Eclipse Arrowhead Reference Model

Framework Description

Abstract

This document provides authoritative definitions for the most fundamental concepts of relevance to *Eclipse Arrowhead*, a framework designed to facilitate the effective creation of highly dynamic automation systems. It is meant to serve as foundation for other documents with relevance to the framework, providing a precise vocabulary untied to any specific practices or technologies. While presented in the form of a model, the document does not in and of itself build upon or endorse any particular modeling language.

Contents

1 Introduction

We expect the automation system of today to keep becoming more and more computerized, digitized and interconnected. By this we mean that more aspects of and surrounding machines will be handled by computers, more information will be made available to those computers and, finally, comparatively more computers will be given the opportunity to collect, communicate and act on that information. Manufacturing, transportation, energy distribution, medicine, recycling, and all other industrial sectors concerned with automation will be affected by this development. It will lead to increased efficiency and flexibility, as machines become able to perform more of the work traditionally assigned to humans. However, it will also lead to new magnitudes of complexity, not the least because of the renewed incentive to use more and more of these highly communicative machines.

The *Arrowhead framework* is designed to address this explosion of complexity. It provides a foundation for *service-oriented communication* [1] between automation systems, such that interoperability, security, safety, performance, and other major concerns can be addressed efficiently and effectively. It notably allows for system capabilities to be described, shared and exploited dynamically by communicating devices.

In this document, we, the Eclipse Arrowhead project, present an authoritative set of concept definitions, meant to serve as the fundamental language for discussions about and the modeling of Arrowhead-based designs. These definitions exist to help mitigate compatibility and consistency issues in software, tooling, models, documentation and all other things of relevance to the Arrowhead framework.

1.1 Primary Audiences

This document is being written and maintained for all who may require a precise and rigorous understanding of the Arrowhead framework, which we understand is likely to include the following groups:

- System designers, standardization engineers, developers, integrators and operators.
- Researchers.
- Technical managers and advanced users.

1.2 Scope

The concepts described in this document make up a so-called *reference model*, which we understand to be a set of definitions for technical concepts of fundamental importance to a specific problem domain. Such a document does not specify how its definitions should be used to design systems, either abstract or concrete. Reference models can be used as vocabularies for defining reference architectures, which in turn can be used to derive concrete architectures and, finally, software implementations, as illustrated in Figure 1.

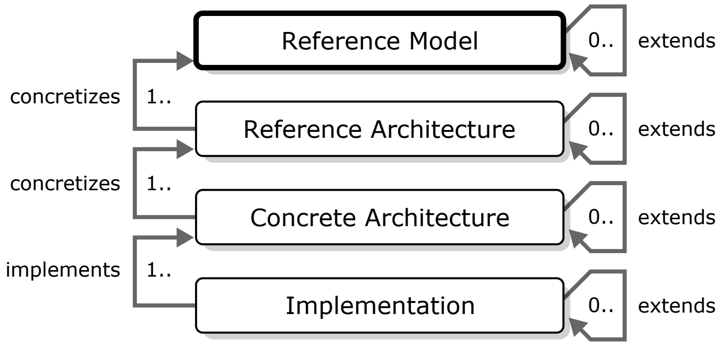

Figure 1: The steps from reference model to implementation, going from highly abstract to entirely concrete. Reference models define fundamental concepts, reference architectures add abstract and/or concrete design constraints rooted in real-world observations or experiences, concrete architectures makes those design constraints practically realizable as one or more artifacts, while implementations represent their actual realization.

1.3 Notational Conventions

The following conventions regarding diagrams, references and requirements are adhered to throughout this document. All three of them were selected by virtue of being deemed unsurprising to our primary audiences.

1.3.1 Diagrams

A box with a name inside it denotes a named entity or stakeholder. A named arrow between boxes denotes the relationship implied by the name. If a named arrow has an associated positive integer or range, which we refer to as a *quantifier*, the relation is to be considered as extending to the number of distinct entities indicated by that integer or range. A range is denoted by $x..y$, where x and y are positive integers and $x < y$. Omitting y when using the range notation (e.g. "$1..$") means that the range is infinite from x. If two or more arrows are combined such that their source or target end is shared, a difference is made if a quantifier is closest to a shared or non-shared arrow part. If it is closest to a shared part, the quantity must be understood to apply to every arrow part of the combination. If it is closest to a non-shared part, the quantifier must be understood to only apply to that arrow. A box with a dotted border represents a group. The entities explicitly placed within the box may or may not represent all entities that belong to that group. See Figure 1 for an example of this notation being used.

Note that this document does *not* define an Arrowhead profile for SysML [2], or any other modeling language. As we cover later in Section 4, however, we do expect all models based on this document not to contradict any of its definitions.

1.3.2 References

Square brackets around numbers (e.g. [3]) are references to the reference list in Section 6. The number within the brackets of any given reference corresponds to the entry with the same number in the reference list.

References within this document are hyperlinked, which means that those reading it electronically can click the references and immediately be taken to their targets. Special treatment is given to references targeting Section 5, the Glossary. These are displayed as regular text rendered with blue color.

1.3.3 Requirements

Use of the words **must**, **must not**, **required**, **should**, **should not**, **recommended**, **may**, and **optional** are to be interpreted as follows when used in this document: **must** and **required** denote absolute requirements that must be adhered to for a described entity to be considered as compliant to this reference model; **must not** denotes an absolute prohibition; **should**, **should not** and **recommended** denote recommendations that should be deviated from only if special circumstances make it relevant; and, finally, **may** and **optional** denote something being truly optional. These word definitions are derived from and are meant to capture what is outlined in RFC 2119 [4].

1.4 Relationships to Other Documents

When this reference model was produced, care was taken to reuse or build upon the concepts presented in the following works:

1. **Reference Architecture Model Industrie 4.0** (RAMI4.0) [5], which outlines an ontological and architectural view of *Industry 4.0*. The document may be seen as a predecessor to, or major influence on, the conceptual aspects of the Arrowhead framework. In particular, the document describes how to model and design communicating industrial systems such that key Industry 4.0 characteristics can be facilitated, such as high degrees of dynamicity and interoperability. However, as RAMI4.0 is a reference *architecture* rather than a reference *model*, we have only been concerned with what concepts it defines and what problems it frames. This delimitation excludes its "architectural layers", "life-cycle & value-stream" phases and "hierarchical levels", as well as the abstract design of its "asset administrative shell". These excluded aspects are neither condemned nor endorsed by this document. They are simply outside its scope.
2. **Reference Model for Service Oriented Architecture** (SOA-RM) [1], which provides a standardized definition of Service-Oriented Architecture (SOA). As communication between systems of the Arrowhead framework is understood to follow this paradigm, it becomes particularly relevant to consider.

3. **IoT Automation: Arrowhead Framework** (IoTA:AF) [3], which significantly includes an overview of the *local automation cloud* concept in its second chapter, as well as the *Arrowhead framework architecture* in its third chapter. The book most significantly represents the state of the Arrowhead framework up until it was written. Even though the framework has evolved since then, it still represents the most comprehensive view of the framework. While the strictly architectural aspects of IoTA:AF are outside the scope of this document, the two mentioned chapters contain several definition with a high degree of relevance.

Only conformity with IoTA:AF is observed strictly, which means that concept definitions presented here may diverge from those of the other two works. All significant terminology differences are noted in the glossary of Section 5, which briefly defines all concepts of relevance to this document.

1.5 Section Overview

The remaining sections of this document are organized as follows:

Section 1 This section.

Section 2 An informal overview of Arrowhead, serving both to provide a workable summary of the framework and to prepare readers for better understanding Section 3.

Section 3 The formal and normative description of Arrowhead. Each of its subsections is concerned with one major Arrowhead concept, ranging from entities to systems-of-local-clouds.

Section 4 A brief list of requirements, meant to help determine whether or not a given model or document is conforming to this reference model.

Section 5 Lists all significant terms and abbreviations presented in this document in alphabetical order.

Section 6 Lists references to publications referred to in this document.

Section 7 Records the history of changes made to this document.

