ArchiMate to UML mapping

Thomas Gericke

Adocus AB, Stockholm, Sweden
thomas.gericke@adocus.com

Abstract: ArchiMate is a notation for enterprise architecture modeling and its use and popularity is on the rise. However, the more technically oriented modeling notation UML is used as a base in most available modeling tools and UML is also needed for more detailed type of models. To make use of existing modeling tools and for ArchiMate be able to co-exist with other types of models based upon UML there is a need to understand the relationship between ArchiMate and UML. This paper offers one possible and well-founded proposition for a mapping between ArchiMate and UML.

Keywords: ArchiMate, UML, Enterprise Architecture, Modeling.

1 Introduction

ArchiMate® is an enterprise architecture modeling notation from The Open Group®. Enterprise architecture is the top-most architecture as it documents the strategic development intent for whole organizations including all different kinds of sub architectures in that organization, for example business models, application models as well as infrastructure models. The notation ArchiMate is closely related to the widely spread enterprise architecture framework TOGAF, also managed by The Open Group.

Having said that enterprise architecture is strategic in nature, this also means that it is "high level", broad and less deep in nature. The depth/details need to be documented in other types of models and notations such as UML, BPMN etc.

Since UML is the largest existing modeling notation there is, and strategic models should be detailed into other types of models/notations, there is a need to be able to make ArchiMate and UML co-exist and relate to each other. To do this, we need to be able to have ArchiMate and UML in the same modeling tools and that often means that we need to build other modeling notations upon UML which is so widely used in modeling tools today. To be able to build ArchiMate upon UML we need to know the actual relationship between the constituents of both modeling notations – there are a lot of similarities since ArchiMate is influenced by UML, but there are some differences as well.

This paper documents findings and challenges when mapping ArchiMate, version 2.1, to UML 2.5 after an in-depth investigation made when an ArchiMate DSML-extension was created for the open source Papyrus modeling tool.

2 Goals with the mapping

The goal with the mapping is to find the logically closest UML element and/or relation for any given ArchiMate element and/or relation. In other words, mapping shall not be done in a "one element fits all" fashion, for example map all ArchiMate elements to the same kind of UML classifier, for example *class*.

3 Acronyms and abbreviations

UML – Acronym for *Unified Modeling Language*. A widely spread and accepted modeling language, primarily for defining architecture and design of IT systems.

RUP – Acronym for *Rational Unified Process*. A step-by-step process for development of IT systems. RUP covers business modeling, requirements and architecture of IT systems. RUP uses UML extensively for depicting views of different kinds.

BPMN – Acronym for *Business Process Modeling and Notation*. A widely spread and used modeling language for describing business processes.

SAD – Acronym for *Software Architecture Document*. An architectural document describing significant aspects of software. Described in detail in RUP.

4 ArchiMate concepts and layers

In ArchiMate there are three main layers and two extensions. The three main layers are *Business layer*, *Application layer* and *Technology layer*. The extensions are *Motivation* and *Implementation and migration*. The five layers/extensions are listed below in a logical order, with the most strategic layer first and more realization related layers below in a logical, consecutive order.

Motivation extension

The *Motivation extension* in ArchiMate shows the things that drive the whole evolution of the organization´s architecture. In this layer we find things such as *stakeholders*, *drivers*, *goals* etc., i.e. all the things that management persons need to be able to discuss and plan a wanted scenario for the future.

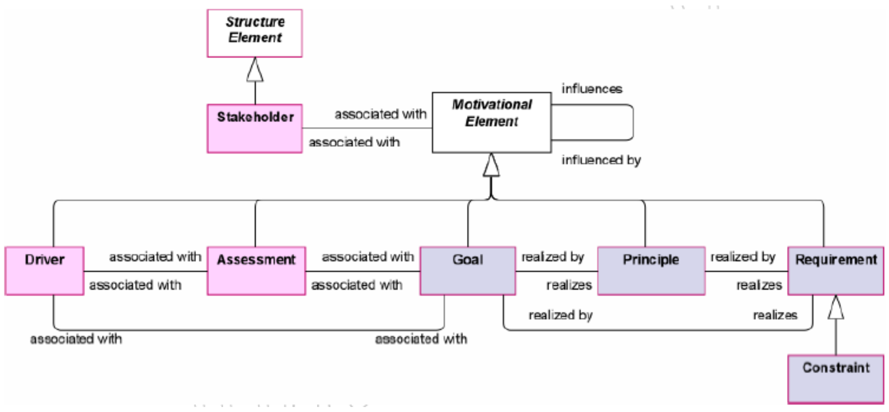

Figure 1 ArchiMate specification for Motivation layer

Naturally, UML does not have similar concepts within this extension because of its base in technology. Therefore the investigation has found that the most reasonable UML element to map to ArchiMate motivation layer elements is the *class* element. The class element must be regarded as the core of UML and it is probably also its most widely used element type.

