industrial internet CONSORTIUM

Industrial Internet Reference Architecture

TABLE OF CONTENTS

1 Rationale and Context ..... - 8 -
1.1 The Industrial Internet ..... - 8 -
1.2 Reference Architecture Concepts ..... - 9 -
1.3 Industrial Internet Reference Architecture ..... - 10 -
1.4 Major Components of the Technical Report ..... - 10 -
1.5 Next Steps ..... - 10 -

2 Key System Characteristics and their Assurance ..... - 11 -
2.1 Key System Characteristics ..... - 11 -
2.2 System Characteristic Assurance ..... - 12 -

3 Industrial Internet Reference Architecture ..... - 14 -
3.1 Industrial Internet Architecture Framework ..... - 14 -
3.2 Industrial Internet Viewpoints ..... - 15 -
3.3 Security across the Viewpoints ..... - 17 -
3.3.1 An Integrated approach to Security ..... - 17 -
3.3.2 Threat Modeling and Secure Design ..... - 17 -

4 The Business Viewpoint ..... - 19 -
4.1 Elements of the Business Viewpoint ..... - 19 -
4.2 Security Concerns in the Business Context ..... - 20 -

5 The Usage Viewpoint ..... - 22 -
5.1 Elements of the Usage Viewpoint ..... - 22 -
5.2 Common Security Activities ..... - 24 -

6 The Functional Viewpoint ..... - 25 -
6.1 Background ..... - 25 -
6.2 The Control Domain ..... - 28 -
6.3 The Operations Domain ..... - 30 -
6.4 The Information Domain ..... - 32 -
6.5 The Application Domain ..... - 34 -
***6.6* The Business Domain ..... - 34 -**
6.7 Common Security Functions ..... - 35 -

7 Implementation Viewpoint ..... - 36 -
7.1 Architecture Patterns ..... - 36 -
7.1.1 Three-tier architecture pattern ..... - 37 -
7.1.2 Gateway-Mediated Edge Connectivity and Management architecture pattern ..... - 39 -
7.2 Secure Implementations ..... - 41 -

8 Safety ..... - 45 -
8.1 Relationships with Other Concerns ..... - 47 -

9 Security, Trust and Privacy ..... - 49 -
9.1 Endpoint Security ..... - 50 -
9.1.1 Secure Boot Attestation ..... - 51 -
9.1.2 Deployment of Security Agent ..... - 51 -
9.1.3 Endpoint Identity ..... - 52 -

9.1.4 Endpoint Attack Response ..... - 52 -
9.1.5 Remote Policy Management ..... - 52 -
9.1.6 Logging and Event Monitoring ..... - 53 -
9.1.7 Application Whitelisting ..... - 53 -
9.1.8 Network Whitelisting ..... - 53 -
9.1.9 Dynamically Deployed Countermeasures ..... - 53 -
9.1.10 Remote and Automated Endpoint Update ..... - 53 -
9.1.11 Policy Orchestration Across Multiple Endpoints ..... - 54 -
9.1.12 Peripheral Devices Management ..... - 54 -
9.1.13 Endpoint Storage Management ..... - 54 -
9.1.14 Access Control ..... - 54 -
9.2 Communication Security ..... - 54 -
9.2.1 Architectural Considerations for Information Exchange Security ..... - 55 -
9.2.2 Security in request-response and publish-subscribe communications ..... - 55 -
9.2.3 Mutual Authentication Between Endpoints ..... - 56 -
9.2.4 Communication Authorization ..... - 56 -
9.2.5 Identity Proxy/Consolidation Point ..... - 56 -
9.2.6 User Authentication and Authorization ..... - 56 -
9.2.7 Encryption in Communication ..... - 56 -
9.3 Management and Monitoring Security ..... - 56 -
9.3.1 Identity Management ..... - 57 -
9.3.2 Provisioning and Commissioning ..... - 57 -
9.3.3 Security Policy Management ..... - 57 -
9.3.4 Endpoint Activation Management ..... - 58 -
9.3.5 Credential Management ..... - 58 -
9.3.6 Management Console ..... - 58 -
9.3.7 Situational Awareness ..... - 58 -
9.3.8 Remote Update ..... - 59 -
9.3.9 Management and Monitoring Resiliency ..... - 59 -
9.4 Data Distribution and Secure Storage ..... - 59 -
9.4.1 Data Security ..... - 59 -
9.4.2 Data Centric Policies ..... - 59 -
9.4.3 Data Analysis and Privacy ..... - 60 -
9.4.4 IT Systems and the Cloud ..... - 60 -

10 Resilience ..... - 61 -

11 Integrability, Interoperability and composability ..... - 66 -

12 Connectivity ..... - 70 -
12.1 Architectural Role ..... - 70 -
12.2 Key System Characteristics ..... - 71 -
12.3 Key Functional Characteristics of the Connectivity Framework Layer ..... - 73 -
12.4 Key Functional Characteristics of the Communication Transport Layer ..... - 75 -
12.5 Connectivity Gateways ..... - 76 -

13 Data Management ..... - 78 -
13.1 Reduction and Analytics ..... - 78 -
13.2 Publish and Subscribe ..... - 78 -

13.3 Query ..... - 80 -
13.4 Storage, Persistence and Retrieval ..... - 80 -
13.5 Integration ..... - 81 -
13.6 Description and Presence ..... - 81 -
13.7 Data Framework ..... - 82 -
13.8 Rights Management ..... - 82 -

14 Analytics and Advanced Data Processing ..... - 83 -
14.1 Advanced Data Processing ..... - 83 -
14.2 Advanced Data Processing Pattern and Properties ..... - 84 -
14.3 Advanced Analytics ..... - 86 -
14.4 IIS RA Alignment ..... - 87 -

15 Intelligent and Resilient Control ..... - 88 -
15.1 Motivation ..... - 88 -
15.2 Considerations ..... - 88 -
15.3 Functional Components ..... - 90 -

16 Dynamic Composition and Automated Interoperability ..... - 94 -
16.1 Motivation ..... - 94 -
16.2 Considerations ..... - 95 -
16.3 Functional Components ..... - 96 -

17 References ..... - 98 -

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

IIC Issue Reporting

All IIC documents are subject to continuous review and improvement. As part of this process, we encourage members to report any ambiguities, inconsistencies, or inaccuracies they may find in this Document or other IIC materials by providing comments in the Technology Working Group Kavi workspace.

Acknowledgements

This document is a work product of the Industrial Internet Consortium Technology Working Group, co-chaired by Shi-Wan Lin (Intel) and Bradford Miller (GE), in collaboration with the Security Working Group, co-chaired by Sven Schrecker (Intel) and Jesus Molina (Fujitsu).

EDITORS

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).

AUTHORS

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

Shi-Wan Lin (Intel), Bradford Miller (GE), Jacques Durand (Fujitsu), Rajive Joshi (RTI), Paul Didier (Cisco), Amine Chigani (GE), Reinier Torenbeek (RTI), David Duggal (EnterpriseWeb), Robert Martin (MITRE), Graham Bleakley (IBM), Andrew King (University Of Pennsylvania), Jesus Molina (Fujitsu), Sven Schrecker (Intel), Robert Lembree (Intel), Hamed Soroush (RTI), Jason Garbis (RSA), Mark Crawford (SAP), Eric Harper (ABB), Kaveri Raman (AT&T), and Brian Witten (Symantec)

CONTRIBUTORS

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

Farooq Bari (AT&T), Tom Rutt (Fujitsu), Jack Weast (Intel), Lin Nease (HP), Ron Ambrosio (IBM), Omer Schneider (Cyber-X Labs), Pete MacKay (Wurldtech), Lance Dover (Micron)

We are also grateful to everyone who made comments on the draft version available to the Architecture Task Group, and will thank in advance anyone who provides further constructive comments on the current version.

We acknowledge the work by the members of the Vocabulary Team in the Technology Working Group led by Tom Rutt (Fujitsu), who have authored the Industrial Internet Vocabulary that is a companion document to this one.

Finally we are grateful for the ongoing support of Aaron Soellinger (IIC), whose diligence with supporting the tools and the mechanics of the document writing have made this process possible.

This document is the first version of the '*Industrial Internet Reference Architecture Technical Report'*. It initiates a process to create broad industry consensus to drive product interoperability and simplify development of Industrial Internet systems that are better built and integrated with shorter time to market, and at the end better fulfill their intended uses.

The Industrial Internet is being shaped by many participants from the energy, healthcare, manufacturing, transportation and public sectors, each with complex and fast-changing architectures. To avoid fragmentation and a loss of interoperability, and the concomitant increases in cost and length of development, it is important and urgent to build early consensus among the participants on major architecture questions.

Accordingly, we have identified what we believe are the major architecture issues and we have examined these issues based on a formal architecture framework. We believe we have arrived at a reasonable statement of what the most important architecture components are, how they fit together and how they influence each other. This first version of the document contains our initial findings and its publication is an opportunity to gather early feedback from industry participants, especially across industrial sectors, so we can improve it quickly in its subsequent versions.

Consequently, this document is broad, rather than deep. It is intended to be informative and to broaden understanding of the issues. It is not a normative technical specification that would require conformance or compliance. As we proceed, this document will be supported by a collection of other documents that discuss topical subjects in greater depth.

The audience for this document includes component and system providers who can use this document to guide the development of interoperable technologies and solutions, and system implementers who can use it as a common starting point of system conception and design. Interested parties may also use the framework to build and select interchangeable technology and solution components.

1 RATIONALE AND CONTEXT

This document describes a Reference Architecture for Industrial Internet Systems. It defines Industrial Internet Systems, and specifies an Industrial Internet Architecture Framework to aid in the development, documentation and communication of the Industrial Internet Reference Architecture. We begin by describing our scope.

1.1 THE INDUSTRIAL INTERNET

The Industrial Internet is an internet of things, machines, computers and people, enabling intelligent industrial operations using advanced data analytics for transformational business outcomes. It embodies the convergence of the global industrial ecosystem, advanced computing and manufacturing, pervasive sensing and ubiquitous network connectivity.

There are many interconnected systems deployed today that combine hardware, software and networking capabilities to sense and control the physical world. These industrial control systems

contain embedded sensors, processors and actuators that provide the capability to serve specific operational or business purposes, but, by and large, these systems have not been connected to broader systems or the people who work with them.

The Industrial Internet concept is one that has evolved over the last decade to encompass a globally interconnected network of trillions of ubiquitous addressable devices and collectively representing the physical world.

The Industrial Internet effort will bring industrial control systems online to form large end-to-end systems, connecting them with people, and fully integrating them with enterprise systems, business processes and analytics solutions. These end-to-end systems are referred to as Industrial Internet Systems (IISs). Within these IISs, operational sensor data and the interactions of people with the systems may be combined with organizational or public information for advanced analytics and other advanced data processing (e.g., rule-based policies and decision systems). The result of such analytics and processing will in turn enable significant advances in optimizing decision-making, operation and collaboration among a large number of increasingly autonomous control systems.

Industrial Internet systems cover energy, healthcare, manufacturing, public sector, transportation and related industrial systems. As such, many IISs operate in mission critical environments and so demand high standards of security, safety and resiliency, different from those in the consumer and commercial sectors. We do not draw a clear boundary between these types of systems. Rather, we focus on the reference architectures required for industrial systems with the understanding that many concepts defined here may be applied in other types of systems.

To be effective, an IIS requires significant increases in levels of performance, scalability and efficiency. For rapid and widespread deployment, the IISs must be easily understandable and supported by widely applicable, standard-based, open and horizontal architecture frameworks and reference architectures that can be implemented with interoperable and interchangeable building blocks. These are the key motivations of this document.

1.2 Reference Architecture Concepts

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

> 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.

1.3 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE

The Industrial Internet Reference Architecture (IIRA) is a standard-based open architecture for IISs. To maximize its value, the IIRA has broad industry applicability to drive interoperability, to map applicable technologies, and to guide technology and standard development. The description and representation of the architecture 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 well understood at this time, prominently those that have been defined in the Industrial Internet Consortium (IIC). The IIRA design is intended to transcend today's available technologies and in so doing is capable of identifying technology gaps based on the architectural requirements. This will in turn drive new technology development efforts by the Industrial Internet community.

1.4 MAJOR COMPONENTS OF THE TECHNICAL REPORT

This technical report is divided into two parts. The first part (Chapters 1~7) provides rationale and context for the types of system under consideration (Chapter 1), and their key characteristics (Chapter 2), the architectural framework we use to describe the reference architecture (Chapter 3), and each of the component parts ('viewpoints') of the reference architecture (Chapter 4~Chapter7). The second part (Chapters 8~16) provides an analysis of key system concerns that require consistent analysis across the viewpoints to ensure concerted system behaviors. We conclude with references (Chapter 17).

The force behind the Industrial Internet is the integration of Information Technologies (IT) and Operational Technologies (OT). That integration is now realized in several domains as described in Chapters 4~7. Further exploration of IT/OT issues is deferred to later versions.

The IIC Vocabulary defines terms used in this document and other IIC documents [1].

1.5 NEXT STEPS

The publication of this first version of this Technical Report enables feedback. As more use cases are developed, more real systems built, and more IIC testbeds deliver results, we will continuously explore the general architecture of Industrial Internet Systems, guided by the feedback from these activities, both to refine this document and produce new works built on it. Meanwhile, we shall define which standards and technologies fit into the building blocks of the reference architecture, identify gaps and define requirements for improvement.

2 KEY SYSTEM CHARACTERISTICS AND THEIR ASSURANCE

2.1 KEY SYSTEM CHARACTERISTICS

An Industrial Internet System will exhibit end-to-end characteristics, such as safety, security or resilience. Such characteristics are emergent properties or behaviors of the IIS resulting from the properties of its various components and the nature of their interactions. Because IISs are large scale, heterogeneous, built with multi-vendor components, often broadly distributed and continuously evolving, it is a challenge to define, measure, enforce and maintain the system characteristics over time. For these reasons, we give prominent treatment to a few key system characteristics in this document.

A system characteristic is delivered through four elements:

- the system's functional components and their interactions,
- the engineering process by which the components are created and the system assembled (system engineering),
- the way the system is used and maintained (system operations) and
- the evidence gathered throughout the full lifecycle of the components and systems. [4]

Common system characteristics include upholding privacy expectations, reliability, scalability, usability, maintainability, portability and composability but three are key to the discussion of IISs because of their criticality in ensuring the core functions, rather than the efficiency, of these functions, of the system:

Safety: the condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment.[5]

Security: the condition of the system operating without allowing unintended or unauthorized access, change or destruction of the system or the data and information it encompasses.

Resilience: the condition of the system being able to avoid, absorb and/or manage dynamic adversarial conditions while completing assigned mission(s), and to reconstitute operational capabilities after casualties.

[4] Broadly speaking, a lifecycle of a system includes activities for requirement specification, design, development, manufacturing, test, assembly, integration, delivery, deployment, verification, evolution and upgrade, un-deployment, and disposal of the system. We will not hereafter refer to the specific activities unless it is necessary.

[5] IEC 61508, Functional Safety [25]

Desired system characteristics and the degree they are needed vary by system. They may be motivated by business context and values, be mandated by regulations and contractual agreements, or simply be commonly expected behaviors for a specific type of system. Note that some of these desired characteristics may be negated by unintended side effect of other unrelated system behaviors.[6]

To preserve these system characteristics in the evolving IISs, continuous tracking and auditing should be provided throughout the lifecycle of these systems.

The number of reports of security attacks on large enterprise and infrastructure systems is rapidly increasing, along with the level of damage they have wrought. These attacks are better organized, and employ attack techniques of ever-increasing levels of sophistication. The actors in these attacks are becoming more diverse and include anyone from insiders to casual hackers, terrorists and state-sponsored actors. Security for Industrial Internet systems from design, development, and deployment to operations must be heightened to mitigate the increased security risks. We therefore give security, a key system characteristic, a strong emphasis in this document.

2.2 System Characteristic Assurance

System characteristics are often subject to regulations, compliance requirements and contractual agreements and thus need be measured and assessed.[7] The foundation for claiming such characteristics covers three major aspects:

System engineering: Evidence supporting system characteristics must be gathered and provided, with proper tracking and certification, on the components and the system formed from their combination throughout the entire lifecycle of activities. It may include specific attributes of the software and hardware, documentation on the way it was constructed, where it was developed and by whom, and records of the types of component testing and analysis carried out, and the results thereof.

System operations: A system characteristic may depend on how a system is being used independent from the quality of the system design and components. Evidence must be available to demonstrate that the system operational processes support the characteristic. This may include evidence that the system is being used for its intended purpose and the qualification of the personnel who play a role in its operations.

[6] For example, a system that tracks an individual's location on a ship to provide a man-overboard alert may also violate privacy expectations of that individual.

[7] Examples of security related compliances include Common Criteria [23] and various Federal Information Processing Standards (FIPS) specifications. Examples of safety related compliance include the application of DO-178B/C to avionics software systems certification by the FAA. Compliance will not be taken up by the RA but may be addressed by future more detailed documents, as they are greatly dependent on the specific vertical and deployment scenario.

System auditing: The system and its processes shall support the collection and the evaluation of metrics and logs necessary for monitoring and auditing the important characteristics. Monitoring and audit processes should be established and audit records must be maintained.

Because IISs are built from multi-vendor components and solutions, possibly composed dynamically after deployment, engineers must be able to access recorded claims and their supportive evidences of specific system characteristics in components to evaluate, select, acquire and assemble qualified components into the desired IIS.

> Example: In the residential house example above, information about fire resistance, load bearing capabilities and structural integrity of the materials must be available to support the claimed "key system characteristic" of safety for individually constructed homes.
>
> Similarly, the software in a Bluetooth-enabled device that controls the Bluetooth interface and communicates with external entities should have information available to support the assessment of the key characteristics in its fitness for a given use case. This information may include coding practices and reviews, tests and assessments done with respect to its strength to accidental inputs, malicious attacks, and the known issues and their risks in coding, design and architecture if any. It may also include regular and contingent update process and methods to address newly discovered vulnerabilities.
>
> Supported by the information along with testable results as evidence, we can ascertain that it is, for example, safe and secure enough to use in an interface to an insulin pump.

An assurance case [2] [3] [4] is an argument supporting a claim that an IIS satisfies given requirements, in a traceable manner. Assurance cases provide a means to structure the reasoning about safety, security or resiliency in a given environment, so that system creators can gain confidence that systems will work as expected. The assurance case also becomes a key element in the documentation of the system and provides a map to more detailed information.

The activities required to construct an assurance case are similar to those that would normally occur when developing a system with required behaviors and characteristics. Assurance cases highlight and explicitly present the claims, evidence and reasoning that relates them together in a reviewable form. The explicit connections between what is claimed and the evidence used to support that claim makes the assurance case a useful tool for third parties who need to have confidence that the IIS exhibits the desired characteristics.

Assurance cases are also often used to support the iterative review and revision of the implementation of the claimed characteristics until the stakeholder gains sufficient confidence about the system displaying the claimed behaviors.

3 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE

3.1 INDUSTRIAL INTERNET ARCHITECTURE FRAMEWORK

Many stakeholders are involved when considering complex systems such as those expected of Industrial Internet Systems (IISs). These stakeholders have many intertwining concerns pertinent to the system of interest. Their concerns cover the full lifecycle of the system, and their complexity calls for a framework to identify and classify the concerns into appropriate categories so that they can be evaluated and addressed systematically.

To address this need, the Industrial Internet Consortium has defined an *architecture framework* that describes the conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders. Based on ISO/IEC/IEEE 42010:2011 [5], the IIC Architecture Framework facilitates easier evaluation, and systematic and effective resolution of stakeholder concerns, and serves as a valuable resource to guide the development and the documentation of, and the communication about, the Industrial Internet Reference Architecture (IIRA).

The ISO/IEC/IEEE 42010:2011 standard specification codifies the conventions and common practices of architecting and provides a core ontology for the description of architectures. The IIAF adopts the general concepts and constructs in this specification, e.g. *architecture* and *architecture framework*, *concern, stakeholder*, and *viewpoint*.

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 system. A *viewpoint* consists of conventions framing the description and analysis of specific system concerns.

The stakeholders, concerns, viewpoints and their relationship, as shown in Figure 3-1, form the basis of the Architecture Framework. The Industrial Internet Reference Architecture is fully described by the analysis on the set of specific concerns in viewpoints.

> Example: 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 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 viewpoint.

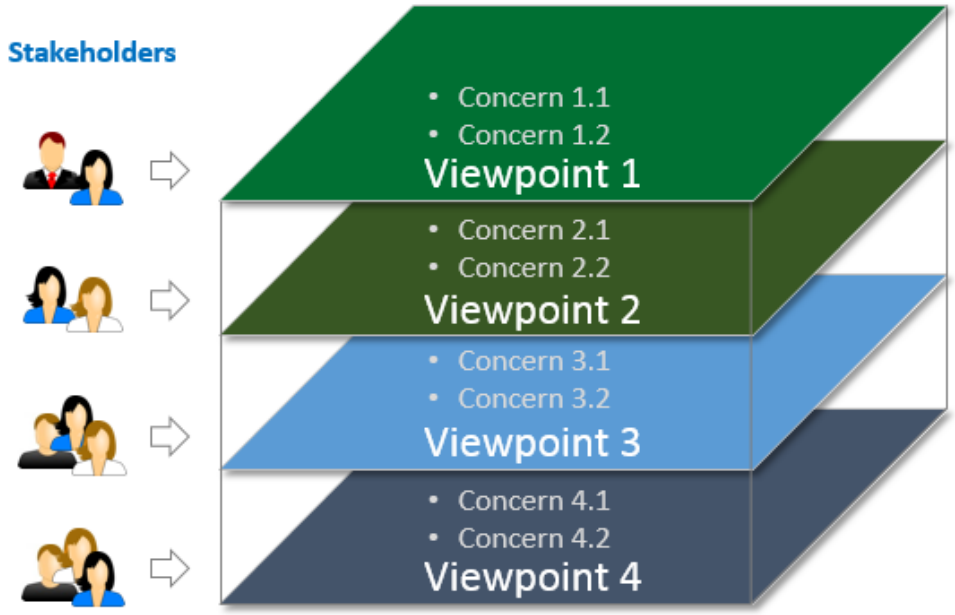


Figure 3-1 Architecture Framework

3.2 INDUSTRIAL INTERNET VIEWPOINTS

The various concerns of an IIS are classified and grouped together as four viewpoints:

- Business
- Usage
- Functional
- Implementation

As shown in Figure 3-2, these four viewpoints form the basis for a detailed viewpoint-by-viewpoint analysis of individual sets of Industrial Internet System concerns.

