

ArchiMate[®] 1.0 Specification



ArchiMate® 1.0 Specification

Other publications by Van Haren Publishing

Van Haren Publishing (VHP) specializes in titles on Best Practices, methods and standards within four domains:

- IT management,
- Architecture (Enterprise and IT),
- Business management and
- Project management

These publications are grouped in series: *ITSM Library, Best Practice and IT Management Topics*. VHP is also publisher on behalf of leading companies and institutions: The Open Group, IPMA-NL, PMI-NL, CA, Getronics, Pink Elephant.

Topics are (per domain):

IT (Service) Management / IT Governance

ASL
BiSL
CATS
CMMI
COBIT
ISO 17799
ISO 27001
ISO/IEC 20000
ISPL
IT Service CMM
ITIL® V2
ITIL® V3
ITSM
MOF
MSF

Architecture (Enterprise and IT)

Archimate
GEA
TOGAF™

Business Management

EFQM
ISA-95
ISO 9000
ISO 9001:2000
SixSigma
SOX
SqEME®

Project/Programme/ Risk Management

A4 Project management
ICB / NCB
MINCE®
M_o_R®
MSP
PMBok
PRINCE2®

For the latest information on VHP publications, visit our website: www.vanharen.net.

ArchiMate® 1.0 Specification

THE *Open* GROUP
www.opengroup.org



Colofon

Title: **ArchiMate® 1.0 Specification**

A publication of: The Open Group

Publisher: Van Haren Publishing, Zaltbommel, www.vanharen.net

ISBN: 978 90 8753 502 5

Print: First edition, first impression, April 2009

Layout and design: CO2 Premedia, Amersfoort-NL

Copyright: © 2009, The Open Group

For any further enquiries about Van Haren Publishing, please send an e-mail to: info@vanharen.net

Trademarks

TOGAF™ is a trademark and Archimate® and The Open Group® are registered trademarks of The Open Group in the United States and other countries. All other brand, company, and product names are used for identification purposes only and may be trademarks that are the sole property of their respective owners.

Comments relating to the material contained in this document may be submitted by email to: ogspecs@opengroup.org

Although this publication has been composed with most care, neither Author nor Editor nor Publisher can accept any liability for damage caused by possible errors and/or incompleteness in this publication.

No part of this publication may be reproduced in any form by print, photo print, microfilm or any other means without written permission by the publisher.

Contents

Chapter 1 Introduction	1
1.1 Intended Audience	2
1.2 Structure	3
Chapter 2 Enterprise Architecture.....	5
2.1 Why Enterprise Architecture?	5
2.2 Definitions.....	6
2.3 ArchiMate and TOGAF	7
Chapter 3 Language Structure	11
3.1 Design Approach.....	11
3.2 Core Concepts	12
3.3 Collaboration and Interaction.....	13
3.4 Relationships.....	14
3.5 Layering.....	14
3.6 The ArchiMate Framework.....	15
Chapter 4 Business Layer	17
4.1 Business Layer Metamodel	17
4.2 Structural Concepts	18
4.2.1 Business Actor.....	19
4.2.2 Business Role.....	19
4.2.3 Business Collaboration	21
4.2.4 Business Interface	22
4.2.5 Business Object.....	23
4.3 Behavioral Concepts	24
4.3.1 Business Process	26
4.3.2 Business Function.....	27
4.3.3 Business Interaction	29
4.3.4 Business Event.....	31
4.3.5 Business Service.....	32
4.4 Informational Concepts	34
4.4.1 Representation	35
4.4.2 Meaning.....	36
4.4.3 Value.....	38

4.4.4	Product.....	39
4.4.5	Contract.....	40
4.5	Summary of Business Layer Concepts	41
Chapter 5 Application Layer.....		45
5.1	Application Layer Metamodel	45
5.2	Structural Concepts	45
5.2.1	Application Component.....	46
5.2.2	Application Collaboration.....	47
5.2.3	Application Interface.....	48
5.2.4	Data Object	50
5.3	Behavioral Concepts.....	51
5.3.1	Application Function	51
5.3.2	Application Interaction.....	53
5.3.3	Application Service	54
5.4	Summary of Application Layer Components.....	55
Chapter 6 Technology Layer		57
6.1	Technology Layer Metamodel.....	57
6.2	Structural Concepts	57
6.2.1	Node.....	58
6.2.2	Device.....	59
6.2.3	Infrastructure Interface.....	60
6.2.4	Network	61
6.2.5	Communication Path.....	62
6.3	Behavioral Concepts.....	63
6.3.1	Infrastructure Service	63
6.3.2	System Software	65
6.4	Informational Concepts	66
6.4.1	Artifact.....	66
6.5	Summary of Technology Layer Concepts.....	67
Chapter 7 Cross-Layer Dependencies		69
7.1	Business-Application Alignment	69
7.2	Application-Technology Alignment	70

