Using the ArchiMate® Language with UML®

A White Paper by:

Chris Armstrong, APG
J.D. Baker, APG
Iver Band, The Standard
Sam Courtney, APG
Henk Jonkers, BiZZdesign
Veer Muchandi, Hewlett-Packard
Martin Owen, Corso

September, 2013
Using the ArchiMate® Language with UML®

Copyright © 2013, The Open Group

The Open Group hereby authorizes you to use this document for any purpose, PROVIDED THAT any copy of this document, or any part thereof, which you make shall retain all copyright and other proprietary notices contained herein.

This document may contain other proprietary notices and copyright information.

Nothing contained herein shall be construed as conferring by implication, estoppel, or otherwise any license or right under any patent or trademark of The Open Group or any third party. Except as expressly provided above, nothing contained herein shall be construed as conferring any license or right under any copyright of The Open Group.

Note that any product, process, or technology in this document may be the subject of other intellectual property rights reserved by The Open Group, and may not be licensed hereunder.

This document is provided "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Any publication of The Open Group may include technical inaccuracies or typographical errors. Changes may be periodically made to these publications; these changes will be incorporated in new editions of these publications. The Open Group may make improvements and/or changes in the products and/or the programs described in these publications at any time without notice.

Should any viewer of this document respond with information including feedback data, such as questions, comments, suggestions, or the like regarding the content of this document, such information shall be deemed to be non-confidential and The Open Group shall have no obligation of any kind with respect to such information and shall be free to reproduce, use, disclose, and distribute the information to others without limitation. Further, The Open Group shall be free to use any ideas, concepts, know-how, or techniques contained in such information for any purpose whatsoever including but not limited to developing, manufacturing, and marketing products incorporating such information.

If you did not obtain this copy through The Open Group, it may not be the latest version. For your convenience, the latest version of this publication may be downloaded at www.opengroup.org/bookstore.

ArchiMate®, DirecNet®, Jericho Forum®, Making Standards Work®, OpenPegasus®, The Open Group®, TOGAF®, and UNIX® are registered trademarks and Boundaryless Information Flow™, Dependability Through Assuredness™, FACETM, Open Platform 3.0™, and The Open Group Certification Mark™ are trademarks of The Open Group.

MDA®, Model-Driven Architecture®, Object Management Group®, OMG®, UML®, Unified Modeling Language®, and XMI® are registered trademarks, and BPMN™, Business Process Modeling Language™, and MOF™ are trademarks of the Object Management Group, Inc. in the United States and/or other countries.

All other brands, company, and product names are used for identification purposes only and may be trademarks that are the sole property of their respective owners.

Using the ArchiMate® Language with UML®

Document No.: W134

Published by The Open Group, September 2013.

Any comments relating to the material contained in this document may be submitted to:

The Open Group, 44 Montgomery St. #960, San Francisco, CA 94104, USA

or by email to:

ogpubs@opengroup.org
# Table of Contents

- Executive Summary .......................................................... 4
- Introduction ................................................................................. 5
- The TOGAF Framework .......................................................... 6
  - Structure .................................................................................. 6
  - Architecture Development Method (ADM) ............................. 7
- The ArchiMate Language ......................................................... 9
  - Core Concepts ......................................................................... 9
  - Extensions .............................................................................. 10
  - Viewpoints ........................................................................... 11
  - The ArchiMate Modeling Language and the TOGAF Standard ... 11
- UML ....................................................................................... 12
  - Core Concepts and Evolution ............................................. 12
  - Current Usage and White Paper Positioning ....................... 13
  - Applying UML to Solution Architecture ............................ 14
- Using UML with the ArchiMate Language ............................... 15
  - Informal Methods ................................................................. 15
  - Approaches to Implementing Formal Methods .................. 19
- Case Study .............................................................................. 22
  - Scenario .................................................................................. 22
  - Modeling the Scenario with ArchiMate Views .................... 22
  - Elaborating the ArchiMate Views with UML Diagrams ....... 25
  - Requirements and Use-Case Views .................................... 25
- Conclusion .............................................................................. 33
  - Benefits of using the ArchiMate Language and UML Together ... 33
  - Opportunities for Further Work ........................................ 33
- References ............................................................................... 34
- About the Authors .................................................................. 36
- About The Open Group .......................................................... 38
Boundaryless Information Flow™
achieved through global interoperability
in a secure, reliable, and timely manner

Executive Summary

Enterprises committed to aligning their process and systems investments with their business strategies must work at multiple architectural levels defined by the TOGAF® standard, including enterprise-wide strategic architectures, and more granular segment and capability architectures. Solution Architectures may occur at the segment and capability level. Enterprise Architectures are used for high-level direction setting and strategic planning, while Solution Architectures guide lower-level individual project efforts.

The ArchiMate® visual modeling language standard is a natural choice for Enterprise Architectures while, for Solution Architectures, the Unified Modeling Language® (UML®) provides a wide range of views, concepts, and relationships. When architects make these common and workable choices, understanding how to use the ArchiMate language and UML together is necessary for efficient and precise alignment between the Enterprise and Solution Architectures.

This White Paper focuses on the use of the ArchiMate language for Enterprise Architectures and UML for IT Solution Architectures, even though both languages have other uses. However, this White Paper is also relevant to other uses of these languages. It explores several ways of using the two languages together.

This White Paper is written for Enterprise and Solution Architects and the analysts, designers, and developers who must consume their work. It supports The Open Group vision of Boundaryless Information Flow by enabling the definition and elaboration of complex and inter-related architectures using industry-standard visual modeling languages that foster productivity and comprehension among architecture creators, influencers, and consumers.
Introduction

Enterprises committed to aligning their process and systems investments with their business strategies must work at multiple architectural levels. TOGAF, an Open Group standard, provides for three types of Enterprise Architectures: strategic architectures, and more granular segment and capability architectures. Strategic architectures support executive-level direction setting and planning, while segment architectures support these activities at the program and portfolio level. Capability architectures support the development of architecture roadmaps realizing increments in the abilities of organizations, persons, or systems [1:20.2, 3.26]. Solution Architectures, as considered in this White Paper, may occur at the segment and capability level. They describe particular business activities and how IT supports them, typically as a result of a single project or release [1:3.65]. Enterprise Architectures are used for high-level direction setting and strategic planning, while Solution Architectures guide lower-level individual project efforts.

