interface for data exchange among several systems? Security: How can each system guard against improper modification of the system, and include ensuring non-repudiation and authenticity? Autonomy: Can systems connect to other systems even after entering each domain operation into service?	Interface in a system collaboration network like Websocket, REST, MQTT, CoAP, HTTP, ADS-net, and OPC UA Security functions like firewall, virus detection, and so on Interface independent from a intra-network
3	Temporary data repository	Relevance: Can systems recognize the meaning of data collected from each system and translate one system's semantics to other system's semantics? Privacy: How can each system protect its local information from other systems?	Temporary data store for data conversion Secure data management like data encryption
4	Data processing	Relevance: Can systems recognize the meaning of data collected from each system and translate one system's semantics to other system's semantics? Privacy: How can each system protect its local information from other systems?	Data modeling and context provisioning for system collaboration Data abstraction for protecting private data
5	Protocol conversion	Networkability: Can systems and the Collaborative Platform connect other systems easily and reliably? Collaboration: Can systems and the Collaborative Platform have a mechanism or a common interface for data exchange among several systems?	Protocol conversion for a system collaboration network Communication security like encryption, authentication, authorization, and accounting
6	Intra-system routing management	Responsibility: Can systems and the Collaborative Platform identify systems to control and manage the operation of each system?	Routing for intra-system management
7	Intra-system resource management	Responsibility: Can systems and the Collaborative Platform identify systems to control and manage the operation of each system?	Resource management in the system

Table A.2 shows the function list at the Collaborative Platform. Table A.2 indicates a function name, concerns solved by this function, and technical requirements for the function.

Table A.2—Function list at the Collaborative Platform

#	Function name	Concern to be solved	Technical requirement
1	Cross-system connector	Collaboration: Can systems and the Collaborative Platform have a mechanism or a common interface for data exchange among several systems? Security: How can each system guard against improper modification of system and include ensuring non-repudiation and authenticity?	Interface in a system collaboration network like Websocket, REST, MQTT, CoAP, HTTP, ADS-net, and OPC-UA System authentication
2	Cross-system data repository	Data semantics: How can systems share the meaning of each system's data? Identity: Can systems and the Collaborative Platform recognize systems and data? Evolvability: How can an autonomous system evolve or be functional with newer technology to collaborate with other autonomous system?	Data identification Data authenticity check Scalable design to extend semantics
3	Analytics	Cost: How can each system reduce cost by sharing knowledge and resources between systems? Utility: How can each system enhance operation efficiency by sharing knowledge and resources between systems? Evolvability: How can an autonomous system evolve or be functional with newer technology to collaborate with other autonomous system?	Simulation function Evaluation function Scalable design to extend analytics function
4	Cross-system resource management	Optimization: How can systems have a mechanism and an algorithm for sharing resources optimally like devices, calculation resources, energy and human resources among several systems? Evolvability: How can an autonomous system evolve or be functional with newer technology to collaborate with other autonomous system?	Management of resources of systems connected to the Collaborative Platform Optimization algorithm Scalable design to extend resource management algorithm

5	Cross-system routing management	Responsibility: Can systems and the Collaborative Platform identify systems to control and manage the operation of each system?	Routing for information exchange between systems

A.3 Examples

A.3.1 System collaboration between Smart Manufacturing domain and Smart Grid domain

This subclause introduces the energy interchange between Smart Manufacturing domain and Smart Grid domain as a use case of system collaboration.

Figure A.3 shows the use case of energy interchange between Smart Manufacturing domain and Smart Grid domain. Conventionally, to realize energy management between Smart Manufacturing domain and Smart Grid domain, a developer has to design the collaborative system from the start and develop the dedicated API as shown in Figure A.3 part (a). Therefore, the collaborative system is fixed, and it is expensive to add other systems into the system.

Figure A.3 part (b) shows the energy interchange by symbiotic ADS. The feature of symbiotic ADS is to create the platform for system collaboration even after entering each system operation into service. In Figure A.3 part (b), initially, Smart Manufacturing domain and Smart Grid domain work autonomously and execute energy management internally. Then, by the cross-system connector in the ADS function, these two systems send the energy demand and supply data to the Collaborative Platform. The two systems also send the operation data of machines in factories or power plants to the Collaborative Platform.

The Collaborative Platform creates an energy reallocation plan by analyzing shared data, and sends feedback information to each application system in order to adjust imbalances between energy demand and energy supply.

Based on the feedback information, each application system executes operation control of machines or power plants and realizes energy interchange.

In this way, by connecting different application systems at the Collaborative Platform, the symbiotic ecosystem can make optimal allocation of energy consumption and supply.

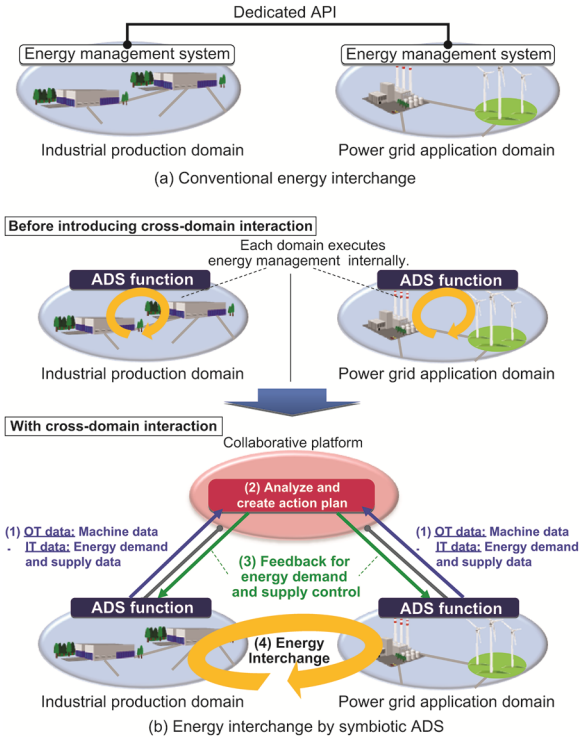


Figure A.3—System collaboration between Smart Manufacturing domain and Smart Grid domain

Figure A.4 shows the relationship between the function model shown in Figure A.2 and the system collaboration use case shown in Figure A.3 part (b).