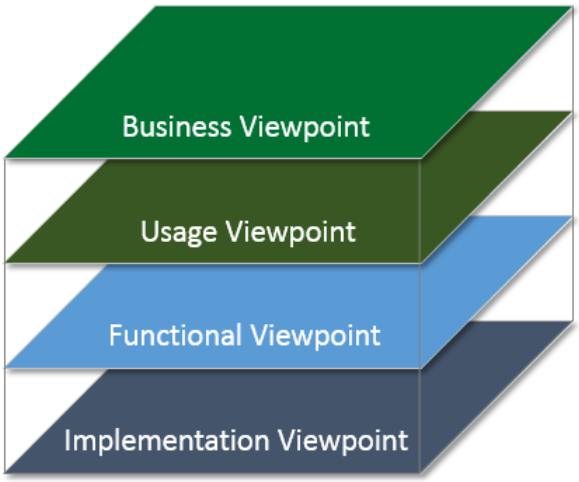


Figure 3-2 Architecture Viewpoints

The *Business Viewpoint* attends to the concerns of the identification of stakeholders and their business vision, values and objectives in establishing an IIS in its business and regulatory context. It further identifies how the IIS 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.

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

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

The *functional viewpoint* focuses on the functional components in an IIS, their interrelation and structure, 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.

The *implementation viewpoint* deals with the technologies needed to implement functional components, their communication schemes and their lifecycle procedures. These components 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.

3.3 SECURITY ACROSS THE VIEWPOINTS

Security of industrial control systems today often relies on physical security, the isolation of the systems and the obscurity of proprietary communication protocols. Industrial Internet Systems, on the other hand, are, by nature, connected and distributed. They continually exchange data; they are deeply integrated with enterprise systems; and they evolve over their lifetimes, converging with other IISs. Consequently, their attack surface is significantly larger than isolated industrial control systems.

IISs call for an integrated approach to security spanning the physical world (including direct observability), the network world (including preservation of rights to the use of data), and the business world (including property rights and rights to make contracts). They simply cannot treat security as a separate, add-on design concern.

3.3.1 AN INTEGRATED APPROACH TO SECURITY

The IIC Reference Architecture therefore integrates security policies for physical plant, hardware, software and communication as core to system design, across all the viewpoints:

- The business viewpoint establishes the return on investment for security, in the context of other considerations such as performance or consumer satisfaction. It also defines requirements for security compliance backed by security metrics that are collected.
- The usage viewpoint strives to make security transparent to the user, minimizing their involvement, and to establish a strong differentiation between machine-to-machine protocols and human interaction.
- The functional viewpoint defines what security functions must be provided for each functional domain and how they work in concert to provide consistent security for the system as a whole.
- The implementation viewpoint applies security technologies in respect to common architecture patterns and system components.

3.3.2 THREAT MODELING AND SECURE DESIGN

All elements in an IIS are subject to various threats from various kinds of actors—anyone from employees and other insiders to casual hackers, terrorists, and state-sponsored actors, including from one's own state when it exceeds its authority. Comprehensive threat modeling of users, assets, data and entry points considers: [8]

- the solution as a whole,
- the security and privacy features,
- the features whose failures are security relevant and

[8] A structured analysis to identify, quantify, and address the security risks associated with an application or a system.

- the features that cross a trust boundary separating different trust levels or domains.

Secure system design requires consideration of not only threats and the typical software issues, but also hardware design at chip and device level, physical plant design, a robust personnel security program and supply chain security.

The functional capabilities that address these threats manifest differently in each viewpoint: they need to have a business rationale and value (business viewpoint), be coordinated by specific activities and roles (usage viewpoint), have specific security functions (functional viewpoint), and dictate some architectural and deployment properties while relying on specific technologies (implementation viewpoint).

4 THE BUSINESS VIEWPOINT

4.1 ELEMENTS OF THE BUSINESS VIEWPOINT

Business-oriented concerns such as value proposition, expected return on investment, cost of maintenance and product liability must be evaluated when considering an Industrial Internet System as a solution to business problems. To identify, evaluate and address these business concerns, we introduce a number of concepts and define the relationships between them, as shown in Figure 4-1. [9]

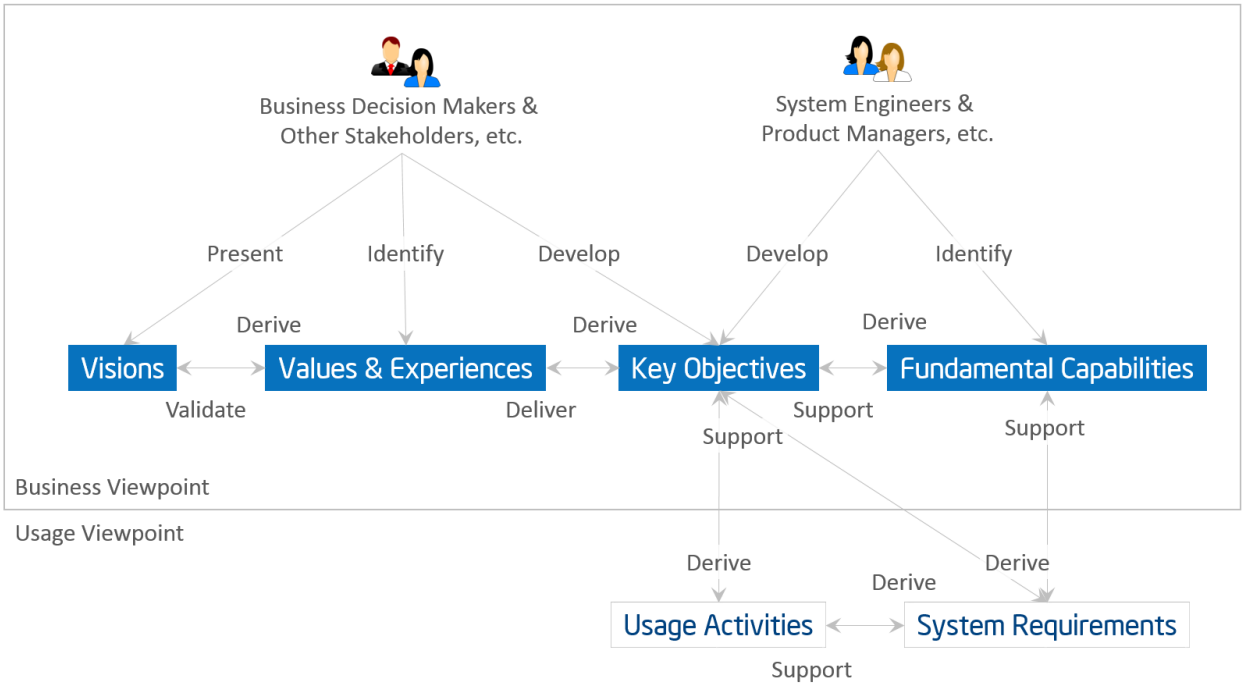


Figure 4-1 Value and Experience Model

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

[9] This approach is based on the work, An IoT Experience Framework (IoTEF), presented to the Industrial Internet Consortium by Intel Corporation and the Business Motivation Model (BMM) [24] 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.

Vision describes a future state of an organization or an industry.[10] It provides the business direction toward which an organization executes. Senior business stakeholders usually develop and present an organization's vision.

Values and experiences 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 and experiences 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 and experiences. 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 core business tasks.[11] 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 first to identify the vision of the organization and then how it could improve its operations through the adoption of an IIS. From the vision, the stakeholders establish the values and experiences of the IIS 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.

4.2 Security Concerns in the Business Context

In rationalizing business values and establishing key system objectives for IISs, security inevitably stands out as a key consideration in today's environment. Security considerations are driven by two main factors:

[10] 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).

[11] 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). 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.

Regulatory and compliance mandate controlling access to financial systems, protecting credit card information, upholding privacy expectations and protecting critical infrastructure. These are requirements the business must meet, no matter the cost.

Business value requires safeguarding the business investment in IISs and protecting their operations against the risk of damage brought about by security breaches. This damage may include interruption or stoppage of operations, destruction of systems, leaking sensitive business and personal data, and intellectual property, harming business reputation, and loss of customers. Heightened security objectives and standards are required to address these risks. However, they would lead to additional investment and likely extended time for system development and deployment. In some cases, they may affect user experience negatively. These additional costs must be justified to stakeholders in the context of the business risks they are addressing, sometime in terms of costs saved by averting damages.

Just like other key system objectives, there must be a way to measure and validate the continued effectiveness of security capabilities implemented in the system in meeting the security objectives. Derived from the security objectives and requirements, Security Metrics and Key Performance Indicators provide reports on compliance, regulatory, contractual or business driven, and create continuous feedback loops to increase accountability, improve effectiveness and provide quantified inputs for effective decision-making.

Security has a strong dependency on the quality of the design and implementation of a component or a system—attackers often gain access to systems by exploiting vulnerabilities in design (flaws) or implementation (bugs). The principle tenets of secure development are the same as in quality development: to apply rigor in documenting, tracking, and validating the desired features and to use known best practices to avoid introducing unknown or undesired side effects. Secure development lifecycle (SDL) is a security development process that is widely used by many organizations to avoid software vulnerabilities.[12] It begins by benchmarking existing quality practices and improving any shortcomings found during initial assessments. Benefitting from the identified process improvements, the resultant product is typically both more secure and of higher quality by being more reliable and having fewer issues reported by the customers. Investments in security therefore can have measurable and sometimes immediate returns.

[12] Microsoft: Security Development Lifecycle, SDL Process Guidance Version 5.2, 2012 [26]

5 THE USAGE VIEWPOINT

5.1 ELEMENTS OF THE USAGE VIEWPOINT

The *usage viewpoint* is concerned with how an Industrial Internet 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 IIS.

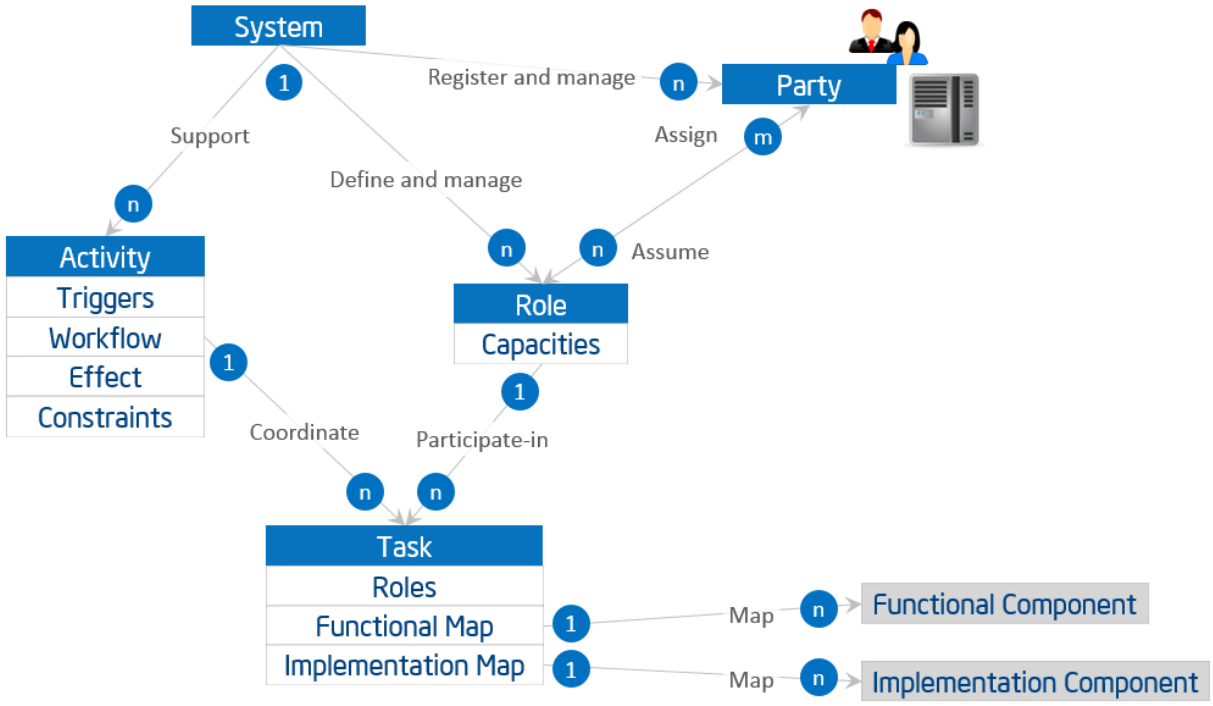


Figure 5-1 Role, Party, Activity and Task

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 IIS 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.

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 IIS, it is often associated with certain security properties (such as privileges, permissions, etc.). This association in turn may require a more refined notion of role—e.g. reflective of an organization chart, user groups, etc.—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* describes, 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.

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 IIS. 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 IIS 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.

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.[13]

The design of the IIS 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.

5.2 COMMON SECURITY ACTIVITIES

The enforcement of security policies requires the ability to control the various endpoints and their communications involved in an activity in a generic and consistent way to ensure complete end-to-end coverage. Four common security activities are described below.

Security monitoring gathers and analyzes security-related data continuously as activities are performed. It may take different forms depending on the context of operations and on security events. For example, different tasks are appropriate before, during and after an attack.

Security auditing collects, stores and analyzes of security information related to an IIS.

Security policy management manages both automated and human-driven administrative security tasks by documenting their usage and constraints.

Cryptographic support management consists of globally interoperable key management, secure credential storage and revocation.

[13] Defined in the functional viewpoint (6.1)

6 THE FUNCTIONAL VIEWPOINT

6.1 BACKGROUND

Industrial Control Systems (ICS) have been widely deployed to enable industrial automation across industrial sectors.[14] 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.[15,16] 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.[17,18] In OT (operations 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

[14] ICSs typically include supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as skid-mounted Programmable Logic Controllers (PLC) that are often found in the industrial control sectors [30].

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

[16] 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.

[17] http://en.wikipedia.org/wiki/Symbol_grounding_problem

[18] http://en.wikipedia.org/wiki/Chinese_room

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. 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 holds forth a great possibility of advancement - embodied cognition - to a system that can avoid the symbol grounding problem by basing its representation on the world (and not on programmer supplied models) and only its own episodic experience (and thus not be limited to human conceptions of epistemology). However even nearer term breakthroughs that will support advanced analytics based on actual world data rather than engineering models may well yield substantial improvements. The key obstacle is safety and resilience. Mission-critical OT applications are important enough that the typical levels of software reliability that are acceptable in the IT market will not be sufficient for OT. Moreover, actions in the physical world generally cannot be undone, which is a consideration that IT systems normally do not have to address.

Riding on continued advancement of computation and communication technologies, the Industrial Internet can 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. Moreover, 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, we can gain insight to a business's operations. With these insights, we can improve decision-making and optimize the system operations 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.

We create the concept of *functional domain* to address the key concerns surrounding the functional architecture of Industrial Internet Systems. A *Functional Domain* is a top-level functional decomposition of an Industrial Internet System, each providing a predominantly distinct functionality in the overall system.

This functional domain model is intended to be generally applicable to IISs in many industrial verticals. However, a use case in a certain industrial vertical may place stronger emphasis on functions in one or more functional domains than in the others. In some cases, the functional domains can be implemented as a single system, combined into a single one, or split up and implemented as a part of other domains.

The decomposition of functional domains highlights the important building blocks that have been seen to have wide applicability. The set of building blocks is neither a complete nor a minimum set that any system must possess. Rather, it is intended to be a starting point for conceptualizing a concrete functional architecture within these functional domains. A use case under consideration and its specific system requirements will strongly influence how the functional domains are decomposed, so in a concrete architecture derived and extended from this reference architecture, additional functions may be added, some of the functions described here may be left out or combined and all may be further decomposed as needed.

We decompose a typical IIS into five functional domains:

- Control domain
- Operations domain
- Information domain
- Application domain
- Business domain

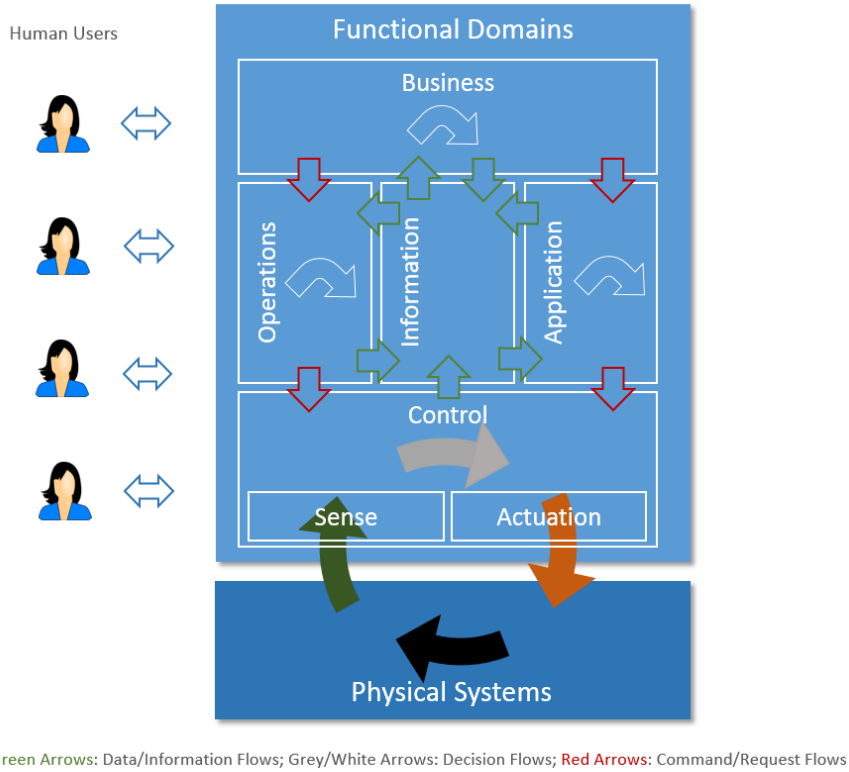


Figure 6-1 Functional Domains

Data flows and control flows take place in and between these functional domains. Fig 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.

565 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
570 derived, and new intelligence may emerge in the larger contexts.

We describe each in turn, starting from the bottom of Figure 6-1, above the physical systems.

6.2 THE CONTROL DOMAIN

The *control domain* represents the collection of functions that are performed by industrial control systems. The core of these functions comprises fine-grained closed-loops, reading data
575 from sensors ("sense" in the figure), applying rules and logic, and exercising control over the physical system through actuators ("actuation").[19] Both accuracy and resolution in timing is usually critical. 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. They may not be easily accessible physically by maintenance
580 personnel, and physical security of these systems may require special consideration.

> Examples: Simple examples of functional components in this domain include a control room 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.[20] Their
585 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. We describe each in turn.

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
590 system, for example, an attention element to tell the sensor what is needed.

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.

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

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

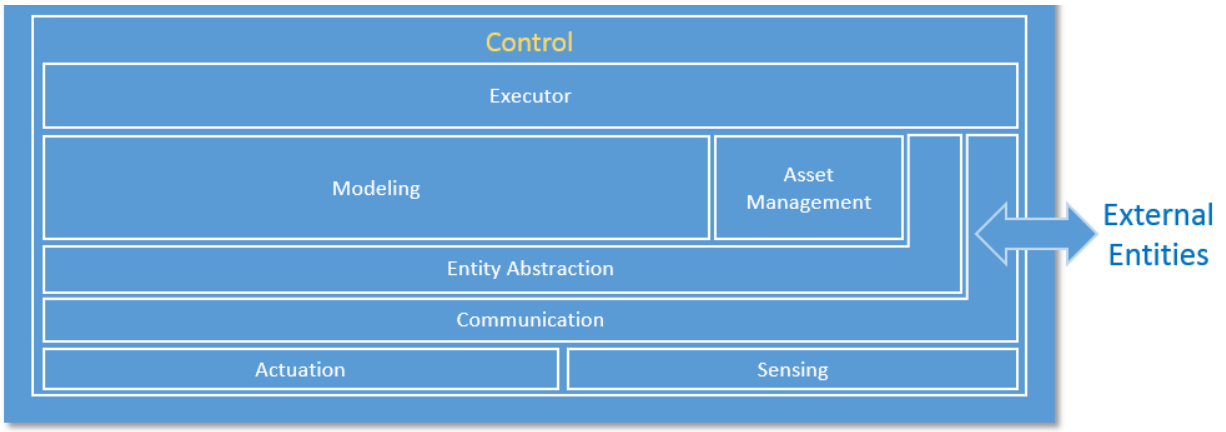


Figure 6-2 Functional Decomposition of Control Domain

595 *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
600 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
605 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,
610 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
615 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
620 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?[21]

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

6.3 THE OPERATIONS DOMAIN

The *operations domain* 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 up opportunities for added business and customer value as set out by higher-level, business-oriented domains.

> Examples: 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 IIS can be supported through a suite of interdependent operations support functions.