The ArchiMate visual modeling language, also a standard of The Open Group, is a natural choice for Enterprise Architectures, since it supports representations that span the business, information systems, and technology domains. For Solution Architectures, the Unified Modeling Language (UML) [4], a standard of the Object Management Group, is a choice that provides a wide range of views, concepts, and relationships. When architects make these common and workable choices, systematic approaches are necessary for efficient and precise alignment between the Enterprise Architectures that guide direction setting, planning, and roadmaps, and the Solution Architectures that guide individual projects or releases to deployment and implementation. As architects evolve an Enterprise Architecture iteratively, these approaches ensure a smooth transition of architecture models between the ArchiMate and UML notations.

This White Paper focuses on the use of the ArchiMate language for Enterprise Architectures and UML for IT Solution Architectures, although it is also generally applicable to other combinations of the two languages. This work is not intended to constrain the use of the two languages. Indeed, the ArchiMate language is used for Solution Architecture, and UML is used for Enterprise Architecture [1:35]. But this White Paper is based on the intentional alignment between the TOGAF Enterprise Architecture framework, and the ArchiMate visual modeling language. This alignment is discussed later. This paper is also based on the extensive use of UML for system architecture across many industries [12:38].

This White Paper identifies several approaches for the benefit of Enterprise and Solution Architects and the analysts, designers, and developers that must consume their work. Informal approaches include general descriptions of relationships between the languages, shared repositories with separate meta-models, and basic inter-model relationships such as equivalence and refinement. Formal approaches include UML profiles for the ArchiMate language and formal mappings between the two languages. Modeling tools can support both informal and formal approaches. After summarizing these approaches, this White Paper explores in greater depth the use of UML diagrams to elaborate ArchiMate views using a case study.

The remainder of this section describes the TOGAF standard, as well as the ArchiMate and UML modeling languages. It also discusses the relationship between these. Readers who are familiar with these standards may wish to skip directly to the section Using UML with the ArchiMate Language.
The TOGAF Framework

The TOGAF standard [1, 2] is an architecture framework. Put simply, the TOGAF standard is a tool for assisting in the acceptance, production, use, and maintenance of architectures. It is based on an iterative process model supported by best practices and a re-usable set of existing architectural assets. In the context of the TOGAF standard, “architecture” has two meanings:

1. A formal description of a system, or a detailed plan of the system at a component level to guide its implementation
2. The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time

The TOGAF standard is developed and maintained by The Open Group Architecture Forum. The first version of the TOGAF standard, developed in 1995, was based on the US Department of Defense Technical Architecture Framework for Information Management (TAFIM). Starting from this sound foundation, The Open Group Architecture Forum has developed successive versions of the TOGAF standard at regular intervals and published each one on The Open Group public web site. This White Paper refers to TOGAF Version 9.1, which was first published in December 2011.

The TOGAF standard can be used for developing and applying a broad range of different Enterprise Architectures. These architectures can be categorized into four related domains:

• Business Architecture, which encompasses an organization’s business strategy, governance, and key business processes
• Data Architecture, which is the structure of an organization’s logical and physical data assets and data management resources
• Application Architecture, which is a blueprint for individual applications to be deployed, their interactions, and their relationships to the core business processes of the organization
• Technology Architecture, which is the logical software and hardware capabilities that are required to support the deployment of business, data, and application services; this includes IT infrastructure, middleware, networks, communications, processing, and standards

The TOGAF standard complements, and can be used in conjunction with, other frameworks that are more focused on specific deliverables for particular vertical sectors such as government, telecommunications, manufacturing, defense, and finance. The key to the TOGAF standard is the method – the TOGAF Architecture Development Method (ADM) – for developing an Enterprise Architecture that addresses business needs.

Structure

The TOGAF 9.1 standard consists of:

• Part I: Introduction, which provides a high-level introduction to the key concepts of Enterprise Architecture and, in particular, to the TOGAF approach. It contains the definitions of terms used throughout the TOGAF standard and release notes detailing the changes between this version and the previous version of the TOGAF standard.
• Part II: Architecture Development Method (ADM) (Figure 1), which is the core of the TOGAF standard.
The ADM is a full lifecycle approach for creating, applying, and evolving Enterprise Architectures.

- Part III: ADM Guidelines and Techniques, which provides methods for adapting the ADM to different process styles and specialized architectures, as well as techniques for specific ADM tasks.

- Part IV: Architecture Content Framework (ACF), describes and inter-relates the work products that architects produce as they execute the ADM, including formally specified, delivered, and reviewed deliverables, more granular, viewpoint-specific artifacts, and potentially re-usable building blocks.

- Part V: Enterprise Continuum and Tools, which describes how architects can classify artifacts and structure a repository that supports straightforward management, reference, and re-use. Part V also provides guidance on the evaluation and selection of Enterprise Architecture tools.

- Part VI: Reference Models, which consists of a Technical Reference Model (TRM), and an Integrated Information Infrastructure Reference Model (III-RM). The TRM models a generic application platform that emphasizes application portability and interoperability. The III-RM expands the TRM Infrastructure Applications and Business Applications components to model integrated information delivery and access.

- Part VII: Architecture Capability Framework, which discusses the organization, processes, skills, roles, and responsibilities required to establish and operate an architecture practice within an enterprise.

- Appendices A and B, which respectively cover Supplementary Definitions and Abbreviations.