2 The Arrowhead Framework

The Arrowhead framework is two things, as depicted in Figure 2. Firstly, it is a framework of assumptions, concepts, values and practices that frame the problem domain of *dynamic device coordination in the context of automation*. Secondly, it is a set of software specifications, implementations and other artifacts meant to help address that problem domain. In this section, we provide an overview of the primary *concepts* of the Arrowhead framework. While *assumptions* and *values* may be possible derive from this overview, neither of these, nor the other parts of the framework, are considered directly.

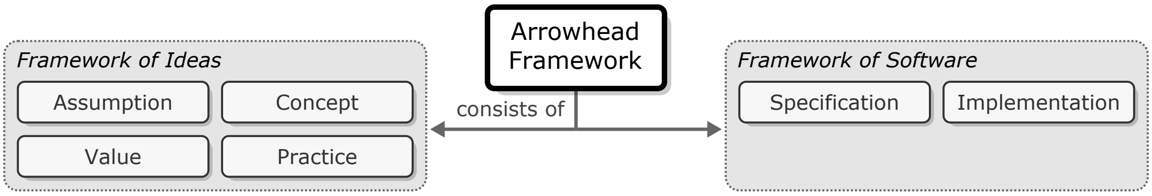

Figure 2: The two main categories of concerns of the arrowhead framework: ideas and software.

2.1 Stakeholders and Artifacts

There are two kinds of citizens in the world of Arrowhead, (1) stakeholders and (2) artifacts, as depicted in Figure 3. The former denotes a person or organization engaged in an Arrowhead enterprise, while the latter is any thing or object, tangible or intangible, that could be relevant to consider as part of such an enterprise. Stakeholders own, design, develop, operate, and use artifacts, among many other possible activities. It is their business needs and ambitions that dictate what and how Arrowhead artifacts will be employed.

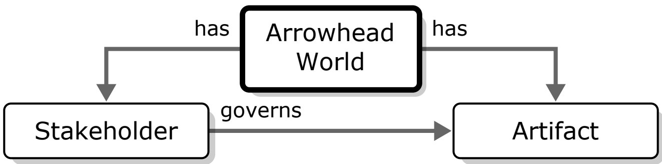

Figure 3: The two kinds of citizens of the Arrowhead world: stakeholders and artifacts.

2.2 Devices, Systems and Services

The most essential types of Arrowhead artifacts are (1) devices, (2) systems and (3) services, as shown in Figure 4. *Devices* constitute the physical machines that make up the industrial complexes, vehicles, tools, and other things that could be made operational via Arrowhead. Each device hosts one or more *systems*, which are communicating software instances that make their devices work toward whatever goals are set for them. Finally, a *service* is a set of related functions that a system can make its device perform for a person or another system. Services can be concerned with manufacturing, repair, analysis, or any other physical or virtual activity. The service is the primary means whereby systems coordinate to fulfill their assignments.

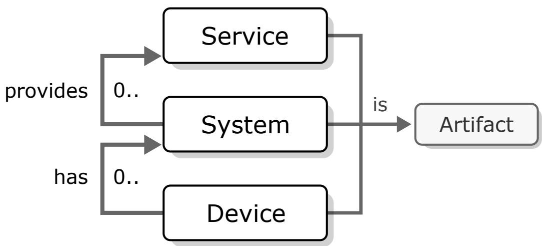

Figure 4: Hardware devices have, or *host*, software systems, which provide services. Each of these is an artifact.

2.3 Service Provision and Consumption

Communication between systems is formulated in terms of the provision and consumption of services, as depicted in Figure 5. When a system *provides* a service, it makes it available to other systems through service interfaces. Other systems can *consume* its services by sending messages to the interfaces of those services. When a service interface receives a message, it routes it to the specific function it targets. That functions must ensure that the message conforms to its protocol and policies before 4concretely handling it. Function protocols establish what messages must contain and when they may be sent in relation to other messages, while policies establish what other conditions must be satisfied for the function invocation to be permitted. In other words, a message satisfying a function protocol guarantees that the message is understood by its receiving function, while a message satisfying certain policies guarantees that the activity it would trigger is occurring under desirable conditions. A policy may require that certain quality-of-service guarantees can be met, or that the sender is properly authorized, among many other possible examples.

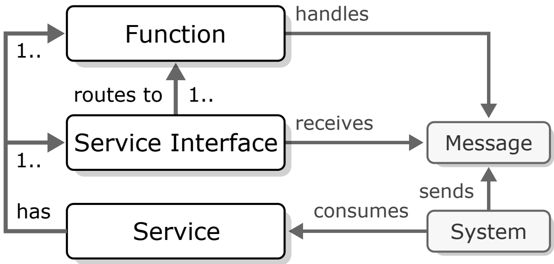

Figure 5: A system consumes a service by sending a message to its interface, which routes it to a function.

2.4 System Composition

When certain systems consume each other's services, those systems form a system-of-systems, as illustrated in Figure 6. That system-of-systems becomes able to perform activities none of its constituent subsystems could perform on its own. While there are many ways it could be relevant to structure systems in relation to each other, two are of particular significance in the context of the Arrowhead framework. These are (1) the local cloud, and (2) the system-of-local-clouds. A *local cloud* is a pool of resources, managed by systems, where at least one pooled resource is local. A *local resource* derives its value from its physical properties. Furthermore, all local clouds have at least one *local boundary*, which is a physical property that distinguishes artifacts inside the cloud from those outside it. A local cloud may also have virtual resources and virtual boundaries. Raw materials, drones, and other systems are a few examples of resources. Boundaries may be formed by physical location, electronic certificate issuance, among many other possible examples. A system-of-local-clouds emerges when distinct local clouds consume each other's services to perform new activities neither of the local clouds could perform on its own.

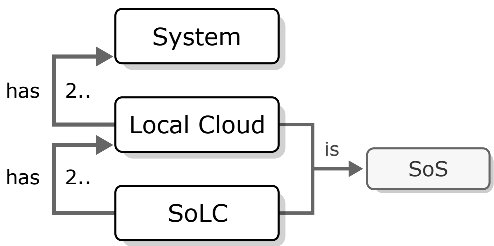

Figure 6: The two primary kinds of systems-of-systems (SoS): the local cloud and the system-of-local-clouds (SoLC).

3 The Reference Model

This section is what constitutes the reference model of this document. Each of its subsections describes a primary Arrowhead concept, which are as follows:

3.1	**Stakeholder**	A person or organization with stake in an entity or undertaking.
3.2	**Entity**	An artifact that can be distinguished from all other artifacts.
3.3	**Device**	A physical entity with the significant capability of being able to host systems.
3.4	**System**	A software instance able to exercise the capabilities of its hosting device.
3.5	**Service**	A set of functions provided by a system for other systems to consume.
3.6	**System-of-Systems**	A set of systems together facilitating new capabilities.
3.6.1	**Local Cloud**	A cloud with a local boundary and local resources.
3.6.2	**System-of-Local-Clouds**	A set of local clouds that jointly facilitate new capabilities.
3.7	**Network**	A set of devices whose systems can communicate.
3.8	**Interface**	A boundary that can be crossed by messages of certain protocols.
3.9	**Policy**	A set of constraints that must be satisfied for an activity to be permitted.
3.10	**Protocol**	A description of how messages may be sent between entities.

3.1 Stakeholder

A stakeholder is a person or organization with stake in an entity or undertaking with relevance to the Arrowhead framework. In this context, we understand *stake* to refer to any type of engagement or commitment. The concept is illustrated in Figure 7, which also lists five reasons why a given person or organization could be considered to be a stakeholder. We refer to these reasons as *roles*.