[21] 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., [33]

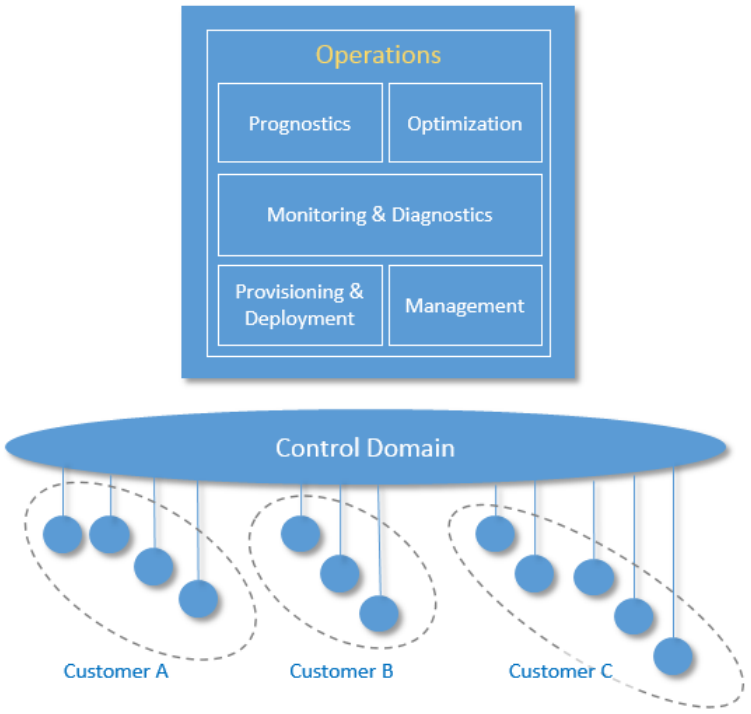


Figure 6-3 Operations Domain decomposition showing support across various customers

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. Provisioning and deployment has a strong dependency on functions in connectivity and security, trust and privacy.

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. Management has a strong dependency on functions in intelligent and resilient control.

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. Monitoring and diagnostics has a strong dependency on functions in data services and analytics.

Prognostics consists of the set of functions that serves as a predictive analytics engine of the IISs. 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. It may use the analytic functions in the information domain to realize its functions.

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 key automation and analytics features including:

- Automated data collection, processing and validation.
- Ability to capture and identify major events, such as downtime, delay, etc.
- Ability to analyze and assign causes for known problems.

Optimization has a strong dependency on functions in dynamic orchestration and automatic integration. It may use the data ingestion and processing functions and analytic functions in the information domain to realize part of its functions.

6.4 THE INFORMATION DOMAIN

The Information Domain 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.[22] 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.

Examples:

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

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

- 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.

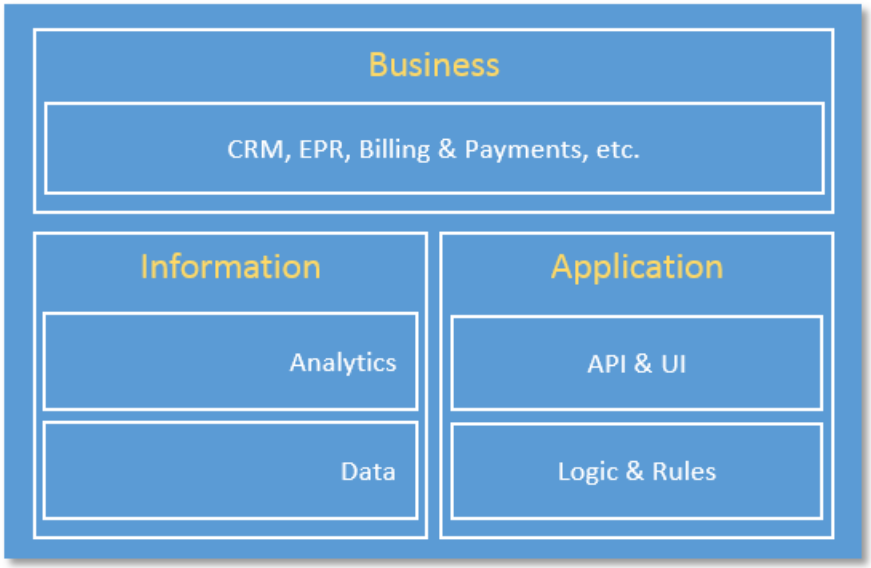


Figure 6-4 functional decomposition of Information, Application & Business Domains

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 IIS 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.

6.5 THE APPLICATION DOMAIN

The *application domain* 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 core logics, e.g., 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.

6.6 THE BUSINESS DOMAIN

The business domain functions enable end-to-end operations of the Industrial Internet Systems by integrating them with traditional or new types of Industrial Internet specific business functions including those supporting business processes and procedural activities. Examples of these business functions include enterprise resource management (ERP), customer relationship management (CRM), asset management, service lifecycle management, billing and payment, human resource, work planning and scheduling systems.

> Examples: 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.

6.7 COMMON SECURITY FUNCTIONS

We summarize below a set of common security functions that may be needed in each of the functional domains and in the interactions between the functions in different domains. The common security functions are based on the international standard '*Common Criteria for Information Technology Security Evaluation.*[23] Note that this document does not cover security compliance, which will be addressed in a future Security Reference Architecture document.

- *Security audit* involves the collection, storage and analysis of security information related to the industrial system.
- *Identity verification in communications* assures the integrity of both the originator and the recipient of a communication in the industrial system.
- *Cryptographic support* provides the resources necessary to support the encryption and decryption operations (software and hardware).
- *Data protection and privacy* assures the confidentiality of data in transit and at rest. If the data contains information owned or related to a third party, it provides protection of the privacy of records as described by a security policy.
- *Authentication and identity management* ensures the components are only accessed by the roles specified in the security policy.
- *Physical protection* provides necessary protections against physical tampering and unauthorized observation as described by a security policy.

These functions contribute to various system security capabilities including:

- *secured booting* to a known secure state enabled through cryptographically signed software and firmware,
- *enhanced trust* by network and application whitelisting and reputation-based approaches (or dynamic approvals),
- *enhanced privacy* through mutual authentication in communication integrated with means to automatically ensure the privacy of data.
- *early attack detection* through rigorous anomaly detection over established norms of traffic patterns and simplified system,
- *secure management* of all systems and their update processes and
- *automatic threat containment* to minimize the damage of a successful attack.

[23] The Common Criteria for Information Technology Security Evaluation (abbreviated as Common Criteria or CC) is an international standard (ISO/IEC 15408) [23] for computer security certification.

7 IMPLEMENTATION VIEWPOINT

The *implementation viewpoint* is concerned with the technical representation of an Industrial Internet System and the technologies and system components required to implement the activities and functions prescribed by the usage and functional viewpoints.

An IIS 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.[24] 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 IIS.

The implementation viewpoint therefore describes:

- The general architecture of an IIS: 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.
- An implementation map for the key system characteristics.

7.1 ARCHITECTURE PATTERNS

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

- Three-tier architecture pattern
- Gateway-Mediated Edge Connectivity and Management architecture pattern
- Edge-to-Cloud architecture pattern (This pattern contrasts with the gateway-mediated pattern as it assumes a wide-area connectivity and addressability for devices and assets.)
- Multi-Tier Data Storage architecture pattern (This pattern supports a combination of storage tiers (performance tier, capacity tier, archive tier.)
- Distributed Analytics architecture pattern.

An architecture pattern is a simplified and abstracted view of a subset of an IIS implementation that is recurrent across many IIS, yet allowing for variants. For example, an implementation of the three-tier pattern in a real IIS 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

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

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 first two patterns in the previous list because of their stronger prevalence in IISs.

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 section 6) involved in usage *activities*. They are connected by three networks, as shown in Figure 7-1.

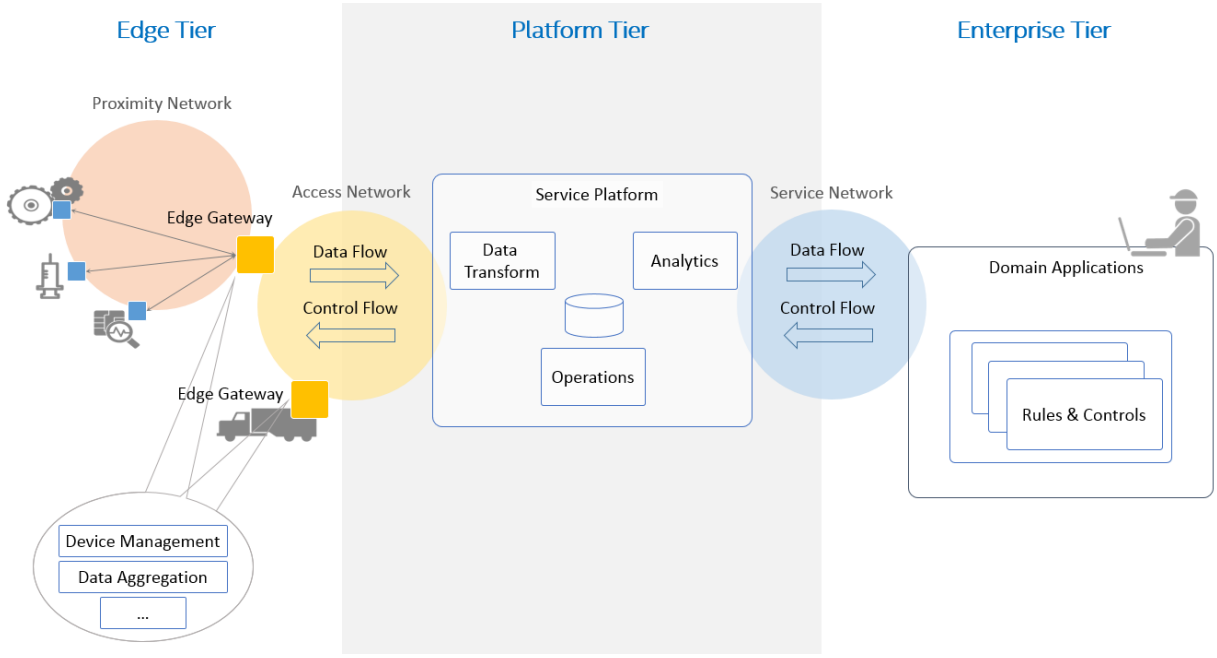


Figure 7-2 Three-tier IIS Architecture

The *edge tier* collects data from the edge nodes, using the proximity network. The architectural characteristics of this tier, 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 originates 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 vocation 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. It may be a corporate network, or an overlay private network over the public Internet or a 4G/5G network.

The *service network* enables connectivity between the services in the platform tier and the enterprise 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.

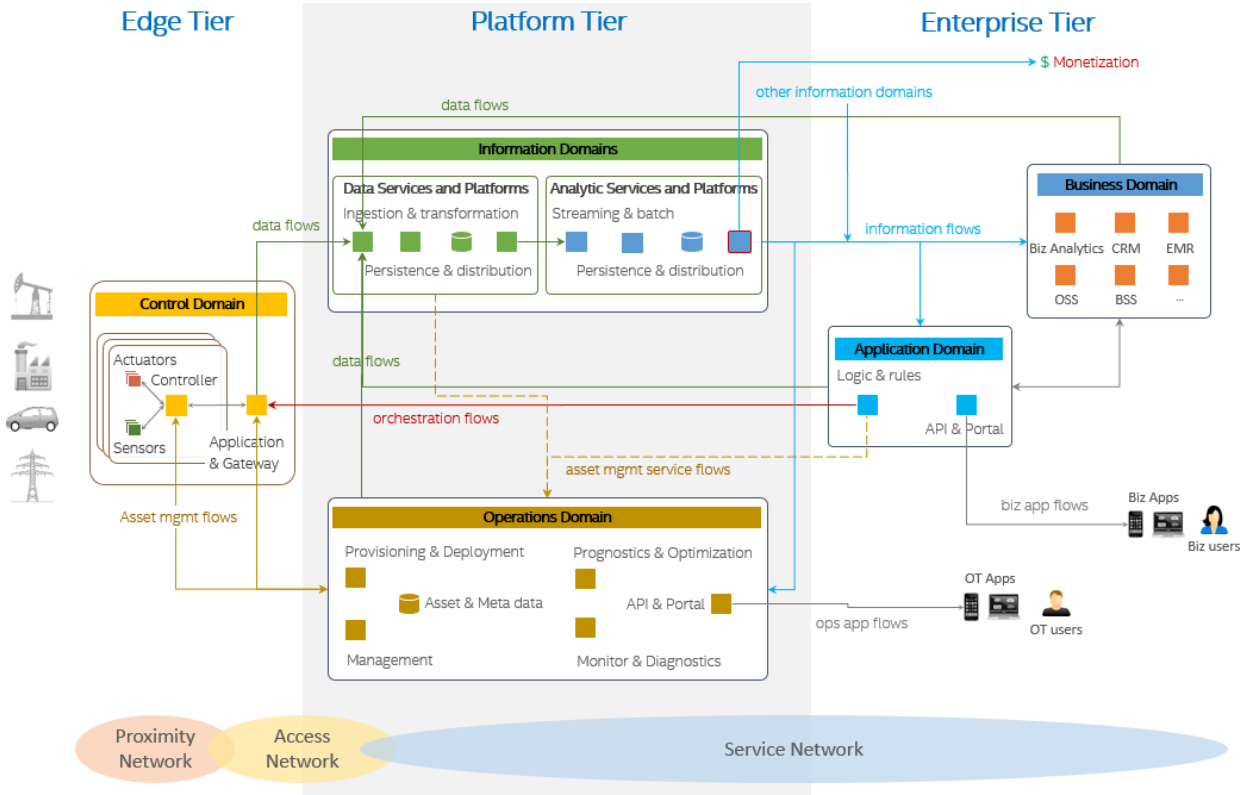


Figure 7-3 Mapping between a three-tier architecture to the Functional Domains

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 7-3. 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. The actual functional mapping of IIS tiers is usually not as simplistic and is highly depends on the specific 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 IIS. These services—e.g. data analytics from the *information* functional domain—then become supportive of other functional domains in other tiers. For example:

The *asset management flows* (see fig. 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 in fig. 7-2*) to other components, either in the same tier or in another.

For example, the data services (*information* domain) component of the *platform* tier may request services from the *operations* domain component for:

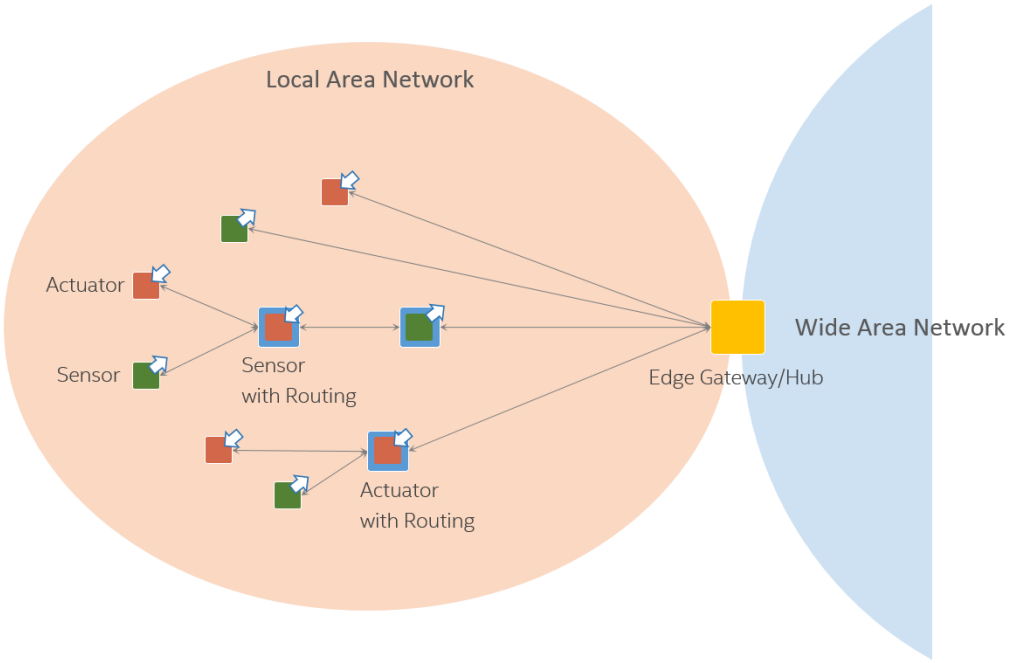
- The verification of asset credentials it receives in the data flows from the edge tier.
- The query of asset metadata so it can augment the data received from the assets before the data are persisted or fed into analytics in the next stage of processing.

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 IIS, with a gateway that bridges to a wide area network as shown in Figure 7-4. 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 IISs, 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.



915

Figure 7-4 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.

920 The local network may use different topologies.

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.

925 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
930 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.

7.2 SECURE IMPLEMENTATIONS

To secure an Industrial Internet System, we outline a number of important and common security issues to be addressed in its implementation.

End-to-end security: To achieve end-to-end security in an IIS, its implementation must provide:

- protected device-to-device communications,
- confidentiality and privacy of the data collected,
- remote security management and monitoring,
- simultaneously addressing both existing technologies as well as new technologies, and
- seamlessly spanning both information technology (IT) and operational technology (OT) subsystems and processes without interfering with operational business processes.

This effort requires building in security by design rather than the often-tried and often-failed paradigm of bringing it in as an afterthought.

Securing legacy systems: Most IISs incorporate legacy systems due to the effort and capital expense involved in replacing or retrofitting these systems in industrial plants, hospitals and infrastructures. Often the legacy endpoints in these systems implement limited or no security capability in processing and in the protocols they use, and they are not modifiable to add the requisite security capability. Security of the overall system requires minimizing the attack surface of these legacy systems.

The use of security gateways is an approach to secure legacy endpoints and their protocols. A security gateway acting as proxies for the legacy endpoints bridges the legacy protocols supported by the legacy endpoints and their counterparts used by new endpoints. A security gateway isolates the attack surface introduced by the legacy endpoints and their protocols to the links from these endpoints. However, isolating the attack surface is insufficient, as we still need to detect attacks. We want to detect security attacks by analyzing the data for anomalies and abnormal behavior in a way that can be done within the threat surface, that is not all attacks will be routed through the gateway.

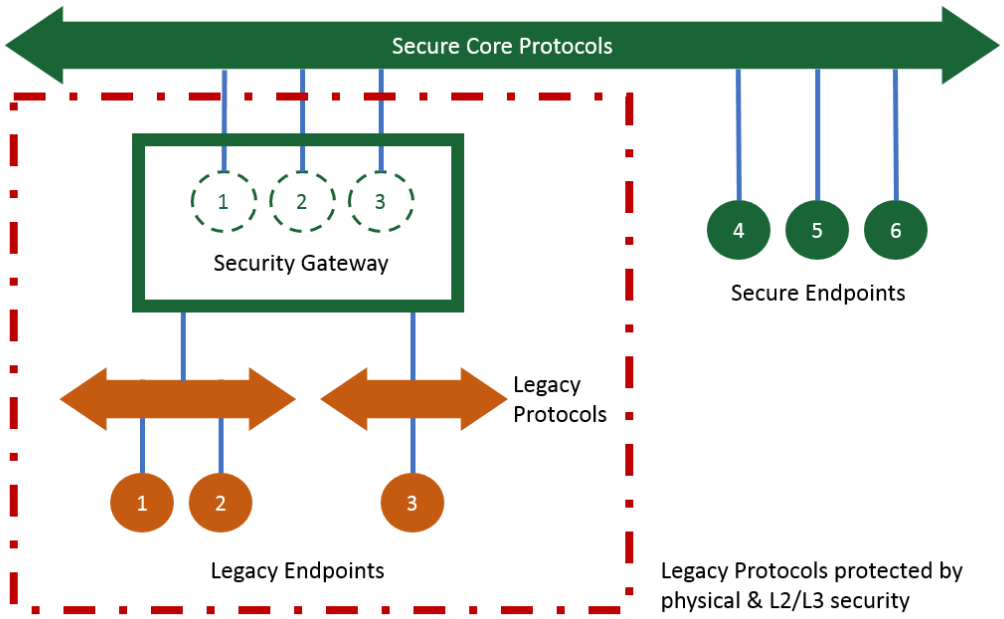


Figure 7-5 Security Gateway Deployment Pattern

Security for architectural patterns: Every architecture pattern has its own specific security requirements and challenges. In the three-tier architecture pattern, for example, there are four critical areas and operations to secure:

- end-points
- information exchange
- management and control
- data distribution and storage

End-point security: Many IISs need to embed security capabilities and policy enforcement directly in end-point devices. It includes the enablement of the remote management and monitoring of end-points for near real-time security responses during an attack as well as for proactive security measures prior to an attack. The end-point should have the ability and autonomy to defend itself and must remain resilient even when disconnected from the external security management systems. Endpoints must be able remain secure and resilient even when adjacent peer endpoints are compromised.[25] The embedded security measures should include mitigating controls, countermeasures and/or remediation actions defined by security policies to minimize the risk of being compromised and the impact when being compromised.

Information exchange security: Communication and data exchanges within an IIS must be protected for authenticity, confidentiality, integrity and non-repudiation. Security solutions and practices in information technologies can be applied to network segments and applications that are built on information technologies-based infrastructures in an IIS. In some industrial

[25] Both in the physical and the process sense.

environments, legacy communication technologies, protocols and processing capability may limit the full security implementation for information exchange. In these environments, the security
995 gateway approach discussed above may be employed to protect the information exchange between the local legacy environments and the broad systems while enforcing logical isolation and physical protection for the local environments until the inadequate legacy systems are phased out over time.

Part II: Analysis of Key System Concerns

Part I described the several viewpoints and the various functional domains needed to evaluate Industrial Internet Systems. However, there are common concerns that cannot be assigned to a particular viewpoint or functional domain, such as the key system characteristics that we discussed in Chapter 2. Addressing these concerns requires consistent analysis across the viewpoints and concerted system behaviors among the functional domains and components, ensured by engineering processes and assurance programs. We call these *key system concerns*.

In this part, we highlight a few of these key system concerns in Industrial Internet systems as special topics and provide additional analysis on them. For some of these topics, we summarize their key elements based on prevailing and matured technologies and practices most relevant to Industrial Internet Systems. In others, we introduce some forward-looking ideas bridging what is in place now and what is needed in the near future to support the kind of IISs that we envision. These topics are:

Safety highlights a number of important considerations for safety in IISs.