**Architecture Development Method (ADM)**

The TOGAF ADM, a result of contributions from many architecture practitioners, forms the core of the TOGAF standard. It is a method for deriving organization-specific Enterprise Architectures and is specifically designed to address business requirements. The ADM describes:

- A reliable, proven way of developing and using an Enterprise Architecture

- A method of developing related Business, Data, Application, and Technology Architectures that enable the architect to ensure that a complex set of requirements is adequately addressed

- A set of guidelines and techniques for architecture development

The ADM consists of a number of phases that cycle through a range of architecture domains that enable the architect to ensure that a complex set of requirements is adequately addressed. The basic structure of the ADM is shown in Figure 1. This White Paper is most relevant to Phase G: Implementation Governance, since it explores methods for supporting alignment of Solution Architectures with Enterprise Architectures.
Figure 1: The TOGAF Architecture Development Method (ADM) (Source [1])
The ArchiMate Language

Core Concepts

ArchiMate [3], an Open Group standard, is an open and independent modeling language for Enterprise Architecture that is supported by different tool vendors and consulting firms. It provides uniform representations for diagrams that describe Enterprise Architectures. Its core concepts (Figure 2) specify three main types of elements that are in turn often used to represent classes of real-world entities. These element types are:

- Active Structure Elements, which are entities capable of performing behavior
- Behavior Elements, which are units of activity performed by one or more Active Structure Elements
- Passive Structure Elements, upon which Active Structure Elements perform behavior

The ArchiMate language specializes two of these core element types to enable service-oriented architectural viewpoints:

- Behavior Elements, known as services, are units of functionality that systems expose to their environments. Services deliver value to their consumers while concealing the internal operations of the systems that expose them.
- Active Structure Elements, known as Interfaces, are points of access where systems expose one or more services to their environments.

![Diagram of ArchiMate Core Concepts](source.png)

Figure 2: ArchiMate Core Concepts (Source [3])

Note that, in this diagram, relationships are read using the verb closest to the first element; e.g., an Active Structure Element uses an Interface, and a Service accesses a Passive Structure Element.

The ArchiMate language contains a core set of relationships that fall into three categories:

- Structural relationships model the structural coherence between structural or behavioral concepts of the same or different types. They include association, access, used by, realization, assignment, aggregation, and composition.
Using the ArchiMate® Language with UML®

- Dynamic relationships model dependencies between behavioral concepts. They include flow and triggering. In addition, the ArchiMate language enables the derivation of dynamic relationships between structural elements to which the behavioral functions are assigned. For example, modelers can depict a flow relationship between two application functions as a flow relationship between separate Application Components that perform those functions.

- Other relationships are neither structural nor dynamic. They include grouping, junction, and specialization.

The ArchiMate language defines three main layers based on specializations of its core concepts:

- The Business Layer models products and services available to external customers of the organization that is being described. These services are realized by business processes that are performed by business actors.

- The Application Layer provides the Business Layer with application services that are realized by software applications.

- The Technology Layer provides the infrastructure services such as data processing, storage, and communications necessary to run applications. These services are realized by hardware and system software.

The ArchiMate language combines its three layers with its three core element types for a nine-cell framework (Figure 3).

![Figure 3: The ArchiMate Core Framework (Source [3])](image)

Extensions

To its nine-cell core framework, the ArchiMate 2.0 standard adds two extensions:

- The Motivation extension models the elements that motivate enterprise design and operation. Its concepts include: stakeholder, driver, assessment, goal, requirement, and principle.

- The Implementation and Migration extension models the implementation of all aspects of Enterprise Architectures, as well as the migration between generations of implemented architectures. Its concepts include: work package, deliverable, plateau, and gap.
**Viewpoints**

The ArchiMate language also includes a set of architecture viewpoints and classifies them in two ways:

- **Purpose**, which may be designing a solution, deciding on a course of action, or informing employees, customers, or other stakeholders.

- **Abstraction levels**, which may embody the details needed by stakeholders such as software and process engineers, the systemic coherence needed by operational managers who must understand key relationships to solve problems and implement change, or the overview needed by executives, Enterprise Architects, and others who must make key decisions and manage change.

**The ArchiMate Modeling Language and the TOGAF Standard**

The ArchiMate modeling language, together with its two extensions, can be used to model architectures developed using the TOGAF ADM. Figure 4 shows the correspondence between the activities of the ADM phases and the parts of the ArchiMate language. Previous Open Group White Papers [5, 6, 7] demonstrate this correspondence.

![Figure 4: Correspondence between ADM Phases and the ArchiMate Framework (Source [3])]
UML

Core Concepts and Evolution

The Unified Modeling Language (UML) is a visual language for specifying, constructing, and documenting the artifacts of systems. It is the most widely used modeling notation in the software industry today, and is an Object Management Group (OMG) specification. UML emerged from the Object-Oriented Analysis and Design (OOAD) methods of Booch, Rumbaugh, Jacobson, and many others that were all standardized during late 1990s and adopted by OMG in November 1997. Currently, UML is the de facto modeling standard for software development. UML is used throughout the entire Software Development Life Cycle (SDLC), covering system requirements, application architecture, software and infrastructure design, quality assurance, and deployment models.

Initial versions of UML (UML 1.x) were focused on creating models and blueprints of software systems. They provided a common language to represent software models, enabled effective communications between software architects, business analysts, end users, and development teams, and achieved their primary goal of standardizing software modeling. To facilitate standardization, the UML specifications distinguish between the terms concrete (notation) and abstract (UML specification) syntax.

Notation provides graphical representations and it is the concrete syntax, not the abstract syntax, of the modeling language.

The UML specification provides:

“A formal definition of a common MOF-based meta-model that specifies the abstract syntax of the UML. The abstract syntax defines the set of UML modeling concepts, their attributes and their relationships, as well as the rules for combining these concepts to construct partial or complete UML models.”

UML is one of the primary specifications that supports Model Driven Architecture (MDA), which has also been standardized by the OMG. MDA has Platform-Independent Models (PIMs) to represent business functionality without regard to the technical platforms, and one or more Platform-Specific Models (PSMs) that are specific to the target technology platform for the software being designed. UML can be used to represent both PIMs and PSMs.

Tools that support MDA allow modelers to transform PSMs into implementations, usually realized as code. Also, some tools can analyze existing source code and reverse engineer it into a set of UML diagrams. The ultimate aim of UML is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes. UML 2 builds upon UML 1 to fully support the model precision and detail needed to support round-trip engineering; i.e., generation of code from models and vice versa.

In UML 2, UML has been enhanced with significantly more precise definitions of its abstract syntax rules and semantics, a more modular language structure, and a greatly improved capability for modeling large-scale systems. As of this writing, the current version of UML is 2.4.1 and that specification is split into superstructure and infrastructure. UML 2.5 is in beta with a publicly available specification. The 2.5 simplification effort has eliminated the infrastructure specification. There is intentionally very little difference between the UML 2.4.1 and UML 2.5 meta-models.