In each system, the function collects OT data and IT data from factories or power plants via the intra-system connector, and stores collected data in the temporary data repository. Next, the function converts collected data into the data for sharing at the Collaborative Platform by the data-processing function and the protocol conversion function. Then, the function sends the data to the Collaborative Platform via the cross-system connector.

The Collaborative Platform analyzes the shared energy demand and supply data by the analytics function, and creates an action plan for energy interchange. The Collaborative Platform sends the feedback information to each system via the connecting connector.

Each system changes the energy management at the intra-system resource management based on the received feedback information. Then, each system controls factories or power plants internally by the intra-system routing management.

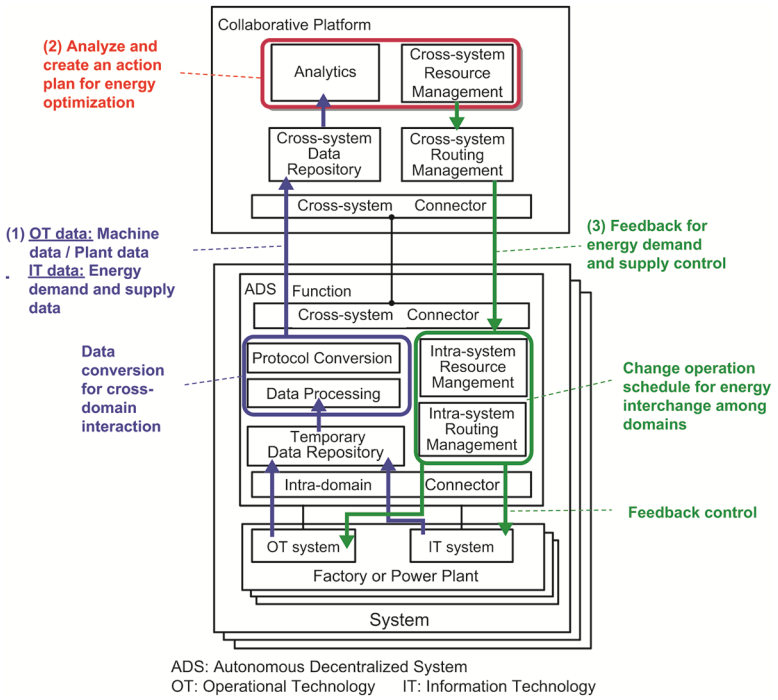


Figure A.4—Relationship between the functional model shown in Figure A.2 and this system collaboration use case shown in Figure A.3 part (b)

A.3.2 Supply chain optimization in Smart Manufacturing domain

This clause introduces the supply chain optimization in Smart Manufacturing domain as a use case of system collaboration. In this use case, the system collaboration means the interaction among several kinds of factories.

Figure A.5 shows the supply chain optimization in Smart Manufacturing domain. This figure shows automobile production as an example. Factory A provides car bodies, Factory B provides tires, Factory C and D provide engines, and Factory X assembles automobiles. Conventionally, each factory executes its task based on the prepared production plan as shown in Figure A.5 part (a). When a certain component factory has a sudden breakdown, it causes the degradation of the productivity in the supply chain.

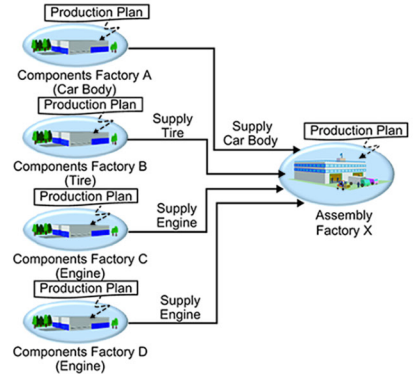
Figure A.5 part (b) and part (c) show the system collaboration by symbiotic ADS. These factories are connected to the Collaborative Platform. Each factory sends machine operation data as OT data and progress reports of the production line as IT data to the Collaborative Platform. The Collaborative Platform creates a new production plan by simulating the manufacturing process based on shared data, and sends feedback information to each factory.

Figure A.5 part (b) shows the simulation of automotive production volume. During the early days of the month, the Collaborative Platform simulates the monthly production volume at the end of the month. The Collaborative Platform predicts the production outage due to a machine failure, and predicts that the factories cannot achieve the target volume. To recover productivity, the Collaborative Platform executes the re-planning of production such as adding temporary machines or workers into some bottleneck processes impacted by the machine failure. Then, the Collaborative Platform sends the feedback information to each factory.

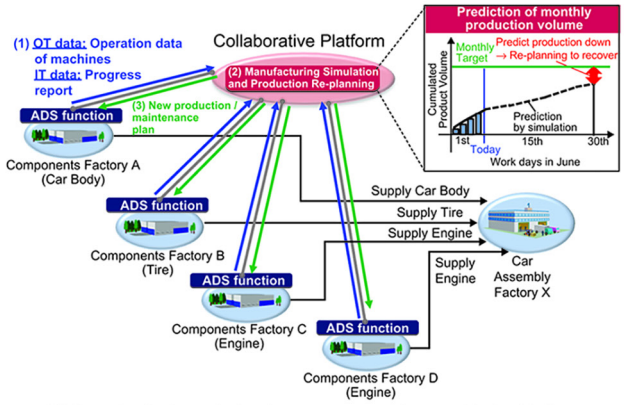
Figure A.5 part (c) shows an example of the simulation of component production volume in each factory. Here, the production volume of engines in Factory C and D is the focus. The Collaborative Platform

predicts the production downtime of Factory C due to a machine failure, and the Collaborative Platform changes the production plan. The Collaborative Platform instructs Factory C to service machines ahead of schedule. To make up for the decreased productivity due to machine maintenance in Factory C, the Collaborative Platform also instructs Factory D to produce more components than the previously assigned amount.

