**The Adaptive Service Model  
Architectural concepts, modelling language and principles**

# Introduction

Future best practice for governing, managing and operating services will be dynamic, emerging, empirical and holistic. Future best practices should not be published as static publications, but should be bodies of knowledge that emerge every day based on input from real people consuming, brokering or providing real services. Different bodies of knowledge should be aligned with each other to form a well-defined and consistent domain of service management.

Such complex emerging practices need underpinning structures. There is a need for a ‘spinal cord’ for the practices to develop and grow. That is the intention of the Adaptive Service Model. The Taking Service Forward initiative sees the need for a basis for co-creation and crowd-sourcing of future service management best practices. And we hope that our initiative may give birth to the structures supporting that vision.

# Contents

[Introduction](#h.gjdgxs)

[Contents](#h.1fob9te)

[Overview](#h.3znysh7)

[Scope](#h.2et92p0)

[Basic architectural concepts](#h.tyjcwt)

[Modelling language](#h.3dy6vkm)

[Views and viewpoints](#h.17dp8vu)

[Constraints](#h.3rdcrjn)

[Modelling tool](#h.26in1rg)

[Architecture principles](#h.lnxbz9)

[Further reading](#h.94eyfpn92f8v)

[Connect with us](#h.tdbyl1ajl8ez)

[Appendix – Other ArchiMate concepts](#h.dmoqw0r8dbjy)

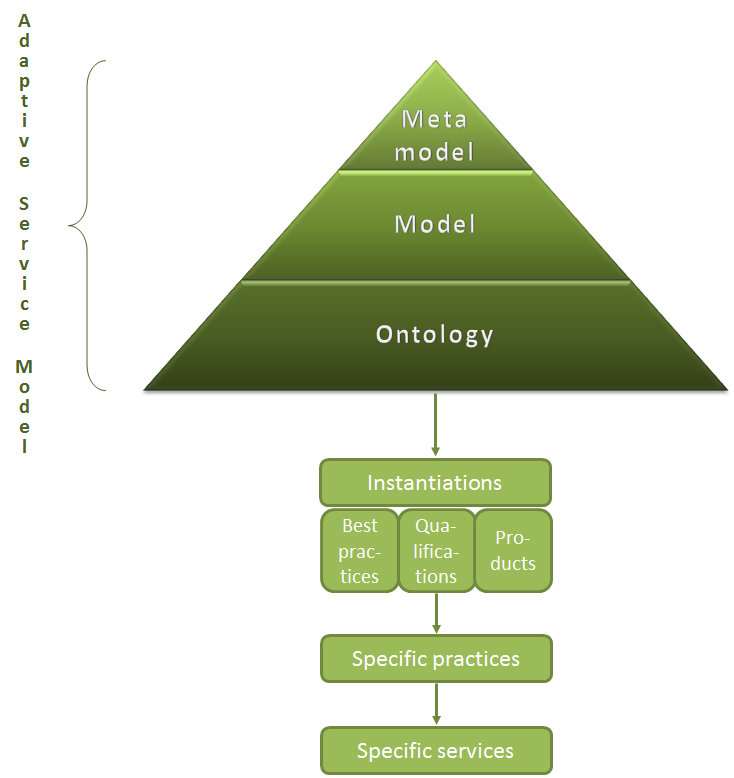
[Document Change Control](#h.z337ya)

# Overview

The Adaptive Service Model (ASM) is a generic reference architecture for service governance, management and operation. It is intended that organizations use it as a basis for creating more detailed domain and industry-specific architectures and ontologies, amongst other applications.

The service model is adaptive in several ways:

* It is adapted as the community participates in its design and evolution
* It needs to be adapted, by instantiation, to individual stakeholder situations
* Innovation among stakeholders contributes to the adaptation of the model
* Changes in industry knowledge, maturity, technology and other factors may lead to portions of the model becoming obsolescent.