Business layer

The *business layer* in ArchiMate more practically shows things that the organization offers to its end customers to be able to reach the goals in the *Motivation layer*. In the business layer we see things such as *values*, *products* and *contracts* etc.

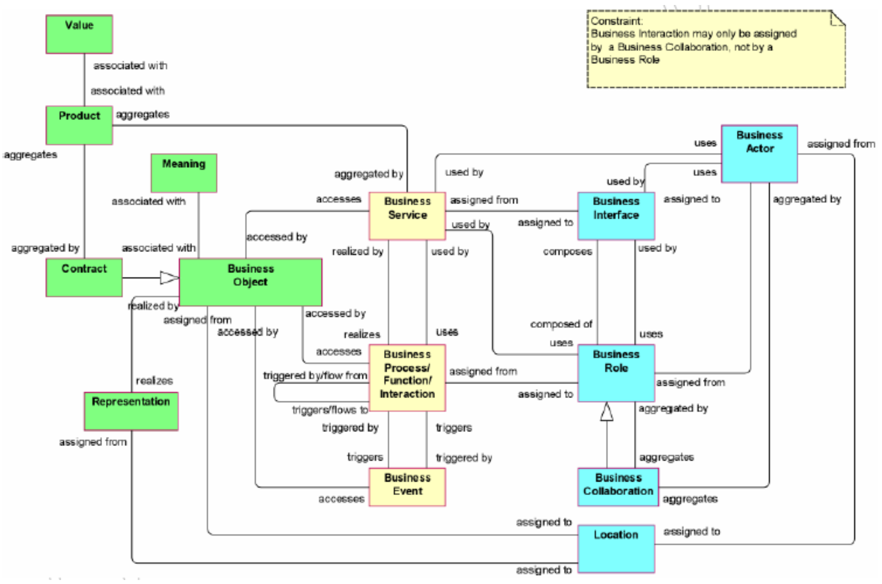

Figure 2 ArchiMate specification for Business layer

In UML we see more natural mappings to the business layer in ArchiMate because we start to reach the boundaries for what UML traditionally has tried to address. For example we have the concept *business actor* in ArchiMate which naturally maps to the RUP *business actor*. Therefore, all actor-oriented ArchiMate elements are recommended to map to the closely related UML element actor.

In the business layer in ArchiMate we also find a *role* concept. Since there is no such thing in UML we recommend to use the default classifier *class* in UML for role-related ArchiMate elements.

Collaborations as concept exist in both languages. Collaborations are packaging of elements that together collaborate to create some kind of greater value than each element can do of its own. The UML element that should be used to carry ArchiMate collaborations is for natural causes the standard UML collaboration since they both have similar meaning.

In the business layer we also see the concept *interface* in the ArchiMate element business interface. Interfaces in both languages have the same meaning why it is natural to use the UML interface when mapping ArchiMate interfaces in general.

In this layer a “geographical” concept *location* is introduced. The closest UML element is the *node* element which in turn often has a physical relation in the real world why it is chosen as the UML element to map to.

Processes do not exist in UML but they do in ArchiMate. The closest to process elements in UML is the *activity* element why this should be the recommended element to be used. However, activities require the context of activity containers and activity diagrams and our ArchiMate diagram types are all based upon class diagrams. Hence, the ArchiMate *business process* element is mapped to UML opaque behavior which is the closest element type that allows itself to be added to class diagrams.

Business functions in ArchiMate are groupings of behavior based on a chosen set of criteria. The options in UML are in this case the *package* or the *collaboration* element. The package because the business function can be seen as a business unit and therefore packages organizational elements such as roles or employees. However, the package element does not describe dynamics why the recommended UML element should be the collaboration which packages both structure and behavior.

Business interactions in ArchiMate show where the organization has interaction with the outside world. Since these interactions follow a behavior unknown or uninteresting in detail the recommendation is to map this kind of element to the UML element type *opaque behavior* which set a name but otherwise abstracts all interior of the behavior. The same UML classifier is used for the ArchiMate *business event*, which also has behavior unknown in detail.