Chapter 8 Relationships	73
8.1 Structural Relationships	73
8.1.1 Composition Relationship.....	73
8.1.2 Aggregation Relationship	74
8.1.3 Assignment Relationship.....	75
8.1.4 Realization Relationship	76
8.1.5 Used By Relationship	77
8.1.6 Access Relationship	77
8.1.7 Association Relationship	78
8.2 Dynamic Relationships.....	79
8.2.1 Triggering Relationship	79
8.2.2 Flow Relationship	80
8.3 Other Relationships	81
8.3.1 Grouping.....	81
8.3.2 Junction.....	82
8.3.3 Specialization Relationship	82
8.4 Summary of Relationships.....	83
8.5 Derived Relationships.....	84
Chapter 9 Architecture Viewpoints	87
9.1 Introduction.....	87
9.2 Views, Viewpoints, and Stakeholders.....	89
9.3 Viewpoint Classification	90
9.4 Basic Viewpoints in ArchiMate	93
9.4.1 Introductory Viewpoint.....	94
9.4.2 Organization Viewpoint	96
9.4.3 Actor Co-operation Viewpoint.....	97
9.4.4 Business Function Viewpoint	99
9.4.5 Business Process Viewpoint	100
9.4.6 Business Process Co-operation Viewpoint	101
9.4.7 Product Viewpoint	103
9.4.8 Application Behavior Viewpoint	104
9.4.9 Application Co-operation Viewpoint	105
9.4.10 Application Structure Viewpoint.....	106
9.4.11 Application Usage Viewpoint.....	108
9.4.12 Infrastructure Viewpoint.....	109

9.4.13	Infrastructure Usage Viewpoint	110
9.4.14	Implementation and Deployment Viewpoint	112
9.4.15	Information Structure Viewpoint.....	113
9.4.16	Service Realization Viewpoint	114
9.4.17	Layered Viewpoint.....	116
9.4.18	Landscape Map Viewpoint	118
Chapter 10 Language Extension Mechanisms.....		121
10.1	Adding Attributes to ArchiMate Concepts and Relations	121
10.2	Specialization of Concepts.....	123
Chapter 11 Future Directions		125
11.1	Extending and Refining the Concepts.....	125
11.1.1	Strategy, Goals, Principles, and Requirements.....	125
11.1.2	Evolution and Realization	126
11.1.3	Design Process	127
11.1.4	Architecture-Level Predictions.....	127
11.1.5	Other Improvements.....	127
11.2	Linking to Other Modeling Languages and Frameworks.....	128
11.3	How to Proceed	128
A Summary of Language Notation		131
B Overview of Relationships.....		133
Index		139

Preface

The Open Group

The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow™ will enable access to integrated information within and between enterprises based on open standards and global interoperability. The Open Group works with customers, suppliers, consortia, and other standards bodies. Its role is to capture, understand, and address current and emerging requirements, establish policies, and share best practices; to facilitate interoperability, develop consensus, and evolve and integrate specifications and Open Source technologies; to offer a comprehensive set of services to enhance the operational efficiency of consortia; and to operate the industry's premier certification service, including UNIX® certification.

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

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification.

More information is available at www.opengroup.org/certification.

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/bookstore.

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards-compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.

- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Readers should note that updates – in the form of Corrigenda – may apply to any publication. This information is published at www.opengroup.org/corrigenda.

This Document

This document is the ArchiMate 1.0 Specification, an Open Group Technical Standard.

Trademarks

Boundaryless Information Flow™ and TOGAF™ are trademarks and Making Standards Work®, The Open Group®, UNIX®, and the “X” device are registered trademarks of The Open Group in the United States and other countries.

ArchiMate® is a registered trademark of The Open Group.

Java™ is a trademark of Sun Microsystems, Inc. in the United States and other countries.

MDA®, Model Driven Architecture®, OMG®, and UML® are registered trademarks and BPMN™, Business Process Modeling Notation™, MOF™, and Unified Modeling Language™ are trademarks of the Object Management Group.

Telelogic™ is a trademark of Telelogic AB.

The Open Group acknowledges that there may be other brand, company, and product names used in this document that may be covered by trademark protection and advises the reader to verify them independently.

Acknowledgements

The Open Group gratefully acknowledges the contribution of the following people in the development of this Technical Standard:

- Maria-Eugenia Iacob, University of Twente
- Henk Jonkers, BiZZdesign BV
- Marc M. Lankhorst, Telematica Instituut¹
- Erik Proper, Radboud University Nijmegen & Capgemini

The results presented in this Technical Standard have largely been produced during the ArchiMate project, and The Open Group gratefully acknowledges the contribution of the many people – former members of the project team – who have contributed to them.