The UML 2 superstructure specification divides the 14 diagram types into two key categories:

• Part I – Structural Diagrams: These diagrams are used to define static architecture. They consist of static
Using the ArchiMate® Language with UML®

Constructs such as classes, objects, and components, and the relationships between these elements. There are seven structural diagrams: Package Diagrams, Class Diagrams, Object Diagrams, Composite Structure Diagrams, Component Diagrams, Deployment Diagrams, and Profile Diagrams.

- Part II – Behavioral Diagrams: These diagrams are used to represent dynamic architecture. They comprise behavioral constructs such as activities, states, timelines, and the messages exchanged between different objects. These diagrams are used to represent the interactions among various model elements and instantaneous states over a time period. There are seven behavioral diagrams: Use-Case Diagrams, Activity Diagrams, State Machine Diagrams, Communication Diagrams, Sequence Diagrams, Timing Diagrams, and Interaction Overview Diagrams.

UML 2 has introduced Composite Structure, Object, Timing, and Interaction Overview Diagrams. The remaining diagrams were borrowed from UML 1.x, although some of them were changed significantly.

Current Usage and White Paper Positioning

Over time, UML usage has been extended to business modeling and modeling other non-software systems. Although UML is a natural fit for object-oriented languages and methodologies, it is used widely for non-object-oriented applications as well. Several UML profiles have evolved to provide extensions to UML and to customize UML models for various business domains such as health, finance, telecom, aerospace, as well as platforms such as JEE and .NET. Due to its versatility, acceptance, and maturity, UML can be used to model Enterprise Architectures as well. However, Figure 5 below compares the positioning of the ArchiMate language and UML for the purposes of this White Paper, which focuses on the use of the ArchiMate language for Enterprise Architecture and the use of UML for Solution Architecture.

Figure 5: Positioning of the ArchiMate Language and UML for the Purposes of this White Paper
Applying UML to Solution Architecture

An effective method to represent Solution Architecture using UML is to use the 4+1 View Model of architecture approach proposed by Philippe Kruchten [9]. Figure 6 shows how the UML 2 diagrams map to the views from the 4+1 View Model based on the White Paper Applying 4+1 View Architecture with UML 2 [10].

Figure 6: UML 2 Diagrams Mapped to the 4+1 View Model of Architecture

---

1 Process View: In practice the Process View is often used to represent business processes and workflow in addition to the concurrency and synchronization aspects described in Mr. Kruchten’s publication [9]. We adopted that convention.
Using UML with the ArchiMate Language

We distinguish two main categories of approach for the combined use of the two languages:

- Informal approaches, providing methodological support and guidelines for using the ArchiMate language with UML; this may include general descriptions of relationships between the languages
- Formal approaches, providing a precise definition of mapping between concepts from the two standards

Both informal and formal approaches may be supported by modeling tools. For example, a shared model repository may be used that allows for storing both ArchiMate language and UML models, and establishing global links between these models.

This White Paper presents a number of informal and formal approaches. The informal approaches are generally ready for use by modelers, although they are easier to use if they are supported by modeling tools. Formal methods, however, require further development and standardization before they can be unambiguously implemented in commercial modeling tools.

Informal Methods

Inter-Model Relationships

Figure 7 below sketches an example of informal relationships between an ArchiMate view and a number of UML diagrams. An element from the ArchiMate model may be related to a single element in the UML diagram, but also to, for example, a complete diagram. In the example, an ArchiMate node is linked to a node from a UML deployment diagram, an ArchiMate Application Component is linked to a component in a UML component diagram, and an ArchiMate Business Process is linked to a complete UML Activity Diagram. The relationships between these concepts in the ArchiMate language also suggest implied relationships between the elements in the UML diagrams, although the exact nature and details of these relationships are not precisely expressed.
Although these relationships are informal, the use of modeling tools is still very helpful and nearly mandatory to establish and manage them. If the different types of models are stored in a shared model repository, inter-model relationships may also be established within the repository, which (depending on the capabilities of the specific tools that are used) makes it possible to trace, manage, and visualize them.

**Copying versus Linking Between Models**

Visibility and traceability are key capabilities for combined use of multiple modeling languages. This is especially so since many environments typically use multiple modeling tools in support of different roles and usage patterns. One way of creating visibility and traceability is to use a single global repository. However, often this approach is not practical given diverse technologies, different user needs, and a lack of unified repository management. Other methods include copying, cloning, and linking content across repositories:

- Copying involves exchanging a model with another tool.
- Cloning involves taking a template such as a TOGAF Architecture Building Block (ABB) and using it as a starting point for another activity.
- Linking content is often the best way to express relationships between models and to preserve them as model content changes over time.
Using the ArchiMate® Language with UML®

The Open Services for Lifecycle Collaboration (OSLC) [18] industry initiative is providing standardized linking semantics designed to provide transparency, bi-directional visibility, version sensitivity, and robustness against change.

Now that we have examined general methods for relating ArchiMate and UML models, we can now examine some more specific examples of how the ArchiMate language and UML can be used together.

**Elaborating ArchiMate Views with UML Diagrams**

UML modelers can specify Solution Architectures by creating more detailed versions of ArchiMate views of an Enterprise Architecture. Elaboration of ArchiMate views with UML diagrams requires the modeler to understand what the ArchiMate view specifies about the Enterprise Architecture, and to develop the appropriate UML diagram to implement those specifications within the Solution Architecture. In this section, we illustrate how Solution Architecture views using standard UML 2 diagrams can be built based upon the Enterprise Architecture views created using standard ArchiMate 2.0 viewpoints.

The ArchiMate Core defines 18 different viewpoints. The ArchiMate Motivation extension defines six additional viewpoints, and the Implementation and Migration extension defines another three. As discussed before, a commonly accepted technique for representing Solution Architecture is the 4+1 View Model approach using UML diagrams. Table 1 illustrates how the Enterprise Architecture representations from ArchiMate views feed into Solution Architecture representations in the 4+1 View Model approach. Note that this table shows conceptual relationships between these views representing how the content from the Enterprise Architecture views is consumed while defining the Solution Architecture. Sometimes the Solution Architecture may need more views than the 5 views in the 4+1 View Model approach. In such cases these relationships will differ.

