The Industrial Internet of Things
Volume G1: Reference Architecture

Version 1.9

June 19, 2019

CONTENTS

FIGURES

TABLES

1 OVERVIEW

This technical report carries on the work of the Industrial Internet Consortium (IIC) by refining and advancing the '*Industrial Internet Reference Architecture*' Version 1.7 [1]. To assist the user community and increase usability, the Industrial Internet Reference Architecture (IIRA) is now published as Volume G1 of the comprehensive and all-inclusive '*The Industrial Internet Consortium: Industrial Internet of Things (IIC IIoT)*'. Please refer to '*IIC IIoT Volume G0*: Overview'[1] for the IIC IIoT structure.

This definitive IIC IIoT collection will continually expand to represent the broad spectrum of technical, business and best-practices content providing guidance and recommendations, and sharing success stories and lessons learned, in the form of IIC technical reports, white papers and green papers. It will collectively provide guidance to IIoT architects, business leaders, implementers and users at every level to optimize their endeavors at establishing IIoT systems, consummating the convergence of Operational Technology (OT) and Information Technology (IT) to achieve the tremendous economic benefits IIoT has to offer.

1.1 INTRODUCTION

This technical report describes the Industrial Internet Reference Architecture (IIRA) for Industrial Internet of Things (IIoT) systems. It specifies an Industrial Internet Architecture Framework (IIAF) comprising viewpoints and concerns to aid in the development, documentation and communication of the IIRA. The reference architecture uses a common vocabulary and a standard-based framework to describe business, usage, functional and implementation viewpoints that it has defined.

1.2 PURPOSE

This Industrial Internet Reference Architecture technical report addresses two primary purposes. For all IIC work efforts, it is the foundational framework for all other technical documents and technical activities of the consortium. For IIC members and the broader IoT community, it provides guidance and assistance in the development, documentation, communication and deployment of IIoT systems.

1.3 SCOPE

The scope of this document is the Industrial Internet Architecture Framework (IIAF) and the Industrial Internet Reference Architecture (IIRA), built on the architecture framework, IIAF. This

[1] *IIC IIoT Volume G0*: Overview comprises two parts:

- Part 1 contains a detailed explanation of what constitutes the Industrial Internet of Things and how the IIoT and corresponding Industrial Internet of Things systems (IIoT systems) are unique.
- Part 2 defines the IIC IIoT publishing scheme, document numbering scheme and contains a manifest of currently published IIC Documents.

document presents an architectural view of Industrial Internet of Things systems using ISO '*ISO/IEC/IEEE 42010:2011'* [2] architecture concepts. This document is not limited to existing technologies but rather thinks ahead to include technology concepts that the IIC is experimenting with through its testbed programs.

1.4 STRUCTURE

This document is organized as follows:

- Chapter 2: Reference Architecture Concepts
- Chapter 3: Architecture Framework
- Chapter 4: Business Viewpoint
- Chapter 5: Usage Viewpoint
- Chapter 6: Functional Viewpoint
- Chapter 7: Implementation Viewpoint

1.5 AUDIENCE

This document is primarily for IIoT system architects. We assume the reader is familiar with the general architecture concepts, architecture frameworks and reference architectures. This document can also be used by, and provides value for, plant managers, IT managers, business managers and others who want to understand better how the convergence of OT and IT is an important part of achieving the promised benefits of IIoT.

1.6 USE

System architects can use this IIRA systematically as an architectural template to define their unique IIoT system requirements and design concrete architectures to address them. Using this common approach to architecture design assists in consistent architecture implementation across different use cases in various industrial sectors meeting unique system requirements. Equally importantly, it assists in achieving a common understanding and communication of the overall system among its diverse stakeholders, which will aid in system deployment and significantly enhance system interoperability across industrial sectors.

1.7 TERMS AND DEFINITIONS

The following unique terms and definitions are used in this document[1]:

- *Concern*: any topic of interest pertaining to a system.
- *Architecture viewpoint* (viewpoint for short): conventions framing the description and analysis of specific system concerns.
- *Stakeholder*: an individual, team or organization having an interest in a concern and, by extension an interest in, the viewpoint and system.

[1] Many of these terms will be elaborated in the context where they are introduced and used.

- *Model kind*: a set of conventions describing, analyzing and resolving concerns in a specific way. A viewpoint may specify one or more model kinds.
- *Model*: the outcome of applying the conventions of a specific model kind in a viewpoint to describe, analyze and resolve a specific set of concerns in that viewpoint.
- *Architecture view*: the collection of ideas describing, analyzing and resolving the set of specific concerns in a viewpoint using the conventions set forth in that viewpoint. A view includes one or more models.
- *Architecture frame*: the collection of conventions (or constructs) for identifying, describing, analyzing and resolving concerns in association with their respective stakeholders for a system of interest. An architecture frame can be considered as the collection of concepts and conventions of concern, stakeholder, viewpoint and model kind.
- *Architecture representation*: the collection of outcomes of applying an architecture frame to a concrete or abstract system, expressed as a view and its models.
- *Architecture framework*: consisting of architecture frame and representation.
- *Reference architecture*: the outcome of applying the architecture framework to a class of systems to provide guidance and to identify, analyze and resolve common, important architecture concerns. A reference architecture can be used as a template for concrete architecture of systems of the class.

1.8 SYMBOLS

None.

1.9 CONVENTIONS

1.9.1 TYPOGRAPHICAL AND LINGUISTIC CONVENTIONS AND STYLE

Terms that require definition are rendered in *italics*. (As the usage immediately preceding demonstrates, italics may be also be used as example, or for emphasis.)

Generally, only the first use of the term is italicized. However, when a term can be read in its usual English language mode, the first use of the term may be italicized as the discussion becomes technical. In the first example below, “safety” and “security” are used informally. In the second, it introduces a definition.


Example

“Among the system characteristics that must be considered, safety is perhaps the most important, followed by security.”

“*Safety* is 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.”

1.9.2 ABBREVIATIONS

The following abbreviations unique to this document are used:

IIAF—Industrial Internet Architecture Framework

IIoT—Industrial Internet of Things

IIRA—Industrial Internet Reference Architecture

1.10 RELATIONSHIP WITH OTHER IIC DOCUMENTS

This document fits in to the IIC Technical Publication Organization, see Figure 1-1 (please refer to '*IIC IIoT Volume G0*: *Overview'* for details).

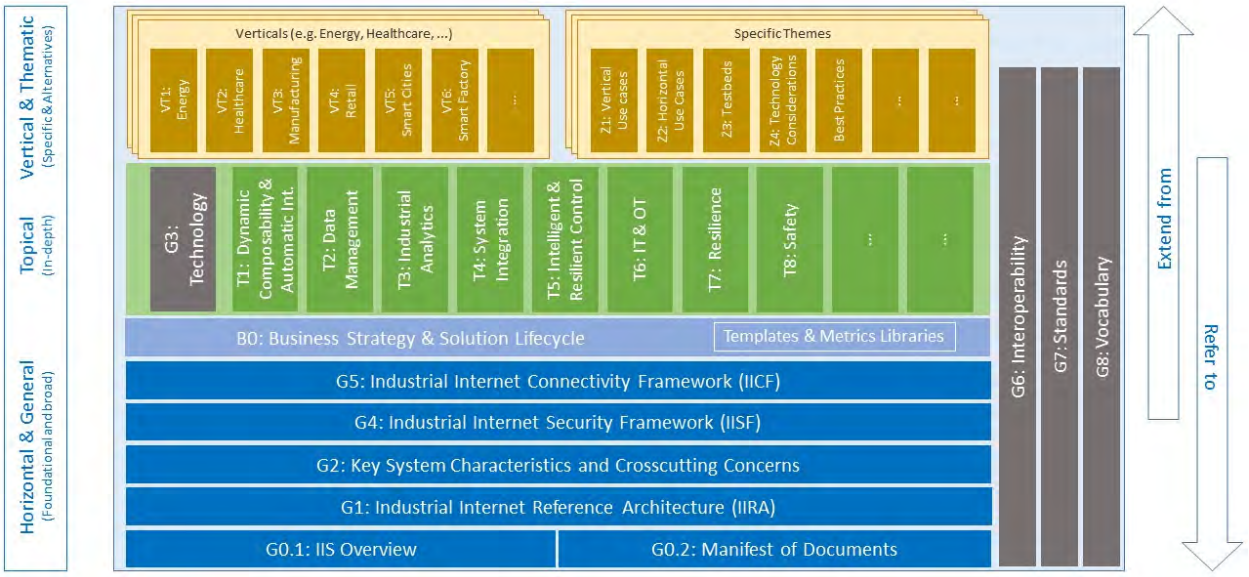

Figure 1-1: IIC Technical Publication Organization

2 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE CONCEPTS

A reference architecture provides guidance for the development of system, solution and application architectures. It provides common and consistent definitions for the system of interest, its decompositions and design patterns, and a common vocabulary with which to discuss the specification of implementations and compare options.


Example

A reference architecture for a residential house states that all residential houses need to provide one or more bedrooms, bathrooms, a kitchen and a living area. This set of rooms is accessible inside the house through doors, hallways and stairways, and from outside through a main and a back door. The house provides a safe environment against threats such as fire, hurricanes and earthquakes. The structure of the house needs to sustain snow and wind load that may be found in its local environment. The house needs to provide reasonable measures to detect and prevent unauthorized intrusions.

A reference architecture provides a common framework around which more detailed discussions can center. By staying at a higher level of abstraction, it enables the identification and comprehension of the most important issues and patterns across its applications in many different use cases. By avoiding specifics, a reference architecture allows subsequent designs to follow the reference architecture without the encumbrance of unnecessary and arbitrary restrictions.

2.1 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE

The IIRA is a standards-based open architecture for IIoT systems. The IIRA maximizes its value by having broad industry applicability to drive interoperability, to map applicable technologies, and to guide technology and standard development. The architecture description and representation are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA distills and abstracts common characteristics, features and patterns from use cases defined in the IIC as well as elsewhere. It will be refined and revised continually as feedback is gathered from its application in the testbeds developed in IIC as well as real-world deployment of IIoT systems. The IIRA design is also intended to transcend today's available technologies and so can identify technology gaps based on the architectural requirements. This will in turn drive new technology development efforts by the industrial internet community.

3 INDUSTRIAL INTERNET ARCHITECTURE FRAMEWORK

Many stakeholders are involved when considering complex systems such as those expected of IIoT systems. These stakeholders have many intertwining concerns pertinent to the system of interest. Stakeholder's concerns cover the full lifecycle of the system. System complexity requires a framework to identify and classify stakeholder concerns into appropriate categories. Such a framework allows a systematic evaluation of such systems, as well as the resolution necessary to architect and build such systems.

To address this need, the Industrial Internet Consortium used '*ISO/IEC/IEEE 42010:2011*' [2] to define its Industrial Internet Architecture Framework (IIAF). The IIAF identifies conventions, principles and practices for consistent description of IIoT architectures. This standard-based architecture framework facilitates easier evaluation, and systematic and effective resolution of stakeholder concerns. It serves as a valuable resource to guide the development and the documentation of, and the communication about, the IIRA.

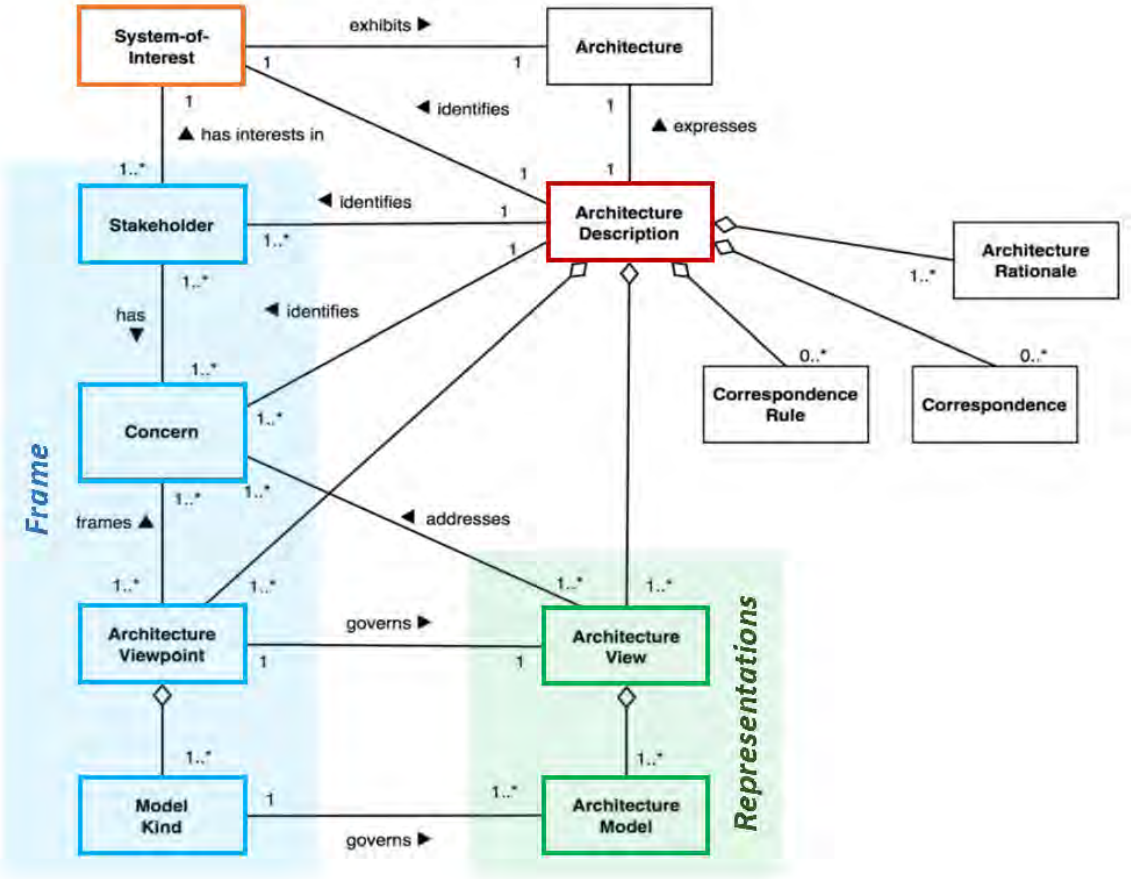

Figure 3-1: ISO/IEC/IEEE 42010:2011—Architecture Description[1]

[1] The colored highlights are added by this document to indicate the architectural constructs that are described by IIRA.

3.1 UNDERLYING STANDARD

The '*ISO/IEC/IEEE 42010:2011 Systems and Software Engineering–Architecture Description*' standard codifies architecting conventions and common practices, and provides an ontology for the description of architectures. Taken from ISO/IEC/IEEE 42010:2011, Figure 3-1 expresses the content of an architecture description and the relations between the terms and concepts therein. The architecture description enables the system architect to express an architecture.

3.2 ARCHITECTURE FRAMEWORK

An architecture framework contains information identifying the fundamental architecture constructs and specifies concerns, stakeholders, viewpoints, model kinds, correspondence rules and conditions of applicability. System architects can use an architecture framework to discover, describe and organize topics of interest (concerns) about the system at hand; they can further use architecture representation to clarify, analyze and resolve these concerns.

It is useful to consider an architecture framework in this context to include an *architecture frame* and a collection of *architecture representations*.

At the core of the ISO/IEC/IEEE Architecture Description standard are viewpoints. A *viewpoint* comprises conventions framing the description and analysis of specific system concerns. A viewpoint frames one or more concerns. The term *concern* refers to any topic of interest pertaining to the system. A *stakeholder* is an individual, team, organization or classes thereof, having an interest in a concern and by extension an interest in the viewpoint and system[1]. To aid the tasks of describing, analyzing and resolving concerns, one or more modeling constructs can be defined as the *model kinds*[2] for each viewpoint. The constructs of viewpoints and their corresponding stakeholders, concerns and model kinds can be considered as the architecture frame.