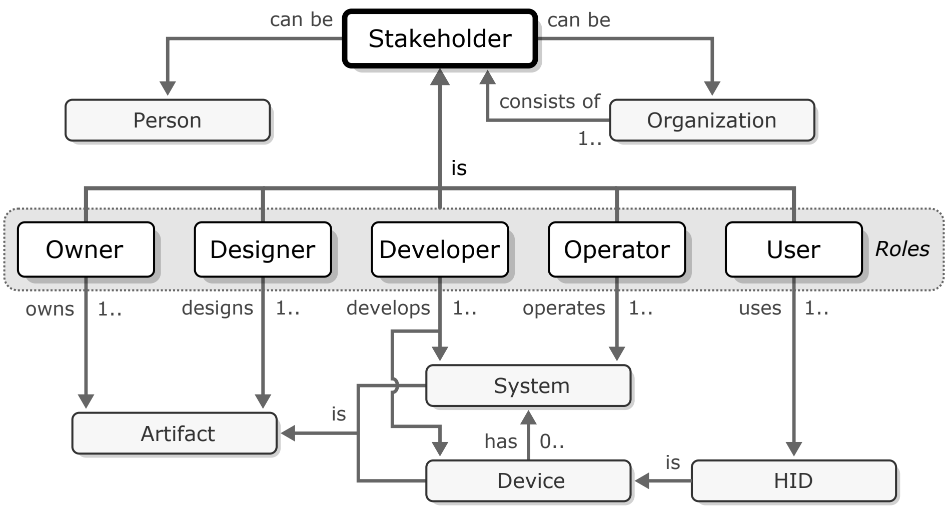

Figure 7: The stakeholder as either a person or organization, where each such stakeholder takes on one ore more distinct roles. The depicted roles is not an exhaustive list. HID is an abbreviation for Human Interface Device.

The roles occupied by a given stakeholder dictates what entities that person or organization will interact with, as well as the nature of that interaction. In Figure 7, (1) owner, (2) designer, (3) developer, (4) operator and (5) user are named explicitly, but more roles are likely to be relevant, such as (6) integrator and (7) standardization engineer, (8) researcher, and (9) manager. The listed nine names should be used rather than any synonyms when referring to these particular roles. Please refer to the glossary for their definitions.

3.2 Entity

An entity is an artifact that can be distinguished from all other artifacts. We use the word *artifact* to refer to any object or thing, physical or intangible. As depicted in Figure 8, this means that an entity always has an identity.

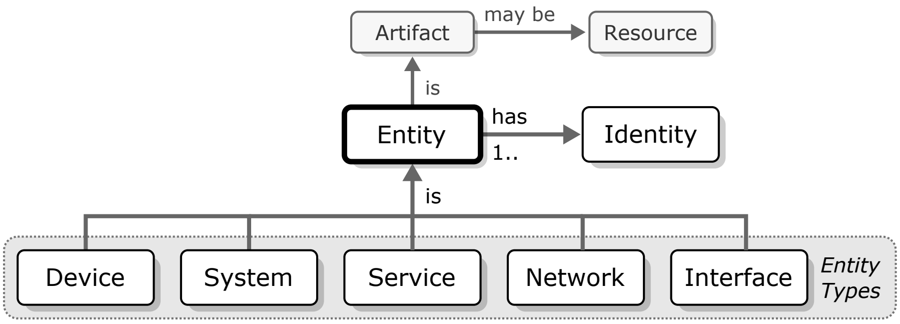

Figure 8: The entity as an artifact with an identity. An entity or artifact may or may not be considered to be a resource, in which case it is deemed to be of value to a stakeholder. The group of entity types is not exhaustive. Other examples are local clouds, certain data, policies, protocols and profiles.

Note that having an identity is not the same as being associated with an identifier, which is a name, number or other value referring to the entity in question. It is enough that such an identifier could be produced for an artifact to count as an entity. That being said, certain identification requirements, perhaps related to security, performance or discoverability, may make it practically unfeasible not to use identifiers.

3.3 Device

A device is a physical entity with the significant capability of being able to host at least one system, each of which may be given the opportunity to exercise the capabilities of that device. Examples of capabilities include moving robotic arms, reading from sensors, running software procedures, and sending messages. Every device consists of hardware components. While there are no limits to what components can make up a device, each device must always have (1) memory, (2) compute and (3) interfacing components, as shown in Figure 9.

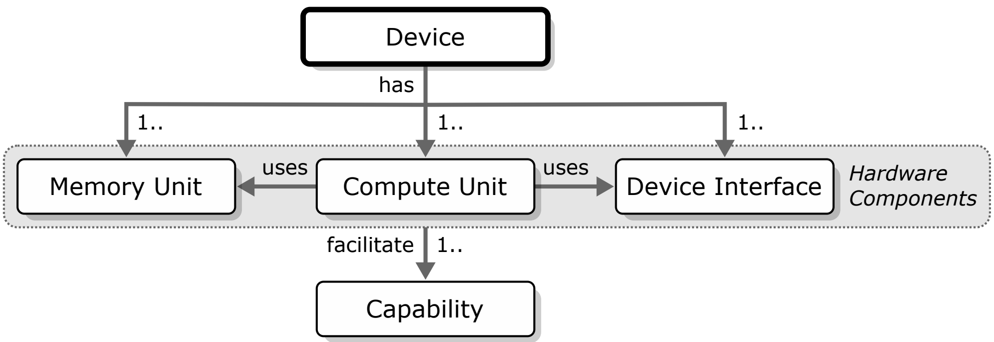

Figure 9: The device as an entity with hardware components, together facilitating one or more capabilities. Devices must be able to host systems, even if not made explicit by this figure. The group of hardware components is not exhaustive. Other examples of such components could be sensors, actuators, compute accelerators, or batteries.

Devices must be able to host systems, or they must be considered as hardware components. While it may seem unintuitive to consider certain machines as components, such as large pumping complexes or vehicles with only manual controls, the Arrowhead framework is meant to facilitate automation through the use of interconnected devices with compute capabilities. If a machine cannot run software, making it able to host systems, that capability must be added before it can play a meaningful role in an Arrowhead context. Consequently, machines without system hosting capabilities must either be considered as components or not be considered from the perspective of Arrowhead at all.

3.4 System

A system is an identifiable software instance that is able to exercise the capabilities of its hosting device. As shown in Figure 10, systems consists of software components. Some significant such components are (1) states, (2) procedures and (3) system interfaces, which are facilitated by the (1) memory, (2) compute and (3) interfacing components of a device. A system with these components should be able to consume and/or provide services, or it must be referred to as an *isolated system*.

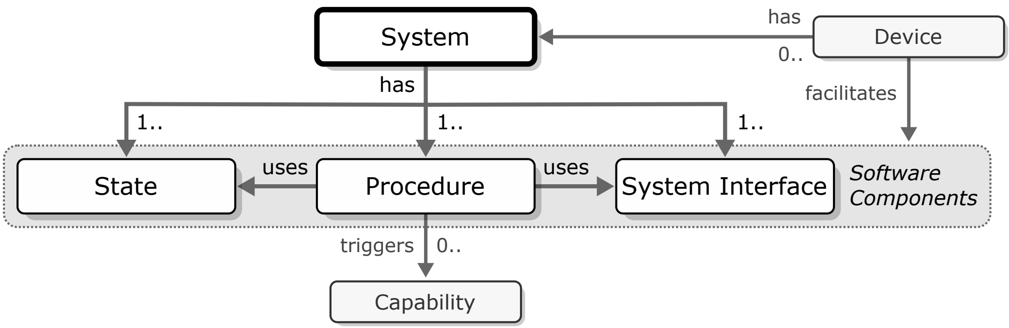

Figure 10: The system as a collection of related software components, able to trigger zero or more capabilities of its hosting device. The group of software components is not exhaustive. Other examples of such components could be operating systems, file systems, software libraries, programming language runtimes, databases or virtual machines.

This rather open-ended definition of “system” makes it possible for such to be realized in many ways. A system may or may not run in its own operating system process, use a certain virtual machine, and so on.

3.5 Service

A service is an identifiable set of functions provided by a system, allowing for other systems to trigger its capabilities by sending messages to, or *invoking*, those functions. The act of sending a message to a certain function is referred to as *consuming* its service. As depicted in Figure 11, every service consists of at least three software components. These are (1) states, (2) functions and (3) service interfaces.

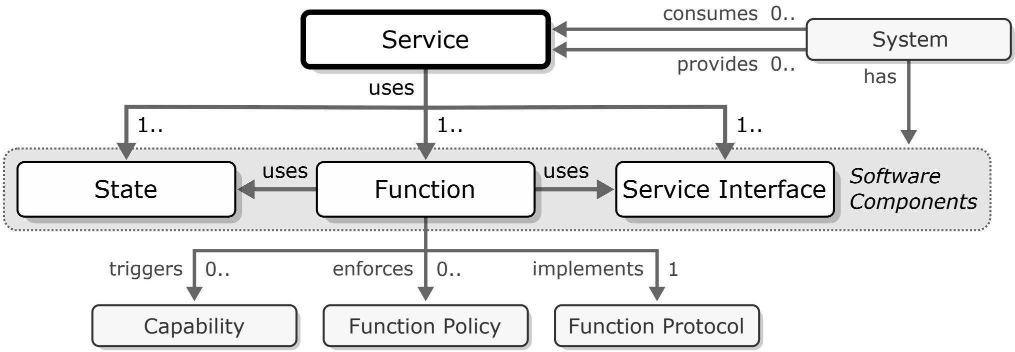

Figure 11: The service as means for a consuming system to trigger capabilities of a providing system. The group of software components is not exhaustive. See the caption of Figure 10 for additional examples.