Security, Trust & Privacy provides additional (to those presented in Part I) details on how to secure IISs end-to-end.

Resilience presents a few ideas on how to establish a resilient system, in reference to some of the learning from the military programs and operations.

Integrability, Interoperability and Composability suggests the direction in which IISs components should be built to support the dynamic evolution of components, including self-assembling components. It also serves as a unifying reference topic for some of the topics, such as Connectivity, Data Management, and Dynamic Composition and Automatic Integration, all of which are to follow.

Connectivity discusses a foundational aspect of Industrial Internet—how to connect the numerous components (sensors, controller, and other systems) together to form IISs.

Data Management concerns the basic approaches for exchanging and management of data among the components in IISs.

Analytics concerns the transformation of vast amounts of data collected in an IIS into information that be used to make decisions and system optimization.

Intelligent and Resilient Control presents a conceptual model and some key ideas on how to build intelligent and resilient control.

Dynamic Composability and Automatic Integration concerns flexible adaptation to optimize services as environments change and to avoid disruptions as components are updated.

8 SAFETY

The Industrial Internet and the systems it comprises manage a variety of *safety-critical* processes. *Safety* is a key concern of IISs that must be considered and analyzed throughout their lifecycle. Depending on the operating domain, regulatory requirements may mandate that a target safety assurance level be established for IISs using a risk assessment process. While there are existing safety standards that may apply to IISs under development for different domains (e.g., nuclear, rail, medical, automotive, process, maritime, machinery, and industrial process control), many are based on the fundamentals established in ISO/IEC 61508 *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems* and do not explicitly address safety issues related to the cross-cutting concerns, architecture, integration and overall lifecycle of IISs. A complete dissertation of techniques for establishing and meeting IIS safety goals is beyond the scope of this document, but some of the relevant concerns are presented here.

Safety is an *emergent property* of the system in question and that has two major implications for systems engineering:

- Safety is not compositional: safety of every component in the system does not *necessarily* imply safety for the system as a whole.
- One cannot predict the safety of a system in a particular situation without first predicting the behavior of the system in that situation.

Therefore, IIS design must focus on not only mandating general notions of safety but also providing mechanisms that enable systems integrators to measure, predict and control the behavior of the system. For systems integrators to ensure safety, they must understand the intended behavior of the system and at the same time employ mechanisms that can constrain unintended behavior. Safety can be addressed either passively (e.g. by adding guards around a process to make sure nothing escapes the guarded area) or actively (e.g. by adding components that adjust the systems behavior to assure it is safe). Mechanisms include, but are not limited to:

Support for independent functional safety features: A functional safety feature is a feature the rest of the system relies upon to ensure safe operation. Examples include airbags in automobiles, ejection seats in military aircraft, and the automatic shutdown system in nuclear reactors. It is not possible to prescribe specific functional safety features in general because a functional safety feature for one system or context may result in unsafe behavior in another. That being said, each safety-critical IIS must implement the functional safety features necessary to its safety requirements and usage context. Architecturally, functional safety features should be isolated and independent of the rest of the system to the maximum possible extent. This simplifies system safety validation and allows system integrators to mitigate costs associated with ensuring safe system behavior (see subsection “The Role Reliability and Resilience in Safety-Critical Systems” below).

Well-defined, verified and documented interfaces: The system components used in IISs must have well-defined, verified and documented interfaces. Systems integrators can leverage these

specifications, and the evidence that demonstrates that components conform to its interface, to make predictions about the emergent behavior of the composite system. Interfaces of concern include software, such as APIs, and relevant physical and processing characteristics that include the resource usage requirements and how the component will behave when used in its intended environment. Component manufacturers should make available any evidence used to verify that a component conforms to its specification. This helps system integrators predict how two or more components may interact when composed.

Enforceable separation of disparate functions and fault containment: Component manufacturers cannot provide complete assurance of component behavior because testing cannot cover all eventualities. Additionally, system integrators may opt to control costs by using components that come with less assurance (providing those components will not be used to support a safety critical function).

Systems integrators must ensure that low assurance components will not negatively impact safety critical system functions. Thus, IIS design must have mechanisms to enforce separation between disparate functions and components. The enforcement mechanism must isolate faults and prevent *unintended* interactions between different system components. Examples of unintended actions include:

- A software component stealing CPU resources from another.
- A software component corrupting the data or instructions of another.
- A device on the network becomes a "babbling idiot" and preventing other components from communicating in a timely manner.
- A can of liquid on a conveyer spilling and causing the floor near the machine to become slippery, causing a mobile robot to lose traction.
- A motor drawing more power than expected causing a brownout affecting other devices on the same branch circuit.

Runtime monitoring and logging: Engineering is a human activity, and our knowledge of engineering is constantly improving. System failures, when they occur, should be looked upon as an opportunity to learn more. Mechanisms for gathering and preserving the episodic chain of events that has led to a failure may be useful to determine the underlying causes of a particular failure incident. Runtime monitoring and logging is one approach to gather and preserve such information.

In addition to supporting post-accident forensic activities, runtime-monitoring and logging can help *prevent* accidents. Runtime monitoring can detect if the system under scrutiny has entered or is trending towards an unsafe state and generate an alert. Some systems are equipped with special safety functions that automatically activate a safety mode in response to this alert. These safety modes are designed to either drive the system to a safe state, or prevent the system from entering unsafe states in the first place (e.g., the automatic shutdown system of a nuclear reactor). The runtime monitor can trigger such mode changes.

8.1 RELATIONSHIPS WITH OTHER CONCERNS

Safety interacts with other system concerns.

The role of reliability and resilience in safety-critical systems: Neither 'reliability' nor 'resilience' imply system safety. Indeed, there can be reliable systems that are unsafe in that they reliably perform unsafe behavior, and there can be unreliable systems that are safe. Even so, reliable and resilient *infrastructure* is useful and sometimes necessary to support the safety-critical functions of a larger system. Examples of features that support reliability and resiliency include fault tolerant computation and communication, replicated communications, distributed consistency protocols, adaptive control algorithms, and 'hardened' components, such as radiation-hardened CPUs and memory. Unfortunately, increased reliability may increase cost. This can be mitigated at the architecture level: The IISs could partition the infrastructure to allow for separation between the infrastructure utilized by safety and non-safety functions. The safety functions reside on a high-reliability (or high-resilience, and high-security) partition while non-safety functions are deployed on a less reliable (but cheaper) partition.

The relationship between safety and security: Often, system safety requirements impose system security requirements. Sometimes safety depends on the presence of a security feature. For example, if a platform cannot protect application code from unauthorized modification, malicious actors can corrupt safety-critical control algorithms and drive the system into an unsafe state. Sometimes, safety depends on the absence of a security feature. For example, one may actually want unauthorized or unauthenticated users, such as emergency responders to have the ability to initiate emergency shutdown procedures. Safety and security requirements (and their possible implementations) must be carefully balanced.

Implications of dynamic composition and automated interoperability for safety: Traditionally, safety-critical systems have been designed, manufactured and integrated by a single systems integrator. This model of integration allows the systems integrator to ensure the safety of their system by decomposing system safety requirements to sub-systems in a top-down manner and by verification and validation to ensure that no unsafe interactions were introduced during integration. Dynamic composition and automated interoperability functionality enable two models of system integration, each with their own set of implications for safety and the IIRA.

*Accelerated traditional***:** Here systems integrators still design, manufacture, and integrate the system prior to its delivery to the customer. The systems integrator understands the top-level system safety requirements and the collection of specific system components that will be composed to comprise the system. In this model, dynamic composition and automated interoperability features, such as active inter-component interface checking, can reduce integration effort by the systems integrator *if* those features can be trusted. If the dynamic composition and automated interoperability features cannot be trusted, then it would be possible to compose components with incompatible interfaces, and other integration verification activities must be performed by the systems integrator to ensure the system components interact properly with each other. Therefore, if dynamic composition and automated interoperability features are used to support safety-critical functions, those features themselves must be verified and validated to the same level of assurance as the safety critical function.

User assembled: Here component manufacturers market a variety of interoperable components. Users can buy those components and compose them into a system designed to meet the user's specific needs. Users effectively act as the integrator of these systems. Unlike the traditional integration model, here the integrator (the users) would not have the necessary engineering expertise or resources to ensure the safety of the composite system. Indeed, the user may not even have a comprehensive understanding of the top-level safety requirements for the composite system. Instead, the dynamic composition and interoperability features of both the IIS infrastructure and IIS components must be designed to enforce safe system integration.

9 SECURITY, TRUST AND PRIVACY

To address the concerns of security, trust and privacy in Industrial Internet Systems, end-to-end security capability must be provided to harden endpoints, secure device-to-device communications, enable remote management and monitoring, and secure data distribution. This end-to-end security capability with real-time situational awareness should seamlessly span the functional domains, and the information technology (IT) and operational technology (OT) subsystems and processes without interfering with the operational business processes.

Today, we build IISs with technology from multiple vendors who provide heterogeneous components with various levels of security. This is fertile ground for weak links in the assembled system that must be addressed by building security in by design rather than the often-tried and often-failed paradigm of bringing in security as an afterthought.

Secure design requires establishing the relevant security concerns for endpoints, the communication between them, the management of both the endpoints and the communication mechanisms, and for processing and storing data. The following security concerns must be considered for each of the viewpoints:

- business viewpoint, for an assessment of business risks, cost factors, regulatory and audit requirements, and what is the ROI on security investments,
- usage viewpoint, for a description of security procedures, and of how to secure end-to-end activities in an IIS, including privileges assigned to their roles,
- functional viewpoint, for a detailed assessment of security functions required to support end-to-end secure activities and operations and
- implementation viewpoint, for ensuring secure architectures and the best use of relevant security technologies.

Security requirements in each of these viewpoints can be analyzed and addressed separately, but a comprehensive security solution requires considering the interplay between them, for example, how some system designs (implementation viewpoint) need to comply with costs aspects (business viewpoint), or how some security functions (functional viewpoint) may not be appropriate for some end-user requirements (usage viewpoint).

Flaws introduced during the design of IISs can be exploited by hackers and intruders leading to data leakage, business disruption, financial losses and damage to products and company brands. This mandates that security be integrated from the outset with a comprehensive development lifecycle encompassing not only the software design lifecycle, but also hardware design at chip and device level for hardware-backed security, secured physical design (e.g. tamper-resistant/proved) for the devices and equipment, and physical plant design along with a robust personnel security program. Each of these requires training and data gathering through the development, deployment and operations of these complex systems.

It is important to keep in mind that security metrics are difficult to define a priori for a system. In order to provide assurance for security the IIS needs to implement security best practices

appropriate to the application according to stakeholder policies. Some of the proposed best practices and implementation patterns are described in this section.

To build a comprehensive security solution, the IIS needs to address the following relevant security concerns:

- endpoint security,
- communication security between the endpoints,
- management and monitoring security of both the endpoints and the communication mechanisms and
- data distribution and secure storage.

The following subsections examine each of these concerns.

9.1 ENDPOINT SECURITY

The security of the endpoint is fundamental to the security of the data and control of IISs. The exact nature of the endpoint security is heavily dependent upon the type of endpoints and what interfaces they expose. However, the security measures that are required to protect them share many common security functions. These common security functions can be organized and implemented consistently as self-contained modules that can be deployed in the endpoints to enforce uniform security policies. Once deployed, such a security agent can monitor and perform security management on the activities within an endpoint and its communication with other endpoints.

There are many ways to attack an endpoint and therefore many issues to address. The issues to address include:

- Secure boot attestation
- Separation of security agent
- Endpoint identity
- Endpoint attack response
- Remote policy management
- Logging and event monitoring
- Application sandboxing
- Application whitelisting
- Network whitelisting
- Endpoint and configuration control to prevent unauthorized change to the endpoints
- Dynamically deployed countermeasures
- Remote and automated endpoint update
- Policy orchestration across multiple endpoints
- Peripheral devices management
- Endpoint storage management
- Access control

9.1.1 Secure Boot Attestation

An endpoint must start from a known secure state, following only a prescribed boot sequence of steps, with no modification of intended execution function. To ensure this, remote attestation to the integrity of the boot sequence (via the secure agent), as well as policy to describe how to proceed when deviation from the expected boot sequence is detected may be used.

In the event that an endpoint's boot sequence has been found altered in an unexpected way, the boot process shall fail, and optionally report failure via the secure agent. The endpoint should either be stopped or be quarantined, depending on policy. This ensures that an endpoint that has been tampered with does not participate in the IIS, thus preventing an attack entry point to the overall system.

9.1.2 Deployment of Security Agent

Four primary security separation models exist to deploy the secure agent at the endpoint: *process, container, virtual* and *physical*.

Process-based security agent: If the security agent resides in a process, then it shares the operating environment with other processes. This is the traditional security model, common in the home environment in the form of anti-viruses and miscellaneous security software. This model is well understood and widely implemented, but suffers from severe security weaknesses. For example, if a process on the device is compromised, it may serve as an attack vector for the agent to be compromised. [26]

Container-based security agent: The security agent can also be implemented with a secured container within the endpoint. With this approach, the separation is implemented using hardware- and software-enforced boundaries. Container-based security agents include operating system containers (software), Trusted Platform Module, hardware co-processors, secure memory mapping and code execution crypto operations. [27]

Virtualization-based security agent: Hypervisors in virtualized environment are widely used to enforce security policies transparently on enterprise and cloud applications in enterprise IT and cloud computing environments. Applied to security management of devices in the OT environment, this approach allows the security agent to function independently in its own environment without changing the existing endpoint functions and its OT operating environment. However, operating within the same physical endpoint as the OT environment does, the security agent gains increased visibility to the activities of the OT environment and is thus able to control security activities such as embedded identity, secure boot attestation, communications (implementing firewall, on-demand VPN connections, mutual authentication,

[26] A path or means (e.g. viruses, e-mail attachment, Web pages, etc.) by which an attacker can gain access to a computer or network server in order to deliver malicious payloads or outcome.

[27] Such as the ARM TrustZone and Intel Software Guard Extensions.

communication authorization, data attestation, IDS/IPS, etc.), all transparent to the OT environment.

Gateway-based security agent: When security cannot be added to an endpoint, as is the case for legacy systems, a security gateway or bump-in-the-wire implementing the security agent function as a physically separate network node can be deployed to secure these type of endpoints and their communications. Because the security agent is not physically on the same endpoint that it protects, advanced security functions such as secure boot attestation or application whitelisting in that endpoint cannot be easily implemented.

9.1.3 Endpoint Identity

Endpoints and other controllable assets in an IIS must have a unique identity so they can be managed and tracked via the secure agent. Ideally, this identity is hardware-embedded so that it cannot be altered. Identifiers traditionally used in applications, such as IP address, MAC address, host name Bluetooth address and IMEI, are not sufficiently secure for they can be changed easily and spoofed trivially.

Credentials may be issued to the holder of each identity to prove that its identity is genuine. These credentials, such as cryptographic keys, must be secured by hardware. These measures are required to prevent logical attacks by "impersonating" a legitimate identity, and physical attacks by replacing a genuine asset with a forgery.

9.1.4 Endpoint Attack Response

When an endpoint is attacked, it should defend itself, report the attack and reconfigure itself to thwart the attack based on policy. The responsible security management system (see section 9.1.10) should provide the policy to the secure agent in the endpoint in response to the attack, or *a priori* for use when communication with the server is severed.

Endpoints must remain resilient and secure even when their peer endpoints have been compromised. If an endpoint is able to recognize that a peer has been compromised, it must report the event to the security management system. The security management system should then quarantine that compromised endpoint to contain the damage and diminish the risk of the compromise being spread.

Upon the detection of an attack, an endpoint may increase the level of security monitoring and analysis, and stop suspicious processes and services. As the threat subsides, a decay algorithm slowly should reduce the risk assessment, as appropriate, to bring the system back to the steady state, resetting appropriate policy along the way.

9.1.5 Remote Policy Management

A central security management system defines the configuration of the security controls and functions as a form of a security policy for each endpoint. The security policy is communicated to the secure agent that authenticates and enforces the policy at the endpoint. Policies can be

modified and updated to the security agent on-demand to address new vulnerabilities or changing concerns in response to changing circumstances.

9.1.6 Logging and Event Monitoring

The security agent must be able to monitor and record events as they occur at the endpoint including events pertinent to security violation, user login/logout, data access, configuration update, application execution and communication.

The endpoint policy defines the events of interest and how specific event records are persisted. This includes location of the storage and the rule for retention to guard against premature deletion of event records. For example, event logs can be stored in a known location on the local file system, or at a remote location that can survive endpoint tampering or failure. The policy should contain provision on access control to prevent unauthorized access and tampering and privacy control to prevent the leaking of personally identifiable information.

9.1.7 Application Whitelisting

Mechanisms should be in place at the endpoint to ensure that only known and authorized application code (whitelist) including binaries, scripts, libraries are allowed to execute on the endpoint to prevent the endpoint from being compromised by malicious code. All other execution attempts should be halted, logged and reported. The security management system may update the application whitelist in the policy at the secure agent for its enforcement at the endpoint.

9.1.8 Network Whitelisting

Mechanisms should be in place at the endpoint to ensure that only a defined set of source/destination, port and protocol tuples is allowed to communicate to/from the endpoint. All other communication attempts should be terminated, logged and reported. Trusted and secured mutual authentication is desirable and required in some cases to prevent masquerading, man-in-the-middle attacks and other network-based attacks. The security management system may update the network whitelist in the policy at the secure agent for its enforcement at the endpoint.

9.1.9 Dynamically Deployed Countermeasures

The security management system should be able to deploy trusted new countermeasures and other mitigating controls as part of the endpoint security policy to the security agent for its enforcement at the endpoint.

9.1.10 Remote and Automated Endpoint Update

The security management system must be able to remotely update the endpoint with trusted software updates via the secure agent through an automated and secure process. The firmware and software updates must be first authorized by the security management system before distributing them to the security agents at the endpoints. Upon receiving the updates, the

security agents must validate the update based on its policy before allowing them to be implemented at the endpoints.

9.1.11 Policy Orchestration Across Multiple Endpoints

Policy orchestration is the coordination of security policy across multiple endpoints to enable secure, trusted operation workflow across these endpoints. For example, a data-generating sensor endpoint and a storage endpoint must have synchronized a consistent policy for the data generated in the former to be stored in the latter.

9.1.12 Peripheral Devices Management

Peripherals on an endpoint must be managed based on security policy concerning whether to allow a peripheral to be connected to or disconnected from the endpoint. Any violation of this policy, such as unauthorized removal of a peripheral, may cause the endpoint to be considered compromised and thus subject to quarantine. The security policy should disable by default all communication ports such as USB or other console ports at an endpoint unless they are used for operations. The security agent may allow a port to be opened temporarily for diagnostic purposes, however the port should be closed immediately when it is no longer used.

9.1.13 Endpoint Storage Management

Data storage and file systems at an endpoint must be managed based on security policy. The security management function includes file integrity monitoring, file reputation tracking (blacklisted, gray-listed, and whitelisted), data, file, file system or device-level encryption, file and data access right management, remote access to file system, data loss prevention, and alerting policy violations reporting.

9.1.14 Access Control

Network access to endpoints must be controlled based on security policy that allows connections required by the operations and deny all other connections. The unauthorized access attempts may be logged and reported to the security management system for analysis. These unauthorized access attempts may be the result of a misconfiguration or an indication of attack that requires appropriate response.

Threats enacted through physical access to endpoints must be considered. Disconnecting power or network cables to an endpoint should not result in vulnerability beyond an endpoint going offline.

9.2 Communication Security

In addition to communication solutions with well-known and mature security features, such as Ethernet and IP-based connectivity, industrial systems use an assortment of industrial-specific and often vendor-specific legacy solutions that have limited or no security features. In some cases, a number of legacy communication solutions are used in separate segments of the network

where new systems are meshed with legacy ones. Securing communications consistently between legacy endpoints and those using new solutions presents special challenges in IISs.

This section describes how communications between IIS components must be secured, and presents scenarios where security must be considered in the following areas:

- Architectural considerations for information exchange security
- Security in request-response and publish-subscribe communications
- Mutual authentication between endpoints
- Communication authorization
- Identity proxy/consolidation point
- User authentication and authorization
- Encryption communication

9.2.1 Architectural Considerations for Information Exchange Security

When designing security solutions, consideration must be given to requirements in confidentiality, integrity, availability, scalability, resilience, interoperability and performance for both transport layers (the *communication transport layer* and the c*onnectivity framework*, as described in Chapter 12*).* Protecting communication links at each layer requires corresponding security controls and mechanisms applicable to that layer. An important design question, therefore, is which layers to protect, and how to protect them for a given industrial application. Providing security controls in all layers may be necessary for some applications but may bring unacceptable performance costs for others.

9.2.2 Security in request-response and publish-subscribe communications

Two common patterns in IIS communications are request-response and publish-subscribe. The request-response pattern is common in industrial systems. Examples of the implementation of this pattern include Java Remote Method Invocation (Java RMI) [6], Web Services/SOAP [7], RPC-over-DDS [8], RESTful Servers, OPC [9], Global Platform Secure Channel Protocol and Modbus [10]. As the protocols of this pattern vary in degrees of support for security, they should be independently and carefully evaluated with regard to confidentiality, integrity and availability requirements. As an example, Modbus, a popular application-level fieldbus protocol within industrial systems, lacks support for authentication and encryption, and does not provide message checksums, and lacks support for suppressing broadcast messages.

Some implementations of the publish-subscribe pattern, such as MQTT or AMQP, rely on an intermediary message broker that performs a store-and-forward function to route messages; others such as DDS may be broker-less. Endpoint security policy should be applied both the publish-subscribe endpoints as well as the message broker (for the former case). For the publish-subscribe pattern the primary categories of threats are: unauthorized subscription, unauthorized publication, removal and replay, tampering and unauthorized access to data.