Following the approach defined by the ISO/IEC/IEEE Architecture Description standard, the ideas in describing, analyzing and resolving the set of specific concerns in each of the viewpoints are expressed as the *architecture view* for each viewpoint. Applying the model kinds defined in each viewpoint to describe, analyze and resolve the concerns consequently result in the creation of architecture models that make up the respective architecture view. Together, the architecture views with their architecture models can be considered as the representations of the architecture.


Example

A common approach for designing a complex system is to decompose it into constituent subsystems. Suppose we want to address the concerns of what the functional subsystems are, across what interfaces they interact and how they interact to realize the desired system behaviors. A functional decomposition of the system

[1] An IIoT system may become a stakeholder of itself as it becomes intelligent, capable of learning and making decisions itself as an autonomous agent.

[2] A model kind captures conventions for a type of model.

> can make each of the subsystems easier to conceive, understand, design, implement, reuse and maintain. A component diagram may be used to describe structure of the subsystems and their interfaces, sequence diagrams the way in which the subsystems interact, and state diagrams the way in which the system or one of its subsystems behaves in response to external events. These diagrams and their associated documentation collectively describe and address the concerns of the functional decomposition. The component, sequence and state diagrams are said to be the model kinds for addressing the concerns of functional structure of the system. The resultant concrete models by applying these *model kinds* to system decomposition are the part of the architecture models[1].

The key ideas of architecture framework, its frame and representations, are outlined in Figure 3-2: Architecture Framework.

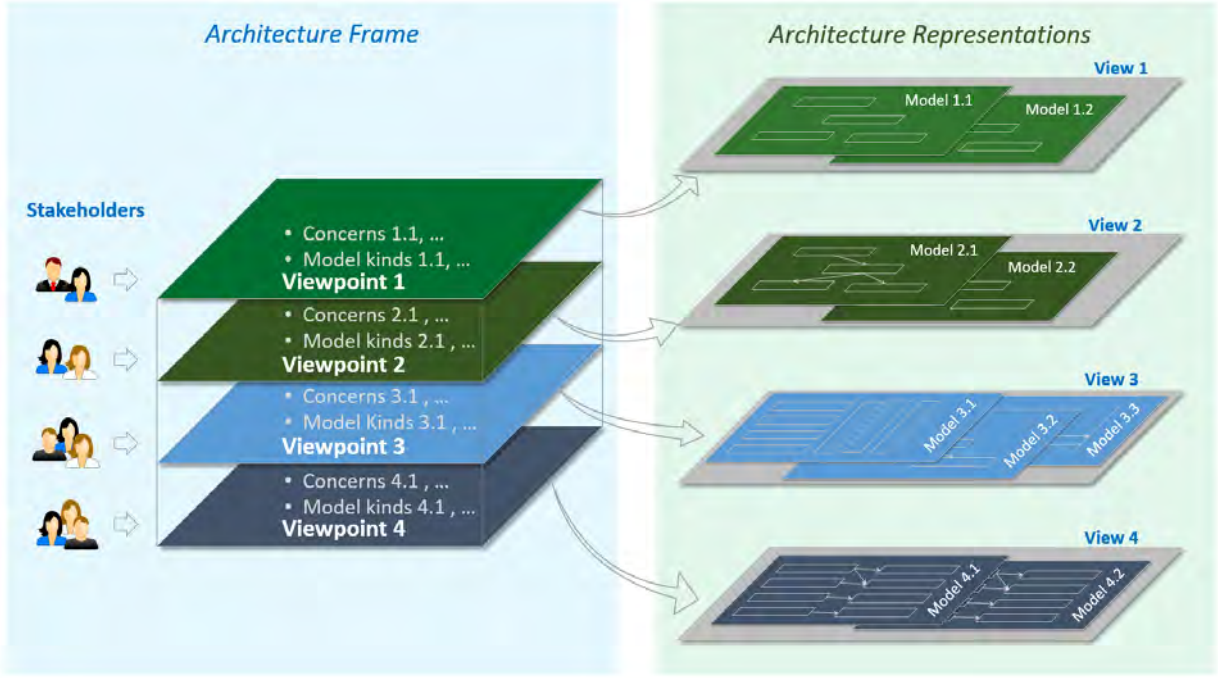

Figure 3-2: Architecture Framework

3.3 INDUSTRIAL INTERNET ARCHITECTURE FRAMEWORK

The IIAF adopts the general concepts and constructs in the ISO/IEC/IEEE architecture specification, specifically, *concern, stakeholder* and *viewpoint* as its architecture frame, and *views* and *models* as its architecture representation in describing and analyzing on important common architecture concerns for IIoT systems. The architecture frame and architecture representation together constitute the Industrial Internet Architecture Framework (IIAF), as shown in Figure 3-3. IIAF is a foundational framework to analyze industrial internet of things systems.

[1] There will be other architecture models addressing other concerns.

3.4 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE

The IIRA documents the outcome of applying the IIAF to its intended class of systems of interest: Industrial Internet of Things systems. It first identifies and highlights the most important architectural concerns commonly found in IIoT systems across industrial sectors and classifies them into viewpoints along with their respective stakeholders. It then describes, analyzes and, where appropriate, provides guidance to resolve these concerns in these viewpoints, resulting in a certain abstract architecture representations.

Figure 3-3 illustrates the key ideas about the constructs of the Industrial Internet Reference Architecture and its application.

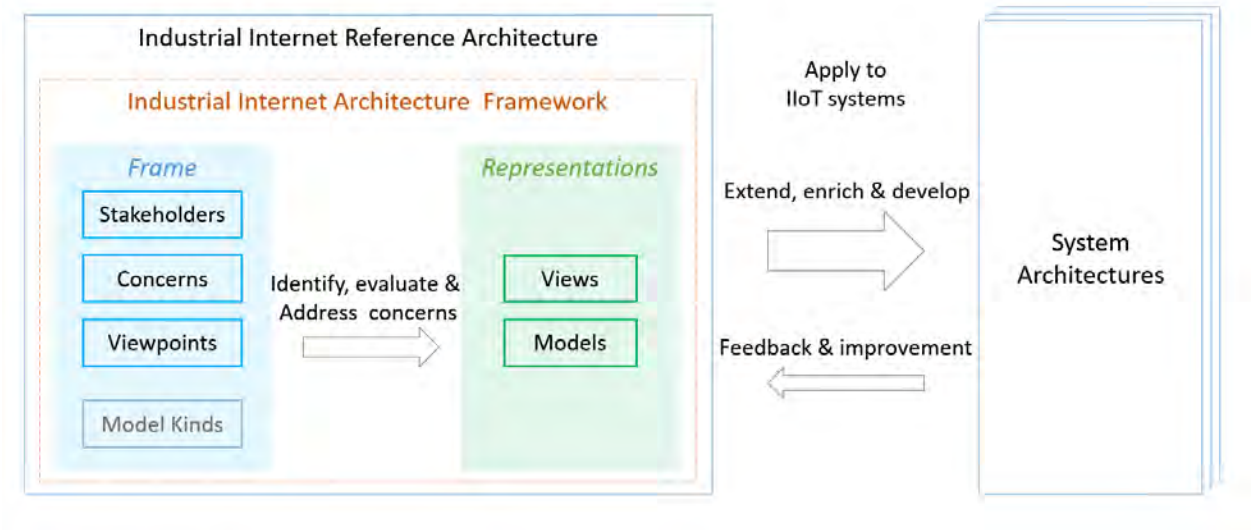

Figure 3-3: IIRA constructs and application

The IIRA is at a level of abstraction that excludes architectural elements whose evaluation requires specificities only available in concrete systems. It does not describe all the architecture constructs as outlined in Figure 3-3: IIRA constructs and application. Instead, it adapts the ISO architecture specification (ISO/IEC/IEEE 42010) with minor adjustments in two aspects:

- IIRA does not explicitly identify the model kind as a key construct of its framework; instead, it uses the concept loosely during the analysis of concerns in developing its representation.
- IIRA does not explicitly discuss certain architecture constructs, Correspondence Rules, Correspondence and Architecture Rationale; it leaves them to the development of concrete architectures as needed.

Within the IIRA, these models and their representations in the views are chosen because they address the respective concerns at the appropriate level of abstraction and demonstrate the key ideas of this reference architecture. They are not, however, the sole models and views for addressing concerns in the viewpoints; nor are they at a depth sufficient to implement a real system. The views can be used as a starting point for concrete architecting, and then extended,

enriched, supplemented, or replaced with better ones in accordance with the needs of the specific IIoT system at hand.

System architects interested in following the ISO/IEC/IEEE 42010 Architecture Description to develop their concrete architectures can first apply IIRA as its base framework, extend and enrich the constructs provided in IIRA based on their specific system requirements where necessary, and develop those constructs not described in IIRA as part of the architecting process.

The following sections define the Industrial Internet Architecture Viewpoints and identify their key concerns. These viewpoints are then described in more details in Chapters 4 through 7.

3.5 INDUSTRIAL INTERNET VIEWPOINTS

The IIRA viewpoints are defined by analyzing the various IIoT use cases developed by the IIC and elsewhere, identifying the relevant stakeholders of IIoT systems and determining the proper framing of concerns. These four viewpoints are:

- Business Viewpoint
- Usage Viewpoint
- Functional Viewpoint
- Implementation Viewpoint

As shown in Figure 3-4, these four viewpoints form the basis for a detailed viewpoint-by-viewpoint analysis of individual sets of IIoT system concerns. Architects who adapt the industrial internet viewpoints as the basis of their architecture may extend them by defining additional viewpoints to organize system concerns based on their specific system requirements.

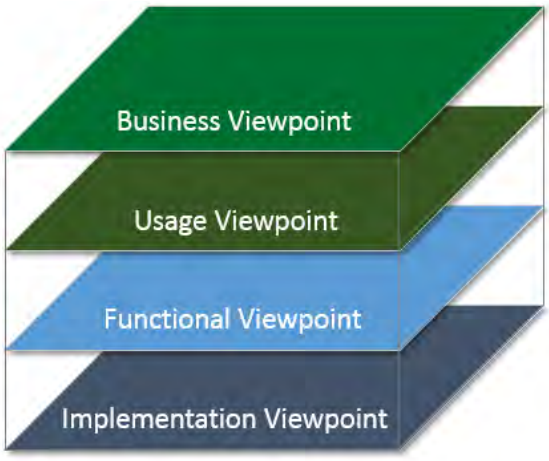

Figure 3-4: Industrial Internet Architecture Viewpoints

3.5.1 BUSINESS VIEWPOINT

The *business viewpoint* attends to the concerns of the identification of stakeholders and their business vision, values and objectives in establishing an IIoT system in its business and regulatory context. It further identifies how the IIoT system achieves the stated objectives through its mapping to fundamental system capabilities.

These concerns are business-oriented and are of particular interest to business decision-makers, product managers and system engineers.

Chapter 4 details the Business Viewpoint.

3.5.2 USAGE VIEWPOINT

The *usage viewpoint* addresses the concerns of expected system usage. It is typically represented as sequences of activities involving human or logical (e.g. system or system components) users that deliver its intended functionality in ultimately achieving its fundamental system capabilities.

The stakeholders of these concerns typically consist of system engineers, product managers and the other stakeholders including the individuals who are involved in the specification of the IIoT system under consideration and who represent the users in its ultimate usage.

Chapter 5 details the Usage Viewpoint.

3.5.3 FUNCTIONAL VIEWPOINT

The *functional viewpoint* focuses on the functional components in an IIoT system, their structure and interrelation, the interfaces and interactions between them, and the relation and interactions of the system with external elements in the environment, to support the usages and activities of the overall system.

These concerns are of particular interest to system and component architects, developers and integrators.

Chapter 6 details the Functional Viewpoint.

3.5.4 IMPLEMENTATION VIEWPOINT

The *implementation viewpoint* deals with the technologies needed to implement functional components (functional viewpoint), their communication schemes and their lifecycle procedures. These elements are coordinated by activities (usage viewpoint) and supportive of the system capabilities (business viewpoint).

These concerns are of particular interest to system and component architects, developers and integrators, and system operators.

Chapter 7 details the IIoT implementation viewpoint.

3.6 CROSSCUTTING CONCERNS, SYSTEM CHARACTERISTICS AND THEIR ASSURANCE

The business, usage, functional and implementation viewpoints facilitate a systematic way to identify IIoT system concerns and their stakeholders, to bring similar or related concerns together so they can be analyzed and addressed effectively. The deliberation of the concerns is often performed within each of the viewpoints to which they belong. However, this is not to suggest that system concerns are always to be resolved within each of the viewpoints, in isolation to those in other viewpoints.

The order in which the business, usage, functional and implementation viewpoints are arranged, from top to bottom, as depicted in Figure 3-5, reflects a general interaction pattern between the viewpoints. Broadly speaking, decisions from a higher-level viewpoint guide and impose requirements on the viewpoints below it. For example, the decisions resulting from the business viewpoint has direct influence to the deliberations in the usage viewpoint and so forth. On the other hand, the deliberation of the concerns in a lower viewpoint, including implementing requirements from the viewpoints above it, validate and in some cases cause revisions to the analysis and possibly the decisions in the viewpoint above it. For example, the deliberation in the usage viewpoint may validate if some of the fundamental system capability proposed in the business viewpoint can be realized.

Moreover, there are classes of system concerns, such as those related to safety and security, which may require consistent consideration across the viewpoints. These are sometimes referred to as crosscutting concerns. These classes of concerns are often related to system properties resulting not just from its components but also the interactions among these components—the emergent properties of the system. Emergent system-wide properties are called system characteristics in the context of this reference architecture. More formally, *system characteristics* are system properties and behaviors of an IIoT system resulting from those of its constituent sub-systems and the nature of their interactions with each other, the context and the environment in which they operate. These properties usually have contractual value, e.g. supporting Service Level Agreement (SLA), for the system stakeholders.

System concerns related to safety and security are of crucial importance to IIoT systems. To ensure the system is capable of demonstrating the expected system characteristics, it is essential to have a clear understanding of the business drivers for strong safety and security requirements and their potential effect on the business objectives in case they are not met. It also requires detailed consideration of how these requirements affects the usages of the system. Finally, the requirements and usage considerations must be reflected in the design of the functional components and the choice of technologies and actual implementation of the system.

Furthermore, system characteristics are often subject to regulations, compliance requirements and contractual agreements and thus need be measured and assessed. Because IIoT systems are built from multi-vendor components and solutions, possibly composed dynamically after deployment, it may be required to provide recorded claims and their supportive evidence of specific system characteristics in components to evaluate, select, acquire and assemble qualified components into the desired IIoT system.

More detailed discussions on the key crosscutting concerns and their associated system characteristics, such as safety and security, and their assurance [12], including the introduction of the concept of trustworthiness to address the entwined nature of security, safety, reliability, resilience, and privacy appropriately, can be found in '*G2—Key IIoT System Concerns*', '*G4—Industrial Internet Security Framework (IISF)'* [3] and other volumes.

3.7 SCOPE OF APPLICABILITY AND RELATIONSHIP TO SYSTEM LIFECYCLE PROCESS

3.7.1 SCOPE OF APPLICABILITY

The reference architecture purposely starts from a generic framework and seeks common architecture patterns to ensure wide applicability to industrial internet applications across all industrial sectors. For this reason, this general framework stays at a high level in its architecture descriptions, and its concepts and models are at a high degree of abstraction. The application of this general architecture framework, as a reference architecture, to real-world usage scenarios transform and extend the abstract architectural concepts and models into detailed architectures addressing the specificity of the industrial internet usage scenarios, thereby guiding the next level of architecture and system design.

The IIC will evaluate feedback from practical implementations across the industrial sectors, including the various IIC testbed initiatives, to ascertain the soundness and usefulness of IIRA in aiding the system-design process of real-world systems and may revise and improve this reference architecture as deemed necessary. IIC fully expects IIoT system implementers to identify additional common architecture patterns, especially those carrying next level details in the architecture. From their feedback, IIC will document and make available additional architecture patterns where appropriate to aid in future system designs.