Services receive messages via their interfaces, which must route them to the functions they target. A function receiving a message must guarantee that it adheres to its protocol and satisfies all of its policies, after which it must concretely handle the request described in the message. The function protocol dictates what data must be in the message, among other things, while the function policies may require that certain authorization tokens can be presented, that a future time slot in a real-time network has been allocated, and so on.

If it becomes relevant to distinguish the functions of programming languages with those introduced here, they should be referred to as *program functions* and *service functions*, respectively. Otherwise the unqualified use of the word *function* must be understood to refer to service functions.

3.6 System-of-Systems

A system-of-systems is an identifiable set of at least two systems, together facilitating one or more capabilities none of the constituent systems could have on its own. While this definition may initially seem rather strict, it is enough for a system to consume a service of another system for a system-of-systems to emerge. This as service consumption can only be motivated by the desire to facilitate new capabilities.

As illustrated in Figure 12, there are two types of systems-of-systems of particular relevance to the Arrowhead framework, (1) the local cloud and (2) the system-of-local-clouds, which we describe in the following sections.

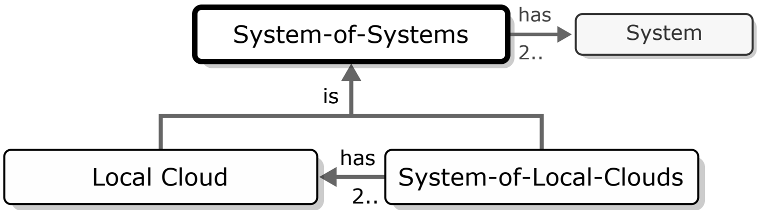

Figure 12: The system-of-system as a group of two or more systems, together facilitating new capabilities.

3.6.1 Local Cloud

A local cloud is a system-of-systems able to execute given tasks through the use of a pool of resources. As depicted in Figure 13, the local cloud is distinct from other types of clouds by having at least one local boundary and one local resource, which means that it is physically tied to a concrete location. A local cloud could be engaged in manufacturing, repairs, heating, electricity distribution, workspace monitoring, drone fleet control, among many other possible kinds of physical activities. A local cloud may be stationary or mobile. A cloud that has no resources or boundaries tied to any particular physical locations should be referred to as a *virtual cloud*.

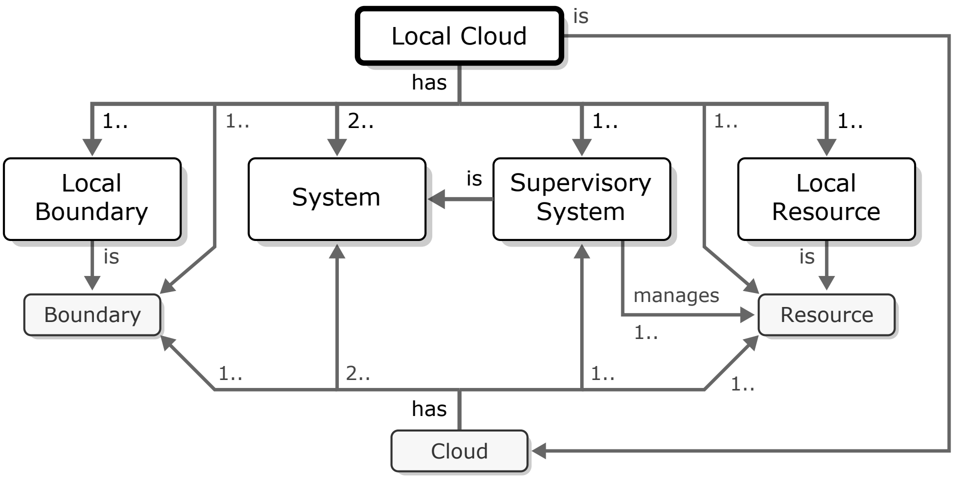

Figure 13: The local cloud as a regular cloud with at least one local boundary and one local resource.

That a local cloud has a boundary means that a distinction is being made between systems inside and outside the cloud. A boundary being local means that the distinction is being made by a physical property, such as device location, type of device, or physical attachment to a certain entity. Boundaries may be protected, which means that measures are in place to guarantee security, safety, real-time characteristics, or other local cloud properties. The resources of a local cloud may be of any type, from virtual compute resources to physical drills or pumps. A system managing a resource may be referred to as a *supervisory system*.

3.6.2 System-of-Local-Clouds

A system-of-local-clouds is two or more local clouds that consume each other's services to facilitate new capabilities. It is similar to the local cloud, with the exception of its subsystems are local clouds instead of plain systems. It has at least one common boundary, but no resources beyond those of its constituent clouds.

3.7 Network

A network is a set of two or more end devices, connected in such a manner that any systems they host are able to communicate. As shown in Figure 14, end devices may be interconnected via intermediary devices.

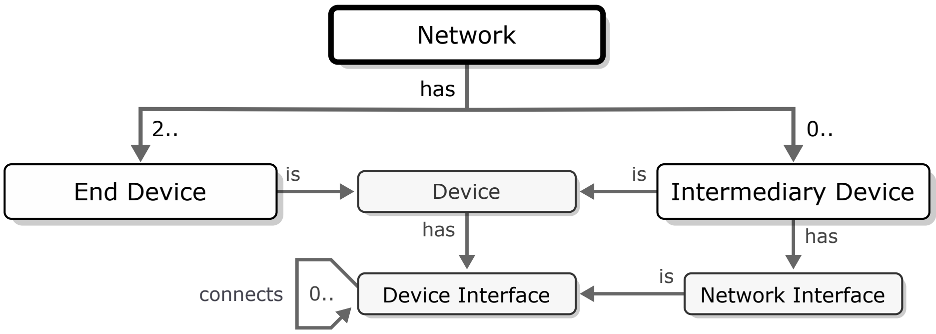

Figure 14: The network as a set of connected end devices, potentially interconnected by intermediary devices.

Both end devices and intermediary devices are regular devices, which means that they have device interfaces through which they can send and receive messages. Only *connected* devices can pass messages between their interfaces, however. Intermediary devices pass on messages toward their intended end devices instead of handling them themselves. Examples of intermediary devices are routers, switches, and firewalls. Device interfaces used exclusively for passing on messages are referred to as network interfaces. Use of the term "end device" is only recommended when a distinction needs to be made between end and intermediary devices.

3.8 Interface

An interface is a boundary where messages of certain protocols can pass between a connection and an entity, between two entities, or between an entity and a person. As outlined in Figure 15, there are three types of interfaces of particular relevance, (1) the device interface, (2) the system interface, and (3) the service interface.

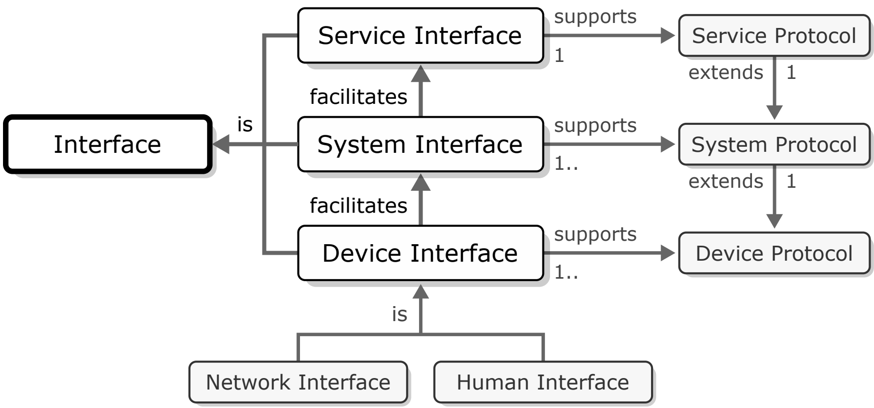

Figure 15: The interface as the boundary where messages pass between mediums and entities.