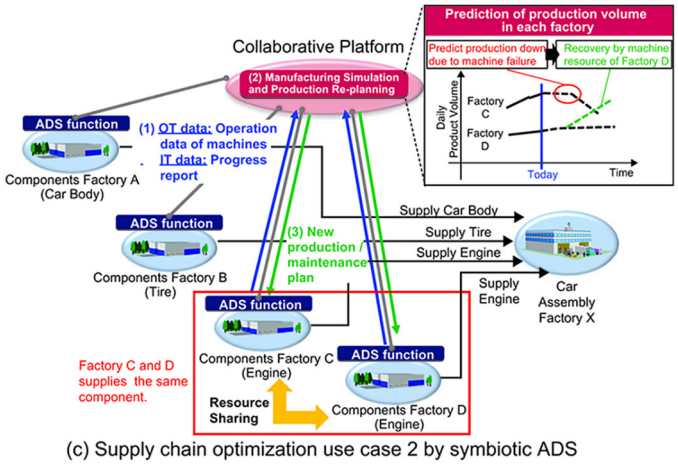
In this way, by connecting several kinds of factories using the Collaborative Platform, the symbiotic ADS can make optimal production management according to the dynamic change of factory status.



(a) Conventional (without symbiotic ADS)



(b) Supply chain optimization use case 1 by symbiotic ADS



(c) Supply chain optimization use case 2 by symbiotic ADS

Figure A.5—Supply chain optimization in industrial production system

Figure A.6 shows the relationship between the functional model shown in Figure A.2 and the system collaboration use cases shown in Figure A.5 part (b) and part (c).

In each system, the function collects OT data and IT data from the OT system and the IT system in a factory via the intra-system connector, and stores collected data in the temporary data repository. Next, the function converts collected data into the data for sharing at the Collaborative Platform by the data-processing function and the protocol conversion function. Then, the function sends the data to the Collaborative Platform via the cross-system connector.

The Collaborative Platform executes manufacturing simulation by the analytics function, and creates a new production plan. The Collaborative Platform sends the feedback information to each factory via the cross-system connector.

Each factory changes the production plan at the intra-system resource management based on the received feedback information. Then, each ADS function controls the OT system and the IT system in the factory internally by the intra-system routing management.

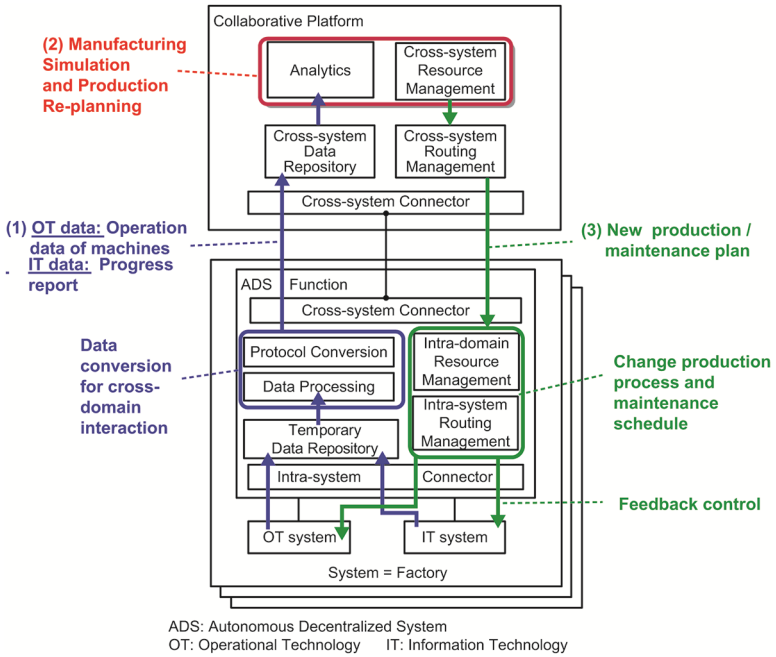


Figure A.6—Relationship between the functional model shown in Figure A.2 and these system collaboration use cases shown in Figure A.5 part (b) and part (c)

Annex B

(informative)

Common concerns and stakholder mapping

B.1 Common concerns mapping

Common concerns mapping is shown in Table B.1.

Table B.1—Common concerns mapping

Common Concerns	Domain							Viewpoint											
	IoT	Smart Manufacturing	Smart Grid	Smart Buildings	Intelligent Transport Systems	Smart cities	E-Health	Conceptual	Compatibility	Lifecycle	Communication	Information	Function	Collaboration	Thread Model	Security and Safety Monitoring	Access Control Architecture	Adequate Design for Required Security	Privacy and Trust

B.2 IoT stakeholder mapping

IoT stakeholder mapping is displayed in Table B.2.