Table 1: Informal Relationships between an ArchiMate View and Several Different Types of UML Diagram

<table>
<thead>
<tr>
<th>ArchiMate Views by Viewpoint Name</th>
<th>4+1 View Model for Elaboration: Logical, Implementation, Process, Deployment, Use-Case</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L</td>
</tr>
<tr>
<td>Introductory, Layered</td>
<td>X</td>
</tr>
<tr>
<td>Organization, Actor Co-operation, Business Function</td>
<td></td>
</tr>
<tr>
<td>Business Process, Business Process Co-operation</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
</tr>
<tr>
<td>Application Behavior</td>
<td>X</td>
</tr>
<tr>
<td>Application Co-operation</td>
<td></td>
</tr>
<tr>
<td>Application Structure</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td></td>
</tr>
<tr>
<td>Infrastructure Usage, Implementation and Deployment</td>
<td></td>
</tr>
<tr>
<td>Information Structure, Landscape Map</td>
<td></td>
</tr>
<tr>
<td>Service Realization, Stakeholder, Goal Realization, Goal Contribution, Principles, Requirements Realization, Motivation</td>
<td></td>
</tr>
</tbody>
</table>
There are several ways in which the ArchiMate language and UML can be used together. We can relate an entire model in one language to a model, view, or even an element in another. Given that the focus of this White Paper is on using the ArchiMate language for Enterprise Architecture and UML for Solution Architecture, we will focus on cases where UML is used to elaborate ArchiMate views. So, there are three cases to explore. We can represent, elaborate, or realize an ArchiMate model, view, or element with one or more UML diagrams.

To apply these techniques it is critical to clearly trace between ArchiMate elements and corresponding UML elements. A few examples are listed below. A complete list of mappings between ArchiMate and UML elements is beyond the scope of this White Paper.

- An ArchiMate Business Role or Business Actor can be represented by a UML Actor. The concepts ArchiMate Business Role and UML Actor represent elements at the same level of detail.

- An ArchiMate Business Process can be elaborated by a UML Activity Diagram. The concept ArchiMate Business Process represents an existence of a business process but does not list all the activities and their flows. UML Activity Diagrams can be used to describe the activities, roles, and decisions required to elaborate an ArchiMate Business Process.

- An application represented as an ArchiMate Application Component can be realized\(^2\) by representing its design using UML Sequence and Class Diagrams. In this case, the requirements for an application may have been represented in the ArchiMate language using the Application Service, Application Component, and Application Interface concepts, while the detailed design is represented using the UML diagrams.

Every ArchiMate element is not necessarily mapped to an element in UML (or \textit{vice versa}). Certain ArchiMate elements can be elaborated or realized using the concepts within the ArchiMate language before they are mapped to UML diagrams. Here are some examples:

- An ArchiMate goal may be realized by ArchiMate requirements. An ArchiMate requirement can in turn be elaborated by UML use-cases. In this case, such UML use-cases are directly related to the ArchiMate requirements and indirectly related to the ArchiMate goals.

- An ArchiMate Business Service may be realized by an ArchiMate Application Component. The component’s design could be represented using UML class diagrams and sequence diagrams. In this case, the UML diagrams have an indirect relationship with the ArchiMate Business Service.

Modelers may often choose between refining model elements in the ArchiMate language or UML. Every ArchiMate concept can be decomposed to some extent within the ArchiMate language. Alternatively, as demonstrated above, those same ArchiMate concepts can be represented with or without additional detail in UML.

UML supports additional specificity that is not available in the ArchiMate language. Here are some examples of important modeling constructs in UML that the standard ArchiMate language does not support:

- UML Class Diagrams express attributes as well as cardinality (one-to-one, one-to-many, etc.) of relationships between instances of classes. They are also used to specify the behaviors that a classifier implements (i.e., operations). They can be used to elaborate ArchiMate Active Structure Elements such as Application Components, and information elements such as business and data objects.

- UML State Machine Diagrams express particular sets of dynamic attributes of objects and precisely how

---

\(^2\) Realization is a specialized relationship between two sets of model elements, one representing a specification (the supplier) and the other representing an implementation of the latter (the client). Realization is different from elaboration. While elaboration describes a concept with more detail (in this example, a business process is elaborated with a more detailed activity diagram), realization represents a logical entity with more concrete structures (in this example, an Application Component is designed using UML).
Using the ArchiMate® Language with UML®

those objects transition between those sets of attributes. They can be used to represent state transitions of
elements defined in ArchiMate models.

• UML Sequence Diagrams express object interactions and can be used to elaborate ArchiMate behavior
elements.

The Case Study provided later in this White Paper further explores these concepts.

Approaches to Implementing Formal Methods

In formal methods, relationships between ArchiMate concepts and UML concepts are precisely defined. One
such method involves the UML profiles package, which contains a mechanism for defining UML models for
particular domains and platforms.

Profiles are defined using stereotypes, tag definitions, and constraints that are applied to specific model
elements, such as classes, attributes, operations, and activities. Profile development is a specific meta-
modeling technique that can be used to customize UML for a particular domain (e.g., aerospace, healthcare,
financial) or platform (J2EE, .NET).

An initial proposal for such a profile was suggested in [8]. Figure 8 below shows an example of an
application of a profile based on the ArchiMate 2.0 standard. The figure expresses the ArchiMate Layered
view (Figure 12) used in the Case Study found later in this White Paper.
A second type of formal method involves the definition of a formal transformation; i.e., a unidirectional or bidirectional mapping, between the concepts in the two languages. This transformation could be done directly or using a standardized intermediary, such as the OMG Meta Object Facility (MOF) [13], which enables the exchange of models across different languages and tools. The OMG recommends the use of the XML
Using the ArchiMate® Language with UML®

Metadata Interchange (XMI) format [14] for exchanging models, and the MOF Query/View/Transformation (QVT) [19] specification for defining relationships used in model transformation. The UML Infrastructure Specification [15] provides a MOF meta-model for UML. Figure 9 shows how a software component could transform XMI models between the ArchiMate language and UML using a specification based on QVT for mapping between the two languages. However, a MOF meta-model for the ArchiMate language does not exist today. This presents an opportunity for standards developers, since a standardized transformation method bridging the two languages would set the stage for tool-facilitated transitions between Enterprise and Solution Architectures. This would be useful when Enterprise and Solution Architects cannot work directly together to facilitate these transitions, architectural models are too complex or extensive for manual transition, or when Solution Architectures are based on industry-standard Enterprise Architectures.