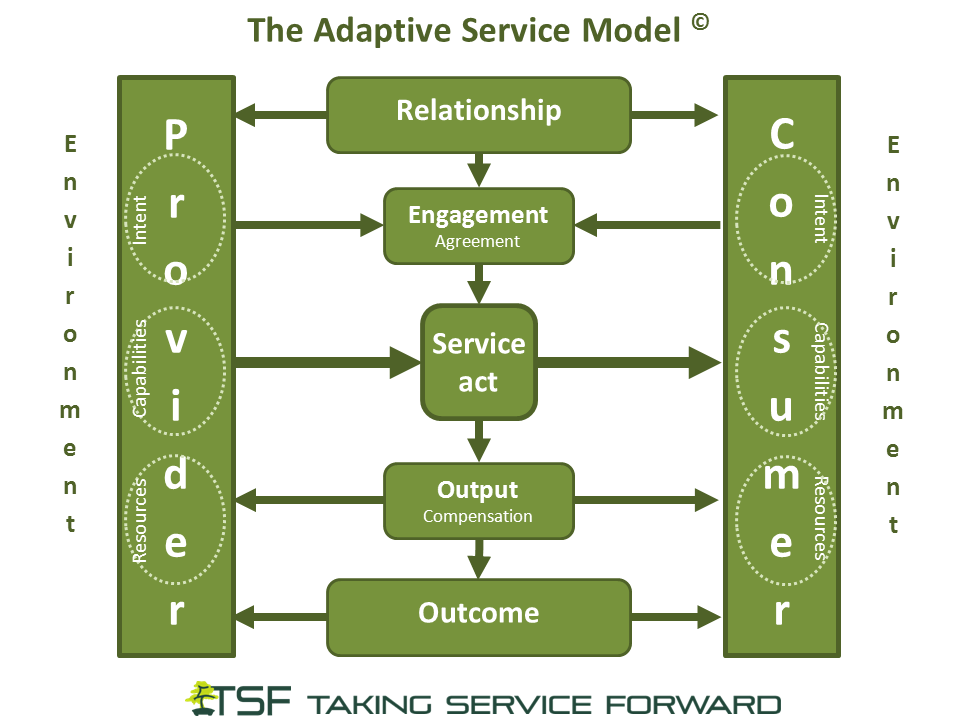


The Adaptive Service Model itself consists of three levels:

* The meta model. Specifies generic classes of entities (e.g. service, relationship, agreement, capability, process, actor, role, plan, information, etc.), their properties and their direct and derived relations
* The detailed model. Specifies specific entities and relations (e.g. specific processes such as Incident Management, specific roles such as Service Level Manager, etc.) for some or all of the classes of entities in the meta model
* The ontology. Defines the detailed protocol of interoperability, i.e. interfaces or exchange formats between the entities in the detailed model. (e.g. detailed message format for incident exchange)

The Adaptive Service Model will not include particular instantiations, such as ITIL, ISO/IEC 20000, USMbok or ISM.

At a very high abstraction level, the Adaptive Service Model can be summarized as follows:



# Scope

The scope of the Adaptive Service Model includes all types of services. The model is therefore not restricted to services enabled by information technology or services that exist in a business context.

Even though the model is not limited to an enterprise context, it is still aligned as far as possible to existing enterprise architecture meta models, such as the ArchiMate meta model, the TOGAF meta model (with the service extension), OBASHI etc.

The model takes the service consumer perspective as much as the service provider’s view and it focuses on service relation, interaction and enabling capabilities as much as on the embedded service resources.

In summary, the model should be as relevant for a service provider such as Google providing services that are heavily dependent on information technology as for you, when you ask your kids to wash the dishes at home.

# Basic architectural concepts

The architecture includes the following elements: Entity classes, entities, entity attributes, relations, relation attributes, viewpoints and views.