The ArchiMate *business service* is close to the RUP element *business use case* which in turn is based upon the UML *use case* classifier why the recommendation is to map ArchiMate business functions to UML use cases.

In ArchiMate, within the business layer, the *business object* is introduced. A business object is an “information” or “conceptual” type of element that best maps to the UML *class* which in turn is the UML type of element to carry information, often with the help of so called attributes.

Representations in ArchiMate are perceivable form of information carried by business objects. The thing that represent and hold information in the physical world in UML are *artifact*, why they are chosen as the UML element for ArchiMate representation elements. Also *products* and *contracts* in ArchiMate map well to the UML artifact element.

Finally we have two concepts that have no natural correspondence in UML – *meaning* and *value*. Since these two concepts have no natural UML element to map to we recommend to use the UML element *class* to map to these constructs.

Application layer

In the ArchiMate application layer we find software related constructs to support the business we want to realize. Here we find things like *application services*, *functions* and *interfaces* etc.

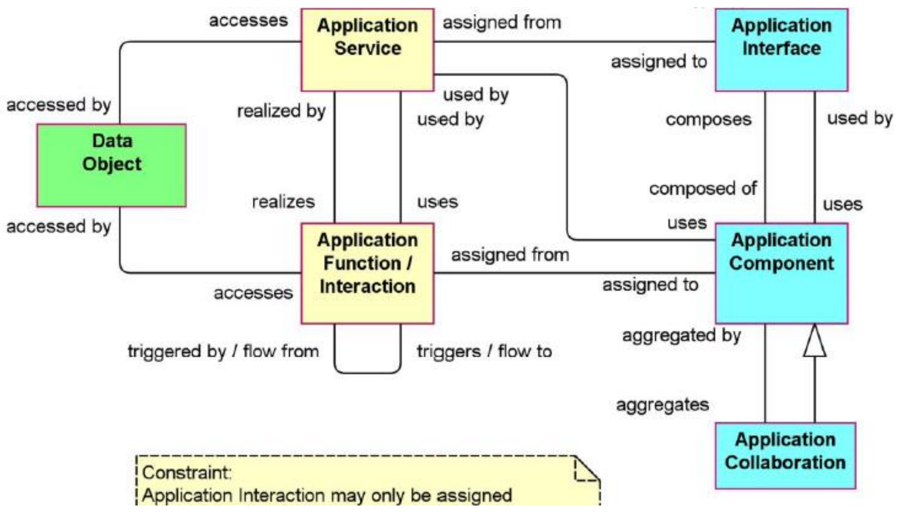

Figure 3 ArchiMate specification for Application layer

The ArchiMate *application layer* is one of the layers within ArchiMate with most natural and best mapping to UML constructs, simply because UML in its nature has a technical (software) focus.

In this layer we find the ArchiMate *application component* which is very close the definition of the UML component. We also find *application collaboration* which maps almost directly to the UML collaboration.

In the business layer section we said that interfaces in ArchiMate in general should map to UML interfaces which we honor by mapping the ArchiMate *application interface* to the UML interface as well. We also follow the same logic as in the business layer for functions and hence map the ArchiMate *application function* to UML collaborations and also map *application interactions* to UML opaque behavior.

In ArchiMate the *application service* functions as the externally visible functionality of systems. This gives that it should be mapped to UML use cases, since use cases is just that – functionality offered to end users without the interior exposed.

At last we have the *data object*, which is close the UML classes why we recommend to map ArchiMate data objects to UML classes.

Technology layer

In ArchiMate, the *technology layer* describes the needed infrastructure for realizing the enterprise architecture migration. This layer mostly relates to the deployment view or diagram in RUP SAD, but at a higher and more abstract strategic level.

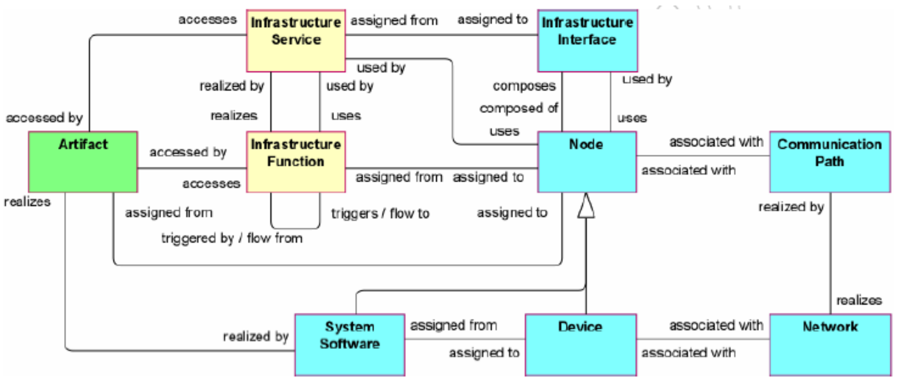

Figure 4 ArchiMate specification for Technology layer

For natural reasons, there is a natural mapping between most ArchiMate elements to UML in this layer. For example, the most significant ArchiMate element *node* maps directly to its UML counterpart with the same name. The same direct mapping should be made between the ArchiMate *device* to the UML device (even though their internal meanings differ slightly).