3.7.2 RELATIONSHIP TO SYSTEM LIFECYCLE PROCESS

The architecture concerns framed by this reference architecture may need to be considered and addressed beyond the design phase of the system into its full lifecycle. This reference architecture, through its viewpoints, provides guidance to system lifecycle processes from IIoT system conception, to design and implementation. Its viewpoints offer a framework to system designers to think iteratively through important common architectural issues in IIoT system creation. It also suggests some common approaches (concepts and models) as views in each of these viewpoints to aid the identification and resolution of important architectural issues. It is not a description of a system lifecycle process, which varies from one industrial sector to another, as that is out of scope. Rather, as shown in figure 3-6, this reference architecture is an architectural framework and methodology for system conceptualization and architecture highlighting important system concerns that may affect lifecycle process. IIoT system lifecycle processes and the expected use of this reference architecture in these lifecycle processes will be covered in a different set of IIC technical reports.

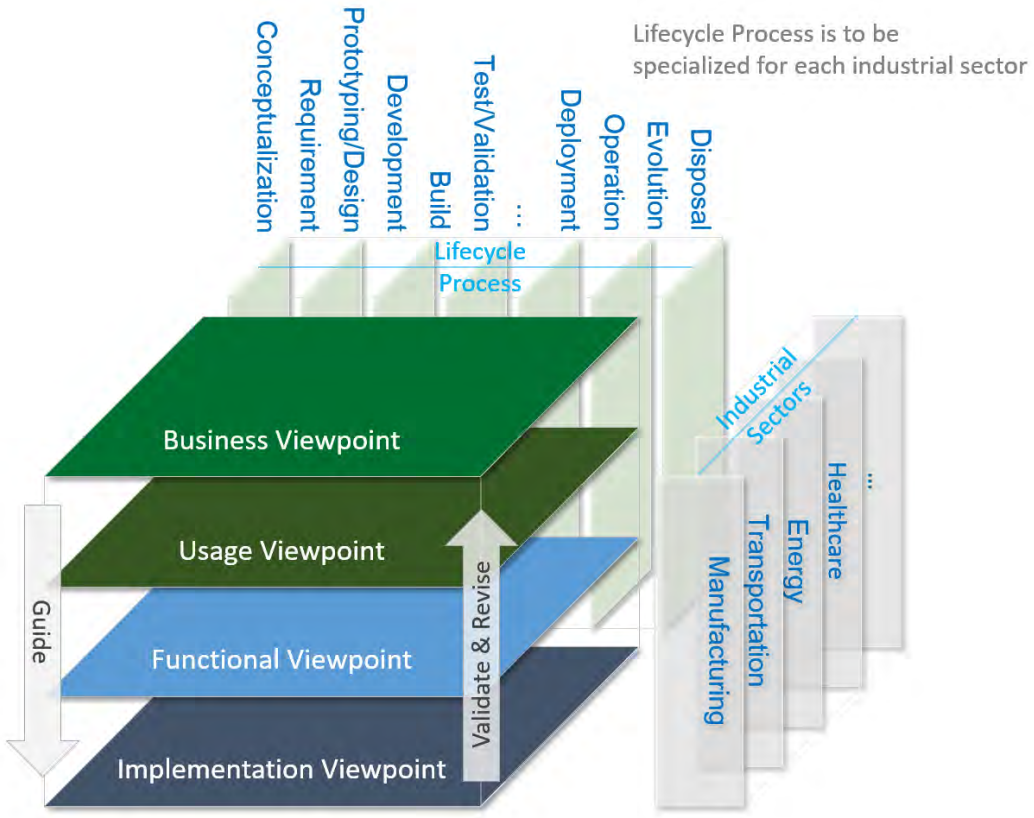

Figure 3-5: Relationship among IIRA Viewpoints, Application Scope and System Lifecycle Process

4 BUSINESS VIEWPOINT

Business viewpoint is an architecture viewpoint that frames the vision, values and objectives of the business stakeholders in establishing an industrial internet of things (IIoT) system in its business and regulatory context.

Business-oriented concerns such as business value, expected return on investment, cost of maintenance and product liability must be evaluated when considering an IIoT system as a solution to business problems. To identify, evaluate and address these business concerns, we introduce several concepts and define the relationships between them, as shown in Figure 4-1.[1]

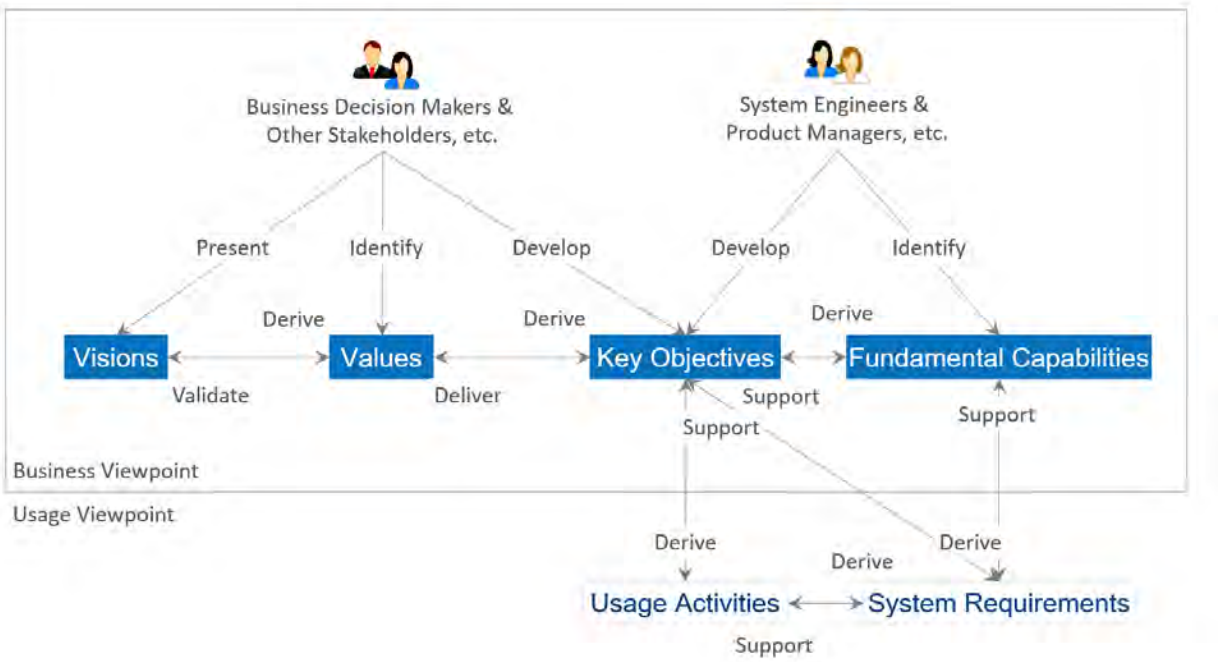

Figure 4-1: A Vision and Value-Driven Model

Stakeholders have a major stake in the business and strong influence in its direction. They include those who drive the conception and development of IIoT systems in an organization. They are often recognized as important strategic thinkers and visionaries within a company or an industry. It is important to identify these major stakeholders and engage them early in the process of evaluating these business-oriented concerns.

In conceptualizing and defining an IIoT system, business stakeholders may take many intervening technological and business factors into consideration, including external influences from

[1] This approach is based on the *Business Motivation Model (BMM)* [11] by the Object Management Group (OMG), consistent with best practices in this domain. Some of the terminology has been changed to be consistent with the ISO/IEC/IEEE 42010:2011 [2].

technological trends, specific market condition and potential, customer inputs, and regulatory requirements (in the areas of, e.g., safety, privacy, environmental and labor).

Vision describes a future state of an organization or an industry.[1] It provides the business direction toward which an organization executes. Senior business stakeholders usually develop and present an organization's vision.

Values reflect how the vision may be perceived by the stakeholders who will be involved in funding the implementation of the new system as well as by the users of the resulting system. These values are typically identified by senior business and technical leaders in an organization. They provide the rationale as to why the vision has merit.

Key objectives are quantifiable high-level technical and ultimately business outcomes expected of the resultant system in the context of delivering the values. Key objectives should be measurable and time-bound. Senior business and technical leaders develop the key objectives.

Fundamental capabilities refer to high-level specifications of the essential ability of the system to complete specific major business tasks.[2] Key objectives are the basis for identifying the fundamental capabilities. Capabilities should be specified independently of how they are to be implemented (neutral to both the architecture and technology choices) so that system designers and implementers are not unduly constrained at this stage.

The process for following this approach is for the stakeholders to first identify the vision of the organization and then how it could improve its operations through the adoption of an IIoT system. From the vision, the stakeholders establish the values and experiences of the IIoT system under consideration and develop a set of key objectives that will drive the implementation of the vision. From the objectives, the stakeholders derive the fundamental capabilities that are required for the system.

To verify that the resultant system indeed provides the desired capabilities meeting the objectives, they should be characterized by detailed quantifiable attributes such as the degree of safety, security and resilience, benchmarks to measure the success of the system, and the criteria by which the claimed system characteristics can be supported by appropriate evidence.

5 USAGE VIEWPOINT

The *usage viewpoint* is an architecture viewpoint that frames the concerns related to implementing the capabilities and structure of an industrial internet of things (IIoT) system.

[1] The concepts of vision, values and experiences and key objectives are related to the BMM concept of Ends (i.e. the results, or what needs to be achieved).

[2] A capability is normally defined as "the ability to do something" although in enterprise architecture terms it is extended to be "a high-level specification of the enterprise's ability" (MODAF (27)). Fundamental Capabilities map to the Means aspect of the BMM, being a starting point for considering how the solution will provide the "means" to deliver the vision.

The usage viewpoint is concerned with how an IIoT system realizes the key capabilities identified in the business viewpoint. The usage viewpoint describes the activities that coordinate various units of work over various system components. These activities—describing how the system is used—serve as an input for system requirements including those on key system characteristics and guide the design, implementation, deployment, operations and evolution of the IIoT system.

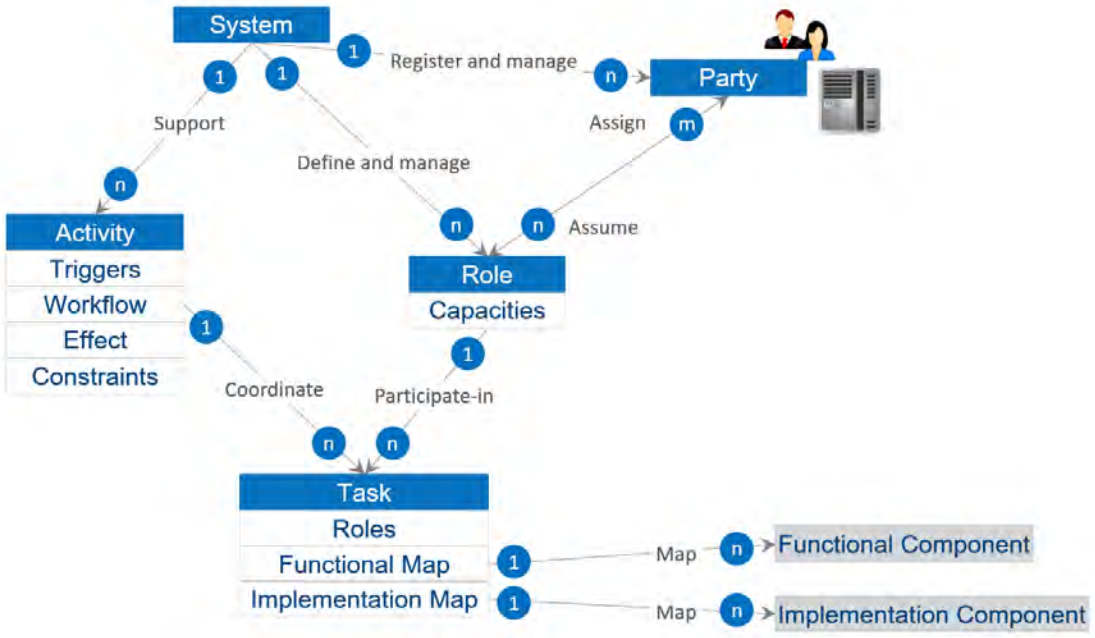

Figure 5-1: Role, Party, Activity and Task[1]

Figure 5-1 depicts the usage viewpoint's main concepts and how they relate to each other.

The basic unit of work is a *task*, such as the invocation of an operation, a transfer of data or an action of a party. A task is carried out by a party assuming a role.

A *role* is a set of capacities assumed by an entity to initiate and participate in the execution of, or consume the outcome of, some *tasks* or functions in an IIoT system as required by an *activity*. Roles are assumed by parties. A *party* is an agent, human or automated, that has autonomy, interest and responsibility in the execution of tasks. A party executes a task by assuming a role that has the right capacities for the execution of the task. A party may assume more than one role, and a role may be fulfilled by more than one party. A party also has security properties for assuming a role.

[1] The numbers and letters in the figure denote the quantitative relations between the elements, e.g. along the arrowed line from System to Activity, they denote a (1) System supports many (n) Activities; along the Party and Role arrowed line, they denote a Party (can) assume many (n) Roles; a Role (can) be assigned to many (m) parties.

Note

The above definition of *role* is primarily operational, based on capacities that qualify an agent from a functional perspective. It does not intend to be by itself an access control model for security purposes. However, because assuming a role implies access to the IIoT system, it is often associated with certain security properties (privileges, permissions, etc.). This association in turn may require a more refined notion of role—e.g. reflective of an organization chart, user groups—that is out of scope for this section. A *party* also has security properties (credentials, ID…) for assuming a *role*, the details of which are beyond the scope of this section.

A task has a role, a functional map and an implementation map.

- *A role* refers to, if applicable, the role(s) responsible for the execution of the task.
- *A functional map* describes to which functions or functional components the task maps. This can be defined only when the functional deposition of the system becomes available to perform the mapping. This mapping includes definition of inputs and outputs in the context where this *task* is to be executed (i.e. of a particular activity).
- *An implementation map* describes the implementation component(s) the task relies on for its execution. If role(s) are associated to the task, the map also defines how these roles map their capacities to the component(s) and related operations. Similarly, this property may be defined only when the implementation architecture of the system become available to perform the mapping.

Example

Examples of tasks and roles are:

- register a new device to the edge gateway (role: administrator),
- run test procedure for passive RFID readers on processing chain X (roles: administrator, QA),
- authenticate user request (role: security agent) and
- summarize data streams from all temperature sensors on asset X (role: same as the role that initiates the edge-to-cloud data flow processing and consolidation activity that this task is part of).

An *activity* is a specified coordination of tasks (and possibly of other activities, recursively) required to realize a well-defined usage or process of an IIoT system. An activity may be executed repeatedly. An activity has the following elements:

- A *trigger* is one or more condition(s) under which the activity is initiated. It may be associated with one or more role(s) responsible for initiating or enabling the execution.
- A *workflow* consists of a sequential, parallel, conditional, iterative organization of tasks.
- An *effect* is the difference in the state of the IIoT system after successful completion of an activity.
- *Constraints* are system characteristics that must be preserved during execution and after the new state is achieved, such as data integrity, data confidentiality and resilience. These characteristics may be affected by the enacting of the tasks beyond what is enforceable by the system design or its functional components alone.


Example

An example of *activity* is of a device on-boarding procedure:

Trigger: Administrator approval of the new addition.

Workflow:

- Task 1: Register new device to the Edge gateway.
- Task 2: Register the new device in the Cloud-based management platform by automatic discovery and querying of all gateways.
- Task 3: Run remote test procedure appropriate for this device type and verify that values generated are within expected range and consistent with similar devices in the proximity.