Device interfaces connect devices, system interfaces connect systems, and service interfaces connect service consumers with service providers. Each of these interface levels depend on the layer below it, with the exception of the device interface at the bottom. As each interface supports its own set of protocols, each interface layer must support a protocol that extends that of the below layer. A device interface may, for example, support the IP [6] protocol via Ethernet [7], a system interface the HTTP [8] protocol via TCP [9], and a service interface a custom HTTP extension enabling it to route messages to the functions of its service. Note that the protocol at each layers may consist of multiple protocols extending each other, as in this example. A particular chain of protocols supported by a device, system or service is referred to as a *protocol stack*. The protocol stack of the system in the previous example would be Ethernet, IP, TCP and HTTP, from the bottom and up.

3.9 Policy

A policy is a set of constraints, of any nature, that must be satisfied for a certain activity to be permitted. Policies may be concerned with authorization, contracts, economic goals, and so on. As depicted in Figure 16, there are two categories of policies of particular relevance, (1) service policies and (2) function policies.

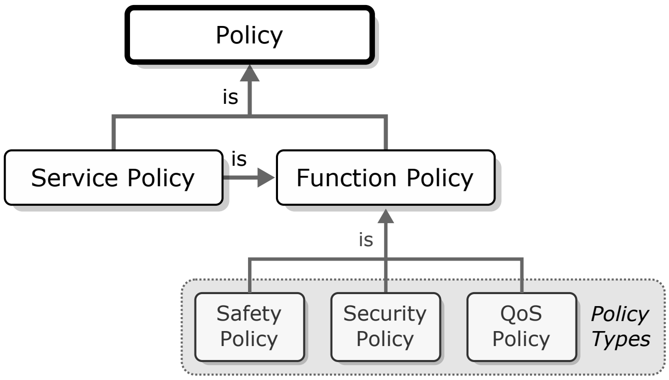

Figure 16: The two categories of policies of highest relevance to this reference model, service and function policies, as well as some examples of possible function policies. QoS is an abbreviation for quality of service. The group of policy types is not exhaustive. Other examples could be real-time, pollution or certificate policies.

A function policy must be satisfied for a system to be allowed to consume a particular service function. A service policy, or a *service-level* policy, applies to all functions of a given service. Failing to satisfy a policy should mean that the consumer in question is notified about the specific policy or policies being violated.

3.10 Protocol

A protocol is a description of how certain messages may be sent between entities as dictated by zero or more states. Received messages may be rejected by violating a current state, or cause a state to be updated. States may also be updated by other events. As illustrated in Figure 17, a protocol may be defined as an extension of another protocol, may be constrained by profiles, and defines its messages in terms of at least one encoding.

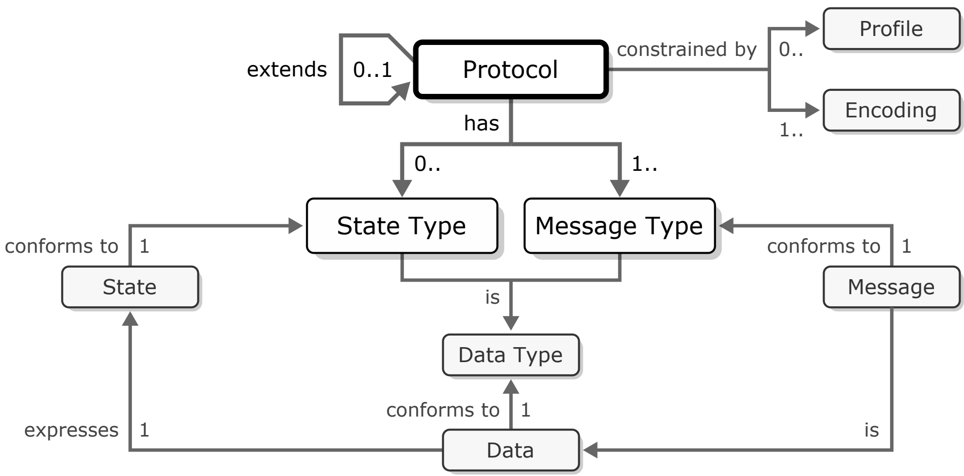

Figure 17: The protocol as set of state and message types, constrained by profiles and encodings.

A profile narrows down what a particular protocol is permitted to express, for example by requiring that authorization tokens be included in requests, or that a certain message payload semantics be observed. Adding a profile to a protocol not already conforming to it produces a new protocol. An encoding introduces a type system in which message payloads can be expressed. If a protocol defines its messages without referring to an encoding, it is considered to have a *custom* encoding.

4 Conformance Requirements

For a document, model, or other artifact, to be considered as conformant to the reference model recorded in this document, the following must be observed by that *derived work*:

1. At least one of the concepts defined in this reference model must be part of that derived work.
2. The derived work must make it explicit what concepts are taken from this reference model.
   (a) How this is done most suitably depends on the type of derived work. A document may include a normative reference to this document, while a model may want to give all relevant entities and relations a property with the identity of this document, for example.
3. Every concept taken from this reference model must be represented by the name it is given here.
   (a) If important to be able to distinguish an Arrowhead concept from other such of relevance, concepts from this reference model may be qualified by the leading word “Arrowhead”, as in, for example, “Arrowhead system“ or “Arrowhead service function”.
   (b) Note that some concepts defined here are given more than one name. For example, program function and software procedure are declared to be synonyms in the glossary. When synonyms exist, only one of their entries in the glossary will have a definition. The name of that definition should be the name being used.
4. Concepts taken from this reference model may be *specialized* or *simplified*, but must never be *contradicted*.
   (a) *Specialization* means that more constraints are applied to it than are presented here. For example, a certain derived work may require that all devices have compute units supporting a certain instruction set, or that every system provides a specific monitoring service, and so on.
   (b) *Simplification* means that entities, relationships or properties introduced here are omitted due to being outside the scope of the derived work. For example, a technical document may not be concerned with stakeholder roles, while a model of certain types of local clouds may not be concerned with whether or not artifacts are resources or not, and so on.
   (c) *Contradiction* means that a property or other constraint is introduced that makes it impossible to reconcile the concepts presented here with those in the derived work. A derived work must not, for example, demand that no devices ever host systems, or that services be provided directly by devices without any services, and so on.

5 Glossary

This section provides an alphabetically sorted list of all significant terms introduced or named in this document. Each term consisting of more than one word is sorted by its final, or qualified, word. This means that the definition of service protocol, for example, is found at Protocol, Service.

Many of the definitions are amended with notes and references to RAMI4.0 [5], SOA-RM [1] and IoTA:AF [3], which are always listed after the definition they amend. Regular notes are numbered, while those making a comment on a definition in RAMI4.0, SOA-RM or IoTA:AF are introduced with the three abbreviations just listed.

Abstract

See Model, Abstract.

Architecture

A concrete model of a system-of-systems defined in terms of certain reference models, reference architectures and other concrete architectures. See Section 1.2.

RAMI4.0 defines architecture as the "combination of elements of a model based on principles and rules for constructing, refining and using it". We consider "combinations of elements of a model" to be a "model of a system-of-systems" and to be "based on principles and rules for constructing, refining and using it" as building upon reference models and architectures. Our definition should be interpreted as being compatible but more specific.

SOA-RM defines software architecture as "the structure or structures of an information system consisting of entities and their externally visible properties, and the relationships among them". That definition is equivalent to our definition of model, with the exception that the thing being modeled has to be an information system. As our definition is concerned with a model and a system-of-systems, which must be an information system, we regard out definition as compatible but more specific.

Architecture, Reference

A significantly useful abstract model of a system-of-systems defined in terms of certain reference models and other reference architectures. See Section 1.2.

RAMI4.0 defines reference architecture as a "model for an architecture description (for I[ndustry]4.0) which is generally used and recognized as being suitable (has reference character)". We consider a "model for an architecture description" to be an "abstract model of a system-of-systems". Our definition should be interpreted as being compatible but more specific.

SOA-RM defines reference architecture as "an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements". While we let the part about requirements be implicit, our definition should be interpreted as being compatible but more specific.