Table B.2—IoT stakeholder mapping

	Viewpoints											
Common Stakeholder	**Conceptual**	**Compatibility**	**Lifecycle**	**Communication**	**Information**	**Function**	**Collaboration**	**Thread Model**	**Security and Safety Monitoring**	**Access Control Architecture**	**Adequate Design for Required Security**	**Privacy and Trust**
acquirers				network designer, owner								
assessors												
builders				network designer								
communicators												
developers				component developer, network designer								
maintainers				network/ system administrator								
operators				owner								
owners				owner								
production engineers												
suppliers				infrastructure supplier								
support staff												
system administrators												
testers												
users				network/ system administrator, owner								

Annex C

(informative)

An example based on threat model viewpoint

An example of attack tree construction for a supervisory control and data acquisition (SCADA) system has been published (Byres, Franz, and Miller [B12]). In the paper for Modbus SCADA the following top-level goals were defined:

a) Gain SCADA system access
b) Identify Modbus device
c) Disrupt master-slave communications
d) Disable slave
e) Read data from slave
f) Write data to slave
g) Program slave
h) Compromise slave
i) Disable master
j) Write data to master
k) Compromise master

As an example the authors defined trees for the attacks:

— Gain SCADA system access
— Identify Modbus device
— Compromise master

Attack: gain SCADA access

OR

1. Gain physical access to remote field site equipment
2. Gain access to SCADA link media

OR

2.1. Intercept wiring leaving building or compound
2.2. Intercept SCADA link in public carrier channel
2.3. Intercept SCADA link over radio link
3. Gain local Process Control Network (PCN) access

OR

3.1. Gain physical access to device on the PCN
3.2. Gain dial-in access to device on PCN

3.3. Gain wireless access to the PCN

4. Gain remote access to PCN via IT network

AND

4.1. Gain network access to IT network

OR

4.1.1. Gain physical access to IT network

4.1.2. Gain remote access to IT network

4.2. Compromise or bypass connection device between IT and PCN

5. Gain access via semi-trusted third party

AND

5.1. Gain access to semi-trusted third party network

OR

5.1.1. Gain physical access to semi-trusted third party

5.1.2. Gain remote access to semi-trusted third party

5.2. Compromise protection between third party system and PCN

6. Gain remote access via untrusted Internet

AND

6.1. Compromise connection device between Internet and IT

6.2. Compromise or bypass connection device between IT and PCN

Attack: identify Modbus device

OR

1. Social engineering (e.g., pretend to be PLC manufacturer's service engineer)

2. TCP/UDP port scan for port 502

AND

2.1. Gain local PCN network access (non-blind)

2.2. Deploy TCP/UDP scanning tool

3. Modbus message scan (only against slave)

AND

3.1. Gain access to remote site or SCADA transmission system

3.2. Deploy Modbus message scanning tool

4. Management/application protocol scan

AND

4.1 Gain local PCN access (non-blind)

4.2 Deploy fingerprinting tool

OR

4.2.1 Scan HTTP/SNMP/Telnet port for identifying characteristics

4.2.2 Scan other identifying ports

5. Sniff existing Modbus session

OR

5.1 Sniff via compromised master

AND

5.1.1 Compromise master

5.1.2 Install packet capture utility

5.2 Sniff via intercepted SCADA media

AND

5.2.1 Gain access to SCADA link media

5.2.2 Install protocol capture tool

Attack: compromise master

OR

1. Physical attack on master

AND

1.1. Gain physical access to the master

1.2. Determine administrator password

2. Network attack on master

2.1 Gain non-blind network access

OR

2.1.1 Compromise master O/S

2.1.2 Compromise primary HMI application on master

2.1.3 Compromise secondary application on master

2.2. Compromise master via slave

OR

2.2.1 Gain physical access to slave

AND

2.2.1.1 Disable real slave device

2.2.1.2 Deploy rogue slave and respond to Modbus requests from master

2.2.1.3 Corrupt master with invalid slave response

2.2.1.4 Load shell app to master

2.2.2 Gain access to SCADA link media

AND

2.2.2.1. Disable real slave device

2.2.2.2. Deploy rogue slave and respond to Modbus requests from master

2.2.2.3. Corrupt master with invalid slave response

2.2.2.4. Load shell app to master

Annex D

(informative)

An example based on an adequate design for required security viewpoint

D.1 Example

This clause fits into 7.3. Figure D.1 shows the evaluation and design flow for security measures using the hardening—adaptivity, responsivity, cooperativity (H-ARC) model.

Timely security measures adapting to security environmental changes are continuously carried out during the lifecycle of IoT systems.

It is also possible to import the result of other security views into the H-ARC model to form the security measures of H-ARC elements.

Figure D.1—Usage example of the H-ARC viewpoint/model

D.2 Information for better understanding of this viewpoint

Figure D.2 shows basic concept of H-ARC.

The security of some IoT systems is achieved through measures taken with respect to the system, time, and organization.

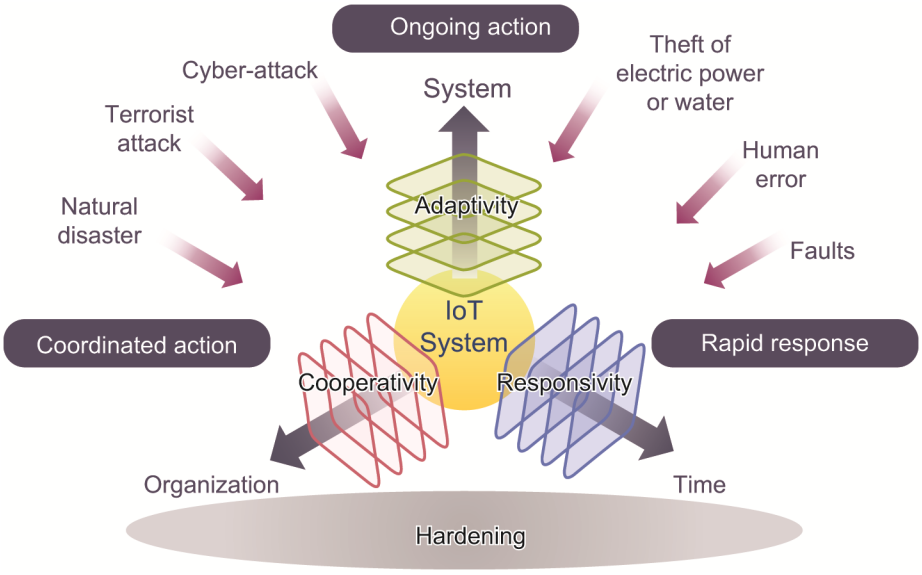


Figure D.2—Basic concept of H-ARC

The security needs to have adaptivity, responsivity, and cooperativity:

a) Adaptivity: ongoing strengthening of preemptive countermeasures and defenses against new threats

b) Responsivity: strengthening of incident response measures for minimizing damage and speeding recovery after an attack or disaster

c) Cooperativity: cooperation between different organizations and infrastructure operators based on a common operational picture

Annex E

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] Ahn, G-J., and R. Sandhu, "Role-based authorization constraints specification," *ACM Transactions on Information and Systems Security (TISSEC)*, vol. 3, no. 4, pp. 492−540, Nov. 2000.