Initially, an abstract description of the activity is sufficient. During design, the activities serve as inputs to the requirements for the system, thus guiding the design of the functional architecture and its components. An activity then requires each task to be mapped to, and supported by, one or more functions. An activity is not restricted to one functional domain but may involve a sequence of tasks that span several functional domains.[1]

The design of the IIoT system now has a concrete representation of the activities by mapping its tasks to the functional and implementation components. The mappings then enable architecture and implementation verification.

[1] Defined in the functional viewpoint (see section 6.9**Error! Reference source not found.**)

6 FUNCTIONAL VIEWPOINT

6.1 BACKGROUND

Industrial Control Systems (ICSs) have been widely deployed to enable industrial automation across industrial sectors.[1] As we bring these automated control systems online with broader systems in the industrial internet effort, control remains a central and essential concept of industrial systems. Control, in this context, is the process of automatically exercising effects on physical systems and the environment, based on sensory inputs to achieve human and business objectives. Many control systems today apply low-latency, fine-grained controls to physical systems in close proximity, without a connection to other systems. Because of this, it is difficult to create local collaborative control, let alone globally orchestrated operations.

Some might argue that the industrial internet is the conjoining of what has been traditionally two different domains with different purposes, standards and supporting disciplines: IT and OT.[2,3] In IT (information technology), everything is reducible to bits that represent ideas in the programmer's head and transformed in a way to produce useful inference—anything from the sum of numbers in a column to email systems to schedule optimization problems (e.g. using Simplex). The essential problem with such an approach, noted as one of the fundamental problems in the artificial intelligence community, is the so-called 'symbol-grounding problem'—that symbols in the machine (the numbers passed around by the processor) only correspond to world objects because of the intentions of the programmer—they have no meaning to the machine.[4,5] In OT (operational technology), 'controls' (traditionally analogue) have been applied directly to physical processes without any attempt to create symbols or models to be processed by the machine. For example, PID (proportional-integrative-derivative) controllers may control the voltage on a line using a particular feedback equation that is defined by the control engineer and demonstrated to work for a particular application—there is no attempt at generality and no need to divide the problem among multiple processing units. The incidence of IT into the OT world has primarily come about due to a need to network larger systems and establish control over hierarchies of machines while also wanting to inject common IT ideas into the OT world (such as scheduling and optimization of resource consumption). There has also been a move toward controls that digitally simulate the physical world and base their control decisions on the simulation model rather than a control engineer's equation. This makes other kinds of approaches that have been examined in IT, such as machine learning, possible to apply to OT.

[1] ICSs typically include supervisory control and data acquisition (SCADA) [5] systems, distributed control systems (DCS) [7], and other control system configurations such as skid-mounted Programmable Logic Controllers (PLC) [6] that are often found in the industrial control sectors.

[2] Here we use the more traditional version of the word 'domain'.

[3] Consider this an introduction to the topic—we will deal with the IT/OT problem in more detail in future versions of this and other documents.

[4] *http://en.wikipedia.org/wiki/Symbol_grounding_problem*; also see '*The Symbol Grounding Problem*', Stevan Harnad [8].

[5] *http://en.wikipedia.org/wiki/Chinese_room*; also see '*Minds, Brains and Programs*', John Searle [9].

This has also led to OT systems to be susceptible to IT problems as well, such as network denial of service attack and spoofing as well as the aforementioned symbol-grounding problem. The combination of IT and OT is a great advance—cognition embodied into an industrial system. This could avoid the symbol-grounding problem by basing its representation on the world (not on programmer-supplied models) and only its own episodic experience (and thus not limited to human conceptions of epistemology). However, even nearer-term breakthroughs that support advanced analytics based on real world data rather than engineering models may yield substantial improvements. The main obstacles are safety and resiliency. Mission-critical OT applications, where failures would jeopardize life or human wellbeing, require reliability at a level much higher than those typically found in IT applications. Therefore, the reliability level typically found in IT systems may not be acceptable in mission-critical OT applications. Moreover, actions in the physical world generally cannot be undone, which is a consideration that IT systems normally do not have to address.

Advancement of computation and communication technologies of recent years can be applied to the industrial internet to dramatically transform industrial control systems in two major themes:

Increasing local collaborative autonomy: New sensing and detection technologies provide more and more accurate data. Greater embedded computational power enables more advanced analytics of these data and better models of the state of a physical system and the environment in which it operates. The result of this combination transforms control systems from merely automatic to autonomous, allowing them to react appropriately even when the system's designers did not anticipate the current system state. Ubiquitous connectivity between peer systems enables a level of fusion and collaboration that was previously impractical.

Increasing system optimization through global orchestration: Collecting sensor data from across the control systems and applying analytics, including models developed through machine learning, to these data, can provide insight to a business's operations. With these insights, improved decision-making and optimized system operations can be achieved globally through automatic and autonomous orchestration.

These two themes have far-reaching impact on the systems that we will build, though each system will have a different focus and will balance the two themes differently.

6.2 THE FUNCTIONAL VIEWPOINT AND FUNCTIONAL DOMAIN

The *functional viewpoint* is an architecture viewpoint that frames the concerns related to the functional capabilities and structure of an industrial internet of things (IIoT) system and its components.

To analyze the functional concerns effectively, the concept of functional domain is introduced as means to decompose the overall functional concerns. A *functional domain* is a (mostly) distinct functionality in the overall IIoT system. A decomposition of a typical IIoT system into functional domains highlights the important building blocks that have wide applicability in many industrial verticals. It is a starting point for conceptualizing a concrete functional architecture. Specific

system requirements will strongly influence how the functional domains are decomposed, what additional functions may be added or left out and what functions may be combined and further decomposed.

We decompose a typical IIoT system into five functional domains:

- Control Domain
- Operations Domain
- Information Domain
- Application Domain
- Business Domain

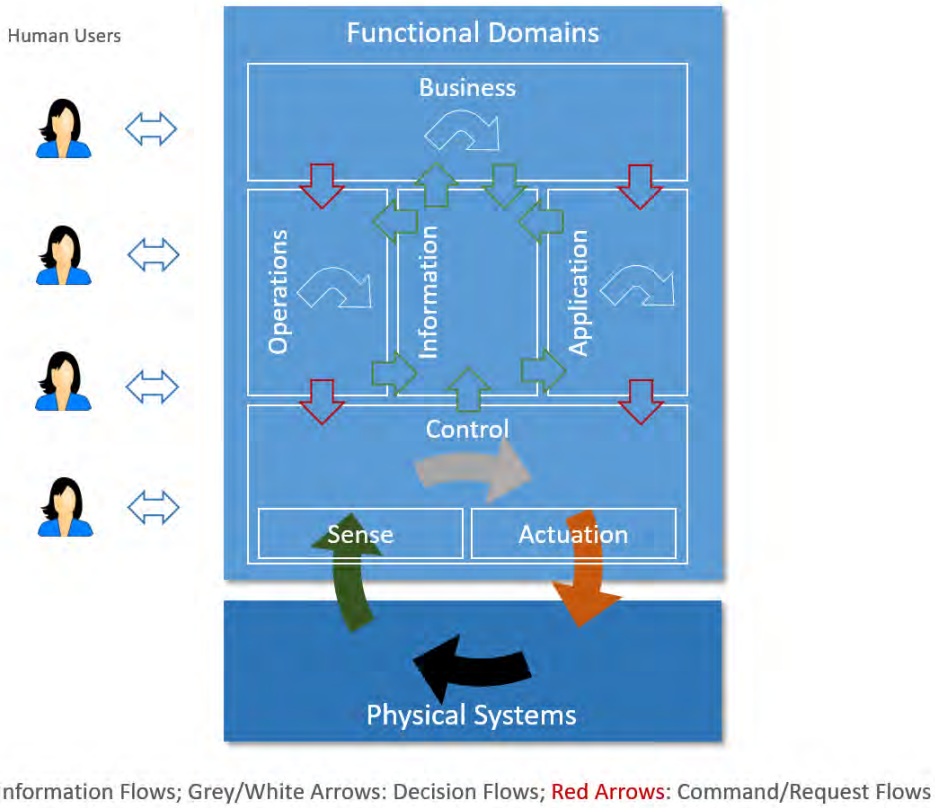

Figure 6-1: Functional Domains

Data flows and control flows take place in and between these functional domains. Figure 6-1 above illustrates how the functional domains relate to each other with regard to data and control flows. Green arrows show how data flows circulate across domains. Red arrows show how control flows circulate across domains. Other horizontal arrows illustrate some processing taking place within each domain, to process input flows and generate new forms of data or control flows.

Controls, coordination and orchestration exercised from each of the functional domains have different granularities and run on different temporal cycles. As it moves up in the functional domains, the coarseness of the interactions increases, their cycle becomes longer and the scope

of impact likely becomes larger. Correspondingly, as the information moves up in the functional domains, the scope of the information becomes broader and richer, new information can be derived, and new intelligence may emerge in the larger contexts.

6.3 THE CONTROL DOMAIN

The *control domain* is a functional domain for implementing industrial control systems. It represents the collection of functions that are performed by industrial control and automation systems. The core of these functions comprises fine-grained closed-loops, reading data from sensors ("sense" in the figure), applying rules and logic, and exercising control over the physical system through actuators ("actuation").[1] Such closed-loop systems usually require high timing accuracy and resolution. Components or systems implementing these functions (functional components) are usually deployed in proximity to the physical systems they control and may therefore be geographically distributed. These components and systems may not be easily accessible physically by maintenance personnel, and physical security of these systems may require special consideration. On the other hand, there are increasing number of industrial automation systems that are mobile, e.g. robotic machines in a manufacturing floor; providing low latency and reliable wireless communication to and between these mobile systems would require additional consideration.

Simple examples of functional components in this domain include control units in electricity utility plant, control units in a wind-turbine, and control units in autonomous vehicles.

The control domain comprises a set of common functions, as depicted in Figure 6-2.[2] Their implementation may be at various levels of complexity and sophistication depending on the systems, and, in a given system, some components may not exist at all. These are abstract functions that many control or automation units provide in various forms and in different architectures. This abstract description aims for general understanding of the control domain function. An IIoT system is not intended to replace the existing implementation of control and automation system in the control domains but seek to establish connectivity, to gather data and provide optimization feedback to them through advanced analytics often in a large context than the individual control and automation system. We describe each of the control domain functions below.

Sensing is the function that reads sensor data from sensors. Its implementation spans hardware, firmware, device drivers and software elements. Note that *active sensing* recursively, requires control and actuation, and may therefore have a more complex linkage to the rest of the control system; for example, an attention element to tell the sensor what is needed.

[1] Possibly in a hierarchy, at several levels.

[2] These set of functions are considered essential to many control systems in the control domain. However, they may exist in other domains as well.

Actuation is the function that writes data and control signals to an actuator to enact the actuation. Its implementation spans hardware, firmware, device drivers and software elements.

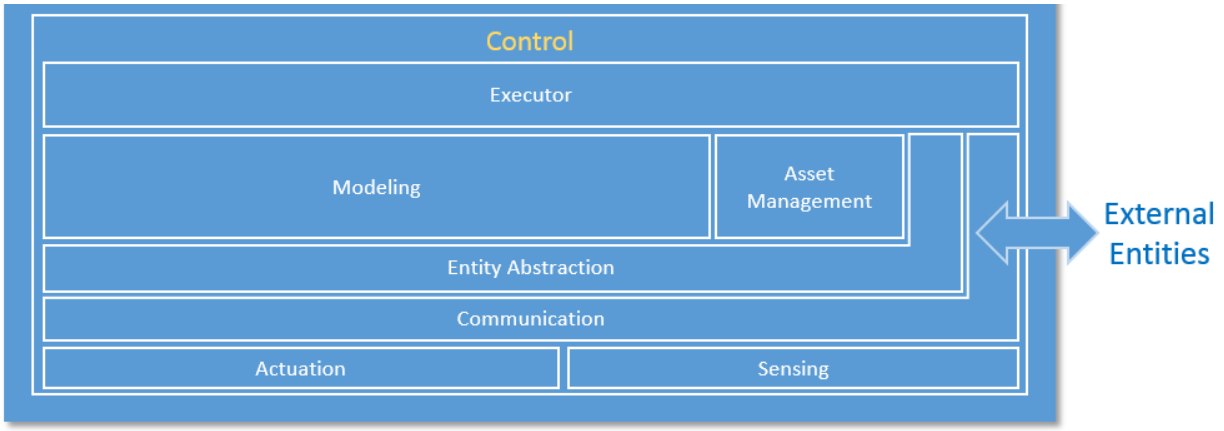

Figure 6-2: Functional Decomposition of Control Domain

Communication connects sensors, actuators, controllers, gateways and other edge systems. The communication mechanisms take different forms, such as a bus (local to an underlying system platform or remote), or networked architecture (hierarchical, hubs and spokes, meshed, point-to-point), some statically configured and others dynamically. Quality of Service (QoS) characteristics such as latency, bandwidth, jitter, reliability and resilience must be taken into account.

Within the communication function, a connectivity abstraction function may be used to encapsulate the specifics of the underlying communication technologies, using one or more common APIs to expose a set of connectivity services. These services may offer additional connectivity features that are not otherwise available directly from the underlying communication technologies, such as reliable delivery, auto-discovery and auto-reconfiguration of network topologies upon failures.

Entity abstraction, through a virtual entity representation, provides an abstraction of scores of sensors and actuators, peer controllers and systems in the next higher tiers, and expresses relationships between them. This serves as the context in which sensor data can be understood, actuation is enacted and the interaction with other entities is carried out. Generally, this includes the semantics of the terms used within the representations or messages passed between system elements.

Modeling deals with understanding the states, conditions and behaviors of the systems under control and those of peer systems by interpreting and correlating data gathered from sensors and peer systems. The complexity and sophistication of modeling of the system under control varies greatly. It may range from straightforward models (such as a simple interpretation of a time series of the temperature of a boiler), to moderately complex (a prebuilt physical model of an aircraft engine), to very complex and elastic (models built with artificial intelligence possessing learning and cognitive capabilities). These modeling capabilities, sometime referred to as edge analytics, are generally required to be evaluated locally in control systems for real-time

applications. Edge analytics are also needed in use cases where it is not economical or practical to send a large amount of raw sensor data to remote systems to be analyzed even without a real-time requirement.

A data abstraction sub-function of modeling may be needed for cleansing, filtering, de-duplicating, transforming, normalizing, ignoring, augmenting, mapping and possibly persisting data before the data are ready for analysis by the models or destroyed.

Asset management enables operations management of the control systems including system onboarding, configuration, policy, system, software/firmware updates and other lifecycle management operations. Note that it is subservient to the executor so as to ensure that policies (such as safety and security) are always under the responsibility and authority of the edge entity.

Executor executes control logic to the understanding of the states, conditions and behavior of the system under control and its environment in accordance with control objectives. The control objectives may be programmed or otherwise set by static configuration, be dynamic under the authority of local autonomy, or be advised dynamically by systems at higher tiers. The outcome of the control logic may be a sequence of actions to be applied to the system under control through actuation. It may also lead to interactions with peer systems or systems at higher tiers. Similar to the case of modeling, the control logic can be:

- straightforward—a set-point program employing algorithms to control the temperature of a boiler) or
- sophisticated—incorporating aspects of cognitive and learning capabilities with a high degree of autonomy, such as deciding which obstacle a vehicle should crash into—the full school bus pulling out in front of the vehicle from the grade school or the puddle of pedestrians in front of the nursing home?[1]

The executor is responsible for assuring policies in its scope are applied so that data movement out of the scope it controls, use of actuators, etc. are within the bounds of such policies.

6.4 THE OPERATIONS DOMAIN

The *operations domain* is functional domain for management and operation of the control domain. It represents the collection of functions responsible for the provisioning, management, monitoring and optimization of the systems in the control domain. Existing industrial control systems mostly focus on optimizing the assets in a single physical plant. The control systems of the Industrial Internet must move up a level, and optimize operations across asset types, fleets and customers. This opens opportunities for added business and customer value as set out by higher-level, business-oriented domains.