ArchiMate´s *system software* element has its counterpart in UML in the *execution environment* element due to the fact that system software can be viewed as an environment for application execution.

In ArchiMate there exist elements that can be depicted as both graphical elements as well as relations between elements. These elements are called *network* and *communication path* and both relate to UML´s structural element *node* as well as its relation type *communication path*.

As before, ArchiMate functions and services map to UML collaboration and use case respectively and hence *infrastructure function* should be mapped to the UML *collaboration* and the *infrastructure service* is best mapped to UML *use cases*.

The ArchiMate *artifact* has a direct and natural relation to its UML counterpart with the same name.

Implementation and migration layer

The *implementation and migration layer* in ArchiMate offers element types handling the actual transformation of the organization´s architecture to a new and higher level. There are no natural, direct mappings in the UML language of natural reasons, but we can choose logical representations in UML to represent the elements in ArchiMate.

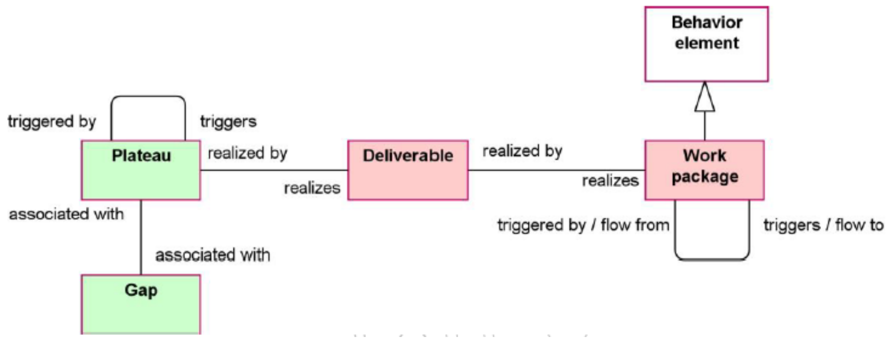

Figure 5 ArchiMate specification for Implementation and migration layer

A *deliverable* in ArchiMate is an outcome of a *work package* and a work package in turn is a set of actions to accomplish a goal of some sort. The deliverable is best mapped to the UML *artifact* by its nature and the work package is best mapped to the UML *opaque behavior* due to its relation to actions (that are not seen or documented in detail in the model).

Plateaus and *gaps* in ArchiMate have a close relationship in the fact that plateaus depict exact levels of an architecture and the gaps show the differences between these plateaus. Since plateau by itself is very abstract and does not have a natural corresponding UML element it is recommended to use the UML *class* when mapping. Gaps are in fact "documentation" of the difference between two plateaus why it is recommended to use the UML *artifact* for mapping.

5 Relations and related topics

In ArchiMate there are twelve different kind of relations and many of them stem from/are highly influenced by the UML language. Therefore there is an easy task to map a lot of the relations, for example *association*, *aggregation* and *composition*.

However, there is a set of relations that do not map directly and an interpretation had to be made, for example the ArchiMate *access* relation is mapped to the UML *usage* and the ArchiMate *derived* relation is mapped to the UML *dependency*.

The *junction* concept in ArchiMate is semantically closest related to the UML *decision node*, but since decision nodes can only exist within activities and ArchiMate is "flat" in nature (see chapter Observations) the junction is mapped to the *opaque behavior* so that modeling can performed in the only-one-dimension.

Last but not the least we have an odd and hard-to-make mapping in the *used by* relation in ArchiMate which has its closest relative in UML in the *usage* relation. Please read the chapter **Challenges in mapping to UML** for details regarding the mapping to this relation.

6 Observations

One major observation that has been made during the analysis of the ArchiMate modeling language is that it, in comparison to UML, is "flat" in nature. This means that when UML can have hierarchies of information (elements within packages, actions within activities etc), ArchiMate is "one-dimensional". Notably, even the *grouping* element in ArchiMate is "flat", i.e. cannot contain other elements "underneath" as its counterpart *package* in UML. This is also the reason why we chose component as base for the grouping element – this makes "flat modeling" easier in practice.