The ArchiMate project comprised the following organizations:

- ABN AMRO
- Centrum voor Wiskunde en Informatica
- Dutch Tax and Customs Administration
- Leiden Institute of Advanced Computer Science
- Ordina
- Radboud Universiteit Nijmegen
- Stichting Pensioenfonds ABP
- Telematica Instituut¹

The Open Group and ArchiMate project team would like to thank in particular the following individuals for their support and contribution to this Technical Standard:

- The Board members of the ArchiMate Foundation
- Mary Beijleveld, UWV
- Adrian Campbell, Ingenia Consulting
- Jos van Hillegersberg, University of Twente
- Andrew Josey, The Open Group
- Louw Labuschagne, Real IRM
- Daniel Moody, University of Twente
- Henk Volbeda, Sogeti
- Egon Willemsz, UWV

¹ From April 2009, Telematica Instituut is called NOVAY

Referenced Documents

The following documents are referenced in this Technical Standard:

- [1] Enterprise Architecture as Strategy, J.W. Ross, P. Well, D.C. Robertson, Harvard Business School Press, 2006.
- [2] ISO/IEC 42010:2007, Systems and Software Engineering – Recommended Practice for Architectural Description of Software-Intensive Systems, Edition 1.
- [3] Enterprise Architecture at Work: Modeling, Communication, and Analysis, M.M. Lankhorst et al, Springer, 2005.
- [4] The Open Group Architecture Framework TOGAF, Version 9, 2009.
- [5] A Framework for Information Systems Architecture, J.A. Zachman, IBM Systems Journal, Volume 26, No. 3, pp. 276–292, 1987.
- [6] ISO/IEC JTC 1/SC 7, Information Technology – Open Distributed Processing – Reference Model – Enterprise Language, October 2006.
- [7] ITU Recommendation X.901 | ISO/IEC 10746-1:1998, Information Technology – Open Distributed Processing – Reference Model – Part 1: Overview, International Telecommunication Union, 1996.
- [8] Unified Modeling Language: Infrastructure, Version 2.0 (formal/05-05-05), Object Management Group, March 2006.
- [9] Extending and Formalizing the Framework for Information Systems Architecture, J.F. Sowa, J.A. Zachman,, IBM Systems Journal, Volume 31, No. 3, pp. 590-616, 1992.
- [10] Enterprise Ontology: Theory and Methodology, J.L.G. Dietz, Springer, 2006.
- [11] Magic Quadrant for Enterprise Architecture Tools IQ06G, A. James, R.A. Handler, Gartner Research Report G00138197, 2006.
- [12] Unified Modeling Language: Superstructure, Version 2.0 (formal/05-07-04), Object Management Group, August 2005.
- [13] A Business Process Design Language, H. Eertink, W. Janssen, P. Oude Luttighuis, W. Teeuw, C. Vissers, in Proceedings of the First World Congress on Formal Methods, Toulouse, France, September 1999.
- [14] Enterprise Business Architecture: The Formal Link between Strategy and Results, R. Whittle, C.B. Myrick, CRC Press, 2004.
- [15] Composition of Relations in Enterprise Architecture, R.v. Buuren, H. Jonkers, M.E. Iacob, P. Strating, in Proceedings of the Second International Conference on Graph Transformation, pp. 39–53, Edited by H. Ehrig et al, Rome, Italy, 2004.

- [16] Viewpoints: A Framework for Integrating Multiple Perspectives in System Development, A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, M. Goedicke, in *International Journal on Software Engineering and Knowledge Engineering*, Volume 2, No. 1, pp. 31–58, 1992.
- [17] Viewpoints for Requirements Definition, G. Kotonya, I. Sommerville, *IEEE/BCS Software Engineering Journal*, Volume 7, No. 6, pp. 375–387, November 1992.
- [18] *Paradigm Shift – The New Promise of Information Technology*, D. Tapscott, A. Caston, New York: McGraw-Hill, 1993.
- [19] The 4+1 View Model of Architecture, P.B. Kruchten, *IEEE Software*, Volume 12, No. 6, pp. 42–50, 1995.
- [20] *Model-Driven Architecture: Applying MDA to Enterprise Computing*, D. Frankel, Wiley, 2003.
- [21] Performance and Cost Analysis of Service-Oriented Enterprise Architectures, H. Jonkers, M. E. Iacob, in *Global Implications of Modern Enterprise Information Systems: Technologies and Applications*, Edited by A. Gunasekaran, IGI Global, 2009.
- [22] A Model-Driven Approach for the Rule-Based Specification of Services, M.E. Iacob, H. Jonkers, in *Proceedings of the 12th IEEE International EDOC Conference*, Munich, Germany, September 2008.
- [23] Petri Nets: Properties, Analysis, and Applications, T. Murata, in *Proceedings of the IEEE*, Volume 77, No. 4, pp. 541–580, April 1989.
- [24] *Business Dynamics: Systems Thinking and Modeling for a Complex World*, J.D. Sterman, McGraw-Hill, 2000.
- [25] *Operations Research: An Introduction*, A.H. Taha, Prentice-Hall, 2006.
- [26] Business Process Modeling Notation Specification (dtc/06-02-01), Object Management Group, February 2006.
- [27] Business Motivation Model (BMM) Specification (dtc/2006-08-03), Object Management Group, August 2006.
- [28] Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0 (formal/08-01-02), Object Management Group, January 2008.
- [29] Business Process Definition Metamodel (BPDM) (bmi/2007-03-01), Object Management Group, March 2007.
- [30] Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering, E. Yu, in *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pp. 226–235, Washington, DC, January 1997.