Figure 9: An ArchiMate Model Depicting how ArchiMate Models could be Transformed to UML and Vice Versa
Case Study

This section explores the elaboration and realization of ArchiMate views with UML diagrams using informal concepts discussed in this White Paper. The case study uses a fictitious scenario explained below. This scenario uses parenthetical expressions to indicate ArchiMate concepts.

Scenario

Joe, a construction worker injured at work (Business Role: Claimant), calls his disability insurance company. He speaks to a Contact Center Representative (Business Role), who connects him to Mary, an insurance company Claim Intake Specialist (Business Role). Joe works with Mary (Business Collaboration) to initiate a disability claim (Business Object). As Joe speaks with Mary, she uses her company's Claim Intake System (Application Component realizing an Application Service called Claim Initiation) to begin the intake process (Business Process). Mary enters basic information about Joe's claim.

Behind the scenes, the Claim Intake System interacts with the insurance company's new Customer Relationship Management (CRM) system (Application Interaction) using an integration service (Infrastructure Service). That service in turn uses the insurance company's Enterprise Service Bus (ESB) (System Software) to send a message to the CRM system. The next time Joe calls the Contact Center (Business Function), the representative that picks up the phone will be able to tell him the status of his claim. The ability to retrieve claim status using a caller's Claim Number or Social Security number is a key feature (Requirement) implemented last year by the insurance company IT organization in order to enable Contact Center Representatives to quickly answer the most common customer questions (Goal: Quick Customer Response).

Modeling the Scenario with ArchiMate Views

Figure 10 shows how the enterprise-level goals and requirements can be expressed with the ArchiMate Motivation extension along with some core language elements. Quick Customer Response is a goal that is realized by the requirements Paperless Claims and Retrieve Claim Status. This goal is associated with the actor Contact Center. The Claim Intake business service realizes the Paperless Claims requirement since it is designed to capture claims electronically. The Claims Inquiry business service realizes the requirements “Paperless Claims” and “Retrieve Claim Status”.
Using the ArchiMate® Language with UML®

Figure 10: An ArchiMate Goal Realization View that Shows Realization of Requirements by Business Services, and Realization of a Goal by those Same Requirements

Figure 11 shows the actors and roles mentioned in the scenario using the ArchiMate Actor Co-operation View. Contact Center Representative is a role assigned to an individual working in the Contact Center (Organization represented as a Business Actor). Claim Intake Contact is a collaboration formed by both Contact Center Representative and Claim Intake Specialist. This collaboration supports the Construction Worker (Business Actor) who is the Claimant (Role) for the disability claim. The ArchiMate actors and roles in this figure are intended to be used in the ArchiMate models only. Later we will show how these ArchiMate roles and actors map to UML Actors.

Figure 11: An ArchiMate Actor Co-operation View Shows Business Actors, Roles, and their Relationships

Figure 12 shows an ArchiMate Layered View. It summarizes an architecture using the Business, Application, and Technology Layers of the ArchiMate language. The Construction Worker (Business Actor) in the role of a Claimant uses the Claim Intake and Claim Inquiry business services. The Claim Intake Service involves the use of a Claim Intake Contact (a Business Collaboration). The Claim Inquiry Service requires a Contact Center Representative. The Claim Intake business process produces a Disability Claim (a Business Object).
Note that the detailed steps of this business process are not relevant at this time and will be handled as part of the Solution Architecture that we will discuss a little later.

The Claim Intake process makes use of the application services Create Claim and Create Case. The Create Claim Service is realized by the Claims Processor Application (represented as an Application Component). The CRM application on the other hand provides the Create Case and Get Case Status services. Note that the details of these applications are not completely known at this time. They are supposed to be hashed out in the Solution Architecture activity for the respective applications.

The Claims Processor application publishes the claims status updates that are received by the CRM system using the Integration service which is an infrastructure service that is used by different applications. This service is realized by an Enterprise Service Bus (ESB) which is system software in the Infrastructure Layer. The technology standards such as ESB that are common across the whole organization are decided in an Enterprise Architecture activity. You will see that such common services are typically depicted in ArchiMate models.

Figure 12: An ArchiMate Layered View Summarizes the Architecture Across the Business, Application, and Technology Layers
Using the ArchiMate® Language with UML®

Figure 13 shows the high-level business object model represented using an Information Structure Viewpoint. The focus here is the key entities and their relationships. This model does not get into the details such as attributes and cardinality of relationships. In this figure, a Case is a business object that aggregates a Claimant and an Incident. Claimant is a business object that represents the data representation of the Claimant role played by the construction worker. The event “Injury at Work” results in an Incident. Case also aggregates one or more claims. Each Claim is composed of claim lines. As you can see, the multiplicity is not shown on the relationships in this figure.

Figure 13: A Simple ArchiMate Information Structure View with an Associated Role Shows Relationships to and between Key Business Objects

Elaborating the ArchiMate Views with UML Diagrams

The above views and the models that they collectively represent demonstrate that ArchiMate models are high-level and just sufficient to the needs of enterprise-level architecture representations. In order to implement elements of this Enterprise Architecture, such as the claims processor application, the architects need to develop a Solution Architecture. The rest of this section shows how this ArchiMate model can be evolved into a Solution Architecture using UML. As the ArchiMate model for Enterprise Architecture is elaborated for Solution Architecture using UML, there is multi-level evolution of various views that result from the modeling effort. The UML models are categorized here into different views adopted from the 4+1 View Model architecture modeling approach for Solution Architecture.

Requirements and Use-Case Views

Figure 14 shows an Actor’s view using a UML use-case diagram. This diagram is derived from Figure 11, the ArchiMate Actor Co-operation View that models Business Actors and Roles. The ArchiMate Role concept directly maps to an Actor in UML. Also, ArchiMate Business Collaborations directly map to Actors
Using the ArchiMate® Language with UML®

in UML. Business modeling uses Business Actors and Business Use-Cases. In Figure 15 and Figure 16, the Claimant does not directly use the Claims Processor application, but participates in the related business process. Hence, Claimant is represented as a Business Actor, whereas the Contact Center Representative and Claim Intake Specialist who use the applications are represented as System Actors in UML.