The architecture uses the following definitions for the elements used to describe the architecture (based on ISO/IEC 42010, TOGAF, ArchiMate 2.1 et. al.):

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Entity | May be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity may be an *object*, an *event*, or a *concept*. Entities can be thought of as nouns. |
| Relation | A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. |
| Attribute | Entities and relationships can both have properties. Every entity and relation must have a minimal set of identifying attributes. The terms *property* and *attribute* are used synonymously. |
| Class | We distinguish between entity and entity-type or class. An entity-type or class is a category. An entity, strictly speaking, is an instance of a given entity-type or class. |
| Viewpoint | The purpose of views and viewpoints is to enable humans to comprehend complex architectures, to organize the elements of the problem and the solution around domains of expertise and to separate concerns.  Each viewpoint satisfies an audience with interest in a particular set of aspects of the architecture.  The term viewpoint describes a partitioning of concerns in system restricted to a particular set of concerns. Adoption of a viewpoint is usable so that issues in those aspects can be addressed separately.  Viewpoints provide the conventions, rules, and languages for constructing, presenting and analyzing views. In ISO/IEC 42010:2007 a viewpoint is a specification for an individual view. A view is a representation of a whole system from the perspective of a viewpoint. |
| View | A view of a system is a representation of the system from the perspective of a viewpoint. This viewpoint on a system involves a perspective focusing on specific concerns regarding the system, which suppresses details to provide a simplified model having only those elements related to the concerns of the viewpoint. For example, a security viewpoint focuses on security concerns and a security viewpoint model contains those elements that are related to security from a more general model of a system  A view allows a user to examine a portion of a particular interest area. For example, an information view may present all functions, organizations, technology, etc. that use a particular piece of information, while the organizational view may present all functions, technology, and information of concern to a particular organization. |

The architecture with its entities and relations is described by using only three simple types of artefacts:

* Entity relationship diagrams
* Tables of attributes
* Descriptive documents (such as this document)

# Modelling language

The Taking Service Forward initiative has decided to use ArchiMate® 2.1 from The Open Group to represent the Adaptive Service Model. ArchiMate is an open and independent modelling language for enterprise architecture. ArchiMate, with its inherent service orientation, is a good match for this model.

Just as an architectural drawing in classical building architecture describes the various aspects of the structure of a building, ArchiMate offers a common language for describing the structure, as well as the operation, of business processes, organizational structures, information flows, IT systems, and technical infrastructure.

The full description of the ArchiMate standard can be found at <http://theopengroup.org/archimate/downloads.htm>.

The core language consists of three main types of elements: active structure elements, behavior elements, and passive structure elements (objects).

An *active structure element* is defined as a subject (e.g., business actors or business roles) that is capable of performing behavior. A *behavior element* is defined as a unit of activity such as a business process or function (capabilities) performed by one or more active structure elements. And a *passive structure element* is defined as an object on which behavior is performed.

ArchiMate also contains a core set of *relationships*.

Part of the ArchiMate definition is an abstraction rule that states that two relationships that join at an intermediate element can be combined and replaced by the weaker of the two. Transitively applying this property allows us to replace a “chain” of structural relationships (with intermediate model elements) by the weakest structural relationship in the chain. With this rule, it is possible to determine the “indirect” or *derived relationships* that exist between model elements without a direct relationship, which may be useful for, among other things, impact analysis. For example it is evident, that the Service Provider and Service Consumer have expectations, even though there only is an indirect relationship between the two objects. Derived relationships are not shown in the “bare bones” meta model, because this would clutter up the diagrams.

The ArchiMate language defines three main layers:

1. The Business Layer offers products and services to external customers, which are realized in the organization by business processes performed by business actors.
2. The Application Layer supports the business layer with application services which are realized by (software) applications.
3. The Technology Layer offers infrastructure services (e.g., processing, storage, and communication services) needed to run applications, realized by computer and communication hardware and system software.

The general structure of models within the different layers is similar. The same types of concepts and relationships are used, although their exact nature and granularity differ.

The Taking Service Forward initiative decided to replace the ArchiMate meta model with our own meta model, which is more generic when it comes to governance, management and operation of services. This meta model makes use of concepts from the ArchiMate Business Layer and Motivational Extension and is aligned as far as possible to the ArchiMate meta model. A description of the concepts used is included below.