- [31] E3-Value: Design and Evaluation of e-Business Models, J. Gordijn, H. Akkermans, IEEE Intelligent Systems, Volume 16, No. 4, pp. 11–17, 2001.
- [32] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification (formal/2008-04-03), Object Management Group, April 2008.
- [33] DoD Architecture Framework (DoDAF), Version 1.5, Volume II: Product Descriptions, US Department of Defense, April 2007.
- [34] FEA Consolidated Reference Model, Version 2.3, US Office of Management and Budget, 2007.
- [35] CBDI-SAE Meta Model for SOA Version 2.0, J. Dodd et al, Everware-DBDI, Inc., 2007.

Chapter 1

Introduction

An architecture is typically developed because key people have concerns that need to be addressed by the business and IT systems within the organization. Such people are commonly referred to as the “stakeholders” in the system. The role of the architect is to address these concerns, by identifying and refining the requirements that the stakeholders have, developing views of the architecture that show how the concerns and the requirements are going to be addressed, and by showing the trade-offs that are going to be made in reconciling the potentially conflicting concerns of different stakeholders. Without the architecture, it is unlikely that all the concerns and requirements will be considered and met.

Architecture descriptions are formal descriptions of an information system, organized in a way that supports reasoning about the structural and behavioral properties of the system and its evolution. They define the components or building blocks that make up the overall information system, and provide a plan from which products can be procured, and subsystems developed, that will work together to implement the overall system. It thus enables you to manage your overall IT investment in a way that meets the needs of your business.

To provide a uniform representation for such architecture descriptions, the ArchiMate enterprise architecture modeling language has been developed. It offers an integrated architectural approach that describes and visualizes the different architecture domains and their underlying relations and dependencies. In a short time, ArchiMate has become the open standard for architecture modeling in the Netherlands, it is also fairly well known in the international enterprise architecture community, and recently it has been brought under the aegis of The Open Group.

ArchiMate is a lightweight and scalable language in several respects:

- Its architecture framework is simple but comprehensive enough to provide a good structuring mechanism for architecture domains, layers, and aspects.

- The language incorporates modern ideas of the “service orientation” paradigm that promotes a new organizing principle in terms of (business, application, and infrastructure) services for organizations, with far-reaching consequences for their enterprise architecture.
- Although it intentionally resembles the Unified Modeling Language (UML), the ArchiMate modeling notation is intuitive and much lighter than currently proposed by UML 2.0. Nevertheless, the language is expressive enough to allow for the modeling of all layers (business, application, and technology infrastructure) and all aspects (structure, behavior, and information) of an organization in an integrated way.
- The two enterprise architecture standards of The Open Group – TOGAF and ArchiMate – complement each other and can be used well in combination.
- Finally, tool support for the ArchiMate language is already commercially available (from BiZZdesign, IDS Scheer, Casewise, Telelogic, and others).

The goal of this Technical Standard is to provide the first official and complete specification of the ArchiMate standard under the flag of The Open Group.

This specification contains the formal definition of ArchiMate as a visual design language with adequate concepts for specifying inter-related architectures, and specific viewpoints for selected stakeholders. This is complemented by some considerations regarding language extension mechanisms, analysis, and methodological support. Furthermore, this document is accompanied by a separate document, in which certification and governance procedures surrounding the specification are specified.

1.1 Intended Audience

The intended audience of this Technical Standard is threefold:

- Enterprise architecture practitioners, such as architects (application, information, process, infrastructure, products/services, and, obviously, enterprise architects), senior and operational management, project leaders, and anyone committed to work within the reference framework defined by the enterprise architecture. It is assumed that the reader has a certain skill level and is effectively committed to enterprise architecture. Such a person is most likely to be the architect – that is, someone who has affinity

with modeling techniques, knows his way around the organization, and is familiar with information technology.

- Those who intend to implement ArchiMate in a software tool. They will find a complete and detailed description of the language in this document.
- The academic community, on which we rely for amending and improving the language based on state-of-the-art research results in the architecture field.