[B2] Al-Ali, A. R., and R. Aburukba, "Role of Internet of Things in the Smart Grid technology," *Journal of Computer and Communications,* vol. 3, no. 5, 229−33, May 2015.

[B3] ANSI INCITS 359-2004, American National Standard for Information Technology Role Based Access Control (RBAC).[13]

[B4] Attribute-based Credentials for Trust (ABC4Trust).[14]

[B5] Axelsson, S., "A Preliminary Attempt to Apply Detection and Estimation Theory to Intrusion Detection," technical report, Geteborg, Sweden: Chalmers University of Technology, 2000.

[B6] Bassi, A., M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange, and S. Meissner, eds., *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model,* New York, NY: Springer Heidelberg, 2013.

[B7] Bishop, M., *Computer Security: Art and Science*, Boston, MA: Addison-Wesley Professional, 2002.

[B8] Baumgrass, A., M. Strembeck, and S. Rinderle-Ma, "Deriving role engineering artifacts from business processes and scenario models," *SACMAT '11 Proceedings of the 16th ACM Symposium on Access Control Models and Technologies,* pp. 11−20, New York, NY, 2011.

[B9] Basin, D., J. Doser, and T. Lodderstedt, "Model driven security: From UML models to access control infrastructures," *ACM Transactions on Software Engineering and Methodology,* vol. 15, no. 1, pp. 39−91, Jan. 2006.

[B10] Building Efficiency Initiative, "What Is a Smart Building?"[15]

[B11] Bulik, M. A., "FoFdation—Foundation for Smart Factory of the Future."[16]

[B12] Byres, E. J., M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," *International Infrastructure Survivability Workshop (IISW'04),* Lisbon, Portugal, 4 Dec. 2004.

[B13] Cannady, J., and J. Mahaffey, "The application of artificial neural networks to misuse detection: initial results," *Proceedings of the 21st National Information Systems Security Conference*, Arlington, VA, 5−8 Oct. 1998.

[B14] Carrez, F., ed., Deliverable D1.5—Final Architectural Reference Model for the IoT v3.0, Internet of Things—Architecture.[17]

[13] ANSI publications are available from the American National Standards Institute (http://www.ansi.org/).
[14] Available: https://abc4trust.eu/download/ABC4Trust-OnePager-About-ABC4Trust.pdf.
[15] Available: http://www.buildingefficiencyinitiative.org/articles/what-smart-building.
[16] Available: https://ec.europa.eu/digital-single-market/en/blog/fofdation-foundation-smart-factory-future.
[17] Available: http://www.meet-iot.eu/deliverables-IOTA/D1_5.pdf.

[B15] Cavoukian, A., "Privacy by Design—7 Foundational Principles." Information and Privacy Commissioner of Ontario, Canada, Aug. 2009, revises Jan. 2011.

[B16] Chaum, D., "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM,* vol. 4, no. 2, Feb. 1981.

[B17] CNSSI No. 4009, Committee on National Security Systems (CNSS) Glossary, 6 Apr. 2015.[18]

[B18] Coyne, J., and T. R. Weil, "ABAC and RBAC: Scalable, flexible, and auditable access management," *IEEE IT Professional,* May/June 2013.

[B19] Conway, M. E., "How do committees invent?" *Datamation,* vol. 14, no. 4, pp. 28−31, 1968.

[B20] Cyber Physical System Public Working Group, Framework for Cyber-physical Systems, Edition 1.0, May 2016.[19]

[B21] Danezis, G., J. Domingo-Ferrer, M. Hansen, J-H. Hoepman, D. Le Métayer, R. Tirtea, and S. Schiffner, *Privacy and Data Protection by Design—From Policy to Engineering.* European Union Agency for Network and Information Security (ENISA), Dec. 2014.

[B22] Denning, D., "An intrusion-detection model," *IEEE Transactions on Software Engineering,* vol. SE-13, no. 2, pp. 222−32, Feb. 1987.

[B23] Dwork, C., "Differential Privacy," in M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds., *Automata, Languages and Programming—ICALP 2006, Volume 4052 of Lecture Notes in Computer Science,* pp. 1–12. Springer-Verlag Berlin Heidelberg, 2006.

[B24] The EU General Data Protection Regulation.[20]

[B25] European Commission, Smart Manufacturing (Digital Single Market Policies), 2017.[21]

[B26] Forrest, S., S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," *Proceedings of IEEE Symposium on Research in Security and Privacy,* Oakland, CA, 1996.

[B27] Floridi, L., "The method of levels of abstraction," *Minds and Machines*, vol. 18, no. 3, pp. 303−29, 2008.

[B28] Garner, G., "Designing Last Mile Communications Infrastructures for Intelligent Utility Networks (Smart Grids)," IBM Australia Limited, 2010.

[B29] Greveler, U., B. Justus, and D. Loehr, "Multimedia Content Identification Through Smart Meter Power Usage Profiles." Steinfurt, Germany: Computer Security Lab Münster University of Applied Sciences D-48565.[22]

[B30] Gu, G., P. Fogla, D. Dagon, W. Lee, and B. Skoric, "Towards an information-theoretic framework for analyzing intrusion detection systems," *ESORICS 2006,* Hamburg, Germany, Sep. 2006.