9.2.3 Mutual Authentication Between Endpoints

Endpoints must be able to perform mutual authentication before exchanging data to ensure data are only exchanged with intended parties and not leaked to malicious or unauthorized entities. The security policy may specify the acceptable authentication protocols and credentials to be used for authentication. Resource-constrained devices, such as sensors, may lack the capability to perform cryptographic intensive operations and implement lightweight authentication protocols instead to limit the vulnerability.

9.2.4 Communication Authorization

Before granting access of any resource, to an authenticated party, authorization must be performed at the endpoint according to the security policy. The security policy may specify fine-grain authorization rules such as what data records to share with whom, under what condition (e.g. encryption, anonymization or redaction of certain data fields) including temporal and spatial conditions.

9.2.5 Identity Proxy/Consolidation Point

In existing industrial deployments ("brown-field"), the identification of endpoints and their authentication may not be achievable in a way that is consistent with higher security standards. In this case, these components may be proxied by an endpoint with capability that meets the higher standard and capable of performing the proxy functions—the proxying endpoint is referred to as a security gateway (see section 7.2). The security gateway, among other functions, provides identity and authentication proxy functions to these brown-field endpoints enabling them to participate securely in the IIS.

9.2.6 User Authentication and Authorization

In addition to the endpoint credentials, user credentials can be used to identify the user of the endpoint uniquely. This is for authentication and authorization of the user for access to the network and resources at an endpoint. The combination of endpoint and user access control can be used to present a unique access profile to request access to resources at an endpoint.

9.2.7 Encryption in Communication

Data exchange between endpoints over communication channels must be encrypted with cryptographic keys of security strength and cipher suite meeting the security policy requirements.

9.3 Management and Monitoring Security

Management and monitoring security involves these areas:

- Identity management
- Provisioning and commissioning
- Security policy management

- Endpoint activation management
- Credential management
- Management console
- Situational awareness
- Remote update
- Management and monitoring resiliency

9.3.1 Identity Management

Hardware-backed identity is required to determine the identity of the endpoint authoritatively. Keys and certificates should be stored in a hardware-secured container (e.g. TPM). These hardware-secured containers generate asymmetric key pairs on chip and never expose private keys outside of the container. They perform crypto-operations on-chip with the private-keys. They export public keys to be distributed to other endpoints or likely signed into PKI certificates by a certificate authority. These containers also perform other crypto-based security operations such as attestation, signing, and sealing on-chip.

9.3.2 Provisioning and Commissioning

An endpoint must be provisioned and commissioned securely before it is allowed to participate in an IIS. In many cases, this process needs to be automated. This requires identity and credentials to be generated, distributed and installed in the endpoints and registered with the security management system.

Endpoint devices receive an identifier either at the time of hardware manufacturing, when the software/firmware image is provisioned to the device or on first boot after that provisioning. Optimally, devices receive an initial set of credentials during a “personalization” step in hardware manufacturing.

Alternatively, devices may generate a credential on first boot then register that credential with a trusted authority; or the device may receive credentials from a trusted authority. The device may register its credentials with additional authorities.

9.3.3 Security Policy Management

Remote policy management involves policy creation, assignment and distribution. Policies are defined on a security management system and communicated to endpoints via the secure agent.

The agent also ensures that the policy is available to the appropriate processes on the endpoint for configuration and enforcement. The agent may pull the policy, or the policy may be pushed to the agent. The agent may interpret the policy for the security processes on the endpoint.

There may be cases where cross-organizational communication is required; therefore, a common set of interfaces and protocols between the endpoints and the security management system, and even among the endpoints is required.

9.3.4 Endpoint Activation Management

Endpoint activation is the event where a new endpoint is recognized, authenticated and then permitted to exchange data with other endpoints in the system. An endpoint may need to activate with the security management system to be considered a legitimate endpoint in the IIS.

Endpoints should be able to activate dynamically and securely on multiple systems simultaneously, and deactivate as circumstances require.

9.3.5 Credential Management

The credential management lifecycle consists of:

- credential provisioning/enrollment/recognition
- additional credential generation (particularly for temporary credentials)
- credential update
- credential revocation/de-recognition

The credentials at an endpoint are managed remotely and securely by one or more security management systems via the secure agent. The endpoint's credentials can be provisioned, updated and revoked by a security management system, and this should be able to be done automatically for many endpoints at once.

By allowing credentials to be managed by multiple security management systems, the endpoint can be authenticated by multiple IIS networks at the same time. This also requires that one or more certificate authorities can be used on any one endpoint.

There are concerns over using the PKI system (i.e. Certificate Authority Model) [11] for managing large number of endpoints in IISs because of its constraints in scalability and reliability, and complexity in management. New schemes such as the DNS-based Authentication of Named Entities (DANE) [12] are emerging to address some of the scalability, reliability and management issues that associated with existing PKI systems.

9.3.6 Management Console

The management console allows human user interaction with the security management system for tasks such as creating and managing security policy and monitoring security activities and events across all the endpoints.

9.3.7 Situational Awareness

The security management system in an IIS must maintain awareness of situations in the network of endpoints including security events, attack attempts, currently deployed mechanisms to thwart such attempts, network health and general endpoint status.

In addition to single-event issues, patterns emerging a sequence of events can contribute to an understanding of the current environment of the IIS. For example, while a single failed login may not be interesting, a series of failed login attempts is significant and contributes to situational

awareness. A detection of a single port scan event is not as interesting as a series of events of port scans. By correlating information, the IIS is able to detect system-wide attacks, whereas less coupled systems would miss them.

9.3.8 Remote Update

Mechanism must be in place to automatically, securely and remotely update software/firmware via the security agent at the endpoint in response to identified vulnerabilities so they can be remedied quickly.

9.3.9 Management and Monitoring Resiliency

Security management needs to manage and monitor the IIS network especially when facing non-optimal network conditions such as when it is under attack, degraded or damaged. The communication mechanisms for management and monitoring must continue to function as well as possible, and be able to restore the network to full function with as little manual intervention as possible.

9.4 Data Distribution and Secure Storage

A core benefit of the Industrial Internet is improving the performance of its operations through data analysis. This process requires the collection of large amounts of data, its analysis in multiple locations and its storage for future use. All these operations must comply with privacy and policy regulations while still providing access to permitted data.

We must therefore consider:

- data security
- data centric policies
- data analysis and privacy
- IT systems and the cloud

9.4.1 Data Security

Sensitive data in an IIS must be protected during communication (see section 9.2.7) and storage. In the case of storage, sensitive data can be protected by employing data encryption at the field, record, file, directory, file system or storage device level. Access control to the sensitive data must be enforced based on authentication, authorization and access control policy.

9.4.2 Data Centric Policies

Data-centric policies include data security, privacy, integrity and ownership, and these policies apply through all stages, including data collection, distribution, processing and storage.

Different actors dictate these security requirements and policies: compliance mandates, such as the Sarbanes-Oxley act; industry standards, such as collecting automotive information; or national security enforcement, to preserve the secrecy of vital parameters in a nuclear reactor.

The information provided by these actors must be translated and implemented automatically to prevent errors.

9.4.3 Data Analysis and Privacy

To protect sensitive data or meet privacy requirements, data access policy may be provided to enforce fine-grained data access rules for example requiring certain data or fields to be removed, encrypted, obfuscated or redacted before distributing them to the data consumers for analysis or other uses.

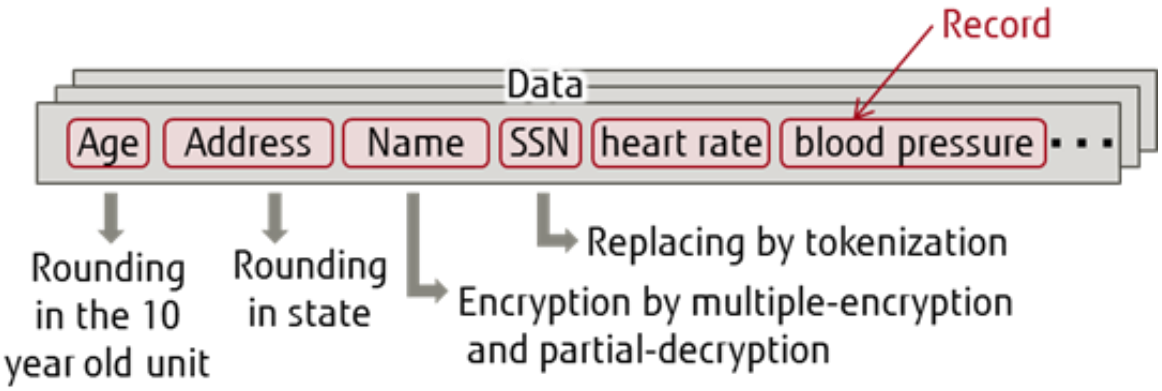


Figure 9-1 Data record encryption, obfuscation or redaction for privacy

The medical record shown in Figure 9-1 requires the enforcement of privacy if it is to be distributed and analyzed by third parties. This is achieved by obfuscation and encrypting at the record level.

9.4.4 IT Systems and the Cloud

In many cases the storage, distribution and analysis of the data needs to be performed at the IT, as opposed to OT environments. To protect the data, provenance information and privacy requirements should be attached to it so that ownership and the custody chain for the data records can be maintained. This applies during communication, when the data is processed by cloud systems (detached from the industrial system), when the analysis is performed by third parties, or moved to storage environments with different privacy requirements.

10 RESILIENCE

Resilience is more than just recovering quickly from pressure. To be resilient is to be able to take "bitter circumstance in stride" and still "get the job done." It might cost more or not be done as well had less (intentional or unintentional) adversity been present, but it will be done. Resilience is a superset of fault tolerance—and very much related to autonomic computing notions of self-healing, self-configuring, self-organizing and self-protecting.[28,29]

No other institutions are more involved directly in bitter or adversarial circumstances than the military. No other institutions have a greater dependence on resilience of its organizations and operations to survive and to succeed. Therefore, the current thinking of the military on resilience and the lessons they have learned in the past will inform us on how to better effect resilience within Industrial Internet Systems.

Military *Command and Control* (C2) has four main functions:

Mission planning is either *strategic* (what resources and programs need to be in place to handle expected major events in the long term), or *tactical* (what resource can be deployed in response to an event that is expected to occur, and what is the overall impact if the resource is diverted from other tasks). Generally, tactical planning is done by units like ships or battalions and by units at higher levels such as a carrier group or a division) for larger engagements. Tactical planning focuses on a set of objectives such as the enemy's next moves and planning for those that pose the deadliest threat. For IISs, this is the difference between how we *plan to fail* (create systems that are robust to expected kinds of failures and have sufficient resources to recover), and how we *enable recovery* (create subsystems that are aware of their own performance and can adjust how they operate, particularly in light of their peers.)

Situation awareness is knowing what needs to be known about a situation in relation to the tactical plan. *Situation understanding* is the larger contextual picture: why things are as they are.

Resource Management balances competing interests and concerns. It balances resources between threats that are current and imminent and those that are future and potential. For an IIS, it addresses questions such as how much computational resource to expend against detecting a security incursion vs. increasing replication of a service to assure that some copy will be able to compute a needed result in time.

Decide and Assess is the execution arm and the part that measures what has been done. If one sets out to neutralize an enemy emplacement, places the order and completes an air strike on

[28] Fault tolerance traditionally addresses system internal faults caused by a bug, hardware failure, or some kinds of internal error states. Resilience has a bigger scope in that it focuses on harmful elements external to the system, often introduced by an adversary, that tend to be unpredictable and unforeseen by the system's designers.

[29] Sometimes called self-optimizing (as part of self-CHOP), but optimization is often too strong a concept—we will accept suboptimal but improved capabilities as part of recovery.

the target, the next question is whether it has in fact been neutralized. This of course has implications for next steps, which could be further targeting or determining the target is too hardened and change the plan. For an IIS, there will be a constant balance of trying to decide if the current result is 'good enough' or if additional resources need to be expended to improve it (such as additional sensor readings to reduce uncertainty, confirmation dialogues to reduce risk of unintended action), or suggesting an in-service part be scrapped after a minor failure as it is more likely to trigger a major failure.

The military generally sorts responsibilities into administration, intelligence, operations, logistics, and communications, all under a common commanding officer. These groups may be replicated at several levels of command, so there may be division level intelligence as well as battalion level (or fleet vs. ship).

Military orders between levels of command have a specific syntax, including a number of sections that must be addressed, but the most important aspect for resilience is the notion of *commander's intent.*[30] This is used as part of the mission planning process where the commander sets up what the mission is about. The commander's intent is important to resilience when used operationally, as it enables an isolated unit at any level devoid of communication to still have a chance of knowing what to do even if the extant plan fails.

> Example: For instance, if all we have is the order to neutralize hill 73, then we must continue until hill 73 is neutralized or we run out of men. If we instead know that we are told this in the context of getting a clear shot at bunker AAA which is blocked by hill 73, then an unit which is (temporarily) under independent command can look at alternatives; perhaps all that is needed is to suppress the enemy or divert them on hill 73 to give the unit the opportunity to take the shot on AAA. So commander's intent pushes decision making down to the level that is best able (in terms of having the best information at the best time) to make the decision. Then when the mission is completed, the unit can attempt to reintegrate, report what they did while out of contact and allow the plans to be changed to address the new situation.

This approach handles uncertainty and failures of communication within 'decide and assess,' which is probably the most time-sensitive part of command-and-control. When executed well, there is a tremendous amount of flexibility and local negotiation possible even with 'disconnected' units to 'get the job done.'[31]

Establishing a global information grid (a military cloud) to provide information to the frontline commanders wherever they were has proved to have unintended consequences. For example,

[30] Of particular importance, "what are the *N* most likely things the adversary may do?" "What are the *M* most dangerous things they may do?" Planning generally has to address all of these contingencies, the longer one has for planning, the larger *N* and *M* can be—which means that the unit can be ready for a wider variety of unfolding circumstances.

[31] The exact amount depends on the service—marines have a lot more doctrinal flexibility than army units do for instance.

providing this information has offered a way to run battles directly from the command headquarters, overriding intermediate command. The military has moved in the other direction in recent years partly because such remote control of battle hurts resiliency. Similarly, moving industrial control to the "cloud" may also hurt resilience—not only does the network itself create an attack surface and a point of failure, but the information available at the scene will always be greater than that which can compressed into the pipe. For resilience, we should instead think of how to improve local decision-making through network services—without introducing new dependencies such as using the cloud for higher-level management and perhaps permission, but not low-level control.

Here then is a list of lessons we can learn from the military C2 structure and doctrine:

Expect to be disconnected from authority. Mechanisms must be in place to allow the mission to succeed, so some level of decision making on the edge is a requirement.

In an IIS, control elements for critical operations must not be dependent on network availability.

Good decisions are not made in a vacuum. Communicate commander's intent so that units in the field understand show their actions fit the bigger picture. The ability to alter plans locally provides a lot more flexibility and resiliency.

The implication for IISs is that local control elements must know more than just their own part of the plan. They must have a bigger picture of what they are responsible for that allows them to reconfigure their operation and maintain mission-level performance when under stress.

Peer-to-peer communication is more important than hierarchical communication. Changing plans and developing new tasks requires the disconnected units to engage in all parts of command-and-control jointly with their neighbors so they can jointly succeed within the constraints of the commander's intent. Once that intent (and an initial plan based on the strategically available resources) is communicated, little more needs to be said from higher chain of command until the mission is completed.

In IISs, this suggests that components must be autonomous, and able to act independently based on the plan and information from other independently operating components nearby.

Take advantage of the hierarchical network to optimize all parts of command-and-control. Do not use the connectivity, when available, to centralize decision making but distribute information to ensure that whole network becomes aware of changes to local plans so they can get an early start on changing too.

In IISs, this suggests that components must be aware of the behavior of other components.

Build a system that does not need the network to work—it only needs it to optimize. This is a given in the 'fog of war.'

In IISs, this partly follows from being able to run disconnected. However, some functions such as safety, should never be compromised just because of a network failure.

Delegate authority but not responsibility. Delegate sufficient authority to the agents to get the tasks done, but assume full responsibility to ensure the tasks are done right.

In IISs, mechanisms must be in place to find the right function for the job, and to validate that it did what it said it would do. It is important, for example, that orchestration elements not just suggest a process, but also suggest how the process can be validated and monitored.

Data without context can never become information. Pushing data around without context is not actionable. Context is hard to transmit as most context is the unwritten aspect of the circumstances and how the data was collected. Again this points to the criticality of 'man on the spot' processing—local to where the context actually is, enabling new techniques like learning to discover local phenomena that can help the particular instance of the problem being solved (rather than the much harder problem of inducing broad general rules).

Plans do not survive first contact with the enemy. Many plans are reworked every time we learn something new. To have a plan is not to have a set of instructions for every situation but to ensure training is in place to handle every conceivable contingency. Battles will always be dynamic, what endures is the ability to see patterns, react quickly and get inside the enemy's OODA loop. [32]

Control systems observe (by reading sensors), decide (using a comparator) and act (using actuators). There is no 'orient' function. In IISs, a resilient control architecture must be able to notice and discover when it is in an unexpected situation (i.e. orient itself), and then work to get itself back into a reasonable operating band, with the cooperation and collaboration of its peers.

Plan and Prepare. Current military thinking tries to go beyond 'react and respond.' This goes back to mission planning. We must both plan to fail and enable recovery. We must also capture the lessons learned to aid future recovery. Recovery of prior operational capacity can mean changes to tactics, techniques and procedure ('doctrine'). That is how the *organization* learns rather than just individuals.

In IISs, analytics can detect both imminent failure of a component, but also circumstances extant across a fleet of components when a component failed. This enables global learning.

Upward communication is often more important than downward. Mechanisms must be in place to communicate knowledge up the chain of command based on actual incidents, including what has been tried and failed.

In IISs, we have to ensure the kinds of properties driven down to the autonomous edge devices are policies rather than plans. That is, they include advice about how to make choices in difficult situations, rather than specific courses of action. The implication of this is that edge devices are

[32] "Observe, Orient, Decide, Act" Col. Boyd's brilliant insight into how fighter pilots operate. If one can execute one's OODA loop faster than the adversary can, they will usually win the battle because they can react to unfolding circumstances and in fact create unsettling circumstances faster than the adversary can. (This has also been applied to business management).

dependent on having appropriate computational elements for the complexity of the kinds of 'reactive plans' they are expected to implement .[33]

I was just following orders. This is never a legitimate excuse in the military.

In IISs, each component to the extent possible must make sure it does not violate local safety doctrine, even if it means ignoring direct orders from chain of command. [34] The 'unit on the spot' is on the spot both in the physical and legal sense.

There is a chain of command. Communications between units must be validated before they are trusted. Obey no commands even if they are issued by a higher-level officer unless they are established in that unit's chain of command. Trust is established before it is needed, and is necessarily hard to change with very formal procedures in place for transfer between commands.

Similarly IISs components should be inherently distrustful of 'changes of ownership,' 'new doctrine,' or even out-of-cycle updates; it is important that they are mechanisms not only for verifying they come from a trusted source, but that they make sense now.

[33] http://en.wikipedia.org/wiki/Reactive_planning

[34] Other doctrine, such as security, privacy, etc. may come into play as well, but safety is the most important as it is the hardest to recover from.

11 INTEGRABILITY, INTEROPERABILITY AND COMPOSABILITY

IISs are assembled from many components from multiple vendors and organizations within a vendor. To be assembled into large systems, these multifarious components must demonstrate:[35]

- *integrability*—the capability to communicate with each other based on compatible means of signaling and protocols,
- *interoperability*—the capability to exchange information with each other based on common conceptual models and interpretation of information in context and
- *composability*—the capability of a component to interact with any other component in a recombinant fashion to satisfy requirements based on the expectation of the behaviors of the interacting parties.

Composability relies on and adds to interoperability and integrability. Integrated components may have a capacity to communicate with each other but there is no guarantee that they can exchange information correctly, let alone whether they would have the intuitively expected behavior. Interoperable components can exchange information correctly but there is no guarantee their behavior is predictable. To look at this in another way: if an integrable component is replaced with another integrable component, the system may stop functioning; if an interoperable component is replaced, the system may behave quite differently; if a composable component is replaced with another with similar specifications, the system behaves in the same way.

> Example: Two people are integrable if both are able to speak and listen; interoperable if they speak the same language; and composable if they share similar culture and educational background and can collaborate for specific tasks.

> Consider a person as a potential pilot in an airplane cockpit. The person is considered integrable with the airplane cockpit if she fits well in the seat, can view the front horizon, see the instrument readings and indicators, and reach to all the controls—this includes any physically fitting adult. The same person is interoperable with the cockpit if she understands the meaning of the instruments and the intended outcome of the controls—this includes any physically fitting enthusiast about piloting. The same person is composable with the airplane cockpit if she is trained for the model of the airplane so that she understands the meaning of the instruments in context and the behavior of the airplane when she exercises control over it. One appropriately trained pilot can replace another in operating a plane.

IISs are large in scale and constructed from many types of components that are each evolving at an increasingly rapid pace. Components will change from being automatic (working by themselves with little or no direct human control) to autonomous (having the freedom to act

[35] This model is based on Page et al: Toward a Family of Maturity Models for the Simulation Interconnection Problem [29]

independently) and they will need to be able to self-assemble. Integrability and interoperability are inadequate to meet the needs of such systems. We also need composability.

Human interaction and communication using natural languages has proven to be a robust and dynamic method for composability. This is evident from observing how well two strangers can communicate on the spot, with minimal preparation for integration and interoperation, and how well they form groups collaborating to complete large tasks (or gossiping on Facebook).

Willingly or not, component designers have different models of reality with different constraints and assumptions. Lacking telepathy or shared memory between designers of different components means models, constraints and assumptions are not easily communicated or shared. It is therefore difficult to support the higher levels of communications and interactions beyond the level of integrability.