7 Challenges in mapping to UML

Even though ArchiMate has had UML in mind when it was created, it has been harder to map between the two notations than anticipated.

One thing is that the ArchiMate specification (the “meta model”) is much less rigid and less documented than its counterpart “the UML specification”. This has the effect that a lot of the interpretation is shifted to the skill, knowledge and experience of the reader. Hopefully we have done the correct assumptions during the mapping process to UML, but the specification still has a lot to wish for.

Another thing is that direct errors were found in the ArchiMate specification, in this case an ArchiMate “role name” at one instance was placed on the wrong side of the association which made the interpretation impossible. This was handled by simply reading the specification “the opposite way” and the way it was probably intended. Even though a way around was found for this problem it is still unsatisfying to rely your work on “work arounds” just to get it work.

A third thing to mention is the illogical and reverse interpretation of the ArchiMate relation *used by* which has an opposite naming than its UML counterpart *usage*. It may not sound a lot, but to say “used by” has another actor than “usage”. A car may be used by its driver but it is the driver that uses (usage) the car. This totally different set of mind is problematic once you are accustomed with UML – you need to reverse all usage relations when you apply ArchiMate used by relations.

8 Deviations from the ArchiMate specification

ArchiMate symbols come in two flavors – “pure symbol presentation” or “rectangular presentation”. The pure symbol presentation presents the ArchiMate elements with a graphic on a transparent background and the rectangular presentation presents elements with a rectangular background with a small symbol up to the right. Some of the rectangles in the specification have “cut corners” and some have not – they are just rectangles. The Papyrus add-in supports both pure symbol presentation and rectangular presentation. However, due to limitations in Papyrus, the cut corners are not offered.

Relations in ArchiMate are sometimes rendered solid and sometimes dotted or dashed. In cases where the base relation type (UML) is dotted/dashed and the ArchiMate relation type based upon this UML relation type is solid, the add-in does not set the presentation to solid because of an exception thrown by Papyrus when saving diagrams as images. The ArchiMate relations related to this are *triggering*, *used by* and *assignment*.

Often relations have end decorations (“arrow heads”), both in UML and in ArchiMate. However, UML does not have small “filled” end decorations as ArchiMate have on some relations and there is no way to customize Papyrus to reflect this. Therefore, the following ArchiMate relations do not have the correct kind of end decoration: *access*, *flow, influence*.

Appendix A – Complete mapping ArchiMate to UML

This appendix shows the complete list of recommended mappings between ArchiMate and UML based upon the findings made during the investigation mentioned in this whitepaper.

Motivation layer

ArchiMate element	UML meta class
Assessment	Class
Goal	Class
Principle	Class
Stakeholder	Class
Driver	Class
Requirement	Class
Constraint	Class

Business layer

ArchiMate element	UML meta class
Business actor	Actor
Business role	Class
Business collaboration	Collaboration
Business interface	Interface
Location	Node
Business process	OpaqueBehavior
Business function	Collaboration
Business interaction	OpaqueBehavior
Business event	OpaqueBehavior
Business service	UseCase
Business object	Class
Representation	Artifact
Meaning	Class
Value	Class
Product	Artifact
Contract	Artifact

Application layer

ArchiMate element	UML meta class
Application component	Component
Application collaboraration	Collaboration
Application interface	Interface
Application function	Collaboration
Application interaction	OpaqueBehavior
Application service	UseCase
Data object	Class

Technology

ArchiMate element	UML meta class
Node	Node
Device	Device
System software	ExecutionEnvironment
Infrastructure interface	Interface
Network	CommunicationPath, Node
Communication path	CommunicationPath, Node
Infrastructure function	InfrastructureFunction
Infrastructure service	InfrastructureService
Artifact	Artifact

Implementation and migration

ArchiMate element	UML meta class
Deliverable	Artifact
Gap	Artifact
Plateau	Class
Work package	OpaqueBehavior

Relationships and packaging elements

ArchiMate relation	UML meta class
Access	Usage
Flow	InformationFlow
Specialization	Generalization
Triggering	InformationFlow
Used by	Usage
Composition	Association
Aggregation	Association
Realization	Realization
Derived	Dependency
Assignment	Dependency
Association	Association
Junction	OpaqueBehavior
Grouping	Component
Influence	Dependency