1.2 Structure

The structure of this Technical Standard is as follows:

- Chapter 1, Introduction (this chapter)
- Chapter 2, Enterprise Architecture, makes the case for enterprise architecture and for the necessity of a modeling standard for enterprise architecture.
- Chapter 3, Language Structure, presents some general ideas, principles, and assumptions underlying the development of the ArchiMate metamodel and introduces the ArchiMate framework.
- Chapter 4, Business Layer, covers the definition and usage of the business layer concept, together with examples.
- Chapter 5, Application Layer, covers the definition and usage of the application layer concept, together with examples.
- Chapter 6, Technology Layer, covers the definition and usage of the technical infrastructure layer concept, together with examples.
- Chapters 7, Cross-Layer Dependencies, and Chapter 8, Relationships, cover the definition of relationship concepts in a similar way.
- Chapter 9, Architecture Viewpoints, presents and clarifies a set of architecture viewpoints, developed in ArchiMate based on practical experience. All ArchiMate viewpoints are described in detail. For each viewpoint the comprised concepts and relations, the guidelines for the viewpoint use, and the goal and target group and of the viewpoint are specified. Furthermore, each viewpoint description contains example models.
- Chapter 10, Language Extension Mechanisms, handles about extending and/or specializing the ArchiMate core language for specialized or domain-specific purposes.
- Chapter 11, Future Directions, identifies extensions and directions for developments in the next versions of the language.

Enterprise Architecture

2.1 Why Enterprise Architecture?

The primary reason for developing an enterprise architecture is to support the business by providing the fundamental technology and process structure for an IT strategy. Further, it details the structure and relationships of the enterprise, its business models, the way an organization will work, and how and in what way information, information systems, and technology will support the organization's business objectives and goals. This makes IT a responsive asset for a successful modern business strategy.

Today's CEOs know that the effective management and exploitation of information through IT is the key to business success, and the indispensable means to achieving competitive advantage. An enterprise architecture addresses this need, by providing a strategic context for the evolution of the IT system in response to the constantly changing needs of the business environment.

Furthermore, a good enterprise architecture enables you to achieve the right balance between IT efficiency and business innovation; in essence, it aligns IT with the business. It allows individual business units to innovate safely in their pursuit of competitive advantage. At the same time, it assures the needs of the organization for an integrated IT strategy, permitting the closest possible synergy across the extended enterprise.

The technical advantages that result from a good enterprise architecture bring important business benefits, which are clearly visible in the bottom line:

- A more efficient IT operation:
 - Lower software development, support, and maintenance costs
 - Increased portability of applications
 - Improved interoperability and easier system and network management
 - Improved ability to address critical enterprise-wide issues like security
 - Easier upgrade and exchange of system components

- Better return on existing investment, reduced risk for future investment:
 - Reduced complexity in IT infrastructure
 - Maximum return on investment in existing IT infrastructure
 - The flexibility to make, buy, or outsource IT solutions
 - Reduced risk overall in new investment, and the cost of IT ownership
- Faster, simpler, and cheaper procurement:
 - Buying decisions are simpler, because the information governing procurement is readily available in a coherent plan
 - The procurement process is faster – maximizing procurement speed and flexibility without sacrificing architectural coherence

Using an architecture framework will speed up and simplify architecture development, and communication with non-architects, ensuring more complete coverage and understanding of the designed solution. The additional understanding across the enterprise enables faster response to changing business needs.

2.2 Definitions

A good definition of *enterprise* in the context of this Technical Standard is any collection of organizations that has a common set of goals and/or a single bottom line. In that sense, an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership.

The term “enterprise” in the context of “enterprise architecture” can be used to denote both an entire enterprise, encompassing all of its information systems, and a specific domain within the enterprise. In both cases, the architecture crosses multiple systems, and multiple functional groups within the enterprise.

The definition of an *architecture* used in ISO/IEC 42010:2007 [2] is:

“The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.”

As in TOGAF, ArchiMate embraces but does not strictly adhere to ISO/IEC 42010:2007 terminology [2]. We use “architecture” in two different meanings, depending on its contextual usage:

1. A formal description of a system, or a detailed plan of the system at 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.

We endeavor to strike a balance between promoting the concepts and terminology of ISO/IEC 42010:2007 – ensuring that our usage of terms is consistent with the standard – and retaining other commonly accepted terminology that is familiar to the majority of the ArchiMate and TOGAF readership.

An *architecture description* is a formal description of an information system, organized in a way that supports reasoning about the structural properties of the system. It defines the components or building blocks that make up the overall information system, and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. It thus enables you to manage your overall IT investment in a way that meets the needs of your business.

An *architecture framework* is a tool which can be used for developing a broad range of different architectures. It should describe a method for designing an information system in terms of a set of building blocks, and for showing how the building blocks fit together. It should contain a set of tools and provide a common vocabulary. It should also include a list of recommended standards and compliant products that can be used to implement the building blocks.

2.3 ArchiMate and TOGAF

TOGAF is an architecture framework – a set of methods and tools for developing a broad range of different IT architectures. It enables IT users to design, evaluate, and build the right architecture for their organization, and reduces the costs of planning, designing, and implementing architectures based on open systems solutions.

The key to TOGAF is the Architecture Development Method (ADM) – a reliable, proven method for developing an IT enterprise architecture that meets the needs of your business. The TOGAF framework considers an overall enterprise architecture as composed of a set of closely inter-related architectures: Business Architecture, Information Systems Architecture (comprising Data Architecture and Application Architecture), and Technology (IT) Architecture. The ADM consists of a stepwise cyclic iterative approach for the development of the overall enterprise architecture.