Facing this challenge, we impose a mental framework for undertaking integrability, interoperability, and composability as different levels of communication or interaction, similar to that taken in *Conceptual Interoperability* [13], an idea grounded in simulation theory—how to make different parts of a simulation system interoperate.[36]

We treat each communication in terms of passing messages containing symbols, similar in concept to natural language. (We will use natural language for examples.) The manner of communication supporting integrability in messages is well understood, so we focus on interoperability and composability.

At the base level, we have vocabulary and syntax—grammar—rules on legal word order and the relationship to meaning. For example, “car race” and “race car” both use the same words, and the meanings of the words themselves to do not change, but the thing that is denoted by the two examples, which differ only in their syntax, is different. Syntax can be fixed, as they are for the order of entries in a form, or variable as they are in a sentence where grammar is expressed as a set of rules.

The next level up is semantics—the meaning of the words themselves. In most languages, each word can have more than one sense, which is taken up by context. For example, “Safety” has one meaning in an industrial setting and another in football; we would not expect a safety engineer to worry about a defensive strategy in a game. Typically, in systems design we try to have each symbol have a unique context-free meaning, but this can also lead to excessive verbosity as well as inflexibility.

Database schemas tend to express the semantics of items in the structure of records, and these can be used as a kind of translation between different systems. For example, one database may talk about ‘names’ while another may segregate ‘last name’ and ‘first name’ but the meaning of

[36] The considerations here are similar to those studied under linguistics and the philosophy of language—to ask how it is possible for humans to communicate, for us to know someone else’s meaning, and how we can learn new things without actually experiencing them through language, by, for example reading a book.

'last name' is culturally dependent, so translating between the two may depend on the culture of the person being denoted. In one culture, the 'last name' is given; in another, it is that of the family.

Therefore, to exchange information with natural languages, we need to share some basic vocabulary (how a word is interpreted in different contexts) and syntax in which the words can be arranged into structures. This semantic understanding in communication is the basis of interoperability.

Next, we have pragmatics, which is the meaning of a sentence *in context.* This generally depends on the conceptual model of the world. Pragmatics allows us to understand the import of an utterance on a particular occasion. In speech act [14] theory (how speech can be treated as a form of action), the base level is the 'locution' or what was actually said—the grunts in the utterance along with the syntax and semantics that are defined by the language (as opposed to the use in this instance). Above that is the 'illocution' or what the speaker intended to say: the pragmatics.[37] The final layer is the 'perlocution,' which is the effect (intended or otherwise) on the other parties who hear the locution.

So one way to think about communication is that we want to specify the illocution (what we intend to say) such that the perlocution (the effect on the listener) can be what we intend. ('Can be' rather than 'will be' because the speaker does not control the mental state of the listener, and the listener may not use the locution for the speaker's intended purposes—let's recall the 'invented the internet' meme among others.)

Therefore, to exchange information with natural languages to achieve the intended effect, we need:

- to share a common or similar world knowledge (the understanding of the natural world and culture),
- the conceptual model, to have the ability of comprehend the meaning in its context (locution and illocution) and
- some general expectation of the other party in their understanding of the information (illocution) and their reaction (perlocution)—behavior.

This pragmatic understanding is the basis for composability.

With this understanding of the different ways components can be assembled to form larger systems and how they are related to the different levels of understanding in communications, we can point to the places in this document where each of these elements are addressed:

[37] Agent-based systems generally spill quite a bit of ink defining 'performative' speech acts (those that are performed by way of saying them, that is, when I say, "I promise you…," I have done something, namely made a promise!). See, e.g., [31]

Levels of Communication	*Levels of understanding in communication*[38]	*Refer to*
Integrability	*Technical*	*Connectivity* (Chapter 12)
Interoperability	*Syntax*—getting the format of the messages right *Semantics*—getting the meaning of the symbols in the messages right	*Connectivity* (Chapter 12) and *Data Management* (Chapter 13)
Composability	*Pragmatics/illocution*—interpreting what was intended by the sender	*Intelligent and Resilient Control* (Chapter 15), *Safety* (Chapter 8), and *Dynamic Composition and Automatic Integration* (Chapter 16)

Table 11-1 Mapping of Levels of Communication to topics in this document

820

[38] The three levels of understanding in communication discussed here loosely map to the corresponding levels of interoperability as defined in the *Conceptual Interoperability* [13], i.e. *technical, syntactic, semantic and pragmatic understanding* to *technical*, *syntactic, semantic* and *pragmatic interoperability,* respectively. We use the term understanding in place of interoperability here to avoid the confusion that might be otherwise caused by also using the term interoperability in the levels of communication (integrability, interoperability and composability). However, phrases such as *semantic interoperability* will be used in other sections of this document.

12 CONNECTIVITY

Ubiquitous connectivity is one of the key foundational technology advances that enable the Industrial Internet. The seven-layer *Open Systems Interconnect (OSI) Model* [15] and the four-layer *Internet Model* [16] do not adequately represent all Industrial Internet connectivity requirements. An Industrial Internet System is more complex and it is necessary to define a new connectivity functional layer model that addresses its distributed industrial sensors, controllers, devices, gateways and other systems.

12.1 ARCHITECTURAL ROLE

Connectivity provides the foundational capability among endpoints to facilitate component integration, interoperability and composability (see Chapter 11).

Technical interoperability is the ability to *exchange bits and bytes* using an information exchange infrastructure and an unambiguously defined underlying networks and protocols. *Syntactic interoperability* is the ability to exchange information in a common data format, with a common protocol to structure the data and an unambiguously defined format for the information exchange. Syntactic interoperability requires technical interoperability.

For IISs, connectivity comprises two functional layers:

- *Communication Transport* layer—provides the means of carrying information between endpoints. Its role is to provide technical interoperability between endpoints participating in an information exchange. This function corresponds to layers 1 (physical) through 4 (transport) of the OSI conceptual model or the bottom three layers of the Internet model (See Table 12-1).
- *Connectivity Framework* layer—facilitates how information is unambiguously structured and parsed by the endpoints. Its role is to provide the mechanisms to realize syntactic interoperability between endpoints. Familiar examples include data structures in programming languages and schemas for databases. This function spans layers 5 (session) through 7 (application) of the OSI conceptual model or the Application layer of the Internet Model (See Table 12-1).

The *data services framework* in the data management crosscutting function builds on the foundation provided by the connectivity framework to achieve syntactic interoperability between endpoints. That, in turn, provides the foundation for *semantic interoperability* required by the "Dynamic Composition and Automated Interoperability" as discussed in Chapter 16.

The table below summarizes the role and scope of the Connectivity functional layers

Scope of IIC Reference Architecture	*Correspondence to OSI Reference*	*Correspondence to Internet Model (RFC 1122)* [16]	*Correspondence to Levels of Conceptual Interoperability* [13]

Crosscutting Function	*Model (ISO/IEC 7498)* [15]		
Connectivity Framework Layer	7. Application	Application Layer	*Syntactic Interoperability Mechanism* introduces a common structure to exchange information. On this level, a common protocol to structure the data is used; the format of the information exchange is unambiguously defined.
	6. Presentation		
	5. Session		
Communication Transport Layer	4. Transport	Transport Layer	*Technical Interoperability*: provides the communication protocols for exchanging data between participating systems. On this level, a communication infrastructure is established allowing systems to *exchange bits and bytes*, and the underlying networks and protocols are unambiguously defined.
	3. Network	Internet Layer	
	2. Data Link	Link Layer	
	1. Physical		

Table 12-1 Role and scope of the connectivity functional layers

12.2 KEY SYSTEM CHARACTERISTICS

In IISs, the connectivity function supports several key characteristics:

Performance: High performance connectivity is expected in IISs. The spectrum of performance ranges from tight sub-millisecond control loops to supervisory control on a human scale. The performance characteristic is measured along two axes.

- *Latency and jitter*: The right answer delivered too late is often the wrong answer. Thus, latency must be within limits and low jitter is needed for predictable performance.
- *Throughput*: High throughput is needed when large volumes of information are exchanged over a short time.

High throughput and low latency are often competing requirements. Low latency and jitter are often more critical than throughput because IISs require short reaction times and tight coordination to maintain effective control over the real-world processes.

Scalability: Large numbers of things in the physical world and endpoints that exchange information about those things must be represented and managed. The connectivity function must support horizontal scaling as billions of things are added into the system.

Resilience: IISs operate continually in a real-world environment prone to failures. Endpoints operate in a dynamic fashion and may fail or become disconnected. Connectivity should support graceful degradation, including localizing the loss of information exchange to disconnected endpoints and restoring information exchange automatically when a broken connection is restored.

Connectivity security—architectural considerations: Information exchange among different actors within a system takes place over the two abstract layers documented in Table 12-1, both of which must consider when designing security solutions, including those of confidentiality, Integrity, availability, scalability, resilience, interoperability and performance.

Different information exchange patterns used in the connectivity framework, such as request-response or publish-subscribe patterns, have different security requirements.

Connectivity Security—building blocks: Information exchange security among connectivity endpoints relies on:

- explicit endpoint information exchange policies
- cryptographically strong mutual authentication between endpoints
- authorization mechanisms that enforce access control rules derived from the policy, and
- cryptographically backed mechanisms for ensuring confidentiality, integrity, and freshness of the exchanged information.

A security management mechanism manages the information exchange policies for connectivity endpoints. They define how to protect exchanged information. For example, they specify how to filter and route traffic, how to protect exchanged data and metadata (authenticate or encrypt-then-authenticate) and what access control rules should be used.

Longevity: IISs have long lifetimes, yet components, especially those in the communication transport layer, are often built into the hardware and hence are not easily replaceable. Where feasible, the connectivity software components should support incremental evolution including upgrades, addition and removal of components. It should also support incremental evolution of the information exchange solutions during the lifecycle of a system.

Integrability, interoperability and composability: IISs comprise components that are often systems in their own right. Connectivity must support the integrability, interoperability and composability of system components (see Chapter 11), isolation and encapsulation of information exchanges internal to a system component, and hierarchical organization of information exchanges. In dynamic systems, connectivity should also support discovery of system components and relevant information exchanges for system composition.

Operation: IISs generally require maintaining continuous operation (see Section 6.3). Hence, it must be possible to monitor, manage and dynamically replace elements of the Connectivity

function. Monitoring may include health, performance, and service level characteristics of the connectivity function; management may include configuring and administering the capabilities; dynamic replacement may include being able to replace hardware and or software while a system is operating.

12.3 KEY FUNCTIONAL CHARACTERISTICS OF THE CONNECTIVITY FRAMEWORK LAYER

The *connectivity framework* layer provides a logical information exchange service to the endpoints participating in an information exchange. It can observe and 'understand' the information exchanges, and use that knowledge to improve information delivery. It is a logical functional layer on top of the communication transport layer and should be agnostic to the technologies used to implement communication transports.

The key role of the *connectivity framework* is to provide *syntactic interoperability* among the endpoints. Information is structured in a common and unambiguous data format, independent of endpoint implementation, and decoupled from the hardware and programming platform. The connectivity framework addresses service discovery, information exchange patterns (such as peer-to-peer, client-server, publish-subscribe), data quality of service, and the programming model.

Discovery and permissions: To support more intelligent decisions, the discovery, authentication and access to services (including information exchanges) must be automated.

A connectivity framework should provide mechanisms to discover:

- The services available and their associated required or offered quality of service
- The data formats associated with the services
- The endpoints participating in an information exchange

The connectivity framework discovery mechanisms should provide a means to:

- Authenticate endpoints before allowing them to participate in an information exchange
- Authorize permissions (e.g. read, write) granted to the endpoints participating in an information exchange

Data exchange patterns: A connectivity framework should support the following information exchange patterns, typical of IIS.

- *Peer-to-peer* is a symmetric information exchange pattern between endpoints without any intermediary or broker. It can provide the lowest latency and jitter information exchange between endpoints.
- *Client-server* is an asymmetric information exchange where endpoints are classified into "client" or "server" roles. A "client" can initiate a service request that is fulfilled by endpoints in the "server" role. An endpoint may operate in both a client and a server role.

This pattern is sometimes also referred to as a “pull” or a “request-reply”, or a “request-response” style pattern.

- *Publish-subscribe* is an information exchange pattern where endpoints are classified into “publishers” or “subscribers”. A publisher can “publish” information on a well-known topic without regard for subscribers. A subscriber can “subscribe” to information from the well-known topic without regards for publishers. Thus, the topic acts as a channel that decouples the publishers form the subscribers. The result is loosely coupled endpoints that can be replaced independently on one another. An endpoint may operate in both a publisher and subscriber role. This pattern is sometimes also referred to as a “push” style pattern.

Data quality of service: *D*ifferent information exchanges have varying requirements on how the information is delivered. This non-functional aspect of the information exchange is referred to as the quality of service (QoS).

A connectivity framework should support many of the following information exchange QoS, categories.

Delivery refers to the delivery aspects of the information. These include

- At most once delivery: Variations include fire-and-forget or best efforts delivery, and “latest update” delivery. This is typical of state updates.
- At least once delivery, sometimes also referred to as reliable delivery. This is typical of events and notifications.
- Exactly once delivery: This is typical of job dispatching, and sometimes referred to as “once and only-once” delivery.

Timeliness refers to the ability of the connectivity framework to prioritize one type of information over another, and inform the endpoints when the delivered information is “late”.

Ordering refers to the ability of the connectivity framework to present in the information in the order it was produced, or received, and collate updates from different things in the system.

Durability refers to the ability of the connectivity framework to make information available to late joiners, expire stale information, and extend the lifecycle of the information beyond that of the source when so desired, and survive failures in the infrastructure.

Lifespan refers to the ability of the connectivity framework to expire stale information.

Fault Tolerance refers to the ability of the connectivity framework to ensure that redundant connectivity endpoints are properly managed, and appropriate failover mechanisms are in place when an endpoint or a connection is lost.

Security refers to the ability of the connectivity framework to ensure confidentiality, integrity, authenticity and non-repudiation of the information exchange, when so desired.

The connectivity function's performance and scalability limits would ultimately be determined by the communication transports layer. Therefore, the connectivity framework layer must introduce minimal overhead in providing the information exchange QoS and must have minimal impact on the overall performance and scalability.

Programming Model: IISs typically involve multiple components, developed by multiple parties over time, with a variety of programming languages.

A connectivity framework must provide an un-ambiguously documented programming model, in multiple programming languages, commonly used in the different parts of an IIS, such as C/C++, Java, C# and so on.

12.4 KEY FUNCTIONAL CHARACTERISTICS OF THE COMMUNICATION TRANSPORT LAYER

The communication transport layer transparently provides *technical interoperability* among the endpoints. The communication transport must address endpoint addressing; modes of communication; network topology, whether endpoints will be connected in a virtual circuit or connectionless, mechanisms to deal with congestion such as prioritization and segmentation, and with timing and synchronization between endpoints.

Network addressing: Each node in an IIS can house one or more components, each with one or more connectivity endpoints. Each node is identified by an address that can be locally unique and possibly globally unique. A node, and hence the endpoints residing on it, may be reachable via multiple addresses. The addressing scheme and associated infrastructure should be able to support billions of devices.

Communication modes: A communication transport can support one or more of the following communication modes:

- *Unicast* for on-to-one communication between two endpoints
- *Multicast* for one-to-many communication between endpoints
- *Broadcast* for one-to-all communication between endpoints, where "all" refers to all the endpoints present on the communication transport network at the time of transmission.

Topology: Communication transport may have one of the network topologies below:

- point-to-point
- hubs-and-spoke
- meshed
- hierarchical
- a combination of the above

It does not preclude others.

Communication transport gateways (see below) can be used to link multiple networks and communication topologies, to form more complex topologies.

Span: A communication transport network in the logical architecture view may span across multiple physical geographies. In the physical view, a logical communication transport network may span just the local area (LAN), or span across large geographic distances (WAN), or span somewhere in between (MAN).

Connectedness: For interactions between endpoints that require high degree of scalability, low latency and jitter, the design of connectivity function should give careful consideration to the choice of connection-oriented and connectionless mode of communication transport and its specific implementation. For example, UDP as a connectionless communication transport is usually chosen for low latency and jitter applications in typical network settings in comparison to TCP as its connection-oriented counterpart, largely due to the retransmission delay and other overheads in TCP. On the other hand, in a network with complex topology and high variation of traffic loads, connection-oriented communication transport may offer less jitter by providing a "virtual circuit" behavior that reduces the variation in routing path.

IISs call for new connection-oriented communication transports that do not suffer the drawbacks that are found in TCP today. When using a connectionless communication transport, the connectivity framework design needs to handle failures in the transport caused, for example, by loss of or out of order packets. Consequently, designing a connectivity framework based on the connection-oriented transport may preclude it from providing a connection-less information exchange.

Prioritization: IISs often need a way to ensure that critical information is delivered first, ahead of non-critical information. The communication transport function may provide the ability to prioritize some byte sequences over others in the information exchange between endpoints.

Network Segmentation: IISs often need a way to separate information from different functional domains over the same communication transport network. The communication transport function may provide the ability to segment a communication transport network, to isolate different functional domains and to isolate one set of information exchanges from another.

Timing & Synchronization: IISs often need a way to synchronize local endpoint clocks over a communication transport network. Many methods are in use today, including NTP or PTP based time synchronization and GPS clocks, and new approaches are in development. The communication transport function may provide ability to synchronize time across the network.

12.5 CONNECTIVITY GATEWAYS

IISs need to integrate multiple technologies, and in the system's lifetime, new connectivity technologies may need to be integrated as well. *Gateways* can be used to bridge one or more connectivity technologies. This gateway concept is shown in the figure below.

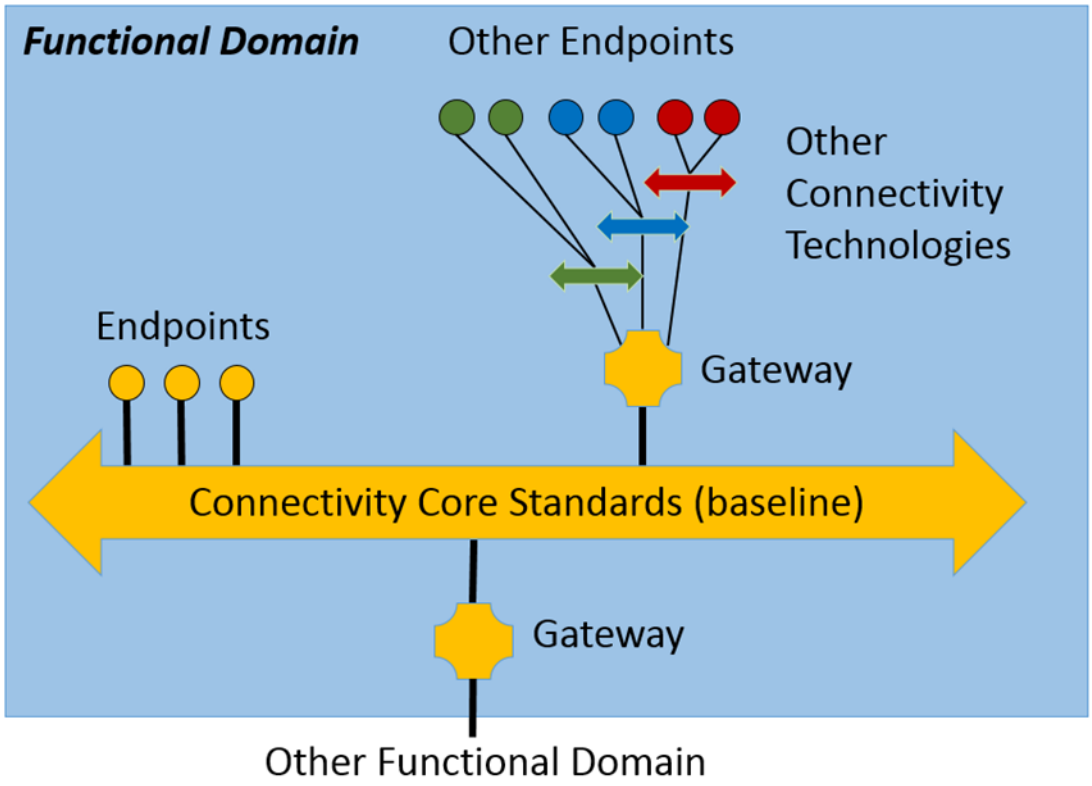


Figure 12-1 Connectivity Gateway Concept. A connectivity core standard technology (baseline) is one that can satisfy all of the connectivity requirements. Gateways provide two functions (1) integrate other connectivity technologies used within a functional domain, (2) interface with connectivity technologies in other functional domains.

To keep the reference architecture manageable, within a functional domain, a connectivity technology standard is chosen as the baseline, and referred to as the "connectivity core standard". Gateways are used to bridge other technologies and to the connectivity core standards used in other functional domains.

There are two types of commonly deployed connectivity gateways:

- *Communication transport gateways* expand the logical span of communications across transport networks. They are transparent to the payload and do not make any logical changes to the payload.
- *Connectivity framework gateways* expand the logical span of connectivity across connectivity framework technologies. They preserve the logical structure of data, but may change the representation (e.g. binary format vs. string format).

Connectivity gateways provide the architectural construct to incorporate new connectivity technologies that will become relevant in the future. They allow the possibility to pivot to a new baseline core standard that better satisfies the requirements, thus providing a stable foundation anchored in "best-of-breed" technologies available today, while allowing for future evolution.

13 DATA MANAGEMENT

Industrial Internet Systems Data Management consists of coordinated activities involving tasks and roles from the usage viewpoint and functional components from the functional viewpoint, specifically:

- Reduction and Analytics
- Publish and Subscribe
- Query
- Storage, Persistence and Retrieval
- Integration
- Description and Presence
- Data Framework
- Rights Management

13.1 REDUCTION AND ANALYTICS

Sensors and other systems in the IIS produce extremely large amounts of data.[39] Transmitting all this raw data over the networks to a central data center is often unnecessary and prohibitively expensive, but insights contained in the raw data must not be lost.