Figure 14: A UML Actor’s View Represents Classifications of Scenario Participants

The Claim Intake business service and the interaction between the Claimant and the Insurance Company are represented in UML using a business use-case. This model is optional as the ArchiMate language already represents these interactions as business services; it is being shown here as an alternative representation using UML. Note that the names of the business use-cases are different from the names of the business services in the ArchiMate language due to a difference in the naming conventions. Figure 15 shows this simple business use-case model. Initiate Claim and Inquire Claim Status are two business use-cases that the Claimant initiates. Both the Claimant and the Contact Center Representative are involved in the Initiate Claim use-case, while the Contact Center Representative is involved in the Inquire Claim Status use-case. Per business modeling discipline, both Claimant and Contact Center Representative can be represented with the stereotype Business Worker. However, we represent them here as system actors as they will be involved in the System Use-Cases in the figures to follow.

3 Business modeling is a concept from the Rational Unified Process (RUP) and is a widely applied technique. It uses stereotypes such as business actors, business use-cases, and so forth. Business modeling has been traditionally used to represent Business Architecture. It fits very well in this example as we showcase transitioning from Enterprise Architecture to Solution Architecture.

4 In business modeling, the actors such as Claim Intake Specialist, Contact Center Representative, and Claim Intake Contact are represented using the internal worker stereotype. When they are used in the system use-case diagram, the (system) actors are created for corresponding business internal workers. In this figure, we have directly used the (system) actors and skipped representing them as internal workers for simplicity.
Using the ArchiMate® Language with UML®

In Figure 15, a UML Business Use-Case View describes how claimants interact with the Insurance Company when they use the Claim Intake business service first described in an ArchiMate view. In ArchiMate modeling, the business processes are represented at a high level that indicates their existence and key relationships. An ArchiMate Business Process can also be decomposed into multiple processes if there are sub-processes involved. But it stops there. There is no step-by-step representation of the details of a Business Process in ArchiMate. UML Activity Diagrams work well for this elaboration. Figure 16 uses a UML Activity Diagram to elaborate the Claim Intake Process from the ArchiMate model. The three actors involved in a business process are shown as swim lanes here to neatly categorize the activities performed by these actors. You will also notice that this business process involves some of the steps that are manual and some decision points are involved in the flow. As discussed earlier, compared to ArchiMate views, UML Activity Diagrams provide many more standard elements that can be used to represent complex business processes.
The ArchiMate view in Figure 12 shows the Claims Processor and CRM applications. UML can be used to define use-cases for these applications. The system use-cases are defined to document the functional requirements for applications. Figure 17 shows the use-case diagram for the Claims Processor Application. Note that this system is used by the internal workers such as Claim Intake Specialist, Claims Adjudicator, etc. It is not directly accessed by the Claimant (Business Actor). The scope of each use-case diagram is a single application, Claims Processor in this case. Other applications that the Claims Processor interacts with are shown as Actors in UML. Hence, the CRM application is an actor in the Claims Processor application's system use-case diagram; and Claims Processor is an actor in the CRM application's system use-case diagram. The Update Case association between the two applications (Figure 12) remains the same.

The use-cases in the system use-case view (Figure 17) should address all the application services provided by the Claims Processor application. The names may be different due to differences in the naming conventions between the service names in the ArchiMate language and use-case names in UML. As system use-case modeling is done during the Solution Architecture phase, which generally happens after the Enterprise Architecture is defined, analysts may identify additional use-cases beyond those corresponding to the application services defined using the ArchiMate language.

---

**Figure 16: A UML Activity Diagram Elaborates the Claim Intake Process First Described in an ArchiMate View**

<table>
<thead>
<tr>
<th>Claimant</th>
<th>Contact Center Representative</th>
<th>Claim Intake Specialist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Start</td>
<td>Start</td>
</tr>
<tr>
<td>Call Insurance Company to make a claim</td>
<td>Verify Credentials</td>
<td>Initiate a Disability Claim</td>
</tr>
<tr>
<td></td>
<td>Create a Case</td>
<td>Capture Claim Data</td>
</tr>
<tr>
<td></td>
<td>Update the claim details on the case</td>
<td>Submit Claim For Processing</td>
</tr>
<tr>
<td></td>
<td>Close the interaction with the user with a Case Reference</td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>Initiate another claim</td>
<td>Stop</td>
</tr>
</tbody>
</table>

The ArchiMate® Language with UML®
Using the ArchiMate® Language with UML®

Figure 17: A UML System Use-Case Diagram Shows how Different Actors Interact with the Claims Processor Application

**Logical View**

The business objects from the ArchiMate model are elaborated into a system-level class diagram in Figure 18, which represents the data entities and their relationships used by an application. The domain model in UML includes attributes, relationships between entities, and their cardinality as opposed to the ArchiMate business model in Figure 13. The UML class diagrams help modelers with detailed understanding of the structural, domain, and data aspects of the solution under development. This model is one of the key aspects of the Solution Architecture phase and it neatly builds upon the top of the business models created using the ArchiMate Information Structure Viewpoint. Although not shown here, the UML class diagrams are used for multiple purposes and the domain model is just a starting point. The database schemas and system object models are derived from the domain model. UML class diagrams are also used to represent entire system design classes involved in building each element of the component model in Figure 19 below.
Using the ArchiMate® Language with UML®

Figure 18: A UML Class Diagram Describes the Structure of the Information Processed by the Claim Processor Application

**Implementation View**

Figure 19 shows the major system components for the Claims Processor system represented using a Component Model in UML. The Application Components from the ArchiMate models such as Application Structure, Application Co-operation, and Application Usage Views are elaborated in this diagram. While each application was represented as a single unit in the ArchiMate view in Figure 12, here the UML diagram shows the applications broken down into multiple components. Each component in this diagram represents an actual physical Application Component that gets implemented and deployed. The actual interfaces with and dependencies on other components are demonstrated in this diagram.
Using the ArchiMate® Language with UML®

Figure 19: A UML Component Model Elaborates the Claims Processor Application Component First Described in an ArchiMate View

**Process View**

UML sequence diagrams show the exact sequence of message exchanges on a timeline between the classes involved in realizing a use-case. These diagrams form the Process View. They can be used in the Solution Architecture to represent the common design patterns applied in the solution. These diagrams can also be used to do detailed system design.