For more information on the ArchiMate Application Layer, the Technology Layer and the Implementation and Migration extension that have not been used in the meta model please refer to the appendix.

***Business layer***

**Active Structure Concepts**

In the Adaptive Service Model, the active structure elements have been colored light blue.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Actor | A business actor is defined as an organizational entity that is capable of performing behavior. |  |
| Role | A business role is defined as the responsibility for performing specific behavior, to which an actor can be assigned. |  |
| Collaboration | Business collaboration is defined as an aggregate of two or more business roles that work together to perform collective behavior. |  |
| Interface | A business interface is defined as a point of access where a business service is made available to the environment. |  |
| Location | A location is defined as a conceptual point or extent in space. |  |

**Behavioral concepts**

In the Adaptive Service Model, the behavior elements have been colored yellow.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Process | A business process is defined as a behavior element that groups behavior based on an ordering of activities. It is intended to produce a defined set of products or business services. |  |
| Function | A business function is defined as a behavior element that groups behavior based on a chosen set of criteria (typically required business resources and/or competences). |  |
| Interaction | A business interaction is defined as a behavior element that describes the behavior of a business collaboration. |  |
| Event | A business event is defined as something that happens (internally or externally) and influences behavior. |  |
| Service | A business service is defined as a service that fulfils a business need for a customer (internal or external to the organization). |  |

**Passive Structure Concepts**

In the Adaptive Service Model, the passive structure elements have been colored green.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Object | A business object is defined as a passive element that has relevance from a business perspective. |  |
| Representation | A representation is defined as a perceptible form of the information carried by a business object. |  |
| Meaning | Meaning is defined as the knowledge or expertise present in a business object or its representation, given a particular context. |  |
| Value | Value is defined as the relative worth, utility, or importance of a business service or product. |  |
| Product | A product is defined as a coherent collection of services, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers. |  |
| Contract | A contract is defined as a formal or informal specification of an agreement that specifies the rights and obligations associated with a product. |  |

***Motivation extension***

**Motivational Concepts**

In the Adaptive Service Model, the motivational elements have retained the below colors (pink and purple).

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Stakeholder | A stakeholder is defined as the role of an individual, team, or organization (or classes thereof) that represents their interests in, or concerns relative to, the outcome of the architecture. |  |
| Driver | A driver is defined as something that creates, motivates, and fuels the change in an organization. |  |
| Assessment | An assessment is defined as the outcome of some analysis of some driver. |  |
| Goal | A goal is defined as an end state that a stakeholder intends to achieve. |  |
| Requirement | A requirement is defined as a statement of need that must be realized by a system. |  |
| Constraint | A constraint is defined as a restriction on the way in which a system is realized. |  |
| Principle | A principle is defined as a normative property of all systems in a given context, or the way in which they are realized. |  |

***Relationships***

**Structural relationships**

|  |  |  |
| --- | --- | --- |
| **Relation** | **Definition** | **Notation** |
| Composition Relationship | The composition relationship indicates that an object is composed of one or more other objects. |  |
| Aggregation Relationship | The aggregation relationship indicates that a concept groups a number of other concepts. |  |
| Assignment Relationship | The assignment relationship links active elements (e.g., business roles or application components) with units of behavior that are performed by them, or business actors with business roles that are fulfilled by them. |  |
| Realization Relationship | The realization relationship links a logical entity with a more concrete entity that realizes it. |  |
| Used by Relationship | The used by relationship models the use of services by processes, functions, or interactions and the access to interfaces by roles, components, or collaborations. |  |
| Access Relationship | The access relationship models the access of behavioral concepts to business or data objects. |  |
| Association Relationship | An association models a relationship between objects that is not covered by another, more specific relationship. |  |