[1] Such ethical choices are the subject of 'trolley problems' and are indicative of deep policy issues that typically should not be left up to a programmer to decide. See, e.g., *https://en.wikipedia.org/wiki/Trolley_problem*; also see '*The Trolley Problem*', Judith J. Thomson [10].

Example

Optimizing the operation of one train has obvious cost savings, but optimizing train operations and routes across a fleet yields more, and combining data from fleets owned by different railroads can optimize the utilization of the rail network within a country.

Figure 6-3 shows how operations in an IIoT system can be supported through a suite of interdependent operations support functions.

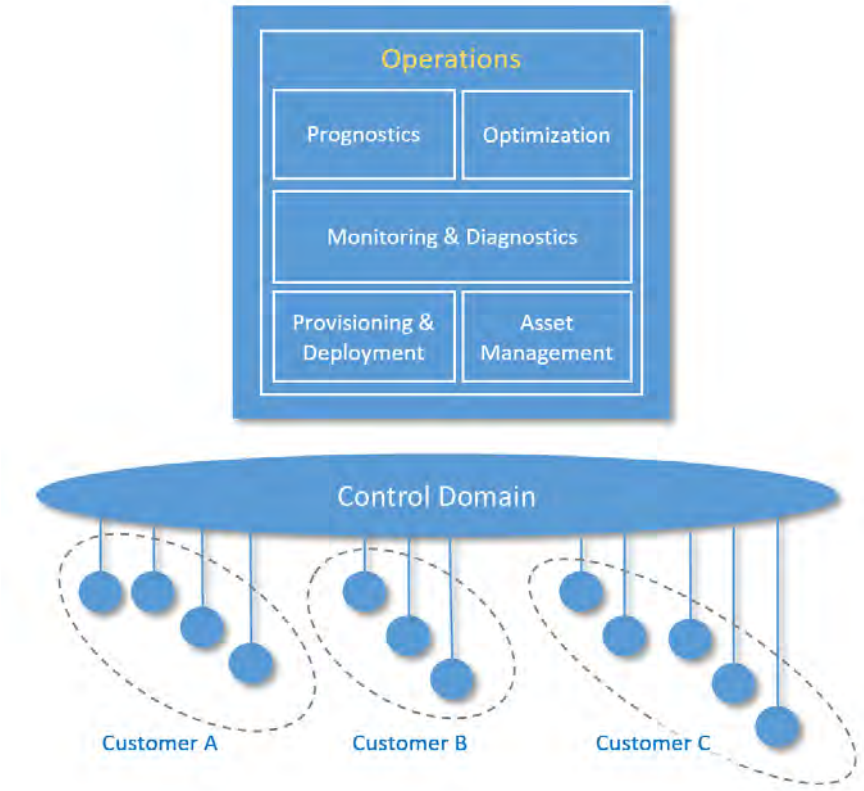

Figure 6-3: Operations Domain decomposition showing support across various customers

Provisioning and Deployment consists of a set of functions required to configure, onboard, register, and track assets, and to deploy and retire assets from operations. These functions must be able to provision and bring assets online remotely, securely and at scale. They must be able to communicate with them at the asset level as well as the fleet level, given the harsh, dynamic and remote environments common in industrial contexts.

Asset Management consists of a set of functions that enable assets management centers to issue a suite of management commands to the control systems, and from the control systems to the assets in which the control systems are installed, and in the reverse direction enable the control systems and the assets to respond to these commands. For this, many of the legacy “dumb” assets need to be retrofitted to have compute, storage and connectivity capabilities.

Monitoring and Diagnostics consists of functions that enable the detection and prediction of occurrences of problems. It is responsible for real-time monitoring of asset key performance indicators, collecting and processing asset health data with intelligence so that it can diagnose

the real cause of a problem, and then alerting on abnormal conditions and deviations. This set of functions should assist operations and maintenance personnel to reduce the response time between detecting and addressing a problem.

Prognostics consists of the set of functions that serves as a predictive analytics engine of the IIoT systems. It relies on historical data of asset operation and performance, engineering and physics properties of assets, and modeling information. The main goal is to identify potential issues before they occur and provide recommendations on their mitigation.

Optimization consists of a set of functions that improves asset reliability and performance, reduces energy consumption, and increase availability and output in correspondence to how the assets are used. It helps to ensure assets operating at their peak efficiency by identifying production losses and inefficiencies. This process should be automated, as much as it is feasible, in order to avoid potential inaccuracies and inconsistencies.

At this level, this set of functions should support major automation and analytics features including:

- automated data collection, processing and validation,
- the ability to capture and identify major events, such as downtime, delay and
- the ability to analyze and assign causes for known problems.

Furthermore, many of the core functions in the operations domain, such as diagnostics, prognostics and optimization, may require performing advanced analytics on potentially large volume of historical asset operational and performance data. Therefore, an optimal approach is to use or share these functionalities that are available by the information domain.

6.5 THE INFORMATION DOMAIN

The *information domain* is a functional domain for managing and processing data. It represents the collection of functions for gathering data from various domains, most significantly from the control domain, and transforming, persisting, and modeling or analyzing those data to acquire high-level intelligence about the overall system.[1] The data collection and analysis functions in this domain are complementary to those implemented in the control domain. In the control domain, these functions participate directly in the immediate control of the physical systems whereas in the information domain they are for aiding decision-making, optimization of system-wide operations and improving the system models over the long term. Components implementing these functions may or may not be co-located with their counterparts in the control domain. They may be deployed in building closets, in factory control rooms, in corporate datacenters, or in the cloud as a service.

Optimizing the electricity generation level of a plant or a generator based on the condition of the facility, fuel cost and electricity price.

[1] Possibly in a hierarchy, at several levels.

> Changing the route of a fleet of freight trucks based on weather, traffic and the condition of the goods in the trucks.
>
> Changing the output of an automated production plant based on condition of the facility, energy and material cost, demand patterns and logistic.
>
> Changing the temperature set-point of a boiler based on energy cost, weather condition and usage pattern.

Figure 6-4 illustrates the functional decomposition of the information, application and business domains.

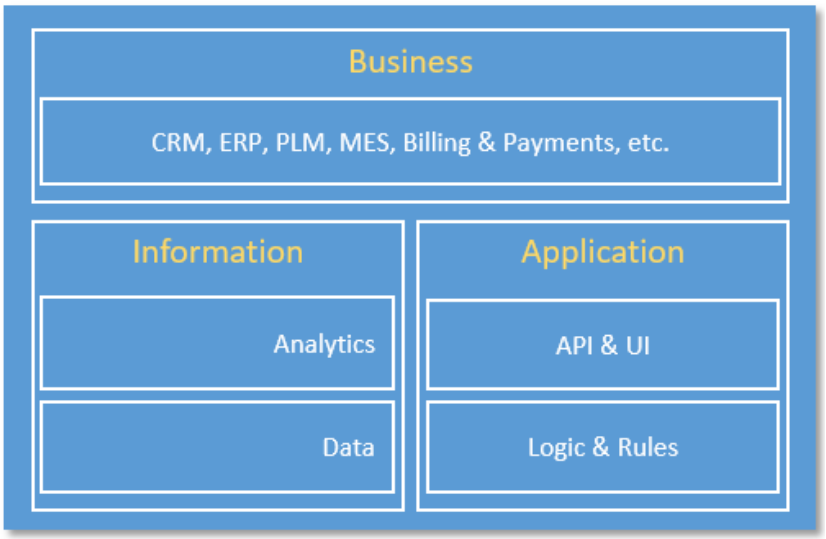

Figure 6-4: Functional Decomposition of Information, Application & Business Domains

Data consists of functions for:

- ingesting sensor and operation state data from all domains,
- quality-of-data processing (data cleansing, filtering, de-duplication, etc.),
- syntactical transformation (e.g., format and value normalization),
- semantic transformation (semantic assignment, context injection and other data augmentation processing based on metadata (e.g. provisioning data from the Operations Domain) and other collaborating data set,
- data persistence and storage (e.g. for batch analysis) and
- data distribution (e.g. for streaming analytic processing).

These functions can be used in online streaming mode in which the data are processed as they are received to enable quasi-real-time analytics in support of orchestration of the activities of the assets in the control domain. They may be used in offline batch mode (e.g. seismic sensor data collected and accumulated in an offshore oil platform that does not have high-bandwidth connectivity to the onshore datacenter).

Data governance functions may be included for data security, data access control and data rights management, as well as conventional data management functions related to data resilience (replication in storage, snapshotting and restore, backup & recovery, and so on).

Analytics encapsulates a set of functions for data modeling, analytics and other advanced data processing, such as rule engines. The analytic functions may be done in online/streaming or offline/batch modes. In the streaming mode, events and alerts may be generated and fed into functions in the application domains. In the batch mode, the outcome of analysis may be provided to the business domain for planning or persisted as information for other applications.

The data volume at the system level in most IIoT systems will eventually exceed a threshold at which the traditional analytic toolsets and approaches may no longer scale in meeting the requirement in performance. “Big Data” storage and analytic platforms may be considered for implementing these functions.

6.6 THE APPLICATION DOMAIN

The *application domain* is a functional domain for implementing application logic. It represents the collection of functions implementing application logic that realizes specific business functionalities. Functions in this domain apply application logic, rules and models at a coarse-grained, high level for optimization in a global scope. They do not maintain low-level continuing operations, as these are delegated to functions in the control domain that must maintain local rules and models in the event of connectivity loss. Requests to the control domain from the application domain are advisory so as not to violate safety, security, or other operational constraints.

The decomposition of the application domain is illustrated in Figure 6-4.

Logics and Rules comprises logics (rules, models, engines, activity flows, etc.) implementing specific functionality that is required for the use case under consideration. It is expected that there are great variations in these functions in both its contents and its constructs among the use cases.

APIs and UI represent a set of functions that an application exposes its functionalities as APIs for other applications to consume, or human user interface enabling human interactions with the application.

6.7 THE BUSINESS DOMAIN

The *business domain* functional domain for implementing business functional logic. It represents business functions supporting business processes and procedural activities business functions that an IIoT system must integrate to enable end-to-end operations of IIoT systems. Examples of these business functions include Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Product Lifecycle Management (PLM), Manufacturing Execution System (MES), Human Resource Management (HRM), asset management, service lifecycle management, billing and payment, work planning and scheduling systems.

A predictive maintenance service for an oilrig may have an application that forecasts failures in the field. To do so, it may require a resource planning system to ensure the required parts are available and reserved, and it may need to connect to internal or partner's service work schedule system and logistics management system, as well as the customer's, to schedule the field service.

6.8 CROSSCUTTING FUNCTIONS AND SYSTEM CHARACTERISTICS

The functional components described so far in the functional domains focus on major system functions that are generally required to support generic IIoT usages and to realize generic IIoT system capabilities for business purpose. However, additional functions must be provided to enable the major system functions. Often these enabling functions, the so-called crosscutting functions, need to be made available across many of the system functional components. For example, system functions need to be connected so they can interact with each other to complete functionality at the system level. Therefore, connectivity is considered a crosscutting function. An important element in the industrial internet is the application of analytics on the data gathered from the industrial assets and control systems to gain insights on their operations. To enable analytics on these asset data, many of the system functional components require concerted effort on data management. Therefore, data management is also considered a crosscutting function.

On the other hand, the aggregate behavior of an IIoT system is not the simple sum of what is provided by its constituent functional components. Like any complex system, there are emergent behaviors or properties resulting from the interactions of the constituent parts. These emergent system-wide properties are called system characteristics. See section 3.6.

The system and crosscutting functional analysis largely concerns how the system works while the analysis of system characteristics emphasizes how well the system works. For example, to ensure security in a system, a certain set of security functions, as crosscutting functions, must be implemented in each of the functional components and their communications, such as encryption and authentication. However, how secure the system as a whole is depends on how these functions are implemented and how securely these functional components are integrated and interact with each other—as an emergent property. In fact, a system is as secure as its weakest component, or link between components. The same is true for safety, resilience and any other system property. The trust into a system based on a set of such system characteristics, including safety, security, resilience, reliability and privacy, is defined as trustworthiness of the system. Please refer to the *'G4: Industrial Internet Security Framework (IISF)*' [3] for detailed analysis of the trustworthiness of IIoT systems.

The realization of a system characteristic to a certain desired level may depend on, constrain—and in some cases run against—those of other system characteristics. For example, one cannot ascertain a system is safe without also ascertaining it is also secure. On the other hand, an

inadequately implemented security measure may be hazardous to safety.[1] Please refer to '*G4: Industrial Internet Security Framework (IISF)*' [3] for detailed analysis on the entwined dependencies and conflicts that can occur among and between system characteristics.

This reference architecture places a strong emphasis on both the functions needed to support the system's business purpose and ensuring adequate system characteristics so that the functions are performed correctly and the business purpose is not compromised. The crosscutting functions and system characteristics are discussed in '*G2—Key IIoT System Concerns'*, '*G4—Industrial Internet Security Framework (IISF)*' [3] and other volumes.

The relationship between the functional domains, and crosscutting functions and key system characteristics are summarized in Figure 6-5:

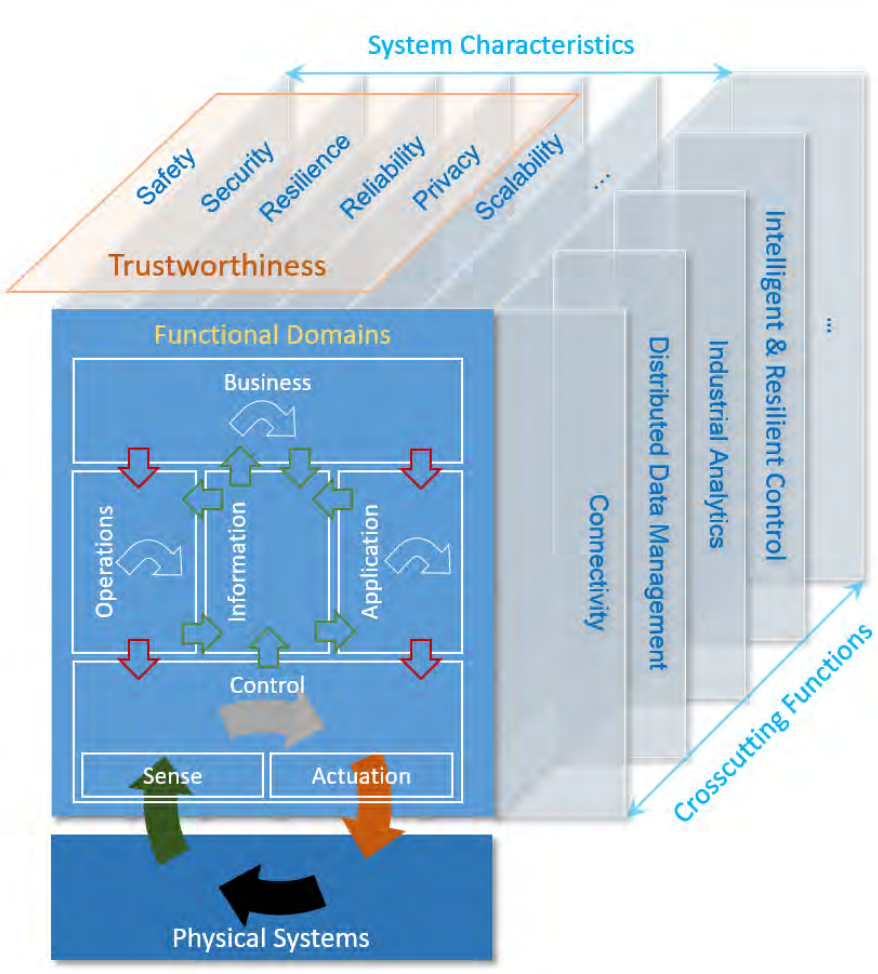

Figure 6-5: Functional Domains, Crosscutting Functions and System Characteristics

[1] For example, a locked-up fire escape door may provide strong security against unauthorized entrance or exit but may also prevent necessary escape in an emergency.