Architecture, Service-Oriented (SOA)

An architecture concerned with service provision and consumption.

Note 1 Any architecture building upon the reference model of this document will become service-oriented. See also Section 1.

Arrowhead

See Framework, Arrowhead.

Artifact

A thing or object, tangible or intangible.

Asset

Synonymous to Resource.

RAMI4.0 defines asset as an "object which has a value for an organization". See Resource for a comparable term.

Boundary

A point or border where either two or more artifacts meet or one artifact ends.

Boundary, Cloud

A boundary separating the artifacts belonging to a cloud from those not belonging to it.

Note 1 A cloud boundary can be local or virtual, depending on if the boundary is formed by physical or virtual properties.

Boundary, Local

A boundary that exists in the physical world.

Note 1 Local boundaries can be facilitated by walls, locations of operation, attachment to certain vehicles or power sources, and so on.

Boundary, Virtual

A boundary that exists only virtually.

Note 1 Virtual boundaries can be facilitated by cryptographic secrets, identifiers, ownership statements, contracts, and so on.

Capability

A task, of any nature, that can be performed by a device.

Note 1 The term must be understood in the most general sense possible. It includes the abilities of hosting systems, reading from sensors, triggering actuators, among many other possible examples.

SOA-RM defines a capability as "a real-world effect that a service provider is able to provide to a service consumer". To leave room for devices to be described as doing other things that providing or consuming services, we made our definition more general. See also Capability, System.

Capability, System

A capability a system can trigger via its hosting device.

Cloud

A bounded system-of-systems able independently execute given tasks through the use of a pool of resources.

Note 1 When the term "cloud" is used elsewhere, it often refers to clouds with only virtual resources, such as compute, storage and software-defined network utilities. Here, we refer to such clouds as virtual clouds. By making the unqualified word "cloud" less specific, it becomes more clear how our local cloud concept shares similarities with other types of clouds.

Cloud, Local

A cloud bound to a physical location due to its acting on or producing local resources. See Section 3.6.1.

IoTA:AF provides an introduction to the local cloud concept in its second chapter, as well as an architectural definition in its third chapter. The following is an excerpt from the introduction:

> *The local cloud concept takes the view that specific geographically local automation tasks should be encapsulated and protected. These tasks have strong requirements on real time, ease of engineering, operation and maintenance, and system security and safety. The local cloud idea is to let the local cloud include the devices and systems required to perform the desired automation tasks, thus providing a local "room" which can be protected from outside activities. In other words, the cloud will provide a boundary to the open internet, thus aiming to protect the internal of the local cloud from the open internet.*

The third chapter contains the following:

> *In the Arrowhead Framework context a local cloud is defined as a self-contained network with the three mandatory core systems deployed and at least one application system deployed [...]*

Both of these descriptions are practical, in the sense that they emphasize engineering aspects. As this document is a reference model, engineering aspects are out of scope. The more general terms "geographically local", "room" and "boundary" clearly highlight the physicality of the local cloud itself, while the depiction of "devices" performing "automation tasks" makes it apparent that some kind of physical activity is involved, such as manufacturing. Finally, the local cloud being "encapsulated", "protected" and "self-contained" indicates that it is understood to exhibit a degree of independence with respect to the tasks it is given, which we expect all kinds of clouds to exhibit. Our definition should be interpreted as a summation of these characteristics.

Cloud, Local Automation

See Cloud, Local.

Cloud, Virtual

A cloud unbound by physical location by only acting on or producing virtual resources.

Code *(verb)*

Transforming data from being expressed in one encoding into another. See also decode and encode.

Coding *(noun)*

Synonymous to Encoding *(noun)*.

Component

An entity that can be part of a device or system and contribute to it facilitating its capabilities.

Note 1 The word "component" should only be used to refer to the constituents of devices and systems. It should never be used to refer to a system being a constituent of a system-of-systems. Such a system should be referred to as being a subsystem.

RAMI4.0 makes no practical distinction between components and systems, as is done here. See System for more details.

Component, Hardware

A physical component that can only be part of a device. See Section 3.3.

Component, Software

A virtual component that can only be part of a system. See Section 3.4.

Communication

The activity of sending and/or receiving messages.

Communication, Service-Oriented

Communication described in terms of the provision and consumption of services.

Concrete

See Model, Concrete.

Concretization

Making an abstract model less abstract by specifying some or all details required to realize it.

Configuration

A set of changeable properties that directly influence how a system exercises its capabilities.

Configure

To update a configuration.

Connection

An active medium through which attached interfaces can communicate.

Constraint

A property that imposes constraints, or limits, on an entity or relationship.

Note 1 The presence of constraints enable validation.

Note 2 Perhaps a bit counterintuitively, a constraint *adds* information to its target by reducing the ways in which it could be realized.

Consumer, Service

A system currently invoking a function provided via a service.

Note 1 If used to refer to a stakeholder, the term must be interpreted as if that stakeholder consumes services via systems.

SOA-RM defines a service consumer as "an entity which seeks to satisfy a particular need through the use [of] capabilities offered by means of a service". We require that the one consuming the service is (1) a system rather than just any entity, as well as (2) that the capabilities of the consumed service be exercised by invoking a function.

Data

A sequence of datums recording a set of descriptions via the structure superimposed by a data type.

Note 1 Let us assume that some data is going to be sent to a drilling machine. The type associated with the data requires that it always consists of 8 bits, organized such that the first 4 bits indicate the speed of drilling in multiples of 100 rotations per minute, while the latter 4 determine how much to lower the drill in multiples of 5 millimeters. A state that could be expressed with those 8 bits is `0100 1101`. If each of the two sequences of 4 bits is treated as a big-endian integer with base 2, they record 4 and 13 in decimal notation. This would indicate that the drill should spin at $4 * 100 = 400$ rotations per minute and be lowered $13 * 5 = 65$ millimeters.

Note 2 Without knowledge of the types and context associated with some data, that data cannot be interpreted.

Datum

A variable expressing one out of a set of possible values. See also State *(noun)*.

Note 1 A familiar example of a datum may be the bit, or binary digit. Its possible set of symbols is $\{0, 1\}$.

Decode

The act of transforming data from being expressed in a encoding suitable for transmission or storage to another encoding suitable for interpretation.

Note 1 Decoding is the reverse of encoding.

Note 2 The term can also be used to express the act of a human interpreting data.

Description

Facts about an entity or class of entities, expressed in the form of a model, a text, or both.

Design *(noun)*

Every document, model and other record describing how a certain artifact can be implemented.

Design *(verb)*

The activity of producing designs.

Designer

A stakeholder involved in the design of artifacts. See Section 3.1.

Developer

A stakeholder developing the components that make up devices and/or systems. See Section 3.1.

Device

A physical entity made from hardware components with the significant capability of being able to host systems. See Section 3.3.

IoTA:AF defines device as "a piece of equipment, machine, hardware, etc. with computational, memory and communication capabilities which hosts one or several Arrowhead Framework systems and can be bootstrapped in an Arrowhead local cloud". The definition provided here should be interpreted as being equivalent.

Device, Connected

A device that is physically attached to at least one other device via their interfaces, enabling them to communicate.

Device, End

A connected device being the intended recipient of a message.

Device, Human Interface (HID)

A device that provides sensors and actuators that together make up an interface through which a human can exchange messages with one or more systems.

Device, Intermediary

A connected device that receives and forwards messages toward end devices.

Encode

The act of transforming data from being expressed in a encoding suitable for interpretation to another encoding suitable for transmission or storage.

Note 1 Encoding is the reverse of decoding.

Note 2 The term can also be used to express the act of a human recording data.

Encoding *(noun)*

A concrete data type used to structure data for transmission, storage and/or interpretation.

Engineer, Standardization

A stakeholder involved in producing and ensuring conformance to standards and other significant specifications. See Section 3.1.

Entity

An artifact with an identity, allowing for it to be distinguished from all other artifacts. See Section 3.2.

Note 1 An entity being uniquely identifiable does not necessarily mean that it is associated with a certificate or identifier. It only means that a description can be rendered that unambiguously refers to the entity in question.

RAMI4.0 defines entity as an "uniquely identifiable object which is administered in the information world due to its importance". Our definition should be interpreted as being equivalent.