**Dynamic relationships**

|  |  |  |
| --- | --- | --- |
| **Relation** | **Definition** | **Notation** |
| Triggering Relationship | The triggering relationship describes the temporal or causal relationships between processes, functions, interactions, and events. |  |
| Flow Relationship | The flow relationship describes the exchange or transfer of, for example, information or value between processes, function, interactions, and events. |  |
| Influence Relationship | The influence relationship models that some motivational element has a positive or negative influence on another motivational element. |  |

**Other relationships**

|  |  |  |
| --- | --- | --- |
| **Relation** | **Definition** | **Notation** |
| Grouping | The grouping relationship indicates that objects belong together based on some common characteristic. |  |
| Junction | A junction is used to connect dynamic relationships of the same type. |  |
| Specialization Relationship | The specialization relationship indicates that an object is a specialization of another object. |  |

# Views and viewpoints

Views are an ideal mechanism to purposefully convey information about architecture areas. In general, a view is defined as a part of an architecture description that addresses a set of related concerns and is addressed to a set of stakeholders. A view is specified by means of a viewpoint, which prescribes the concepts, models, analysis techniques, and visualizations that are provided by the view. Simply put, a view is what you see and a viewpoint is where you are looking from.

A viewpoint in ArchiMate is a selection of a relevant subset of the ArchiMate concepts (and their relationships) and the representation of that part of an architecture that is expressed in different diagrams. In addition to the standard ArchiMate viewpoints the Adaptive Service Model is intended to supports the following viewpoints:

* Governance versus management viewpoints
* Stakeholder viewpoints
* Organizational viewpoints
* Goals cascade viewpoint
* Lifecycle viewpoint
* Requirements viewpoint
* Management system viewpoint
* Capability viewpoint
* Resource viewpoint
* Risk and warranty viewpoints
* Compliance and controls viewpoint
* Outsourced versus insourced viewpoints
* Service provider versus service consumer viewpoints
* In-side-out versus out-side-in viewpoints
* Project viewpoint

# Constraints

Constraints are a reality that all organizations need to deal with and have a significant influence on what a business can do, what services are delivered and/or consumed. Constraints are dynamic, making them a significant challenge to model in an architecture. Any part of the business system may be subject to, or be a constraint in its own right.

For these reasons, we do not provide a statement about which constraints might exist. Rather, we prefer to provide guidance to stakeholders regarding the roles constraints play.

The approach adopted is based on these principles:

1. Almost any element in the Adaptive Service Model may act as a constraint on other elements due to interdependencies.
2. A chain is as strong as its weakest link. Constraints may be active (manifested) or inactive.
3. Active constraints may be bottlenecks. As those bottlenecks are managed and removed, the constraints become inactive. Meaningful metrics can help to identify constraints, but these metrics should not be the sole means for identifying constraints (these metrics can however not be seen as the sole source of identifying constraints)
4. Understanding inactive constraints, which are types of vulnerabilities, helps organizations to be more proactive in managing the influence of constraints on the system. This implies that the organization’s approach to managing risk needs to consider constraints and their influence.
5. A constraint may operate over an entire system, marketplace, set of resources, capabilities and services.
6. Although constraints are perceived to have negative effects, such as slowing the throughput of a service or increasing its costs, they may also be viewed as opportunities to be exploited.
7. It is not desirable to document constraints in a general architectural diagram. Given that virtually any entity may be a constraint, the risk is creating a complex, hard to read diagram. It is better to document constraints via specialized views that focus on the systematic delivery of services.