6.9 FUNCTIONAL DOMAINS AND COMPUTATIONAL DEPLOYMENT PATTERNS

The convergence of operational technology and information technology that enables the industrial internet is driven by technology advances in ubiquitous connectivity and pervasive computation. It is informative to map the IIoT functional domains to connectivity and computational deployment patterns as a high-level guide in how these functional domains could be distributed.

Clearly, connectivity can be considered as the foundation connecting the computational capabilities, enabling information sharing and collaborative operations among computers, machines and people. Near the network peripheries, the advances in connectivity, such as high-performance and low-power wireless communication, make it possible to connect to large numbers of industrial assets without the cost of laying wires to reach them. Within large data centers, Software Defined Network (SDN) is maturing, making it possible for applications to manage their networking dynamically.

Meanwhile, technologies concerning computational deployment patterns, which involve the location and placement of computational capability (including applications, data and services), continue to evolve.

On one hand, large-scale computation capability has become available on demand with unprecedented scalability, accessibility, availability and elasticity at low cost through the economy of scale at large data centers, thereby leading to the advent of cloud computing. This is made possible by advances in virtualization technologies, including containerization technologies, and the maturing of Infrastructure-as-as-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS) technologies.

The cloud computing approach and platforms in virtualizing and managing computation resources are being more broadly adopted in enterprise datacenters. These computational patterns characterized by a high concentration of computing capabilities in public-, private-, hybrid-clouds, on premise and remotely hosed data centers—collectively called the concentrated computational pattern, or the concentrated pattern for short—offer unparalleled scale of computational resources with elasticity at low cost.

On the other hand, computational capabilities ranging from very limited (e.g. in a smart sensor) to reasonably rich (e.g. a cluster of servers) are placed in, attached to or collocated with physical systems (e.g. in smart sensors, devices or machines). These capabilities are connected through a variety of connectivity technologies including wireless. Some of the computational capabilities and connectivity technologies are energy-constrained (e.g. powered by a battery). This computational deployment pattern is called dispersed computational pattern, or dispersed pattern for short.

The traditional industrial control systems can be considered an example of dispersed computational pattern at the network peripheries where computation, albeit mostly embedded, is performed at controllers connected to a network or even in isolation.

There is a renewed movement to distribute computational capability toward the network peripheries, away from concentrated computation centers, to benefit from a reduction in movement of data and communication latency, and enhancement in local intelligent control and, more importantly, resilience.

Moreover, some of the technologies that have been originally developed for large data centers, such as virtualization and SDN, are being applied to manage computational and network resources in dispersed patterns at the network peripheries as well.

Generally, it is an architectural choice in what computational patterns to use and where to place various functions needed for an IIoT system across a network. With the advances in technologies available at the network peripheries, including the availability of smarter sensors and devices, most IIoT systems will involve the dispersed computational deployment pattern in conjunction with a concentrated pattern or on its own. Employing the concentrated pattern provides a higher degree of elasticity of computational capability and a simpler management of the computational resources. The concentrated pattern can be deployed anywhere across a network, including near or at its peripheries to benefit from a reduction in network bandwidth and latency, stronger local control and resilience. In many use cases, functions needing high computational capabilities with less stringent latency and reliability requirements can be placed in the concentrated pattern away from the network peripheries. Employing exclusively dispersed pattern at or near the network peripheries, e.g. through a pure peer-to-peer collaborative model, yields the highest level of resilience.

Some IIoT architectures adapt the concentrated pattern with a relatively flat and often a thin layer at the network peripheries leaving most of the computation being performed away from the peripheries.

For example, it may be efficient to connect a large number of remote sensors that measure air quality and other environmental parameters in a large metropolitan area to a cloud service and perform most of the analytics computation therein.

This computational pattern, however, may not be adequate for many IIoT systems where the computations need to be distributed across multiple, potentially hierarchical, layers close to the industrial assets at or near the network peripheries. For the cases where the industrial assets are dispersed and remote, e.g. turbine engines in a wind farm and oil rigs in an oil field, strong computational capability may be needed at or near the assets for local analytics and control.

As the industrial internet matures, more computation capability is expected to be added to or placed adjacent to the industrial systems to enable local intelligent and autonomous operations thus making computation more dispersed.

Traditionally, operational technology is deployed around the network peripheries while information technology is deployed away from them. Functions in the control and operations domain map to the operational technology and functions in the business, information and application domains map to the informational technology. However, these boundaries are vague

to begin with and will continue to blur as the industrial internet matures and the operational technology and information technology converges.

The IIRA does not constrain how its functional domains are distributed across the extent of the network or the computational patterns used. An optimal distribution pattern of the functional domains will be determined by the specific system context and requirements, the demand on the availability of the computational resource and the available technologies to support such distribution pattern.

6.10 THE HUMAN ROLES IN THE CREATION AND OPERATION OF AN IIOT SYSTEM

A human may play any number of roles in an IIoT system—who may conceptualize, design, build and operate the system, or whom may be monitored by the system as part of the physical system, e.g. being monitored for safety while working in a manufacturing or operating environment. In this section, we focus on the roles a human may play in operating an IIoT system from functional point of view.

As described in the usage viewpoint Figure 5-1, both human and system (or part of a system) can be a party playing a certain role in carrying out some specific tasks to complete an activity. These roles vary and are to be defined for each specific IIoT systems. However, they, as human users, may be abstracted in some way from functional point of view, as depicted in Figure 6-1.

The *human role in the control domain* is largely consistent with what has been in practice in deploying, operating, and maintaining industrial control and automation systems. With the new connectivity IIoT system brought to these systems, increasingly more of these practices can be done remotely and more effectively. With the increasing deployment of robotic machines and their increasingly mixed work environment with human operators in the factory floor, human is playing increasingly a cooperative role with the machines.

The *human role in the operations domain* largely consists of monitoring and maintaining the industrial systems, leveraging the collected data from machines and the advanced analytics performed on these data to move toward prognostic maintenance.

The *human role in the information domain* mostly concerns about data processing and management, and most importantly about building, validating and deploying analytic models into the IIoT system to gain insights from the data. The effectiveness of an IIoT system will increasingly depend on the quality of the analytic models.

The *human role in the application domain* deals with how to apply business logic to the analytic insights to optimize the operation of the machines, and how to use these application functions in daily operational processes.

The *human role in the business domain* from the industrial internet perspective is about how to use the insights gathered from the data to optimize business processes and decision making.

People can potentially take on three types of *roles during the operation* of a given IIoT system: users, entities of interest and participants. First, a person may be the user of the IIoT system, acting as an external entity that interacts with the IIoT system to gain some benefit. Second, a

person may be the (or an) entity of interest that the IIoT system is monitoring and acting upon. Finally, and perhaps the most difficult role to understand—people may participate as part of an IIoT system. In other words, people may interact with other components in the IIoT system to perform some function necessary for the success of the system operation. This final role is especially important due to the challenges of understanding; what capabilities a given person will provide, how those capabilities fit into the system design as a whole, and assuring that person is actually providing those capabilities when needed.

Not all humans support the goals of the IIoT system. *Intentional and unintentional human actions* can also lead to system failures, performance degradation, security problems, privacy breaches, etc. Well-designed IIoT systems should be robust in the face of human-generated problems including design errors and other user errors when using the IIoT system, implementation defects, configuration mistakes, intrusions, hacking, physical attacks and sabotage. For more details for how to deal with these negative aspects of human roles with regard to an IIoT system, please refer to the concepts of trustworthiness described in the Industrial Internet Security Framework.

7 IMPLEMENTATION VIEWPOINT

The *implementation viewpoint* is architecture viewpoint that frames the concerns related to implementing the capabilities and structure of an industrial internet of things (IIoT) system.

The implementation viewpoint is concerned with the technical representation of an IIoT system and the technologies and system components required implementing the activities and functions prescribed by the usage and functional viewpoints.

An IIoT system architecture and the choice of the technologies used for its implementation are also guided by the business viewpoint, including cost and go-to-market time constraints, business strategy in respect to the targeted markets, relevant regulation and compliance requirements and planned evolution of technologies.[1] The implementation must also meet the system requirements including those identified as key system characteristics that are common across activities and must be enforced globally as end-to-end properties of the IIoT system.

The implementation viewpoint therefore describes:

- the general architecture of an IIoT system: its structure and the distribution of components, and the topology by which they are interconnected;
- a technical description of its components, including interfaces, protocols, behaviors and other properties;
- an implementation map of the activities identified in the usage viewpoint to the functional components, and from functional components to the implementation components; and
- an implementation map for the key system characteristics.

[1] This version of the IIRA does not attempt to address regulatory and compliance requirements. These are substantially different by vertical and may be addressed in more detail in future documents.

This reference architecture focuses on system conceptualization and architecture highlighting important system concerns that may affect system lifecycle process that includes system design, development, deploy and operation. Each of this system lifecycle phases have its unique set of concerns that deserve attention. We do not cover concerns about system deployment or operation here, especially given the diverse deployment models in different environments across industrial verticals as it was briefly discussed in section 6.9.

7.1 EXAMPLE ARCHITECTURE PATTERNS

Architecture patterns represent some common, typical and essential features of IIoT implementations that are easy to recognize and understand by practitioners. They are examples and references for conceptualizing real world IIoT architectures, and IIoT architects may arrive final architectures that may substantially different from them.

Coherent IIoT system implementations follow certain well-established architectural patterns, such as:

- Three-tier architecture pattern
- Gateway-Mediated Edge Connectivity and Management architecture pattern
- Layered Databus pattern.

An architecture pattern is a simplified and abstracted view of a subset of an IIoT system implementation that is recurrent across many IIoT systems yet allows for variants. For example, an implementation of the three-tier pattern in an IIoT system does not exclude multiple implementations of every tier—e.g. many instances of the edge tier—as well as many-to-many connections between instances of a tier and instances of the next tier. Each tier and its connections will still be represented only once in the pattern definition.

We describe the three architecture patterns in more detail. The Gateway-Mediated Edge Connectivity and Layered Databus patterns are arguably specific instances or variations of the very general Three-Tier architecture pattern.

7.1.1 THREE-TIER ARCHITECTURE PATTERN

The three-tier architecture pattern comprises edge, platform and enterprise tiers. These tiers play specific roles in processing the data flows and control flows (see chapter 6) involved in usage *activities*. They are connected by three networks, as shown in Figure 7-1.

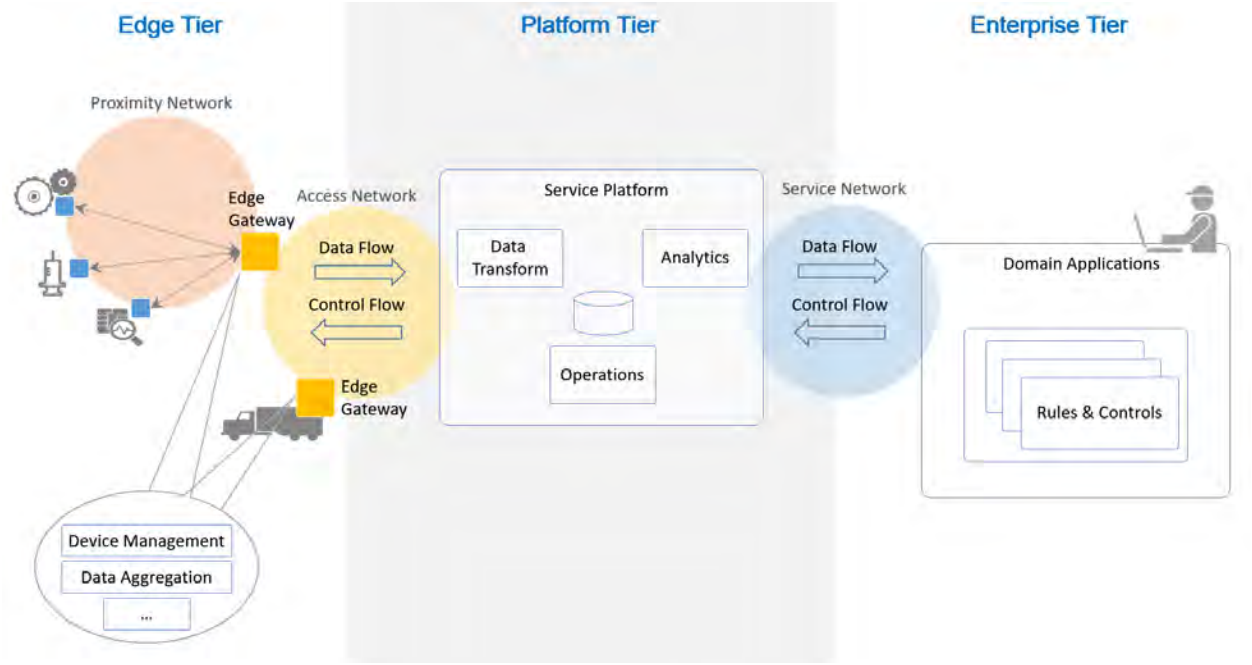

Figure 7-1: Three-Tier IIoT System Architecture

The *edge tier* collects data from the edge nodes, using the proximity network. The architectural characteristics of this tier, including the breadth of distribution, location, governance scope and the nature of the proximity network, vary depending on the specific use cases.

The *platform tier* receives, processes and forwards control commands from the enterprise tier to the edge tier. It consolidates processes and analyzes data flows from the edge tier and other tiers. It provides management functions for devices and assets. It also offers non-domain specific services such as data query and analytics.

The *enterprise tier* implements domain-specific applications, decision support systems and provides interfaces to end-users including operation specialists. The enterprise tier receives data flows from the edge and platform tier. It also issues control commands to the platform tier and edge tier.

Note

In the above figure, functional blocks are shown in each *tier*. These functional blocks are indicative of the primary functional location of the *tier* yet are not exclusively assigned to that *tier*. For example, the 'data transform' function in the *platform tier* could also be found in the *edge tier* (e.g. performed by a gateway) although it would be implemented in a different way and for a different purpose. For example, 'data transform' at the edge is typically done in a device-specific manner through device-specific configuration and interfaces, unlike in the platform tier where it is usually supported as a higher-level service that operates on data that has been abstracted from any device source or type.

Different networks connect the tiers:

The *proximity network* connects the sensors, actuators, devices, control systems and assets, collectively called edge nodes. It typically connects these edge nodes, as one or more clusters related to a gateway that bridges to other networks.

The *access network* enables connectivity for data and control flows between the edge and the platform tiers. For example, it could be a corporate network, an overlay private network over the public Internet or a 4G/5G network.

Service network enables connectivity between the services in the platform tier and the enterprise tier, and the services within each tier. It may be an overlay private network over the public Internet or the Internet itself, allowing the enterprise grade of security between end-users and various services.