The ArchiMate language, as described in this Technical Standard, complements TOGAF in that it provides a vendor-independent set of concepts, including a graphical representation, that helps to create a consistent, integrated model “below the waterline”, which can be depicted in the form of TOGAF’s views.

The structure of the ArchiMate language neatly corresponds with the three main architectures as addressed in the TOGAF ADM. This is illustrated in Figure 1. This correspondence would suggest a fairly easy mapping between TOGAF’s views and the ArchiMate viewpoints.

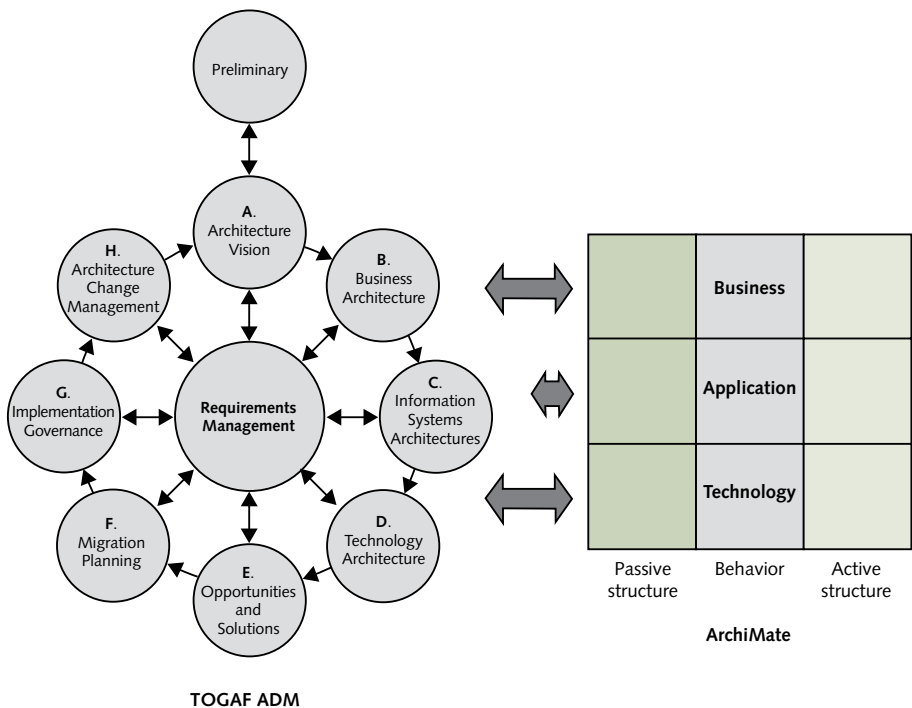


Figure 1: Correspondence between ArchiMate and TOGAF

Some TOGAF views are not matched in ArchiMate, however. Partially, this is because TOGAF's scope is broader and in particular addresses more of the high-level strategic issues and the lower-level engineering aspects of system development, whereas ArchiMate is limited to the enterprise architecture level of abstraction and refers to other techniques both for strategies, principles, and objectives, and for more detailed, implementation-oriented aspects (however, Chapter 11 gives some suggestions for possible extensions of the ArchiMate language in these areas). Secondly, although some of the TOGAF views cannot easily be mapped onto ArchiMate viewpoints, the ArchiMate language and its analysis techniques do support the concepts addressed in these viewpoints. Conversely, ArchiMate viewpoints that deal with the relationships between architectural layers, such as the product and application usage viewpoints, are difficult to map onto TOGAF's structure, in which views are confined to a single architectural layer.

Although there is no one-to-one mapping between them, there is still a fair amount of correspondence between the ArchiMate viewpoints and the views that are defined in TOGAF. Although corresponding viewpoints from ArchiMate and TOGAF do not necessarily have identical coverage, we can see that many viewpoints from both methods address largely the same issues. TOGAF and ArchiMate can easily be used in conjunction and they appear to cover much of the same ground, be it with some differences in scope and approach.

Chapter 3

Language Structure

The unambiguous specification and description of enterprise architecture's components and especially of their relationships requires an architecture modeling language that addresses the issue of consistent alignment and facilitates a coherent modeling of enterprise architectures.

This chapter presents the construction of the ArchiMate architecture modeling language. The precise definition and illustration of its generic set of concepts follow in Chapters 4, 5, 6, 7, and 8. They provide a proper basis for visualization, analysis, tooling, and use of these concepts.

Sections 3.1 through 3.5 discuss some general ideas, principles, and assumptions underlying the development of the ArchiMate metamodel. Section 3.6 presents the ArchiMate framework, which will be used in the remainder of this document as a reference taxonomy scheme for architecture concepts, models, viewpoints, and views.

3.1 Design Approach

A key challenge in the development of a general metamodel for enterprise architecture is to strike a balance between the specificity of languages for individual architecture domains, and a very general set of architecture concepts, which reflects a view of systems as a mere set of inter-related entities. Figure 2 illustrates that concepts can be described at different levels of specialization.