The table below can serve as a guide to identify constraints when adopting the Adaptive Service Model, these are only examples and should by no means be considered as a definitive list!

|  |  |  |  |
| --- | --- | --- | --- |
| **Constraint** | **Explanation** | **Example** | **Where applied** |
| Commitments | Existing commitments tie up resources that may prevent us from using these resources for other purposes | Contracts, agreements, promises, SLAs | supply, demand |
| Regulations | Prevents/limit practices, consume resource, prevent offering of services, require capabilities that we may not have | Laws, regulations, codes of practice | external |
| Dependencies | Any form of input required by an entity to be able to fulfil its function or achieve its stated objective, output or outcome | Processes, practices, cycle times, sequence, suppliers, services | supply |
| Location | The location of an entity or dependent entities may limit its usefulness or availability | Geography, accessibility, physical environment | supply, demand |
| Time | Time influences the ability of an organization to utilize capacity and resources. Organization within this context may also want to be aware of cycles of time | Time available, time of day/week/month/year, business or other cycles | supply |
| Practices | The way that work is done. This should include all activities | Processes, procedures, methods, cycle times, sequence, etc. | supply |
| Resources and capabilities | The ability and availability of resources and capabilities | Volume, utility, cost, level of capability | supply, demand |
| Control risk | Realization of risk and the organizations approach to dealing with realized risk | tolerance of risk, control of risk, management of risk | supply, demand |
| Cost | Availability or use/consumption of resource, competitiveness, profitability |  | supply |
| Demand | Demand in-itself creates opportunities unless demand outstrips the ability to supply when it becomes a constraint | Capacity of resources, consumption of resources, patterns of demand in market | demand |
| Policy strategy | All strategic decisions or choices may prevent/limit practices, consume resource, require capabilities that the organization may not have | Vision, mission, strategy, policy, values | supply, demand |

# Modelling tool

The tool Archi 2.4 has been used for modelling. Archi is a free, open source, cross-platform tool and editor to create ArchiMate models. It is available for download for Windows, Mac and Linux on <http://archi.cetis.ac.uk/>

# Architecture principles

The Taking Service Forward initiative has decided that the following architecture principles are applicable for the design of the Adaptive Service Model:

* The architecture must be simple and elegant. Good architecture is like a bonsai tree: Growing is the easy part, the real art lies in the pruning.
* The architecture must be readable by people without prior knowledge of information models or ontologies, but also informative for architecture experts
* The architecture must be generic and applicable for all service management frameworks and approaches (instances)
* The architecture must include entities that are enabled by services as well as entities that enable services
* The architecture must be available for anyone to modify and reuse without restriction
* The architecture must be vendor neutral and not favor any particular commercial interest
* The architecture should integrate with and take advantage of existing architectures, best practices and standards in areas such as process, people, information, application, and infrastructure
* It must support subsequent application of different views and viewpoints without having to change the contents of the architecture itself (classes, entities, relations and properties).
* An element is defined as a separate individual entity when
  + The properties and/or relationships of the element are significantly different from similar elements
  + The element requires independent control or management
  + It is necessary to break the above rules for pragmatic reasons such as intuition or politics
* An element is probably not a separate entity but a property if it is, by its very nature, a measurement or a value.
* Relationships will only be assigned the following properties to keep the architecture simple: Name and type

# Further reading

[Taking Service Forward - The charter](http://bit.do/TSFcharter)

[Taking Service Forward - The story](http://bit.do/TSFstory)

[Taking Service Forward - The roadmap for the future](http://bit.do/TSFroadmap)

[Adaptive Service Model - The context](http://bit.do/ASMcontext)

[Adaptive Service Model - High level diagram](http://bit.do/ASMabstractiondiagram)

[Adaptive Service Model - Architecture concepts, modelling language and principles](http://bit.do/ASMdescription)

[Adaptive Service Model - Meta model - Diagram](http://bit.do/ASMmetamodeldiagram)

[Adaptive Service Model - Meta model - Objects and attributes](http://bit.do/ASMmetamodelobjects)

# Connect with us

[Web site](http://www.takingserviceforward.org/)

[Document library](http://bit.do/TSFdocuments)

[Google+ community](https://plus.google.com/u/0/communities/108142093643107261658)

 [Facebook](https://www.facebook.com/TakingServiceForward?fref=ts)

[Twitter](https://twitter.com/TSF_ASM)

[Twitter #tag](https://twitter.com/search?q=%23archSM&src=typd&f=realtime)

 [LinkedIn](http://www.linkedin.com/company/taking-service-forward?trk=company_logo)

[LinkedIn group](http://www.linkedin.com/groups/Taking-Service-Forward-Adaptive-Service-7430088/about)

# Appendix – Other ArchiMate concepts

The ArchiMate concepts described in this appendix have not been used in the initial release of the Adaptive Service Model.