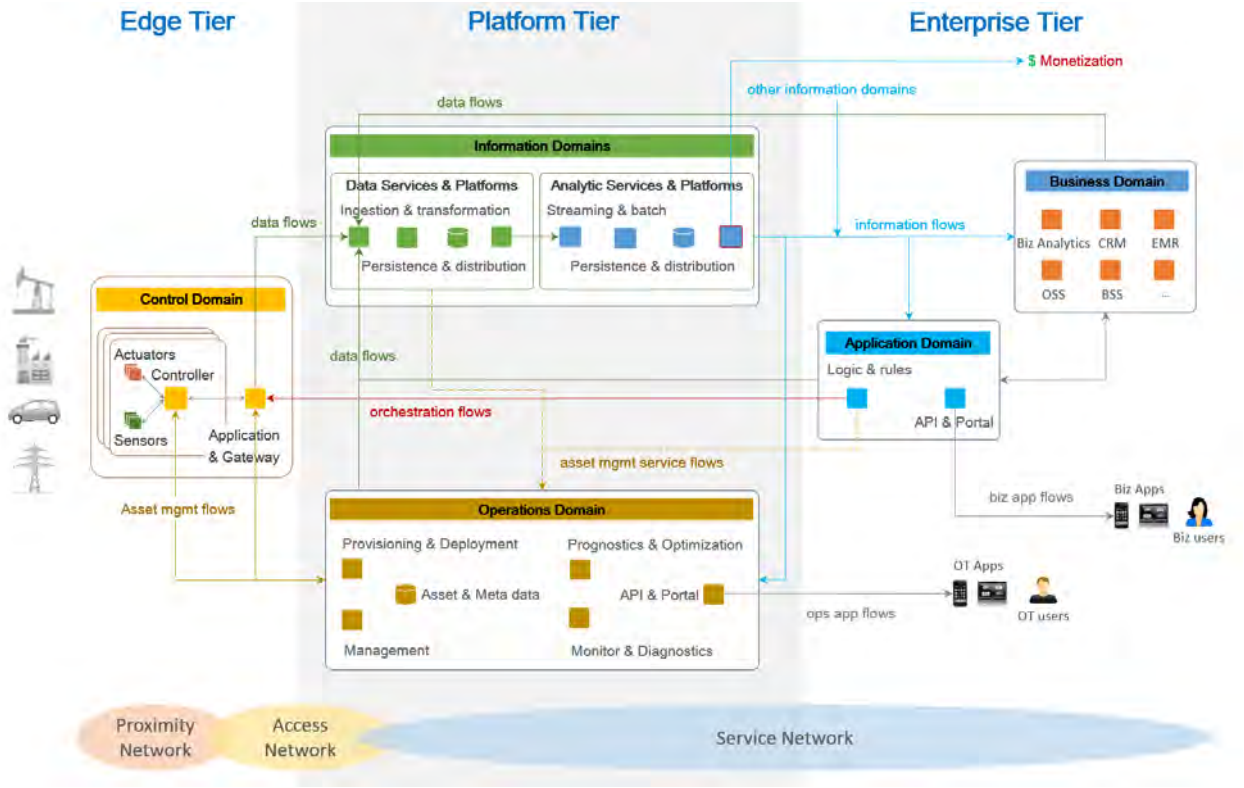

Figure 7-2: Mapping between a three-tier architecture to the functional domains

The three-tier architecture pattern combines major components (e.g. platforms, management services, applications) that generally map to the functional domains (functional viewpoint) as shown in Figure 7-2. From the tier and domain perspective, the edge tier implements most of the control domain; the platform tier most of the information and operations domains; the enterprise tier most of the application and business domains. This mapping demonstrates a simple functional partitioning across tiers. In a real system, the functional mapping of IIoT system tiers depends greatly on the specifics of the system use cases and requirements. For example, some functions of the information domain may be implemented in or close to the edge tier, along with some application logic and rules to enable intelligent edge computing.

Another reason why implementation tiers do not generally have an exclusive mapping to a particular functional domain is that these tiers often provide services to each other to complete the end-to-end activities of the system. These services, for example, data analytics from the information functional domain, then become supportive of other functional domains in other tiers.

The *asset management flows* (see Figure 7-2) is an expression of the operations domain component of the platform tier to manage the assets in the edge tier.

The operations domain component of the platform tier itself provides services (asset management service flows) to other components, either in the same tier or in another.

The data services (information domain) component of the platform tier may request services from the operations domain component for the verification of asset credentials it receives in the data flows from the edge tier, and query of asset metadata so it can augment the data received from the assets before the data are persisted or fed into analytics in the next stage of processing.

Similar operations domain services can be provided to the *application* domain components in the enterprise tier as well. Conversely, the operations domain components may use data services from the information domain component in order to get better intelligence from asset data, e.g. for diagnostics, prognostics and optimization on the assets.

As a result, components from all functional domains may leverage the same data and use analytic platforms and services to transform data into information for their specific purposes.

7.1.2 GATEWAY-MEDIATED EDGE CONNECTIVITY AND MANAGEMENT ARCHITECTURE PATTERN

The gateway-mediated edge connectivity and management architecture pattern comprises a local connectivity solution for the edge of an IIoT system, with a gateway that bridges to a wide area network as shown in Figure 7-3. The gateway acts as an endpoint for the wide area network while isolating the local network of edge nodes. This architecture pattern allows for localizing operations and controls (edge analytics and computing). Its main benefit is in breaking down the complexity of IIoT systems, so that they may scale up both in numbers of managed assets as well as in networking. However, it may not be suited to systems where assets are mobile in a way that does not allow for stable clusters within the local network boundaries.

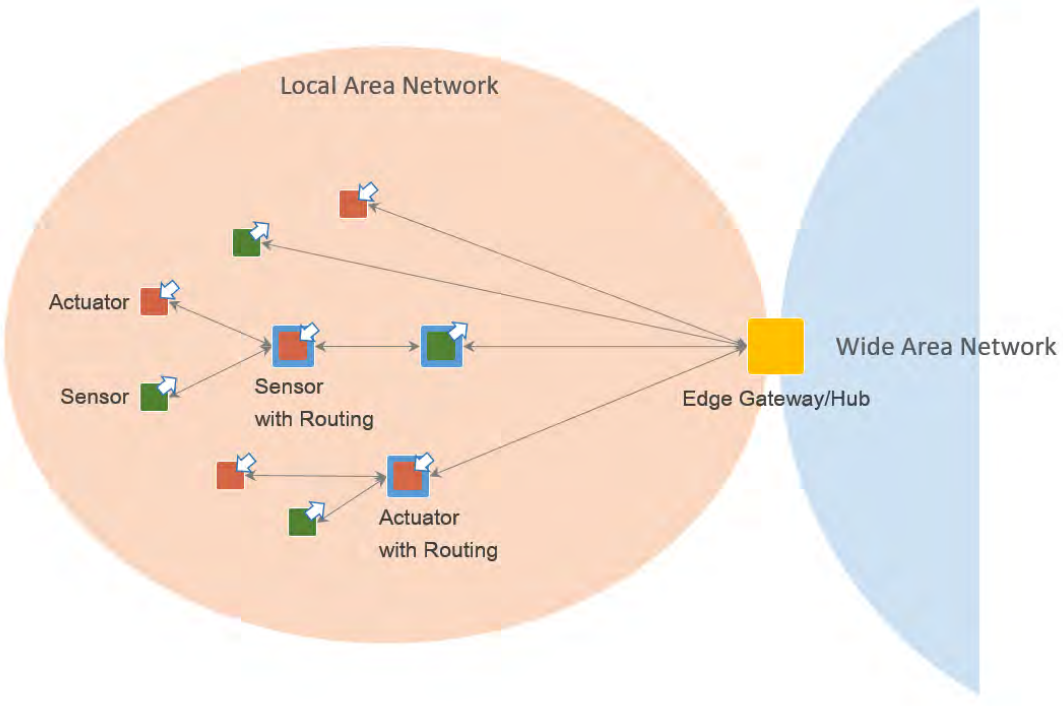

Figure 7-3: Gateway-Mediated Edge Connectivity and Management Pattern

The edge gateway may also be used as a management point for devices and assets and data aggregation point where some data processing and analytics, and control logic are locally deployed.

The local network may use different topologies as described below:

In a *hub-and-spoke* topology, an edge gateway acts as a hub for connecting a cluster of edge nodes to each other and to a wide area network. It has a direct connection to each edge entity in the cluster allowing in-flow data from the edge nodes, and out-flow control commands to the edge nodes.

In a *mesh network* (or peer-to-peer) topology, an edge gateway also acts as a hub for connecting a cluster of edge nodes to a wide area network. In this topology, however, some of the edge nodes have routing capability. As result, the routing paths from an edge node to another and to the edge gateway vary and may change dynamically. This topology is best suited to provide broad area coverage for low-power and low-data rate applications on resource-constrained devices that are geographically distributed.

In both topologies, the edge nodes are not directly accessible from the wide area network. The edge gateway acts as the single entry point to the edge nodes and as management point providing routing and address translation.

The edge gateway supports the following capabilities:

- *Local connectivity* through wired serial buses and short-range wireless networks. New communication technologies and protocols are emerging in new deployments.
- *Network and protocol bridging* supporting various data transfer modes between the edge nodes and the wide area network: asynchronous, streaming, event-based and store-and-forward.
- *Local data processing* including aggregation, transformation, filtering, consolidation and analytics.
- *Device and asset control and management point* that manages the edge nodes locally and acts an agent enabling remote management of the edge nodes via the wide area network.
- Site-specific decision and application logic that are perform within the local scope.

7.1.3 LAYERED DATABUS ARCHITECTURE PATTERN

The layered databus is a common architecture across IIoT systems in multiple industries (see Figure 7-4 below). This architecture provides low-latency, secure, peer-to-peer data communications across logical layers of the system. It is most useful for systems that must manage direct interactions between applications in the field, such as control, local monitoring and edge analytics.

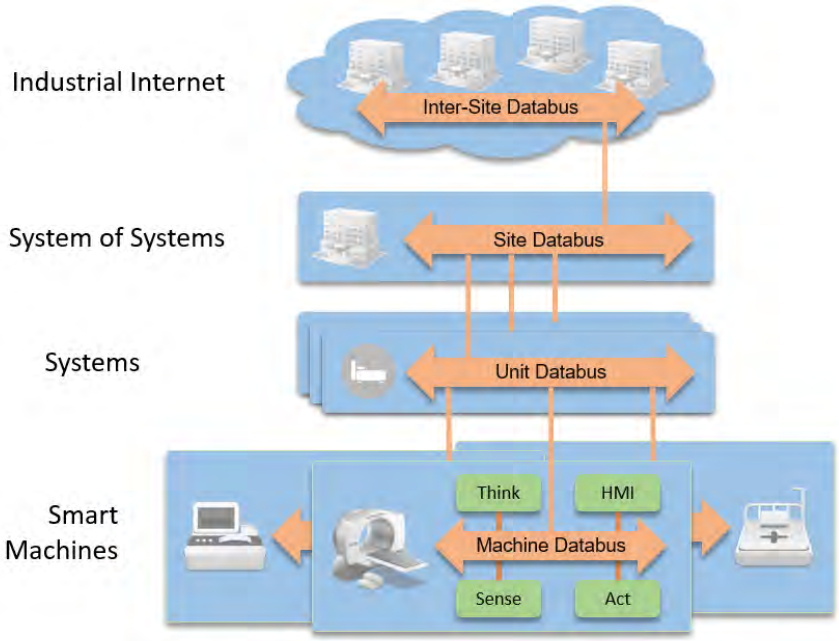

Figure 7-4: Layered Databus Architecture

In Figure 7-4, at the lowest level, smart machines use databuses for local control, automation and real-time analytics. Higher-level systems use another databus for supervisory control and monitoring. Federating these systems into a "system of systems" enables complex, Internet-scale, potentially-cloud-based, control, monitoring and analytic applications.

A databus is a logical connected space that implements a set of common schema and communicates using those set of schema between endpoints. Each layer of the databus therefore

implements a common data model, allowing interoperable communications between endpoints at that layer.

The databus supports communication between applications and devices. For instance, a databus can be deployed within a smart machine to connect its internal sensors, actuators, controls and analytics. At a higher smart system level, another databus can be used for communications between machines. At a system of systems level, a different databus can connect together a series of systems for coordinated control, monitoring and analysis. Each databus may have a different set of schema or data model. Data models change between layers, as lower-level databuses export only a controlled set of internal data.

Adapters may be used between layers to match data models. The adapters may also separate and bridge security domains, or act as interface points for integrating legacy systems or different protocols.

Generally, transitions, occurring between layers, filter and reduce the data. This is important because the scope of control and analysis increases at each layer and the amount of data is generally reduced to match the broader scope, higher latencies and higher level of abstraction. An example use of this architecture for oil well monitoring and operational control, typical for large SCADA (1) systems is represented in Figure 7-5.

In addition to its use in the control, information, application and enterprise domains, this layered databus architecture is useful in the operations domain for monitoring, provisioning and managing devices, applications and subsystems within the system.

Central to the databus is a data-centric publish-subscribe communications model. Applications on the databus simply “subscribe” to data they need and “publish” information they produce. Messages logically pass directly between the communicating nodes. The fundamental communications model implies both discovery—what data should be sent—and delivery—when and where to send it. This design mirrors time-critical information delivery systems in everyday life including television, radio, magazines and newspapers. Publish-subscribe systems are effective at distributing large quantities of time-critical information quickly, especially in the presence of unreliable delivery mechanisms.

The layered databus architecture offers the following benefits:

- fast device-to-device integration, with delivery times in milliseconds or microseconds,
- automatic data and application discovery within and between busses,
- scalable integration, comprising hundreds of thousands of sensors and actuators,
- natural redundancy, allowing extreme availability and
- hierarchical subsystem isolation, enabling development of complex system designs.

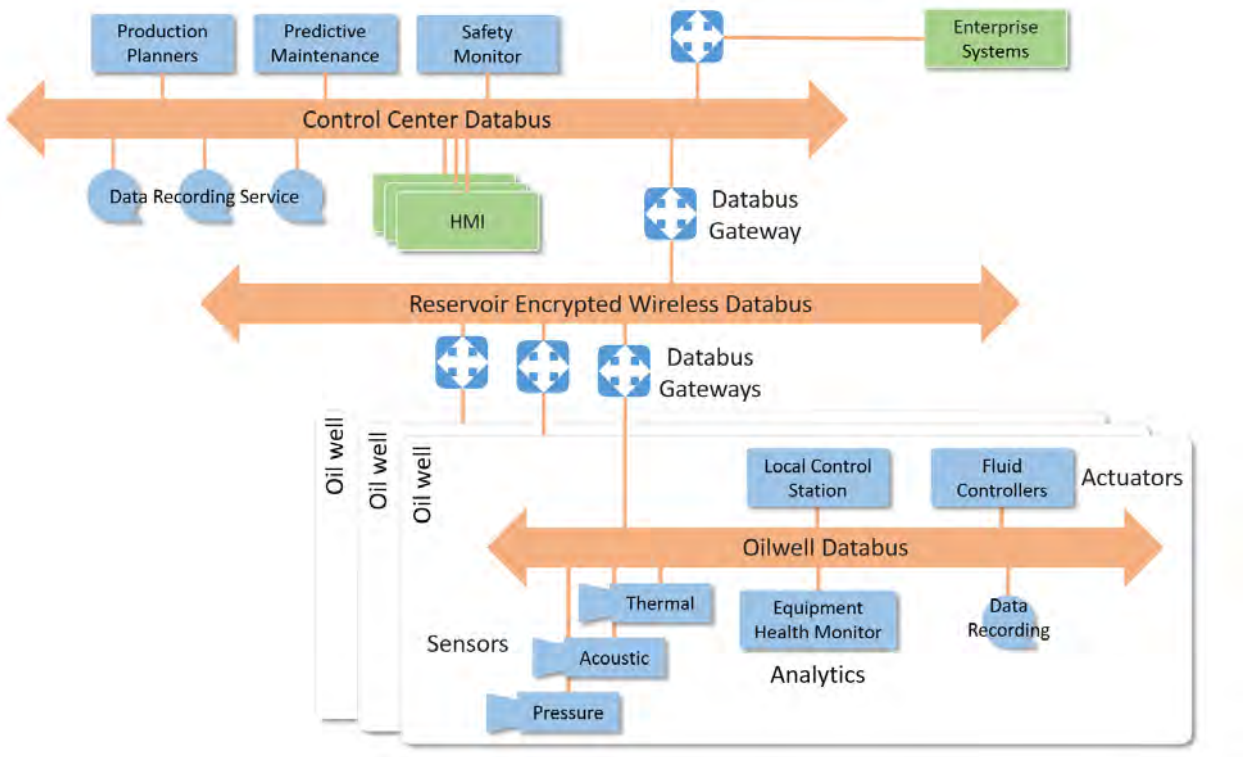

Figure 7-5: A three-layer databus architecture.[1]

7.1.4 System of Systems Architecture Pattern

At a high level, the IIoT system may itself be configured as a component within a larger system, thus creating a system-of-systems. This greatly increases the complexity of the resulting system and the results can have a cascading and comprehensive impact on the system under consideration as well as the neighboring systems, especially in system characteristics, or trustworthiness.