Each use-case from the system use-case diagram can be elaborated using one or more sequence diagrams for detailed design. They collectively form a specification used by the developers to implement the system. The completeness of the design can also be ensured by making sure that every use-case has a realization supported by sequence diagrams. Figure 20 shows one such sequence diagram. The complete set of sequence diagrams and class diagrams (from the Logical View), potentially with other UML diagrams not mentioned here, forms the system design for an application.
Using the ArchiMate® Language with UML®

Figure 20: A UML Sequence Diagram for the Enter Claim Use-Case Realization Details how the Claim Intake Specialist uses the Claims Processor Application.

5 Stereotypes for actor, boundary, control, and entity are based on robustness analysis technique from the Rational Unified Process (RUP). It is a commonly used practice to enhance the sequence diagrams during design.
Conclusion

Benefits of using the ArchiMate Language and UML Together

Most organizations have some basic capabilities for managing their Enterprise and Solution Architectures but they are rarely connected through visual modeling. In this White Paper, we have shown how these capabilities can interact in an organization through carefully co-ordinated use of the ArchiMate language and UML. Modeling tools can assist with this co-ordination.

The benefits of creating a strong interaction between the two languages are:

• A common communication medium and understanding for the pursuit of an initiative to develop organizational capabilities; this understanding may span an organization as well as its business partners
• Custom viewpoints for each stakeholder at various levels of detail that correspond to their own concerns
• Enhanced traceability, re-use, and integration across Enterprise and Solution Architectures for technically sophisticated stakeholders such as architects, designers, analysts, and developers

By modeling a realistic business scenario using the ArchiMate language for Enterprise Architecture and elaborating the ArchiMate model with UML to develop a Solution Architecture, this White Paper demonstrates how the two modeling languages can be used together. This White Paper has only used a small fraction of the capabilities of both languages, and many more relationships are possible. As Enterprise and Solution Architects and their analyst and designer colleagues develop modeling skills and working relationships, they can develop many more ways of using these languages together.

Opportunities for Further Work

This White Paper sets the stage for an additional White Paper proposing a formal mapping between the ArchiMate language and UML.
References


Using the ArchiMate® Language with UML®

[17] W3C Extensible Markup Language (XML) 1.0; refer to: www.w3.org/TR/xml.

[18] Open Services for Lifecycle Collaboration (OSLC); refer to: http://open-services.net.

About the Authors

**Chris Armstrong**, President of Armstrong Process Group, Inc. (APG), is an internationally recognized thought leader in Enterprise Architecture, formal modeling, process improvement, systems and software engineering, requirements management, and iterative and agile development. Chris represents APG at The Open Group, the Object Management Group, and the Eclipse Foundation, where he supports the ongoing development of TOGAF, ArchiMate, UML, SPEM, and EPF Composer.

**J.D. Baker**, Principal Systems Engineer with Armstrong Process Group, Inc., is a recognized expert in software and system architecture, systems engineering, software development, iterative/agile development, object-oriented analysis and design, the Unified Modeling Language (UML), the UML Profile for Systems Engineering (SysML), use-case-driven requirements, and process improvement. J.D. is a member of the Object Management Group Architecture Board and has participated in the development of UML, OMG SysML, the UML Profile for DoDAF and MODAF, and the Software and System Process Engineering Meta-model (SPM).

**Iver Band** is an Enterprise Architect at The Standard, a diversified financial services company based in Portland, Oregon, USA. At The Standard, he oversees a portfolio of insurance, asset management, CRM and contact center solutions, and guides strategic projects. He is both TOGAF and ArchiMate Certified, and applies these standards to his daily work. He is Vice Chair of The Open Group ArchiMate Forum, where he has contributed to the development of the ArchiMate standard. Iver has also co-developed the ArchiMate certification examinations, spoken regularly at Open Group conferences, and has led or contributed to the development of several other Open Group White Papers. Iver has been a security and infrastructure architect, systems engineer, software engineer, and IT director. He is a CISSP and a Certified Information Professional.

**Sam Courtney** is a Senior Sales Engineer with Armstrong Process Group, based in Dallas, TX, with 23 years of experience in the technology field. He specializes in Enterprise Architecture, Business Process Automation, and Application Life Cycle Management among other things. As an IBM Business Partner, they are active members of the IBM Enterprise Architecture and Requirements/Analysis Communities of Practice as well as contributing members to The Open Group and Object Management Group. His highly-focused experience lies in successfully developing strategies for implementation of software and technologies for automation of business processes and software development lifecycles.

**Henk Jonkers** is a Senior Research Consultant, involved in BiZZdesign’s innovations in the areas of Enterprise Architecture and engineering. He participates in multi-party research projects, contributes to training courses, and performs consultancy assignments. Previously, as a member of scientific staff at an applied IT research institute, he was involved in research projects on business process modeling and analysis, Enterprise Architecture, service-oriented architecture, and model-driven development. He was one of the main developers of the ArchiMate language and an author of the ArchiMate 1.0 and 2.0 Specifications, and is actively involved in the activities of the ArchiMate Forum of The Open Group.
Using the ArchiMate® Language with UML®

Veer Muchandi is an Enterprise Architect at Hewlett-Packard. With over 21 years of IT consulting experience he specializes in the fields of Enterprise Architecture, Solution Architecture, Software Development Lifecycle, and Application Development. He is a Master Open CA, TOGAF 9 Certified Practitioner, ArchiMate 2 Certified, and OMG Certified UML Professional. Veer is an active participant in The Open Group Architecture Forum and ArchiMate Forum.

Martin Owen is the CEO and managing partner for strategy and product development at Corso. Prior to Corso, Martin led the product management team at IBM Rational for Enterprise Architecture and Asset Management. He has formerly held roles as Vice-President, Enterprise Architecture at Telelogic, an IBM Company and Consulting Director for Popkin Software. Martin was one of the original authors of the Business Process Modeling Notation (BPMN), now the de facto standard for business process modeling, and continues to work with industry standards bodies.
About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 400 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

• Capture, understand, and address current and emerging requirements, and establish policies and share best practices
• Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
• Offer a comprehensive set of services to enhance the operational efficiency of consortia
• Operate the industry’s premier certification service

Further information on The Open Group is available at www.opengroup.org.