At the base of the triangle we find the metamodels of the architecture modeling concepts used by specific organizations, as well as a variety of existing modeling languages and standards; UML is an example of a language in this category. At the top of the triangle we find the “most general” metamodel for system architectures, essentially a metamodel that merely comprises notions such as “object”, “component”, and “relation”.

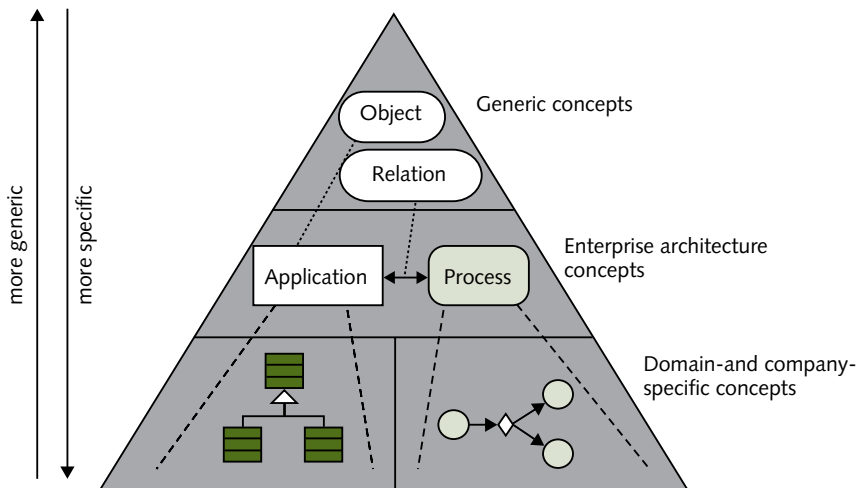


Figure 2: Metamodels at Different Levels of Specificity

The design of the ArchiMate language started from a set of relatively generic concepts (higher up in the pyramid). These were then specialized towards application at different architectural layers, as will be explained below.

The most important design restriction on the language is that it has been explicitly designed to be as small as possible, but still usable for most enterprise architecture modeling tasks. Many other languages, such as UML 2.0, try to accommodate all needs of all possible users. In the interest of simplicity of learning and use, ArchiMate has been limited to the concepts that suffice for modeling the proverbial 80% of practical cases.

3.2 Core Concepts

The language consists of *active structure* elements, *behavioral* elements and *passive structure* elements. The active structure elements are the business actors, application components and devices that display actual behavior, i.e., the ‘subjects’ of activity (right side of the Figure 3). Then there is the behavioral or dynamic aspect (center of Figure 3). The active structure concepts are assigned to behavioral concepts, to show who or what performs the behavior.

The passive structure elements are the objects on which behavior is performed. In the domain of information-intensive organizations, which is the main focus of the language, these are usually information or data objects,

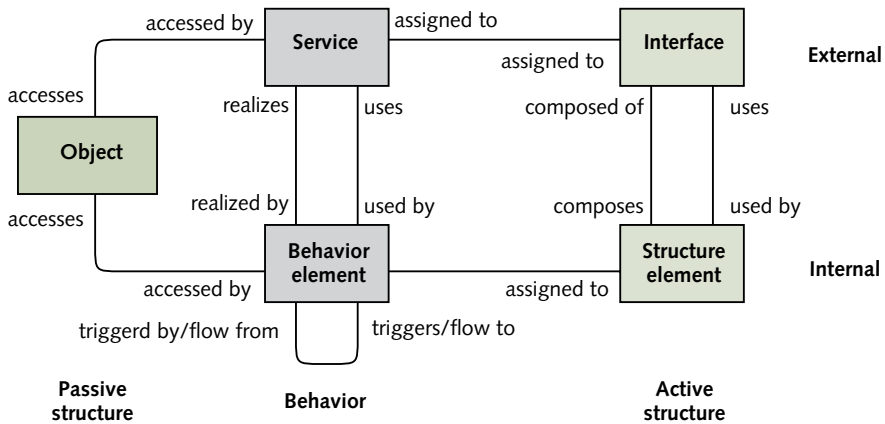


Figure 3: Generic Metamodel: The Core Concepts of ArchiMate

but they may also be used to represent physical objects. These three aspects – active structure, behavior, and passive structure – have been inspired by natural language, where a sentence has a subject (active structure), a verb (behavior), and an object (passive structure).

Second, we make a distinction between an external view and an internal view on systems. When looking at the behavioral aspect, these views reflect the principles of service orientation. The service concept represents a unit of essential functionality that a system exposes to its environment, and it provides a certain value (monetary or otherwise), which thus provides the motivation for the service's existence. For the external users, only this external functionality and value, together with non-functional aspects such as the quality of service, costs, etc., are relevant. These can be specified in a contract or Service Level Agreement (SLA). Services are accessible through interfaces, which constitute the external view on the active structural aspect.