[1] Note that the control center HMIs can access any sensor value from the Oil Well databuses.

Annex A DESIGN SPACE CONSIDERATIONS

Architecture and high-level design are about making consistent choices in a design space, the full range design possibilities spanned by the domains of its multitudes of design parameters, and the multiple combinations of choices in that space remains for the most part unexplored. Indeed, a consequence of the size of the design space for Industrial Internet of Thing (IIoT) systems is that it will remain for the most part unexplored. When applying the Industrial Internet Reference Architecture to real world IIoT systems, it would be beneficial to have a broad view of possible design parameters and their constraints in identifying, describing and resolving system concerns. For this purpose, we include these design space considerations as an Appendix and we may further develop and refine these considerations in the upcoming revisions to making it more useful for the system architects. The specific design exemplars illustrated here are not intended to be proscriptive, and we would like to encourage exploration of the design space for new and consequential combinations that will lead to surprising capabilities and applications. The table is intended to be illustrative.

Topics	Variations			
Location awareness	Applications cannot tell where they are running, unless they can infer from timestamps on data (e.g., that after reading a sensor, the data is several seconds old on arrival)	Applications can tell what clique they are in, and type of device, e.g., control tier, embedded in turbine; enterprise tier, virtual on set of servers owned by client	Applications can be tied to specific types of devices, have access to maps and can infer physical location properties of devices	Applications know exactly in the 4-dimension space (time-space) where they are executing, what hardware they are near and what paths are available by specific route to any local device
Communication paradigm	Stigmergy only (behavior must be observed through environmental changes)	Ad hoc port-based protocols (e.g. point-to-point API based communication)	Pub-sub (having some known structured information)	Message passing using speech acts (with formal semantic logical forms)
Computational assignment	All devices are homogeneous	Devices have specific capabilities, but no particular constraint on where code can run	Code can only be run in appropriate clique of devices	Code can only be run on specific unique device
Execution paradigm	Ad hoc, every device/code combination unique	Data flow—data moves based on interest processing resident	Processing centric—data has specific execution/view history attached and very controlled point to point flow; processing resident	Data resident—processing moves to the data which controls access; limited information may be moved with the process (local state) [e.g. mobile agents]
Generic resource management	Systems engineer performs offline	Automated at device level only (chronologic, preemptive, …, real time)	Automated at clique level	Automated at IIoT system level
Certifications (safety, security, …)	Systems engineer performs offline	Specific sets of flows are certified offline for situations detected online	Automated ‘proof’ for a flow performed online, but ahead of engagement	Automated ‘proof’ performed during execution (e.g., deontic logic)
Addressability	Endpoint address, device name and path needed	Service name	Content addressable (e.g. all services that can perform an IR (information retrieval) observation at point (x,y,z) at time T; all services that know if P)	

Topics	Variations			
Constraint expressivity	No constraints supported	Can specify data handling (e.g. send via different path than last time using 16 new TOR servers)	Can specify processing (e.g. spend no more than 2G nominal CPU cycles on inferences on usage based on the address properties of data or the utilization of device processing for the following conversation marked 'A')	Can specify arbitrary higher order properties (e.g. "there exists a service S, that can predict the likelihood of commodity price shortages" [without specifying where S is, or if it is reachable]
Negotiability	None—take it or leave it	Over specific predetermined system-wide resources, e.g., $ to invoke a service at a fixed quality level	Enter into auction (e.g. Dutch) for specific service availability (e.g., service S has timeslot T available: bid?)	Arbitrary resource/quality negotiation and contract remediation (negotiate 2x quality for 4x time and 8x price, but only 1.5x quality produced so penalty is…)
Resilience	None—single point of failure throughout	Some reliability measures taken (failover redundancy, voting) for predetermined set of critical resources, but general system still has single points of failure (e.g., corrupted security operator)	Resilience at clique level (pool of similar resources allow loss of some without noticeable degradation, pool can be replenished from a system-wide reserve, multiple cross-checks implemented, e.g. vs. insider threat)	Individual device resilience and reconstruction (robot repairs the device and improves it so the same failure will not recur, recovers state of device at time of failure, etc.)

Table 7-1: Architectural Alternative/Design Space

Annex B REFERENCES

[1] Industrial Internet Consortium: “Industrial Internet Reference Architecture Technical Report, version 1.7”, 2015
http://www.iiconsortium.org/IIRA

[2] ISO/IEC/IEEE: “ISO/IEC/IEEE 42010:2011 Systems and software engineering -- Architecture description”, 2011
http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

[3] Industrial Internet Consortium: “Industrial Internet Security Framework Technical Document, version 1.0”, 2016
http://www.iiconsortium.org/IISF

[4] Industrial Internet Consortium: “The Industrial Internet, Volume G8: Vocabulary Technical Report, version 2.0”, 2016
http://www.iiconsortium.org/vocab

[5] National Communcations System: “Supervisory Control and Data Acquision (SCADA) Systems”, 2004
https://www.dhs.gov/sites/default/files/publications/csd-nist-guidetosupervisoryanddataccquisition-scadaandindustrialcontrolsystemssecurity-2007.pdf

[6] F. Petruzella: “Programmable Logic Controllers”, 5th edition, McGraw-Hill Education, 2016.

[7] B. Lydon, “PLC vs. DCS - Competing Process Control Philosophy” automation.com, 2011
http://www.automation.com/automation-news/article/plc-vs-dcs-competing-process-control-philosophy

[8] Stevan Harnad: “The Symbol Grounding Problem”, Princeton University, Department of Psychology 1990
http://users.ecs.soton.ac.uk/harnad/Papers/Harnad/harnad90.sgproblem.html
PDF at *http://courses.media.mit.edu/2004spring/mas966/-Harnad%20symbol%20grounding.pdf*

[9] John R. Searle: “Minds, Brains and Programs”, Behavioral and Brain Sciences 3 (3): 417-457, 1980
PDF at *http://cogprints.org/7150/1/10.1.1.83.5248.pdf*

[10] Judith J. Thomson: “The Trolley Problem”, The Yale Law Journal, Vol. 94, No. 6 (May, 1985), pp. 1395-1415, The Yale Law Journal Company, Inc.
PDF at *http://waleszczynski.pl/wp-content/uploads/sites/-4/2016/02/trolley_j_thomson.pdf*

[11] Object Management Group: “Business Motivation Model (BMM)”, 2015
http://www.omg.org/spec/BMM/

[12] Object Management Group: “Structured Assurance Case Metamodel (SACM)”, 2013
www.omg.org/spec/SACM

ANNEX C REVISION HISTORY

Revision	Date	Editor	Changes Made
V1.7	2015-06-04	Shi-Wan Lin (Lead, Intel), Stephen Mellor (Lead, IIC), Bradford Miller (Lead, V1.5, GE), Jacques Durand (Fujitsu), Mark Crawford (SAP) and Robert Lembree (V1.5, Intel).	Initial Release
V1.8	2016-12-18	Shi-Wan Lin (Thingswise), Mark Crawford (SAP) and Stephen Mellor (IIC).	1. This volume, containing the about roughly the same content as in Part I (from Chapter 1 to 7) of the initial release version, is released under the new IIC document structure. The content in Part II (from Chapter 8 to 16) will be published in separate volumes under the new IIC document structure. 2. A few new boilerplate sections (Chapter 1 and some appendices). 3. Improved and enhanced description of architecture concepts and constructs derived from ISO/IEC/IEEE 42010 Architecture Description standard and their application in IIRA (Chapter 3 Industrial Internet Architecture Framework). 4. Added a section on IIRA viewpoints' Scope of Applicability and Relationship to System Lifecycle Process (Chapter 3 Industrial Internet Architecture Framework, Section 3.6). 5. Added a section on Crosscutting Functions and System Characteristics (Chapter 6 Functional Viewpoint, Section 6.7). 6. Added a section on Functional Domain and Computational Deployment Models (Chapter 6 Functional Viewpoint, Section 6.8). 7. Added a section on Layered Databus Architecture Pattern (Chapter 7 Implementation Viewpoint, Section 7.5). 8. Added an appendix on Design Space Considerations (Appendix A).

V1.9	2019-05-xx	Shi-Wan Lin (Thingswise), Eric Simmon (NIST) and Stephen Mellor (IIC).	1. Added formal definitions from the Industrial Internet vocabulary for the Viewpoints and Functional Domains (across Chapter 4 Business Viewpoint, Chapter 5 Usage Viewpoint, Chapter 6 Functional Viewpoint and Chapter 7 Implementation Viewpoint). 2. Stated the kind of external influences, e.g. regulatory factors, that may need to be considered in conceptualizing an IIoT system (Chapter 4 Business Viewpoint). 3. Brought up the importance of wireless communication in the Control Domain (Chapter 6 Functional Domain, 6.3 The Control Domain). 4. Added the Human Roles in an IIoT System (Chapter 6 Functional Viewpoint). 5. Clarified that the scope of the Implementation Viewpoint does not cover concerns about deployment and operation which are life-cycle issues to be considered separately (Chapter 7 Implementation Viewpoint). 6. Clarified that the patterns are intended as examples and references (Chapter 7 Implementation Viewpoint). 7. A number of editorial changes on sentences for better understanding.

USE OF INFORMATION – TERMS, CONDITIONS AND NOTICES

Copyright © 2019 Industrial Internet Consortium, a program of Object Management Group, Inc. ("OMG").

ACKNOWLEDGEMENTS

This document is a work product of the Industrial Internet Consortium Architecture Task Group, co-chaired by Shi-Wan Lin (Thingswise) and Eric Simmon (NIST).

EDITORS

Shi-Wan Lin (Thingswise) and Eric Simmon (NIST).

AUTHORS

The following persons have written substantial portion of material content in this document:

Shi-Wan Lin (Thingswise/Intel), Bradford Miller (GE), Jacques Durand (Fujitsu), Graham Bleakley (IBM), Amine Chigani (GE), Robert Martin (MITRE), Brett Murphy (RTI) and Mark Crawford (SAP).

CONTRIBUTORS

The following persons have contributed valuable ideas and feedback that significantly improve the content and quality of this document:

Marcellus Buchheit (Wibu-Systems), Anish Karmarkar (Oracle), Sven Schrecker (Intel), JP LeBlanc (Lynx Software Technologies), Chuck Byers (IIC), Sam Bhattarai (Toshiba), Daniel Young (Toshiba).

TECHNICAL EDITOR

Stephen Mellor (IIC staff) oversaw the process of organizing the contributions of the above Authors and Contributors into an integrated document.

IIC ISSUE REPORTING

All IIC documents are subject to continuous review and improvement. As part of this process, we encourage readers to report any ambiguities, inconsistencies or inaccuracies they may find in this Document or other IIC materials by sending an email to *admin@iiconsortium.org*.

USE OF INFORMATION—TERMS, CONDITIONS AND NOTICES

This is an Industrial Internet Consortium document (the "Document") and is to be used in accordance with the terms, conditions and notices set forth below. This Document does not represent a commitment by any person to implement any portion or recommendation contained in it in any products or services. The information contained in this Document is subject to change without notice.

LICENSES

The companies who contributed content to this Document ("Contributing Companies") have granted to the Industrial Internet Consortium (the "IIC"), a program of Object Management Group, Inc. ("OMG") a nonexclusive, irrevocable, sublicensable, royalty-free, paid up, worldwide license to copy and distribute this Document and to modify this Document and distribute copies of the modified version. Each of the Contributing Companies has agreed that no person shall be deemed to have infringed the copyright in the included material provided by any such copyright holder by reason of having copied, distributed or used such material as set forth herein.

Subject to the terms and conditions below, OMG (including its IIC program) and the Contributing Companies hereby grant you a fully-paid up, non-exclusive, nontransferable, royalty-free, worldwide license (without the right to sublicense) to use, copy and distribute this Document (the "Permission"), provided that: (1) both the copyright notice above, and a copy of this Permission paragraph, appear on any copies of this Document made by you or by those acting on your behalf; (2) the use of the Document is only for informational purposes in connection with the IIC's mission, purposes and activities; (3) the Document is not copied or posted on any network computer, publicly performed or displayed, or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (4) no modifications are made to this Document.

This limited Permission is effective until terminated. You may terminate it at any time by ceasing all use of the Document and destroying all copies. The IIC may terminate it at any time by notice to you. This Permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, or at any time upon the IIC's express written request, you will destroy immediately any copies of this Document in your possession or control.

The Licenses and Permission relate only to copyrights and do not convey rights in any patents (see below).

PATENTS

Compliance with or adoption of any advice, guidance or recommendations contained in any IIC reports or other IIC documents may require use of an invention covered by patent rights. *OMG and its IIC program are not responsible for identifying patents for which a license may be required* to comply with any IIC document or advice, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. IIC documents are informational and advisory only. Readers of this Document are responsible for protecting themselves against

liability for infringement of patents and other intellectual property that may arise from following any IIC recommendations or advice. OMG and its IIC program disclaim all responsibility for such infringement.

GENERAL USE RESTRICTIONS

This Document contains content that is protected by copyright. Any unauthorized use of this Document may violate copyright laws, trademark laws and communications regulations and statutes. Except as provided by the above Permission, no part of this work covered by copyright may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping or information storage and retrieval systems—without permission of the copyright owner(s).

DISCLAIMER OF WARRANTY

WHILE THIS DOCUMENT IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. OMG (INCLUDING ITS IIC PROGRAM) AND THE CONTRIBUTING COMPANIES MAKE NO WARRANTIES, REPRESENTATIONS OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, OF MERCHANTABILITY, OF FITNESS FOR A PARTICULAR PURPOSE OR USE, OR AGAINST INFRINGEMENT. OMG (INCLUDING ITS IIC PROGRAM) MAKES NO REPRESENTATIONS, WARRANTIES, GUARANTIES, OR CONDITIONS AS TO THE QUALITY, SUITABILITY, TRUTH, ACCURACY, OR COMPLETENESS OF THIS DOCUMENT.

IN NO EVENT SHALL OMG (INCLUDING ITS IIC PROGRAM) OR ANY OF THE CONTRIBUTING COMPANIES BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGE, OR DAMAGE FOR LOSS OF PROFITS, REVENUE, DATA OR USE, HOWEVER IT ARISES, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, REPRODUCTION, DISTRIBUTION OR USE OF THIS MATERIAL, EVEN IF PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of any software or technology developed using this Document is borne by you. This disclaimer of warranty constitutes an essential part of the Permission granted to you to use this Document.

LIMITED RIGHTS NOTICE

This Document contains technical data that was developed at private expense and (i) embodies trade secrets, or (ii) is confidential and either commercial or financial. This document was not produced in the performance of a government contract and is not in the public domain. The use, duplication or disclosure of this Document by the U.S. Government is subject to the restrictions set forth in 48 C.F.R. 52.227-14–Rights in Data "Limited Rights Notice (Dec. 2007) (a) and (b)," or as specified in 48 C.F.R. 12.211 of the Federal Acquisition Regulations and its successors, as applicable. This data may only be reproduced and used by the U.S. Government with the express limitation that it will not, without written permission of the copyright owners, be used for

purposes of manufacture nor disclosed outside the Government. The copyright owners are as indicated above and may be contacted through Object Management Group, Inc., 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

The trademarks, service marks, trade names and other special designations that appear on and within the Document are the marks of OMG, the Contributing Companies and possibly other manufacturers and suppliers identified in the Document and may not be used or reproduced without the express written permission of the owner, except as necessary to reproduce, distribute and refer to this Document as authorized herein.