[B31] Hassell, J., *RADIUS—Securing Public Access to Private Resources,* Sebastopol, CA: O'Reilly & Associates, 2002.

[B32] Hilliard, R., The Trust Viewpoint, Version 2a, 2014.[23]

[B33] Hoehndorf, R., "What is an upper level ontology?" *Ontogenesis*, 13 Apr. 2010.[24]

[B34] Hommes, L. J., "The Evaluation of Business Process Modeling Techniques," Ph.D. diss., TU Delft, p. 137, 2004.

[18] Available: https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf.
[19] Available: https://s3.amazonaws.com/nist-sgcps/cpspwg/files/pwgglobal/CPS_PWG_Framework_for_Cyber_Physical_Systems_Release_1_0Final.pdf.
[20] Available: https://gdpr-info.eu/.
[21] Available: https://ec.europa.eu/digital-single-market/en/smart-manufacturing.
[22] Available: https://epic.org/privacy/smartgrid/smart_meter.pdf.
[23] Available: http://web.mit.edu/richh/www/writings/hilliard-TrustVP.pdf.
[24] Available: http://ontogenesis.knowledgeblog.org/740.

[B35] Hopcroft J. E., R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* 3rd ed., Boston, MA: Addison-Wesley, 2006.

[B36] Hussain, A., J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," *Proceedings of ACM SIGCOMM,* Karlsruhe, Germany, pp. 99−110, 25−29 Aug. 2003.

[B37] IEC 60050-351:2013, International Electrotechnical Vocabulary—Part 351: Control technology.[25]

[B38] IEC 61131-3:2013, Programmable controllers—Part 3: Programming languages.

[B39] IEC 61360-1:2009, Standard data elements types with associated classification scheme for electric items.

[B40] IEC 61360-6, Standard data element types with associated classification scheme for electric components—Part 6: IEC Common Data Dictionary (IEC CDD) quality guidelines.

[B41] IEC 61499-1:2012, Function blocks—Part 1: Architecture.

[B42] IEC 61512-1, Batch control (multiple parts).

[B43] IEC 61512-1:1997, Batch control—Part 1: Models and terminology.

[B44] IEC 61804-2:2016, Function blocks (FB) for process control—Part 2: Specification of FB concept.

[B45] IEC 61850, Communication networks and systems for power utility automation.

[B46] IEC 61907:2009, Communication network dependability engineering.

[B47] IEC 61987, Industrial-process measurement and control—Data structures and elements in process equipment catalogues.

[B48] IEC 62443-2-1:2010, Industrial communication networks—Network and system security—Part 2-1: Establishing an industrial automation and control system security program.

[B49] IEC 62443-3-3:2013, Industrial communication networks—Network and system security—Part 3-3: System security requirements and security levels.

[B50] IEC 62541, OPC unified architecture.

[B51] IEC TR 62511:2014, Guidelines for the design of interconnected power systems.

[B52] IEC, White Paper, "Factory of the Future," Oct. 2015.

[B53] IEC, White Paper, "IoT 2020: Smart and Secure IoT Platform," 2016.

[B54] IETF, Public-Key Infrastructure.[26]

[B55] IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, Dec. 1998.[27]

[B56] Industrial Internet Consortium (IIC), IIC:PUB:G4:V1.0:PB:20160919, Industrial Internet of Things volume G4: Security framework.[28]

[B57] Industrial Internet Consortium (IIC), IIC:PUB:G8:V2.00:PB:20170719, The Industrial Internet of Things, Volume G8: Vocabulary, July 2017.

[B58] ISO 15686-1:2011, Buildings and constructed assets—Service life planning—Part 1: General principles and framework.[29]

[25] IEC publications are available from the International Electrotechnical Commission (http://www.iec.ch) and the American National Standards Institute (http://www.ansi.org/).
[26] Available: https://datatracker.ietf.org/wg/pkix/.
[27] Available: www.rfc-editor.org/info/rfc2474.
[28] Available: https://www.iiconsortium.org/IISF.htm.
[29] ISO publications are available from the International Organization for Standardization (http://www.iso.org/) and the American National Standards Institute (http://www.ansi.org/).

[B59] ISO/IEC 12207:2008, Systems and software engineering—Software life cycle processes.

[B60] ISO/IEC 15026-1:2013, Systems and software engineering—Systems and software assurance—Part 1: Concepts and vocabulary.

[B61] ISO/IEC 19501-1:2005, Information technology—Open distributed processing—Unified Modeling Language (UML) Version 1.4.2.

[B62] ISO/IEC 7498-1, Information technology—Open systems interconnection—Basic reference model: The basic model, 2nd ed. 1984, corrected and reprinted 1996.

[B63] ISO/IEC 11179-1:2004, Information technology—Metadata registries (MDR)—Part 1: Framework.

[B64] ISO/IEC 27000:2016, Information technology—Security techniques—Information security management systems—Overview and vocabulary.

[B65] ISO/IEC Guide 51:2014, Safety aspects—Guidelines for their inclusion in standards.

[B66] ISO/IEC/IEEE 15288:2015, Systems and software engineering—System life cycle processes.[30]

[B67] ISO/TS 17574:2009, Electronic fee collection—Guidelines for security protection profiles.

[B68] ITU X.509, Information technology—Open systems interconnection—The directory: Public-key and attribute certificate frameworks.

[B69] Jan, J., *Secure Systems Development with UML,* Springer-Verlag Berlin Heidelberg, 2005.

[B70] Kantara Initiative, Kantar Initiative Identity Assurance Framework Operations Program.[31]

[B71] Kuhn, D. R., E. J. Coyne, and T. R. Weil, "Adding Attributes to Role Based Access Control," *IEEE Computer,* vol. 43, no. 6, pp. 79−81, June 2010.