Reduction and analytics can manage data by either reducing the volume or velocity without losing the value or the information content. It is analogous to lossy data compression, as the original IIS data is irretrievable.

Analytics summarize raw data and produce an approximation of the truth that is suitable for downstream communication, processing and storage, while *data sampling* and filtering are examples of data reduction techniques devoid of analytics. Data reduction and analytics services suggest a migration of computing, networking and storage resources from enterprise to edge systems.

13.2 PUBLISH AND SUBSCRIBE

Publish and subscribe is suited for exchanging data updates between loosely coupled components and allows the publish-subscribe framework to optimize the communication path between publishers and subscribers based on their requirements.

Publish-subscribe contributes to IIS reliability, maintenance and resilience by the decoupling of publishing and subscribing components in both location (location transparency) and time (asynchronous delivery). This decreases the likelihood of fault-propagation and simplifies incremental updating and evolution. Interactions on the receiver side can be periodic (time-

[39] Data quality monitoring and sensor health monitoring are common applications for data reduction and analysis.

driven) or responsive (event-driven), depending on the needs of the user. Asynchronous transfers can also handle IIS component failures such as a network failure on the data path by delaying rather than cancelling an ongoing data transfer operation.

Publish-subscribe naturally supports the following kinds of IIS data exchange.

Streaming data: Data is continually or periodically updated at fixed rates ranging from KHz frequencies to multi-second periods, requiring low latencies and jitter with best-effort reliability. Components often check for the reception of data on a periodic basis. When the volume of streaming data is exceedingly large, publish-subscribe offers key advantages for the large numbers of interconnected systems.

Alarm and event: Data is issued when detection of specific IIS system conditions occurs. This spontaneous publication requires the IIS system to provide at a minimum guaranteed at-least-once delivery. IIS alarm and event data should be delivered with low latency and high priority, and pre-empt lower priority data where needed to ensure critical alarms are transmitted within acceptable delays. Parallel processing of a topic by multiple subscribers is essential when large number of spontaneous alarms or events may arrive at once.

Command and control: Control algorithms or people change the behavior or state of IIS components by generating command and control messages. They are typically time sensitive and require delivery by a deadline to allow target IIS component to react in a timely manner. Spontaneous publication is the norm and it requires guaranteed, low-latency and high-priority delivery, pre-empting lower priority data where needed, to minimize response time.

Configuration: Configuration or policy data are exchanged to enable IIS components to adjust their algorithms and behavior. These data change slowly, and typically have low latency and low priority requirements. Persistence is essential to support information requirements of newly joining subscribers even when the original moment of publication is missed. The data may also need to persist beyond the lifetime of the original publisher.

Publish-subscribe serves these purposes:

- Reliable data flow from the edge to a data consolidation and aggregation tier, for example to a cloud-based data services platform.
- Scalable handling of a large, evolving number of data sources such as devices, as well as of a large number of data consumers.
- End-user, application-level data consumption often requires a subscription model, from data consolidated on a platform to application components.
- Reliable control flow from applications or management services to devices: Control commands can be multi-cast in a way that allows devices to get these commands whenever they are ready.

13.3 QUERY

IISs employ two models to make queries. The *one-time query* model is associated with traditional databases and it fits well with the request-response pattern. The *continuous query* model is associated with data stream management systems and in-memory databases. It fits naturally with a publish-subscribe pattern and is better suited to handle infinite and rapidly changing data streams and support real-time analytics.

IISs use a combination of two styles to select a subset of data from a larger data set: *Save Data; Run Query* and its inverse, *Save Query; Run Data.* [40] Both query styles and models may apply at different levels in an IIS architecture, including at the device level.

Addressable devices may support direct queries (e.g. using WebSockets) in pull mode. Alternatively, device data may be pushed to a gateway configured with filtering rules that selects a subset of the device-generated data stream before acting as a data source to a higher-level data broker or data bus.

Query serves the following purposes:

- selection of a subset of device-generated data, either pulled by requests to addressable devices or pushed to a gateway running filters, and
- selective, usage-centric access to consolidated data by end-users and analytics, possibly in the cloud

13.4 STORAGE, PERSISTENCE AND RETRIEVAL

Storage, persistence and retrieval support many IIS functions:

Record supports defining and persisting a subset of IIS data in sequential order. Preserving time-stamping information supports ordering identification and reproduction between different data sets. Record data is typically not queried or reproduced as a time-series. Record is used for meeting record-keeping obligations, post-processing and analysis, replaying of system scenarios and related II use cases.

Replay supports retrieving a collection of IIS data previously recorded by replaying data-items in the order received. Replay supports creating simulation environments, regression tests and related II use cases.

Historian persists selected data for delayed time-series analysis.

[40] Essentially, the difference is in the active element – in the first case, a knowledge base contains the data and a query is run against that data such as SQL. In the latter case, the query is fixed in a stream processor and the data is run through it to filter out anything from the stream that does not fit. The latter can be a better fit when the data being generated is too voluminous to store, however, once filtered the 'filtered out' data is lost.

Big data solutions support voluminous IIS Control Domain system data.

Storage, persistence and retrieval serve these purposes:

- creation of audit records for future auditing (record and historian)
- support for simulations and various forms of testing (record and replay) and
- reliable storage and scalable archiving (Big Data).

13.5 INTEGRATION

Subsystems often have only partially compatible data models, so integration mechanisms between them are essential. An IIS Integration mechanism may use a wide variety of available integration mechanisms, including:

Syntactical transformation, which requires knowledge about the structure of the data and transformation rules in both IIS subsystems. Presence Discovery (see below) partially addresses this requirement. Semantic compatibility is also required and can be achieved via an Open Standards based metadata solution such as ISO 11179.

Domain transformation, which converts a data domain based on one protocol to a data domain based on another.

Integration serves these purposes:

- enabling integration across various middleware and application components and
- supporting functions analogous to conventional ETL, typically occurring in the first stages of data transfer, and preceding initial storage. [41]

13.6 DESCRIPTION AND PRESENCE

Description and presence enable components to discover the kinds, format, structure and metadata of available system data. Both use a variety of available mechanisms including query.

Presence allows components to discover which kinds of data are available using mechanisms such as query.

Metadata description enables components to obtain definitions of the structure of, and other information about, the present data.

Description and presence serve the following purposes:

- dynamic integration of new application components or middleware in a deployed IIS,
- addition of new types of devices with different data models and communication modes,
- design of a system management console applicable to various IISs and
- composition of IIS with different data models.

[41] Extract, transform, load, a process in database usage and especially in data warehousing.

13.7 DATA FRAMEWORK

Data frameworks provide users with insight into state and behavior of data exchange components by exposing diagnostic data, such as data update rates, number of discovered framework participants and detected message loss. Diagnostic data is similar to other data and therefore subject to all regular data mechanisms.

Monitoring and analyzing framework-associated information access is required such as data exposed by 'description and presence'. Framework mechanisms produce regular data that should be accessible via 'publish, subscribe and query'. The data provided enables creation of a dashboard for the data management framework that can track:

*Component presence discover*y identifies IIS component past or present framework participation.

Component activity monitoring monitors IIS component data such as update frequencies, throughput numbers, CPU load and memory usage.

Traffic monitoring monitors data flow characteristics such as data exchange volume, throughput, latencies, and jitters.

Data frameworks serve these purposes:

- design of a system management console applicable to various IIS and not specific to system technologies and components and
- ongoing deployed IIS testing and diagnostics

13.8 RIGHTS MANAGEMENT

IIS data rights management identifies and tracks data ownership. Rights management enables data owners to grant use rights, manage access based on the granted rights, and protect against unauthorized use. Rights management must be built on security functions but are clearly distinct from generic data protection and privacy.

Rights management serves these purposes:

- general data stewardship, in particular in case of consolidation and integration scenarios (between IISs, or IIS integration with enterprise systems),
- out-sourcing of data-related functions of an IIS to third parties such as cloud providers and
- support for regulatory and compliance requirements.

14 ANALYTICS AND ADVANCED DATA PROCESSING

Analytics and advanced data processing transform and analyze massive amounts of data from sensors to extract useful information that can deliver specific functions, give operators insightful information and recommendations, and enable real-time business and operational decisions, as shown in Figure 14-1. This section discusses the middle two boxes: advanced data processing in a system outcomes flow.

Figure 14-1 Advanced Data Processing in a System Outcomes flow

14.1 ADVANCED DATA PROCESSING

Advanced data processing enables a better understanding of system operational states and environments. It identifies and analyzes emerging information patterns to enable control system assessments under varied conditions in different environments. These assessments improve functionality and reduce operational cost and negative effects. For example, they enable utility companies to optimize electricity level output based on dynamic usage patterns that factor in weather, season, events, pricing, resource availability, cost and electricity generation asset availability; support vehicle and equipment fleet management; optimize smart home energy management and other unimagined capabilities. This is called *dynamic operations optimization*.

Advanced data processing can also optimize system missions. For example, metropolitan area real-time traffic pattern analysis combined with roadway conditions, roadway construction and maintenance, weather condition, time and day, seasons, accidents and other events can lead to vehicle control systems determining optimal routes to reduce travel time, congestion, pollution and energy consumption.

Industrial Internet advanced data processing consists of a number of components, with the two primary disciplines being *complex event processing* and *advanced analytics*. They share a common objective in discovering meaningful patterns from data.

Complex event processing receives streaming data from disparate sources to detect, abstract, filter and aggregate event-patterns, and finally to correlate and model them to detect event relationships, such as causality, membership, and timing characteristics. By identifying

meaningful events and inferring patterns that suggest large and more complex correlations, proper responses can be made to these events and circumstances.

Advanced analytics are used to discover and communicate meaningful patterns in data and to predict outcomes. Traditional business analytics are typically applied to business data to describe, predict and improve business performance.

Advanced data processing can reside in various components in an IIS across the breadth of its Functional Domains. It may, for example, be implemented in the information domain to analyze data aggregated from the control domain, other functional domains and external sources to providing analytic results covering the full scope of an end-to-end IIS. It may also be implemented in the control domain to analyze data to realize functionality in a local scope.

14.2 Advanced Data Processing Pattern and Properties

Advanced data processing can be implemented according to a variety of architectural patterns. One common architecture is a pipes and filters architecture [17] that allows pipelines to be created, in parallel and series, based on the application requirements, as depicted in Figure 14-2. The value in this composability is two-fold:

- the problem is divided into parts where each can be solved independently, to support different requirements and
- pipelines need not be co-located allowing the processing to be deployed as appropriate.

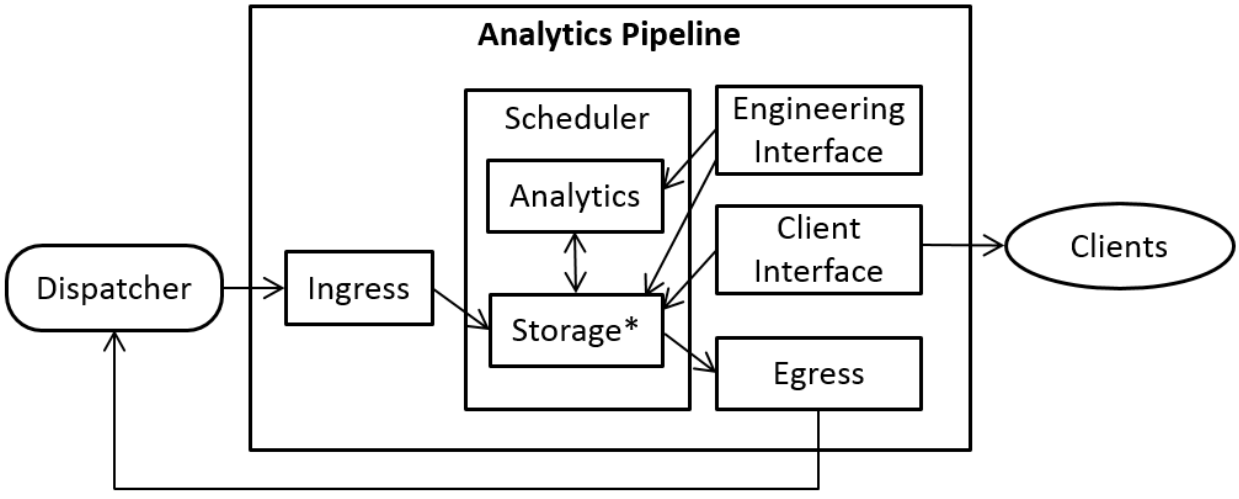


Figure 14-2 Analytics Pipeline Functionality [18]

Two components enable change over time. A *dispatcher* directs the input to the relevant analytics pipeline(s) that supply an environment in which data is analyzed and stored, and *clients* either pull data from a pipeline (as a query), or have data pushed to them (a notification).

A pipeline feeds the dispatcher with data, composed with other pipelines that tap into these data streams. Each analytics pipeline has a model set of features:

- *ingress*: connected to the dispatcher with the responsibility to transform the incoming data stream into something the storage component can handle,
- *storage*: temporary or long-term resource, such as a buffer, memory, disk, storage cluster, distributed file system that makes the data available to other components,
- *analytics*: configured with algorithmic functionality, which is designed using the engineering interface ,
- *outgress*: responsible for exposing the pipeline results,
- *Scheduler*: manages concurrent access to storage and schedules the analytics tasks,
- *engineering interface*: design time environment for specifying and experimenting with analytics algorithms, and
- *client interface*: provides access the data, including analytics results.

Each pipeline should have an ingress and egress specification defining compatibility criteria for the dispatcher and other pipelines, and a quality-of-service specification setting response-time expectations.

To match defined application needs, each pipeline type's properties must be understood.

Property	*Description*
Data Flexibility	New/unknown data types, without data model modification
Algorithm Flexibility	Variety of supporting libraries, query representations
Productivity	Ratio between effort and cost
Static Capacity	Store or configure permanently
Dynamic Capacity	Process or manage data simultaneously with concurrent tasks
Analytics Latency	Time delay experienced in core data processing
Round Trip Response	Time elapsed between request and response
Scalability	Ease, speed and affordability of changing performance qualities
Reliability	MTBF, operation when faults occur, degree of recovery, no data loss

Table 14-1 Analytics Pipeline Properties [18]

14.3 ADVANCED ANALYTICS

Advanced analytics are intended to spot opportunities in real-time, make fast and accurate predictions and act with confidence at the point of decision. These advanced analytics fall into four major categories:

- *descriptive analytics*: gain insight from historical data with reporting, scorecards, clustering and such.
- *predictive analytics*: identify expected behaviors or outcomes based on predictive modeling using statistical and machine learning techniques.
- *prescriptive analytics*: recommend decisions using optimization, simulation etc.

The results of analytics can be used to support human decisions through visual analytics to enhance human understanding and generate confidence in a decision

Advanced Analytics use a combination of the following execution approaches:

- *Automated*: automated data analysis, modeling and result application to automatic and continuous executions (including improving the analytics and modeling themselves in systems capable of learning).
- *Real-time*: near instantaneous analytic results with correct timing information enabling appropriate and timely actions.
- *Streaming*: continuous results flow of on-the-fly analysis of live streaming data in memory without storage persistence until after analysis completion.
- *Active*: active sharing of real-time pattern discoveries with other components enabling fast and accurate responses to discovered system changes.
- *Causal-oriented*: identifying complex, causal relationships in the data anchored in well-understood physical laws enabling better analysis using new approaches such as physical modeling and combining physical modeling with neural network deep-learning capabilities.
- *Distributed*: shared processing and results generation leveraging dynamic inter- and intra-functional domain relationships within and across II systems.

IIS place unique requirements on advanced analytics to include timing constraints, data volume constraints and safety criticality, as described below.

Timing constraints: If network latency cannot meet real-time requirements or the communication is not dependable, analytics must be performed in close proximity to the data sources and the systems that consume the results. For example, the results of image analytics of an autonomous vehicle must be made available to the model executor controlling the driving system within milliseconds.

Data volume constraints: Control systems that operate at high frequency and high speed, such an aircraft engine process large volume of time-series data of high resolution in both time and value. Network bandwidth constraints may make it infeasible to transport such large data

volumes across the network. *Dynamic Composition and Automated Interoperability*, enables dynamic binding of the data to the analytic capabilities at the edge, and bursts of high volume data to the appropriate analytic systems on demand. When low-level, high-resolution data are analyzed locally, summary data can find useful patterns across fleets of systems.

Safety criticality constraint: Safety-critical situations such as the presence of a child in an autonomous vehicle requires instantaneous, zero-fault-tolerant analytic. Failure to execute a safety override because the image analytics failed due to poor image quality is not an option.

14.4 IIS RA ALIGNMENT

Advanced data processing plays a role in each of the four IIS Reference Architecture viewpoints. In the Business Viewpoint, stakeholders have a vision focused on achieving benefits from their investments that requires advanced, real-time data processing activities for continuing measurement of business performance and ultimately the return-on-investment (ROI). These business-driven system objectives can guide the identification of advanced data processing capabilities required in an IIS.

Advanced data processing activities in the Usage Viewpoint must be identified to support the required advanced data processing capabilities and to guide the design, implementation, deployment, operations and evolution of these capabilities.

From an implementation viewpoint perspective, the implementation of analytics must take into account the timing, data volume and safety constraints and the consideration of resilience. Because of these constraints, analytics may be distributed across an IIS architecture, for example, in each of the edge, platform, and enterprise tiers of a *three-tier architecture* pattern.

15 INTELLIGENT AND RESILIENT CONTROL

15.1 MOTIVATION

The control model prevailing in industrial automation systems today tends to be localized in scope and reactive in response, such as the limited control loop feedback mechanisms implemented by proportional-integral-derivative (PID) controllers. When we embark on the task of creating a control, even a simple PID controller, we must consider a number of system engineering factors in respect to the conditions, constraints on operation, and the context. We then build a mechanism that takes some inputs to produce some outputs including engineering data values (voltages, temperatures etc.) and control signals to hardware, (opening or closing a breaker). Most of these factors are kept in the heads of the control (systems?) engineers and are thus a black box.

In IISs, we intend to perform distributed rather than local control, and to make predictions about how the world will change as a result of control. Moreover, this control must be 'intelligent and resilient' so that it can operate within a dynamic and unpredictable environment, using a distributed, collaborative capability to sense, make sense of, and affect the world and so achieve the goals of the specific entity that is acting (the agent). However, to reason and make predictions about how other controllers will work, particularly in unpredictable circumstances and dynamic environments, etc., we need transparency into that black box—the head of the control engineer. We need to understand how those choices were made, and what models of the world, assumptions about the world, understanding of the actions an actuator can take, and so on, prompted those decisions.

By employing models, either explicit or implicit, we can affect the desired intelligent control of the resources available to the agent and enable planning to bring the world to a state more acceptable to our interests. By making these modeling choices explicit, we improve the communication with the users of the system, and enable more advanced approaches to resiliency.

15.2 CONSIDERATIONS

The control engineer makes these choices based on a number of considerations, including:

Is the model of the world fully or partially observable? In a chess game, where the world is completely observable, the rules are known, so a legal move has a deterministic result. We may not be able to precisely determine the opponents move, we know it will be from a list of legal moves, and once it has been made we will know which move in the list has been taken. On the other hand, in a world that is only partially observable, such as many card games where cards are hidden, we are forced to infer world state based on the actions of the opponent. This affects our choices of the decision theory to use, the way we will model the world, the kinds of recovery strategies that are available to us after a fault, etc.

Are actions deterministic or probabilistic? When dealing with actions that can fail, such as a game of billiards, we are forced to consider not only the position of the balls we want to create if our

shot is successful (leading to the next shot we will take being easier), but also that if we are unsuccessful (leading to the next shot our opponent will take being harder). Many models presume sensing the world is 'free' in that it evolves continuously and outside of the influence of our decision making system, but taking a reading may require effort in that we have to move our sensor to observe something. That means that a world that might at one level of analysis appear to be fully observable is really only partially observable (because we don't have the processing capability to digest all of the sensor information we may be receiving) and also that what we think is the 'right' choice may not be, because we didn't see everything—the outcome of a particular action in a fully known state may be deterministic, but if we can't fully know the state—if some portion of it is uncertain, then we may want to say that the outcome of the action is uncertain (probabilistic) as well.

Can we plan all at once, or must we plan-to-plan? One way to deal with uncertainty is to defer planning until we have more knowledge—that is we can 'plan to plan' in that we create a partial plan that includes 'planning' as an action that will be taken under certain circumstances.

> Example: I may not know the train schedule to NY, so I have to plan to get the schedule before I buy the ticket. Since getting the schedule is insufficient to know which ticket to buy, I then have to plan to plan - decide now to postpone my decision as to which train to take, and thus the specifics of what I will do upon arrival. Or I may create a contingency plan, where I do all the work now iterating through every reasonable contingency, e.g., arrival before lunch, arrival after lunch, arrival after dinner, arrival after the subways are shut down, etc. It is a metacognitive action to decide which kind of plan I should create, but that decision can also be fixed at design time (i.e. the system will always generate a non-contingent plan, and if there are sufficient unknowns to prevent generation of such a plan, planning will fail with a list of unknowns to be satisfied).

Can we specify alternative methods to achieve goals? There's usually more than one way to skin a cat, and just as there are several possible routes to travel from point A to point B, by specifying multiple non-redundant ways to achieve a goal we build in a mechanism for the system to have backup strategies. For instance, while raising the house temperature is best achieved by turning on the furnace, a fireplace can be used, or the stove, or electrical heaters - all alternative methods that can be used if the furnace temporarily is non-operational.

Can we specify methods to reclaim or recover resources (particularly after casualty)? Such methods may be as simple as instructions for rebooting the network routers in case the network stops working, to strategies for reducing electrical usage to allow high current machinery to be started. Similar to the alternative methods, such an approach allows us to construct resilient systems - those that can reconstitute their capabilities after a failure. One can easily imagine resources having been assigned to processes (such as a database) to be in an unknown state during use, but should the process fail, we need to have a method to return those resources to some kind of known good state so they can be used again without having to wait for repair. In a database, this might be a rollback; in a nuclear power plant, this might be an orderly shutdown followed by a cold restart.