3.3 Collaboration and Interaction

Going one level deeper in the structure of the language, we distinguish between behavior that is performed by a *single* structure element (e.g., actor, role component, etc.), or collective behavior (interaction) that is performed by a collaboration of multiple structure elements.

A collaboration is a (temporary) grouping (or aggregation) of two or more structure elements, working together to perform some collective behavior.

This collective behavior can be modeled as an interaction.

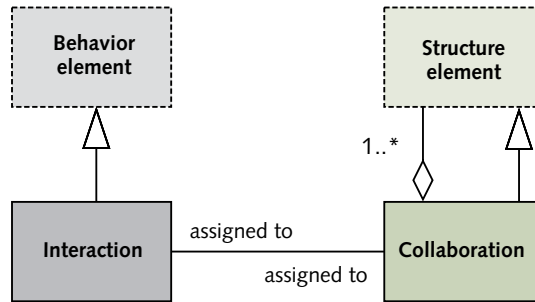


Figure 4: Collaboration and Interaction

3.4 Relationships

Next to the core concepts outlined above, ArchiMate contains a core set of relationships. Several of these relationships have been adopted from corresponding relationship concepts that occur in existing standards; e.g., relationships such as composition, aggregation, association, and specialization are taken from UML 2.0, while triggering is used in many business process modeling languages.

Note: For the sake of readability, the metamodel figures in the next sections do not show all possible relationships in the language. Refer to Section 8.5 on additional derived relationships. Furthermore, aggregation and composition relationships are always permitted between two elements that have the same type.

3.5 Layering

The ArchiMate language defines three main layers (depicted with different colors in the examples in the next chapters), based on specializations of the core concepts described in Sections 3.2 and 3.3:

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 relations are used, although their exact nature and granularity differ. In the next chapters, we will specialize these concepts to obtain more concrete concepts, which are specific for a certain layer. Figure 3 shows the central structure that is found in each layer.

In line with service orientation, the most important relation between layers is formed by “used by” relations, which show how the higher layers make use of the services of lower layers. (Note, however, that services may not only be used by elements in a higher layer, but also by elements in the same layer, as is shown in Figure 3.) A second type of link is formed by realization relationships: elements in lower layers may realize comparable elements in higher layers; e.g., a “data object” (Application layer) may realize a “business object” (Business layer); or an “artifact” (Technology layer) may realize either a “data object” or an “application component” (Application layer).

3.6 The ArchiMate Framework

The aspects and layers identified in the previous sections can be organized as a framework of nine “cells”, as illustrated in Figure 5. The cells in this framework are a subset of the cells in, for example, the Zachman framework [5], [9]. Often used architectural domains can be projected into this framework; Figure 5 shows an example of this.

It is important to realize that the classification of concepts based on conceptual domains, or based on aspects and layers, is only a global one. It is impossible to define a strict boundary between the aspects and layers, because concepts that link the different aspects and layers play a central role in a coherent architectural description. For example, running somewhat ahead of the later conceptual discussions, (business) functions and (business) roles serve as intermediary concepts between “purely behavioral” concepts and “purely structural” concepts.

Besides the core aspects shown in Figure 5 (passive structure, behavior, and active structure), which are mainly operational in nature, there are a number of other important aspects, some of which may cross several (or all) conceptual domains; for example:

- Goals
- Security

- Governance
- Costs
- Performance
- Timing
- Planning and evolution

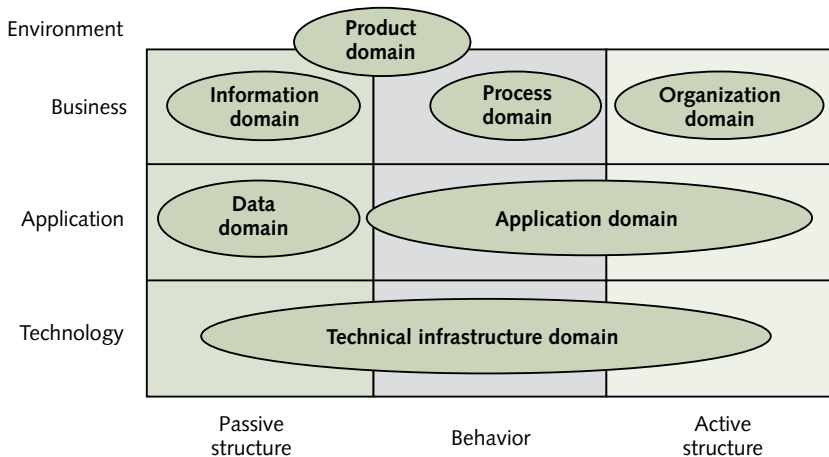


Figure 5: Architectural Framework

The aspects may be added to the models by means of additional concepts, relationships, or attributes. Also, it may be useful to add concepts or attributes related to the design process rather than to the system or organization that is to be described or designed. Examples of such concepts or attributes are requirements and design decisions. These aspects may be addressed in future extensions of the language (see Chapter 1 for a more thorough discussion of this).