SOA-RM mentions the word "entity" nine times, but provides no explicit definition. We assume their definition to match that of a regular English dictionary, such as "something that has separate and distinct existence and objective or conceptual reality" [10]. Our definition should be interpreted as being equivalent.

Entity, Class of

A set of entities that share a common property.

Framework

A set of assumptions, concepts, values and practices that frame a certain problem domain.

SOA-RM defines framework as "a set of assumptions, concepts, values, and practices that constitutes a way of viewing the current environment". Our definition should be interpreted as being equivalent.

Framework, Arrowhead

Either of the framework of ideas and the framework of software maintained by the Arrowhead project. See Section 2.

Framework, Software

A set of software specifications, implementations and other artifacts meant to help address the problem domain of a certain framework.

Function

Typically synonymous to Function, Service. See Section 3.5.

> **Note 1** The term may be used to refer to both program and service functions, if the context makes it clear that both kinds of functions are relevant.

Function, Program

Synonymous to Procedure, Software.

Function, Service

A software procedure that handles messages it receives from a service interface. See Section 3.5.

HID

See Device, Human Interface (HID).

Industry 4.0

The fourth industrial paradigm, primarily characterized by high degrees of computerization, digitization and interconnectivity. See also [5].

Identification

The process through which an entity verifies the identity of another entity.

Identifier

Data associated with an entity that allows for it to be identified.

Identity

The aspect or aspects, such as identifiers, that makes an entity distinct from all other entities.

Implementation

The realization of a design as a set of artifacts.

Implementation, Software

An implementation comprised of executable software artifacts.

Instance, Software

A software artifact currently being executed by a compute unit.

Integrator

A stakeholder tasked with making systems work together by installing and configuring them. See Section 3.1.

Interconnection

A connection that passes through one or more intermediary devices.

Interface

A boundary where messages of certain protocols can pass between a connection and an entity, between two entities, or between an entity and a person. See Section 3.8.

Interface, Device

An interface through which a device may send and/or receive messages to/from a connection.

Interface, Human

An interface through which a person may send and/or receive messages to/from an entity.

interface, Network

An interface of an intermediary device, primarily intended to be used for passing on messages toward their intended end devices.

Interface, Service

An interface through which a certain service can be consumed.

Note 1 Consuming a service requires that messages be passed from its device to its system, and then from its system to the service itself. As the software making up the service is owned by the system, it is the system that is understood to produce any responses. Those are passed on via its device.

SOA-RM defines service interface as “the means by which the underlying capabilities of a service are accessed”. Our definition should be interpreted as being equivalent.

Interface, System

An interface through which a system may send and/or receive messages to/from its hosting device.

Note 1 The device may either send or respond to messages by its own accord, or pass on messages it receives to and/or from any of its device interfaces.

Invocation, Function

The attempt to exercise the capabilities of a system by sending a message to one of its functions.

Message *(noun)*

Data sent or received via a service interface.

Note 1 In the context of Arrowhead, messages are only sent to invoke the functions of services provided by systems.

Manager

A stakeholder with strategic, operational or other key responsibility over other stakeholders and/or significant entities. See Section 3.1.

Metadata

Data describing other data.

Model

A representation of facts in the form of a graph, consisting of entities, relationships and properties.

Note 1 Models can be expressed or recorded in many ways, including as visual diagrams, spoken words, text and binary data.

Note 2 Models can be human-readable, machine-readable, or both.

Model, Abstract

A model that is *insufficiently* specified to be possible to realize as the artifact it represents.

Note 1 Abstract models can be referred to by other models, serving as a form of constraint. They are commonly used to enforce a degree of uniformity across multiple other models.

Model, Concrete

A model that is *sufficiently* specified to be possible to realize as the artifact it represents.

Note 1 Two examples of artifacts that could be produced from a concrete model are concrete protocols, the messages of which can be practically coded, and software implementations.

Model, Information

A model consisting of related data types, messages and/or other information artifacts.

Note 1 For example, all data types used by a certain service make up the information model of that service. In other words, the concept represents a pool of information artifacts that can are useful to consider as belonging to the same group.

Model, Reference

An abstract model defining technical concepts of fundamental importance to a specific problem domain. See also Section 1.2.

RAMI4.0 defines reference model as a “model that is generally used and recognized as being suitable (has recommendation character) for deriving specific models”. We understand their use of the word “specific” to be equivalent to how we use “concrete”. Even though our definition clarifies that the model in question must be abstract, it should be interpreted as being equivalent.

SOA-RM defines reference model as “an abstract framework for understanding significant relationships among the entities of some environment that enables the development of specific architectures using consistent standards or specifications supporting that environment”. It further clarifies that a “reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details”. Our definition should be interpreted as being equivalent.

Network

A set of two or more end devices, connected in such a manner that any systems they host are able to communicate. See Section 3.7.

Operator

A stakeholder responsible for the configuration and oversight of systems and the resources those systems manage. See Section 3.1.

Owner

A stakeholder that owns significant resources and/or other artifacts. See Section 3.1.

Organization

A stakeholder comprised of an organized body of other stakeholders and/or other persons.

Policy

A set of constraints, of any nature, that must be satisfied for a certain activity to be permitted. See Section 3.9.

SOA-RM defines policy as “a statement of obligations, constraints or other conditions of use of an owned entity as defined by a participant”. Our definition should be interpreted as being equivalent.

Policy, Function

A policy that must be satisfied to be permitted to invoke a certain function.

Policy, Service

A policy that must be satisfied to be permitted to invoke any function part of a certain service.

Procedure

See Procedure, Software.

Procedure, Software

A segment of instructions, part of a software artifact, that perform some activity if executed.

Profile

See Profile, Protocol.

Profile, Protocol

A set of constraints superimposed on a protocol.

Note 1 A profile *never* introduces more messages to a protocol. It adds constraints to the existing messages of a protocol.

Note 2 A profile could, for example, introduce an authentication mechanism to a protocol by requiring that a certain type of token be included in each message. It could demand that a certain protocol be extended, or that a particular kind of encoding be used for message bodies, and so on.

Property

A name/value pair of data, associated with either an entity or a relationship.

Note 1 A property is a form of metadata.

Protocol

A model of communication defined in terms of states and messages. See Section 3.10.

Note 1 The states, if any, dictate the outcomes of sending certain messages. For example, let us assume that some state can be either `BUSY` or `READY`. If the former state would be the active when a certain message is received, the designated response could be an error message. If, however, the `READY` state would have been active, the state could be transitioned to the `BUSY` value and a success response be provided to the sender.

Protocol, Device

A protocol implemented by a device.

Protocol, Extensible

A protocol allowing for subprotocols to be formulated in terms of its messages. See also Stack, Protocol.

Note 1 Every new message introduced by a subprotocol must be a valid message of its superprotocol.

Note 2 Many of the currently prevalent protocols are designed with the intent of being extensible. For example, HTTP [8] provides provisions for an extending protocol to define its own set of directory operations, to simultaneously support multiple codecs, and so on.

Note 3 As long as a given protocol provides at least one message whose contents can be arbitrary, a subprotocol can be produced. This means that even protocols not designed to be extended can, in some context, be meaningfully used to define subprotocols.

Protocol, Function

A protocol implemented by a service function.

Note 1 A function protocol is always an extension of a service protocol. See Section 3.10 for more details.

Protocol, Service

A protocol implemented by a service.

Note 1 A service protocol is always an extension of a system protocol. See Section 3.10 for more details.

Protocol, System

A protocol implemented by a system.

Note 1 A system protocol is always an extension of a device protocol. See Section 3.10 for more details.

Provider, Service

A system that makes services available for consumption to other systems.

Note 1 If used to refer to a stakeholder, the term must be interpreted as if that stakeholder provides services via systems.

SOA-RM defines a service provider as "an entity (person or organization) that offers the use of capabilities by means of a service". Our definition is more specific in that it requires the entity be a system.

QoS

See Service, Quality of (QoS).

Relationship

A uni-directional association of two entities, possibly with an associated data name.

Researcher

A stakeholder involved in the analysis or development of significant entities, particularly with the ambition of facilitating properties or use cases that cannot be realized without refining, extending or replacing those entities. See Section 3.1.

Resource

An artifact that is of value to a stakeholder.

Note 1 Any type of artifact can be a resource, which includes everything from local resources, such as raw materials on devices, to virtual resources, such as systems or .