Do we want to learn and adapt to our inputs over time? Our main concern is application to dynamic situations - so the connection between the inputs and outputs may not be known perfectly at design time, and may change over time. For instance, if we are controlling both heat and humidity in a house, we may not know what kind of insulation the house has, and the system may not even know the time of the year it is (so if heating or cooling will be called for, if humidity will need to be added or subtracted, and at what rate). We will know that cooling will dehumidify. But presuming we have (unlike most residential systems) the ability to change the rate of cooling or adjust the balance between cooling and dehumidification, we may want to do so based on how the system has reacted in the past to such controls and furthermore be willing to readapt since seasonal changes will alter the response.

These considerations allow building appropriate models of and the relationships between the following that were previously mapped in the control engineer's head:

- the (relevant) world (context, environment, state of the universe, etc.), [42]
- action (both atomic and compound—e.g. a typical process that does something useful),
- communication (as a kind of action),
- intention (as of other agents),
- sensors,
- actuation and
- ethics (that is, those actions we must and must not do within a context)

15.3 FUNCTIONAL COMPONENTS

Figure 15-1 is a sketch of one way we might architect an intelligent control for a very dynamic environment. As a top-level decomposition, we have the following modules:

Deliberative and reactive planners: Long-horizon plans (typically called "deliberative") set goals. Short-horizon planners ("reactive") make satisficing real-time decisions (addressing resiliency) using a long-term plan to guide executing the plan in the current situation. Thus even when the long-term plan is obsolete, *i.e.* we cannot execute the plan as written, the reactive planner must be able to modify it on the fly to fit the actual circumstances. The planners handle most physical or logical planning constraints. For example, you cannot put down something you are not holding and you cannot use two positives to make a negative.[43]

[42] Modeling invariably involves abstracting away irrelevant detail. Deciding what is and what is not relevant is part of the job and risks of systems engineering.

[43] Yeah, yeah. [32]

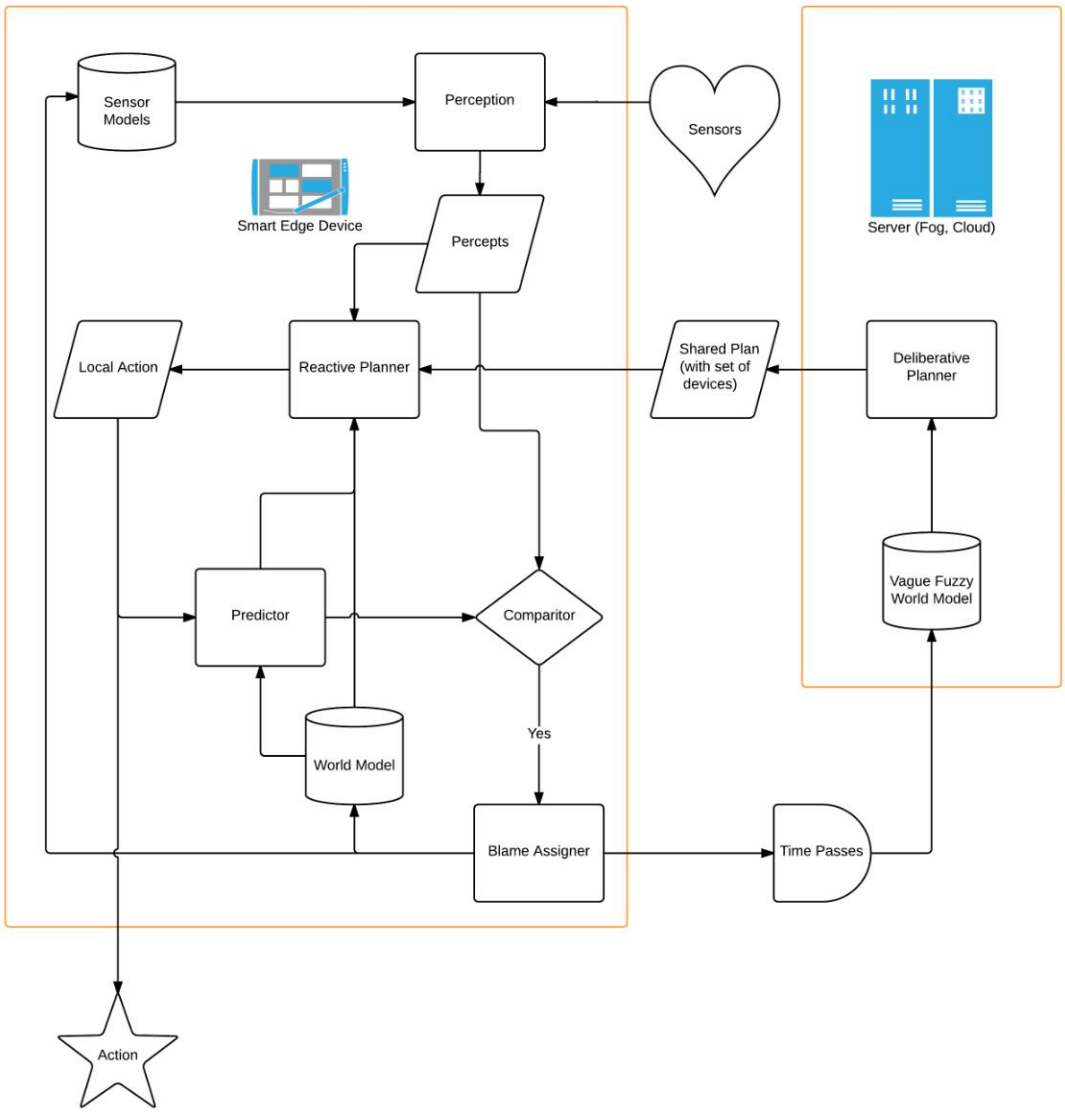


Figure 15-1 Intelligent Control Model

The deliberative planner needs more resources to establish a long-term plan, and is driven on a general understanding of the current world state, but is not the 'man on the scene.' That is the reactive planner, in the tight loop with the sensors and actuators making moment-to-moment decisions—guided by the long term plan, but able to override actions—the long term plan provides the moral equivalent of 'commander's guidance' while the reactive planner is the non-commissioned officer making tactical decisions under fire.

Because planning is a joint activity, and 'the plan' may not be visible to any particular agent or set of agents—since much of the plan is parochial and by the time it is observed by a (remote) agent, it will have been implemented or overtaken by events. A flexible mechanism for planning and implementing plans is therefore called for. One recommendation is a small but flexible reactive planner in the device itself and a deliberative (offline) planner provided as a service by a larger system or through remote service providers.

Predictor and precepts: Because plans may fail in an uncertain and dynamic world, we should expect that any particular agent's plan may fail, and that its models may make mistakes and fail. We therefore need *perception* to look at the relevant part of the world (driven by the plan) and for it to generate *percepts*—individual perceptions of interest to the agent.

The *Predictor* function informs the reactive planner what the likely outcome of planned actions will be, and through the *comparator* can look at what actually happened as the result of taking an action. If there is no difference, the operation continues but when there is a difference, we invoke the Blame Assigner (see below). A predictor function predicts what the state of the world will be at some point in the future, given an action or lack of action by that agent. (We regard not taking an action as an action: waiting). We can break the predictor down into two components: one that uses the models to chain out a possible future, and another that learns from experience. A learning function also requires additional parameters, including at least the inertia (how long to wait until making a prediction) or entropy rate (how likely is the pattern of the next input to be different from the past). (As an example, if we see the time series 0, 1, 1, 2, 3, 5, 8, 13: at what point do we react and say 'Fibonacci'—after the 2? The 3? The 13? At what point do we go back and make sure we are *still* seeing Fibonacci numbers? Every time? Every *xx* numbers? What if the pattern repeats? Stops and changes to some other pattern?).

Blame assigner: Given a world model, a predictor and a plan, we can predict the likelihood our plan will succeed, and then amend the plan to increase the likelihood of success (in probabilistic models). But we may be 'surprised' when our action does not have the intended effect. The 'blame assigner' makes the decision as to why things aren't as we thought they would be by considering a number of possible scenarios to determine the component at fault when something goes awry. It could be bad sensing where the control did generate the expected effect, but our decision to act was based on incorrect sensor data, or the world was not in the state we thought it was when we selected the action. It could be because of a bad model in which we have the wrong effects or likelihood of effects from our action, or because of the conditions under which the action is effective were incomplete, etc. It could also be because of faulty action in which e.g. we intended to press button A, but actually pressed button B, etc.

Ethical governor: An *ethical governor* (not in this diagram) might also be used to vet the action decisions to make sure that the agent does not perform any action it 'must not' perform (for, e.g., safety, security, or other reasons) and does perform any action it 'must' perform. It is a special deontic checker that validates that a course of action is within the scope of agreed upon ethics within appropriately negotiated, communicated, and represented community policies. The ethical governor must have the ability to override the agents' intentions. We give ethical rules special treatment because they do not tend to be an issue at the level of action selection but rather the overall plan pragmatics.[44]

[44] Ron Arkin: Governing Lethal Behavior: Embedding Ethics in a Hybrid Deliberative/Reactive Robot Architecture [22]

Security, safety and other models implemented by the ethical governor would be able to reject a request made by an operator or other agents. The autonomy implemented by an agent would always have 'final say' on accepting or rejecting a request.

Another task of the ethical governor is to determine when it is safe to dynamically change a device's behavior and/or performance through updating the device by deciding if the update is appropriate for the current circumstance and does not violate a safety or security constraint.

The agent should also store its own meta-information, so as to advertise its capabilities to the community, or to describe them in whole or in part in answer to a query about them.[45]

[45] Note that this does not require the agent to understand the meta-information, just be able to report it. The meta-information could then be parsed or interpreted by the receiver, through either pattern matching or general reasoning.

16 DYNAMIC COMPOSITION AND AUTOMATED INTEROPERABILITY

16.1 MOTIVATION

IISs require secure, safe and scalable composition of many diverse components from a variety of sources, often with different protocols, to deliver reliable end-to-end services. Given that distributed Industrial Internet applications are intended to be responsive to dynamic environments and that related technologies and standards are rapidly evolving, resilient IISs need to adapt flexibly to optimize services as environments change and to avoid disruptions as components are updated, upgraded and replaced. IISs present new use-cases in distributed computing that will drive advances in information technology architecture.

Service Orientation defines a logical framework for thinking about exchanging capabilities and data via distributed services and the ability to compose services into high-order applications and business processes. Though implementations vary and evolve, in general practice, service compositions are models of statically connected components. The relationships between components are defined in advance. At run-time, "orchestration engines" merely execute the composition. The method of composition design do not provide for any adaptation in response to change in the environment or of the components themselves. This tight coupling makes the compositions brittle; change requires manual redesign and generation of a new model or else the service is likely to break. The approach is not scalable in dynamic IIS environments. Advance testing of such service compositions can only validate the model in a controlled environment that is not representative of real-world operations.

IISs require a flexible method of composing services, so components can be dynamically integrated at run-time to enable adaptable services. Instead of static point-to-point connections, the demands of IISs need semantic interoperability to support many-to-many connections. In this approach, compositions indirectly link components using metadata references to a domain information model. Compositions represent a set of references to the information model, rather than a fixed set of named components. By separating the model from the implementation, semantic service compositions support metadata-driven policy-controlled orchestration that interprets references at run-time to dynamically discover components and their connections, as well as their transport and transformation details and integrate them on-demand. This loosely coupled approach automates interoperability and enables policy-based optimization for flexible and dynamic IISs that can adapt to change and re-configure network resources accordingly. Since semantically composed services have abstract contracts that are dynamically translated to concrete implementations, simulations can demonstrate the impact of real-time change and provide an execution trace that can be validated.

Many of these concepts were part of the original vision of Service Oriented Architecture (SOA).[46] In the early days of SOA, the software industry focused on static solutions that it could bring to

[46] Using OMG's Model Driven Architecture (MDA) to Integrate Web Services (2002) [20] and New OASIS Committee Organizes to Provide Semantic Foundation for SOA (2005) [21]

market quickly to satisfy pressing business demand for distributed computing. While dynamic features were perhaps ahead of their time, IISs now clearly require 'smart', adaptable composite applications. The demands of IIS drive change in several areas:

Situational awareness: Static solutions do not provide mechanisms for resolving possible incompatible assumptions about, for example, the operating environment, the deployment context, the interacting entities, and so on.

Workload diversity: Static compositions cannot change their stripes. In the real world, an end-to-end process may be a collaboration, choreography or orchestration with elements of various levels of autonomy. Many responses may be taking place simultaneously and would require flexible compositions in response to event.

Complex relationships: In IISs any element may have peer, parent and child relationships and thus varying roles and perspectives. This applies recursively to the participating elements, which may represent complex systems and involve a network of sub-systems. In a static solution, as tightly coupled complexity goes up, resilience goes down.

Dynamic relationships: In IISs relationships are constantly forming and un-forming, as in a 'friend-of-a-friend network'. The dynamics are changing constantly, creating on-the-fly activity and changing state among collaborating components. A self-forming composition may come into being when there is value in coordination, and then disengage or be uninvited from the group as events unfold.

In short, static, compartmentalized and centralized methods are not suited for dynamic optimization that concurrently satisfies multiple constraints for dynamic, diverse and distributed interactions expected in an IIS, and this prompts a shift from static models of integration and orchestration to "Dynamic Composition and Automated Interoperability."

16.2 CONSIDERATIONS

Practices and standards established today may enable or constrain future capabilities, so any approach must provide enduring value, and delay the need for expensive and time-consuming re-evaluation and re-design of the architecture.

By separating models from implementation, dynamic composition and automated interoperability supports a future-proofed IIS architecture that accommodates change by design (i.e. integration decisions are postponed to run-time to allow for optimization and adaptation).

We compose loosely coupled components using metadata references, contained in an information model that captures logical models of services based on abstract contracts. The abstract contracts de-couple system capability and control from the details of the implementation and infrastructure complexity. Consequently, the system capabilities can be declaratively described in the form of policy to bring about the desired service, function, conditions, and so on, without having to understand low-level implementation details.

Since the abstract contracts do not explicitly define the implementation, they need to be interpreted at run-time by an agent that acts as an intermediary responding to events. Every

event is an opportunity for an agent to add value, not just to perform a static script, but to discover patterns, perform functions, run analytics and otherwise 'reason'.

Agents resolve all references to find-and-bind the right resources just-in-time by performing all necessary connections and transformations, and to provide an optimal, context-enhanced response. Moreover, these agents monitor the responses of the employed services to ascertain if they are, in fact, responsive in accordance to the contract and otherwise replace the underperforming services to achieve the desired end-goal.

To maintain Quality of Service (QoS) and Quality of Experience (QoE), applications in an IIS may require visibility to the state of the interacting elements and network resources so it can observe and react to changes in connectivity and faults. This requires information be aggregated and translated into a common abstraction so system management applications can respond to events with any necessary re-configuration of the infrastructure and service implementation. The same high-level abstraction supports change at the application level, with the same QoS and QoE implications, to provide for sustainability of the end-to-end business capability.

In short, dynamic composition and automated interoperability must allow for real-time, data-driven, policy-controlled integration of services, and systematically late-bound at run-time, rather than integrating them in advance at design-time.

This approach for IISs, as a natural advancement in the evolution of system composition and service orchestration, has other significant benefits:[47]

Virtually centralized policy control: Security, business compliance and IT governance policies can be linked to abstract contracts addressing the historic challenge of managing consistent policy enforcement of system-wide concerns across diverse and distributed components.

Service adaptability: Since abstract contracts are not tightly coupled to any resources in advance, they can automatically evolve with updates and upgrades of underlying components without interrupting the operations.

DevOpspProductivity: Automating interoperability eliminates repetitive, error-prone and time-consuming integration work, accelerating service delivery and reducing cost.

16.3 Functional Components

The key functional components required to support the dynamic composition and automated integration are:

[47] For more details, please see "Semantic SOA makes Sense!" [19].

Integration contract management:[48] The integration contract management functional component provides capabilities for managing abstract contracts for automated interoperability, including:

- creation, query, update and deletion of abstract contracts for automated integration
- management of policies that apply to dynamic compositions.

Dynamic composition: The dynamic composition functional component provides run-time capabilities for composing system elements, in adherence to abstract contracts for automated integration, including:

- monitoring the status of the distributed system
- automated addition and removal of system components to a composition, in reaction to changes of system state
- creation and deletion of links between the interfaces of composed components.

[48] Traditional SOA composition patterns, such as orchestration, collaboration and choreography can be supported through static integration contracts, using existing languages such as WS-BPEL. Languages for specifying more dynamic automated integration patterns may require new standardization.

17 REFERENCES

1 Industrial Internet Consortium. [Internet]. 2015 [cited 2015 May 28]. Available from: https://workspace.iiconsortium.org/kws/groups/tech/download.php/818/Industrial%20Internet%20Vocabulary%201.0.pdf.

2 ISO/IEC 15026:2:2011, Systems and Software Engineering - Systems and Software Assurance - Part 2: Assurance Case. [Internet]. 2011 Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=52926.

3 Object Management Group Structured Assurance Case Metamodel (SACM). [Internet]. 2013 Available from: http://www.omg.org/spec/SACM/.

4 Open Group Dependability Through Assuredness™ (O-DA) Framework. [Internet]. 2013 Available from: HTTPS://WWW2.OPENGROUP.ORG/OGSYS/CATALOG/C13F.

5 ISO/IEC/IEEE 42010:2011 Systems and software engineering -- Architecture description. [Internet]. Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508.

6 Java Remote Method Invocation. [Internet]. Available from: http://en.wikipedia.org/wiki/Java_remote_method_invocation.

7 Web Service. [Internet]. Available from: https://en.wikipedia.org/wiki/Web_service.

8 Data Distribution Service (DDS). [Internet]. Available from: http://portals.omg,org/dds.

9 OPC Unified Architecture. [Internet]. Available from: http://www.opcua.us.

10 Modbus. [Internet]. Available from: http://en.wikipedia.org/wiki/Modbus.

11 Public key infrastructure. [Internet]. Available from: http://en.wikipedia.org/wiki/Public_key_infrastructure.

12 DNS-based Authentication of Named Entities. [Internet]. Available from: http://en.wikipedia.org/wiki/DNS-based_Authentication_of_Named_Entities.

13 Conceptual Interoperability. [Internet]. Available from: http://en.wikipedia.org/wiki/Conceptual_interoperability.

14 Wikipedia. Speech act. [Internet]. [cited 2015 May 28]. Available from: https://en.wikipedia.org/wiki/Speech_act.

15 OSI Model. [Internet]. Available from: https://en.wikipedia.org/wiki/OSI_model.

16 Internet Protocol Suite. [Internet]. Available from: https://en.wikipedia.org/wiki/Internet_protocol_suite.

17 Shaw M, Garlan D. Software Architecture: perspectives on an emerging disipline. ISBN 0131829572: Prentice Hall, Inc; 1996.

18 Harper KE, Zheng J, Jacobs SA, Dagnino A, Jansen A, Goldschmidt T, Marinakis A. Industrial Analytics Pipelines. IEEE BigDataService. 2015.

19 Semantic SOA Makes Sense. [Internet]. Available from: http://enterpriseweb.com/semantic-soa-makes-sense-2/.

20 Siegel J. Using OMG's Model Driven Architecture (MDA) to INtegrate Web Service. [Internet]. 2002 Available from: http://enterpriseweb.com/semantic-soa-makes-sense-2/.

21 New OASIS Committee Organizes to Provide Semantic Foundation for SOA. [Internet]. Available from: https://www.oasis-open.org/news/pr/new-oasis-committee-organizes-to-provide-semantic-foundation-for-soa.

22 Arkin RC. Governing Lethal Behavior: Embedding Ethics in a Hybrid Deliberative/Reactive Robot Architecture. [Internet]. Available from: http://www.cc.gatech.edu/ai/robot-lab/online-publications/formalizationv35.pdf.

23 ISO/IEC 15408-1:2009 Information technology -- Security techniques -- Evaluation criteria for IT security -- Part 1: Introduction and general model. [Internet]. Available from: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50341.

24 Business Motivation Model (BMM). [Internet]. Available from: http://www.omg.org/spec/BMM/.

25 Functional Safety. [Internet]. Available from: http://www.iec.ch/functionalsafety/explained/.

26 Microsoft. Security Development Lifecycle, SDL Process Guidance Version 5.2. [Internet]. 2012 Available from: http://www.microsoft.com/en-us/download/confirmation.aspx?id=29884.

27 OWASP. Application Threat Modeling. [Internet]. Available from: https://www.owasp.org/index.php/Application_Threat_Modeling.

28 Microsoft. Uncover Security Design Flaws Using The STRIDE Approach. [Internet]. Available from: https://msdn.microsoft.com/en-us/magazine/cc163519.aspx.

29 Page EH, Briggs R, Tufarolo JA. Toward a Family of Maturity Models for the Simulation Interconnection Problem. In: Proceedings of the Spring 2004 Simulation Interoperability Workshop, IEEE CS Press; 2004.

30 Stouffer K, Falco J, Scarfone K. NIST Special Publication 800-82: Guide to Industrial Control Systems (ICS) Security. National Institute of Standards and Technology; 2011.

31 FIPA/IEEE. FIPA Communicative Act Library Specification. [Internet]. 2002 [cited 2015 Apr 24]. Available from: http://www.fipa.org/specs/fipa00037/index.html.

32 Gumbel A. Professor Sidney Morgenbesser - Obituaries. [Internet]. 2004 [cited 2015 Apr 24]. Available from: http://www.independent.co.uk/news/obituaries/professor-sidney-morgenbesser-6164166.html.

33 Wikipedia. Trolley problem. [Internet]. [cited 2015 May 28]. Available from: http://en.wikipedia.org/wiki/Trolley_problem.