[B72] Kruegel, C., G. Vigna, and W. Robertson, "A multi-model approach to the detection of web-based attacks," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* vol. 48, no. 5, 2005.

[B73] Li, W., Using Genetic Algorithm for Network Intrusion Detection, 2004.

[B74] Lodderstedt, T., D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security." *UML 2002—The Unified Modeling Language: 5th International Conference,* Dresden, Germany, 2002.

[B75] Mendel, T. et al., *Global Survey on Internet Privacy and Freedom of Expression,* Paris, France: UNESCO Publishing, 2012.[32]

[B76] Meyer, S., K. Sperner, C. Magerkurth, and J. Pasquier, "Towards modeling real-world aware business processes." *Proceedings of the Second International Workshop on Web of Things*, San Francisco, CA, p. 8, 16 June 2011.

[B77] Mimura, M., T. Arai, T. Nakano, R. Hattori, and A. Sato, "Hitachi's concept for social infrastructure security," *Hitachi Review*, vol. 63, no. 5, pp. 13−20, July 2014.

[B78] Murdock, P., L. Bassbouss, A. Kraft, and C. Wang, "Semantic Interoperability for the Web of Things," Aug. 2016.[33]

[B79] Nakano, T., K. Shimizu, T. Yamada, and T. Kaji, "Control system security for social infrastructure." *Hitachi Review,* vol. 63, no. 5, pp. 68−73, July 2014.

[30] The IEEE standards or products referred to inAnnex E are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.
[31] Available: https://kantarainitiative.org/idassurance/.
[32] Available: http://unesdoc.unesco.org/images/0021/002182/218273e.pdf.
[33] Available: https://www.researchgate.net/publication/307122744_Semantic_Interoperability_for_the_Web_of_Things.

[B80] Nakano, T., H. Tonooka, M. Sato, T. Kaji, and Y. Nonaka, "International standardization activities for Hitachi system security concept and social infrastructure security based on it," *Hitachi Review,* vol. 65, no. 5, pp. 64−9, June 2016.

[B81] National Computer Security Center (NCSC-TG-003), "A Guide to Understanding Discretionary Access Control in Trusted Systems," 30 Sep. 1987.

[B82] Neumann, P. G., "Principled Assuredly Trustworthy Composable Architectures," A001 Final Report, Menlo Park, CA: SRI International, 28 Dec. 2004.

[B83] NIST SP 800-30 Rev 1, Guide for Conducting Risk Assessments, Sep. 2012.

[B84] NIST SP 800-32, Introduction to Public Key Technology and the Federal PKI Infrastructure, Feb. 2001.

[B85] NIST SP 800-94, Guide to Intrusion Detection and Prevention Systems (IDPS), Feb. 2007.

[B86] NIST SP 800-160, Systems Security Engineering—Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems, Nov. 2016.

[B87] NIST SP 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations, Jan. 2014.

[B88] NIST SP 1500-201, Framework for Cyber-Physical Systems, Volume 1, Overview, June 2017.

[B89] Nonaka, Y., Y. Suginishi, A. Lengyel, S. Nagahara, K. Kodama, and Y. Katsumura, "The S-Model: A digital manufacturing system combined with autonomous statistical analysis and autonomous discrete-event simulation for smart manufacturing," *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1006−11, Aug. 2015.

[B90] OASIS Committee Draft 03, XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0, 11 Mar. 2010.

[B91] OASIS Committee Specification 01, Reference Architecture Foundation for Service Oriented Architecture,Version 1.0. OASIS.[34]

[B92] OASIS Standard WS-BPEL, Web Services Business Process Execution Language, Version 2.0.

[B93] Object Management Group, Business Process Model and Notation, Version 2.0, 2011.

[B94] Object Management Group, UML 2.2.[35]

[B95] Al-Omar, B., A. R. Al-Ali, R. Ahmed, and T. Landolsi, "Role of information and communication technologies in the Smart Grid." *Journal of Emerging Trends in Computing and Information Sciences,* vol. 3, no. 5, pp. 707−16, May 2012.

[B96] oneM2M website: http://www.onem2m.org/.

[B97] oneM2M published specifications: http://www.onem2m.org/technical/published-documents.

[B98] oneM2M latest drafts: http://www.onem2m.org/technical/latest-drafts.

[B99] oneM2M TS-0001, Functional Architecture v3.8.0.

[B100] oneM2M TS-0003, Security Solutions v3.7.0.

[B101] oneM2M TS-0004, Core Protocol v3.4.0.

[B102] oneM2M TS-0005, Management Enablement OMA v3.2.0.

[B103] oneM2M TS-0006, Management Enablement BBF v3.6.0.

[B104] oneM2M TS-0008, CoAP Protocol Binding v2.4.0.

[34]Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html.
[35] Available: https://www.omg.org/spec/UML/2.2/.

[B105] oneM2M TS-0009, HTTP Protocol Binding v2.11.0.

[B106] oneM2M TS-0010, MQTT Protocol Binding v2.6.1.

[B107] oneM2M TS-0012, oneM2M Base Ontology v3.5.0.

[B108] oneM2M TS-0014, LWM2M Interworking v3.1.0.

[B109] oneM2M TS-0020, WebSocket Protocol Binding v2.1.1.

[B110] oneM2M TS-0023, Home Appliances Information Model and Mapping v3.5.0.

[B111] oneM2M TS-0024, OIC Interworking v2.0.1.

[B112] oneM2M TS-0026, 3GPP Interworking v0.5.0.

[B113] oneM2M TS-0030, Ontology based Interworking v0.4.1.

[B114] oneM2M TS-0033, Interworking Framework v0.1.1.

[B115] oneM2M TS-0034, Semantics Support v0.3.0.