Note 2 An artifact stops be a resource when it is perceived as having no value, at which point it may be destroyed, recycled or sold to someone that does perceive it as a resource, for example.

Resource, Local

A resource whose value is inextricably tied to a physical property.

Note 1 Examples of local resources could be raw materials, drills, pumps, power stations, or drones.

Resource, Virtual

A resource whose value is not derived from any physical property.

Note 1 Examples of virtual resources could be compute, storage, or software-defined network utilities.

Role

See Role, Stakeholder.

Router

See Router, Message.

Router, Message

A hardware component or software procedure that receives and passes on messages toward their intended end entities.

Role, Stakeholder

An assignment, objective, or other responsibility, that makes a person or organization into a stakeholder.

Routing, Message

The act of forwarding a message towards the function it is meant to invoke.

Service

A set of functions that can be provided by a system via one or more service interfaces. See Section 3.5.

> **RAMI4.0** defines a service as “separate scope of functions offered by an entity or organization via interfaces”. Our definition restricts service provision to systems.
>
> **SOA-RM** defines a service as “the means by which the needs of a consumer are brought together with the capabilities of a provider”. Our definition is more specific about how the capabilities of a service are made available.
>
> **IoTA:AF** defines a service as “what [is] used to exchange information from a providing system to a consuming system”. It further adds that “in a service, capabilities are grouped together if they share the same context”. The definition presented here should be interpreted as being compatible but more specific about how information is exchanged and capabilities are invoked.

Service, Quality of (QoS)

The degree of performance at which a given service is provided.

SOA

See Architecture, Service-Oriented (SOA).

Software

A set of sequences of instructions that can be executed by a compute unit.

SoLC

See System-of-Local-Clouds (SoLC).

SoS

See System-of-Systems (SoS).

Stack, Extensible Protocol

A protocol stack whose topmost protocol is extensible.

Stack, Protocol

A stack with an extensible protocol as base and $n > 0$ subprotocols layered on top of it.

> **Note 1** Every protocol part of a protocol stack, with the exception of the topmost, must be extensible.
>
> **Note 2** An example of a notable protocol stack is that of HTTP [8]. It is defined as an extension of the TCP protocol, which in turn extends the IP protocol, which can work as a subprotocol of several other lower-level protocols. HTTP is an extensible protocol stack, which allows for an engineer to define an application-specific protocol on top of its stack.

Stake

Any type of engagement or commitment.

Stakeholder

A person or organization with stake in certain entities or enterprises. See Section 3.1.

State *(noun)*

One out of all possible sequences of values that could be expressed by the datums of some data.

Note 1 If the data would consist of a sequence of bits, each of which can only have the values 0 and 1, a state becomes a pattern of zeroes and ones those bits can record. Given four bits, possible states could, for example, be `0010` or `1001`.

Note 2 The term is often used as a wildcard for any kind of storage construct, including bit flags, state machines and graph databases.

State, Protocol

The state of a protocol in active use, determining what messages it currently deems valid. See Section 3.10.

State, Software

The state of a software instance, determining its current activities and its reactions to any future procedure calls.

Subprotocol

A protocol that is realized as an extension of another protocol.

Subsystem

A system or system-of-systems being a constituent of a larger system-of-systems.

Superprotocol

A protocol that is extended by another protocol.

System

An entity capable of providing services, consuming services, or both.

Note 1 The word "system" is more generally understood to be very inclusive, expressing the larger idea of connected components facilitating one or more capabilities. From the perspective of Arrowhead, however, capabilities can only be invoked through services, which means that a system unable to provide or consume services can only be described as a component of another system.

Note 2 A system is practically distinct from a system-of-systems by being represented only by a single identity. In contrast, a system-of-systems does either not have its own identity, or has both its own identity and another identity for each of its subsystems.

IoTA:AF defines a system as "what is providing and/or consuming services". It further adds that "a system can be the service provider of one or more services and at the same time the service consumer of one or more services". The definition presented here should be interpreted as equivalent.

System, Isolated

A system that is unable to either provide or consume services.

System, Supervisory

A system that is tasked with managing one or more resources beyond its direct control.

Note 1 All systems are managing the resources provided to them by their hosting devices, such as primary memory, compute time, and so on. This term is meant to capture the systems that are engaged in overseeing and/or managing resources beyond those directly provided. Examples of such scenarios could be a single system being responsible for provisioning other devices, or a system using its robot device to collect and handle raw materials.

System, Type

A set of data types that can be used together, often in the context of an encoding or programming language.

System-of-Local-Clouds (SoLC)

A set of local clouds that consume each other's services in order to facilitate a capability none of the constituent local clouds could provide on its own. See Section 3.6.2.

System-of-Systems (SoS)

A set of systems that consume each other's services in order to facilitate a capability none of the constituent systems could provide on its own. See Section 3.6.

IoTA:AF defines a system-of-systems as "a set of system, which [...] exchange information by means of services". It further adds that "when Arrowhead compliant systems collaborate, they become a System of Systems in the Arrowhead Framework's definition". While we clarify here that the desired outcome of collaboration is the facilitation of new capabilities, the definitions should be interpreted as being equivalent.

Type, Data

A description of how datums are to be arranged to code certain facts. See also Data.

Note 1 While this definition may seem foreign, it does capture how integer types, classes, enumerators and other general data type are used in the context of a programming language or encoding. In the end, all data are bits or other symbols. From our perspective, types serve to group those symbols and assign them meaning.

Note 2 A data type provides only syntactic, or structural, information about data. While knowing the data type used to code some data is required for its interpretation, contextual knowledge is also needed. For example, a data type may specify a `name`, but it will not indicate when or why that name is useful. That information would have to be provided via documentation or some other means.

Unit, Compute

A hardware component able to execute software adhering to its instruction set.

Unit, Memory

A hardware component maintaining a set of changeable datums.

User

A stakeholder involved in the usage of certain entities. See Section 3.1.

Note 1 The activity of *using* an entity is not related to its coming into existence, maintenance, decommissioning, or any other peripheral activity. When a user engages in an entity it produces whatever value it was designed to produce.

Validation

The process through which it is determined if a model satisfies a constraint.

6 References

[1] C. Bashioum, P. Behera, K. Breininger *et al.* "Reference Model for Service Oriented Architecture". *OASIS Standard soa-rm*, Organization for the Advancement of Structured Information (OASIS), October 2006. URL `https://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html`. Version 1.0, accessed 2021-10-05.

[2] "OMG Systems Modeling Language (OMG SysML™)". *OMG document*, Object Management Group (OMG), November 2019. URL `https://www.omg.org/spec/SysML/1.6/`. Version 1.6, accessed 2021-10-06.

[3] J. Delsing (editor). *IoT Automation: Arrowhead Framework*. CRC Press, 2017. ISBN 9781498756761.

[4] S. Bradner. "Key words for use in RFCs to Indicate Requirement Levels". *RFC 2119*, RFC Editor, March 1997. doi:10.17487/RFC2119.

[5] P. Adolphs, S. Berlik, W. Dorst *et al.* "Reference Architecture Model Industrie 4.0 (RAMI4.0)". *DIN SPEC 91345:2016-04*, Deutsches Institut für Normung (DIN), April 2016. doi:10.31030/2436156.

[6] S. Deering and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification". *RFC 8200*, RFC Editor, July 2017. doi:10.17487/RFC8200.

[7] "Telecommunications and exchange between information technology systems — Requirements for local and metropolitan area networks — Part 3: Standard for Ethernet". *Standard*, International Organization for Standardization (ISO), Geneva, CH, February 2021. URL `https://www.iso.org/standard/78299.html`. Accesses 2021-11-03.

[8] R. Fielding *et al.* "Hypertext transfer protocol (HTTP/1.1): Message syntax and routing". *RFC 7230*, RFC Editor, March 2014. doi:10.17487/RFC7230.

[9] J. Postel. "Transmission Control Protocol". *RFC 793*, RFC Editor, September 1981. doi:10.17487/RFC0793.

[10] Merriam-Webster. "Entity". URL `https://merriam-webster.com/dictionary/entity`. Accessed 2021-10-12.

7 Revision History

7.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1				

7.2 Quality Assurance

No.	Date	Version	Approved by
1			