***Application layer***

**Active Structure Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Component | An application component is defined as a modular, deployable, and replaceable part of a software system that encapsulates its behavior and data and exposes these through a set of interfaces. |  |
| Collaboration | An application collaboration is defined as an aggregate of two or more application components that work together to perform collective behavior. |  |
| Interface | An application interface is defined as a point of access where an application service is made available to a user or another application component. |  |

**Behavioral Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Function | An application function is defined as a behavior element that groups automated behavior that can be performed by an application component. |  |
| Interaction | An application interaction is defined as a behavior element that describes the behavior of an application collaboration. |  |
| Service | An application service is defined as a service that exposes automated behavior. |  |

**Passive Structure Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Data Object | A data object is defined as a passive element suitable for automated processing. |  |

***Technology layer***

**Active Structure Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Node | A node is defined as a computational resource upon which artifacts may be stored or deployed for execution. |  |
| Device | A device is defined as a hardware resource upon which artifacts may be stored or deployed for execution. |  |
| System Software | System software represents a software environment for specific types of components and objects that are deployed on it in the form of artifacts. |  |
| Infrastructure Interface | An infrastructure interface is defined as a point of access where infrastructure services offered by a node can be accessed by other nodes and application components. |  |
| Network | A network is defined as a communication medium between two or more devices. |  |
| Communica-  tion Path | A communication path is defined as a link between two or more nodes, through which these nodes can exchange data. |  |

**Behavioral Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Infrastructure Function | An infrastructure function is defined as a behavior element that groups infrastructural behavior that can be performed by a node. |  |
| Infrastructure Service | An infrastructure service is defined as an externally visible unit of functionality, provided by one or more nodes, exposed through well-defined interfaces, and meaningful to the environment. |  |

**Passive Structure Concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Artifact | An artifact is defined as a physical piece of data that is used or produced in a software development process, or by deployment and operation of a system. |  |

***Implementation and Migration extension***

**Relevant concepts**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Definition** | **Notation** |
| Work Package | A work package is defined as a series of actions designed to accomplish a unique goal within a specified time. |  |
| Deliverable | A deliverable is defined as a precisely-defined outcome of a work package. |  |

# Document Change Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Date of Issue** | **Author(s)** | **Brief Description of Change** |
| **1** | **Oct 31 2013** | **Christian F. Nissen** | **Initial document draft as guiding principles for the Birmingham workshop** |
| **2** | **Nov 10 2013** | **Christian F. Nissen** | **Document updated with concepts and principles agreed on the Birmingham workshop** |
| **3** | **Nov 13 2013** | **Johann Botha** | **Description of ‘Constraints’ added** |
| **4** | **Nov 17 2013** | **Robert Falkowitz** | **Elements from ‘Understanding and using the Adaptive Service Model’ added** |
| **5** | **Dec 19 2013** | **Christian F. Nissen** | **Document updated to ArchiMate 2.1 and aligned with the first version of the meta model** |
| **6** | **Jan 13 2014** | **Christian F. Nissen** | **Document rewritten to align it with the changes to the Adaptive Service Model etc.**  **Document harmonized with other TSF documents** |

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **Jan 15 2014** | **Sharon Taylor, Stuart Rance, Robert Falkowitz** | **Minor edits to resolve grammar issues and many comments added with suggested changes** |
| **8** | **Jan 16 2014** | **Christian F. Nissen** | **Consolidation of review comments** |
| **9** | **June 26 2014** | **Christian F. Nissen** | **The abstraction of the meta model was changed to reflect** [**Chrnis 5062014 RFC**](http://takingserviceforward.org/wiki//index.php?title=Chrnis_5062014_RFC) |