[B116] oneM2M TS-0035, OSGi Interworking v0.1.0.

[B117] OPC Foundation, "Practical Security Recommendations for Building OPC UA Applications," Whitepaper, Security Working Group, 2017.

[B118] Open Identity Exchange (OIX).[36]

[B119] The Open Group. TOGAF Version 9.1, Berkshire, United Kingdom: Van Haren Publishing, 2011.

[B120] The Open Web Application Security Project (OWASP).[37]

[B121] Phillip, T., "What is a smart building and how can it benefit you?" *RCR Wireless News,* 25 Jul. 2017.[38]

[B122] Proposal for a Regulation of the European Parliament and of the Council. On the protection of individuals with regard to the processing of personal data and on the free movement of such data, 102:24, 5 Apr. 2012.

[B123] Pfitzmann, A., and M. Hansen, "A Terminology for Talking About Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management," Version v0.34, 10 Aug. 2010.[39]

[B124] Ptacek, T., and T. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," Calgary, Alberta, Canada: Secure Networks, Inc., 1998.

[B125] RFC 1492, "An Access Control Protocol, Sometimes Called TACACS," 1993.[40]

[B126] Rowley, J. "The wisdom hierarchy: representations of the DIKW hierarchy," *Journal of Information and Communication Science,* vol. 33, no. 2, pp. 163–80, 2007.

[B127] Rozanski, N., and E. Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives,* 2nd ed., Upper Saddle River, NJ: Addison Wesley, 2011.

[B128] Russo, N., "Smart health and IoT—opportunities and challenges," *Medium*, 28 Apr. 2016.[41]

[B129] Saaksvuori, A., and A. Immonen, *Product Lifecycle Management.* Springer Science & Business Media, 2008.

[36] Available: http://www.openidentityexchange.org/.
[37] Available: https://www.owasp.org/index.php/Main_Page.
[38] Available: http://www.rcrwireless.com/20160725/business/smart-building-tag31-tag99.
[39] Available: https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.
[40] Internet Requests for Comments (RFCs) are available on the World Wide Web at the following ftp site: venera.isi.edu; logon: anonymous; password: user's e-mail address; directory: in-inotes.
[41] Available: https://medium.com/@iotap/smart-health-and-iot-68125f95c405#.6k561u434.

[B130] Sandhu, R., E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer,* vol. 29, no. 2, pp. 38–47, Aug. 1996.

[B131] Sandhu, R., and Q. Munawer, "How to do discretionary access control using roles," *Proceedings of 3rd ACM Workshop on Role-Based Access Control (RBAC-98)*, Fairfax, VA, pp. 47–54, Oct. 1998.

[B132] Schneier, B., "Attack trees," *Dr. Dobb's Journal,* Dec. 1999.[42]

[B133] Schneider, F. B., ed., *Trust in Cyberspace.* Washington, DC: National Academy Press, 1998.

[B134] Sebring, M. M., and R. A. Whitehurst, "Expert systems in intrusion detection: a case study," *The 11th National Computer Security Conference,* Baltimore, MD, pp. 74–81, Oct. 1988.

[B135] Sweeney, L., "K-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, no. 5, pp. 557–70, 2002.

[B136] Taylor, B. N., and C. E. Kuyatt, "Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results," NIST Technical Note 1297, 1994.

[B137] Timm, K., "IDS Evasion Techniques and Tactics."[43]

[B138] Tor Project: Anonymity Online.[44]

[B139] United Nations Economic Commission for Europe, *Intelligent Transport Systems (ITS) for Sustainable Mobility*, Geneva, Switzerland: UNECE, 2012.[45]

[B140] U.S. Department of Energy (DOE), Office of Electricity Delivery and Energy Reliability, The Smart Grid: An Introduction, 2008.[46]

[B141] "User-Managed Access (UMA) Profile of OAuth 2.0," Internet-draft.[47]

[B142] Vigna, G., S. T. Eckmann, and R. A. Kemmerer, "The STAT tool suite," *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX'00),* Hilton Head, SC, 25−27 Jan. 2000.

[B143] Vigna, G., and R. A. Kemmerer, "NetSTAT: A network-based intrusion detection approach," *Proceedings of the Annual Computer Security Applications Conference (ACSAC),* Scottsdale, AZ, 1998.

[B144] de Vries, M. J., "Philosophy of Technology," pp. 15−33 in *Technology Education for Teachers,* P. J. Williams, ed., Rotterdam, The Netherlands: SensePublishers, 2012.

[B145] Weaver, W., "The mathematics of communication," *Scientific American,* vol. 181, no. 1, pp. 11−15, 1949.

[B146] Whitney, K., et al., "Systems theory as a foundation for governance of complex systems," *International Journal of System of Systems Engineering,* vol. 6, no. 1/2, pp. 15−32, Jan. 2015.

[B147] Zanella, A., N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal,* vol. 1, no. 1, pp. 22−32, Feb. 2014.

[42] Available: http://www.drdobbs.com/attack-trees/184411129.
[43] Available: http://online.securityfocus.com/infocus/1577.
[44] Available: https://torproject.org.
[45] Available: https://www.unece.org/fileadmin/DAM/trans/publications/Intelligent_Transport_Systems_for_Sustainable_Mobility.PDF
[46] Available: http://energy.gov/oe/downloads/smart-grid-introduction-0.
[47] Available: https://tools.ietf.org/html/draft-hardjono-oauth-umacore-07.

IEEE SA
STANDARDS ASSOCIATION

RAISING THE WORLD'S STANDARDS

Connect with us on:

Twitter: twitter.com/ieeesa
Facebook: facebook.com/ieeesa
LinkedIn: linkedin.com/groups/1791118
Beyond Standards blog: beyondstandards.ieee.org
YouTube: youtube.com/ieeesa

standards.ieee.org
Phone: +1 732 981 0060

IEEE