Supporting Enterprise Cartography through Metamodel Integration

Margarida Guerreiro Morais

Thesis to obtain the Master Degree in
Information Systems and Computer Engineering

Supervisors: Prof. André Ferreira Ferrão Couto e Vasconcelos
Prof. Pedro Manuel Moreira Vaz Antunes de Sousa

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Pedro Manuel Moreira Vaz Antunes de Sousa
Member of the Committee: Prof. Sérgio Luís Proença Duarte Guerreiro

October 2020

Acknowledgements

I would like to show my gratitude to my advisors, Pedro Sousa and André Vasconcelos, for the support, patience and guidance in this work. The deep knowledge and experience they have, contributed to develop my research and obtain a more consolidated work.
I am also very grateful to all my closest friends, they who were always present and always supported me throughout this journey, and with whom I shared some of the most important moments in my university years.
Finally, my biggest words of gratitude go to my entire family, who have always shown interest in my academic successes and have always supported me, especially my parents and my sisters who have always advised me in the best way during all these years, and also to Bruno, for helping me steer through some turbulent times and for his unconditional love and friendship.

Abstract. Enterprise Architecture (EA) has taken on a major importance for the biggest problem Enterprises face today - "change". Enterprise Architecture enables the bridge between Business-Driven strategy and IT-Driven implementations and the establishment of an Enterprise environment suited to change. Our motivation with this work is to understand the main differences between two EA specifications (TOGAF and ArchiMate) and following previous work done on TOGAF Framework and ArchiMate Modeling Language Harmonization achieve through the Enterprise Cartography tool, Atlas, a way to transform ArchiMate models into TOGAF models and vice-versa in order to have all the models consolidated in one EA repository. In this research we explore the TOGAF 9.2 and ArchiMate 3.1 metamodels, its entities and relationships and the harmonization between both metamodels, focusing on their integration. We propose a solution to incorporate models from different specifications, TOGAF and ArchiMate, into a single and enterprise-wide view using the Enterprise Cartography tool Atlas. The approach proposed defines the process for the automation of EA documentation and, using XSLT script technology, provides a mechanism of integration between the EA repository and the modelling specifications, and its different metamodels. The result obtained is a solution to transform TOGAF models to ArchiMate models and vice versa, being able to consolidate these models in the same EA repository. The solution proposed is applied to a simple model example that demonstrates the purpose of this work.

Keywords: Enterprise Architecture · Enterprise Cartography · TOGAF · ArchiMate · Model Integration · Atlas

Table of Contents

1 Introduction

Even though Enterprise Architecture as a discipline is still fairly young, the concept itself remotes to the mid of late 1980s after John Zachman published an article describing a framework for information systems architectures in the IBM Systems Journal. Zachman saw Enterprise Architecture as a set of descriptive representations that are relevant for describing an Enterprise such that it can be produced to management's requirements and maintained over the period of it's useful life [1]. Nowadays Enterprise Architecture is practiced not only as a collection of artifacts by itself but is used to deliver services to improve overall organizational performance. Given we live in a frantic information era and ongoing change, to be able to bring value and benefits from EA, Enterprises have the need to be able to update their EA and manage it in the most automatic and efficient way [2], so its consistency can be preserved while the organization continues to evolve the architecture. A well defined EA is seen as an important asset for innovation and change by providing both stability and flexibility. [13]. Given the size and still immature nature of many Enterprise Architecture efforts in organizations, a number of critical challenges and problems continue to exist on how to develop and manage a EA project so an updated and consolidated EA view can be achieved. [3].

1.1 Objectives

As much benefits as Enterprise Architecture can bring to the table, a study made in 2010 shows that 66 percent of the EA projects expected results are not achieved and that it is not only the implementation of an EA project itself that is a challenge but also the several difficulties in the maintenance and managing of that same EA. [4] The problem of keeping a set of accurate representations (i.e. models) of an enterprise is not an easy one for large enterprises. The origin of this difficulty is that the planning process originates in many of the enterprise's communities without a consistent, coherent and complete systemic view of the enterprise to support them, individually and as a whole as well as the struggle to align the different elements designed by different architects resulting into an unconscious design [5]. Referring to Enterprise Cartography (EC) as the process of representing an Enterprise observed directly from reality. We can differentiate it from Enterprise Architecture (EA) because it focuses on producing representations based on observations and not including the purposeful design, as one expects in EA. [5] Adopting a federated approach Enterprise Cartography demises the constraints and inflexibility that one has on the EA modeling and provides a way to have concise and coherent views of the entire organization.

Considering work already developed in the context of harmonization between TOGAF and ArchiMate specifications [32], the objective of this work is to achieve through the Enterprise Cartography tool, Atlas, a semi-automatic solution to transform ArchiMate models into TOGAF models and vice-versa, being

able to consolidate these models in the same EA repository. In order to obtain the expected results and create a viable solution, we set the following objectives:

1. *Related Work*

   In this phase we will start by analyzing the previous work done in this field. Starting by understanding the TOGAF and ArchiMate specifications, following a good understanding of the Enterprise Cartography discipline and its tool Atlas. Insights on metamodel integration will also be covered, focusing on the XSLT approach, which will be the one in use. Ending with research on the work developed on the TOGAF and ArchiMate harmonization topic [32].

2. *Concepts Mapping*

   To understand which mappings of entities can be used in the transformation between metamodels, in this case between TOGAF entities and ArchiMate entities, an analysis of the meaning of each one is necessary. Following the work done in [32] and research on the new entities added in the TOGAF 9.2 specification and ArchiMate 3.1 specification, the result we want to achieve in this phase is the materialization of the mapping between ArchiMate entities and TOGAF entities and vice versa.

3. *Relationships Mapping*

   To understand which mappings of relationships can be used in the transformation between metamodels, in this case between TOGAF entities and ArchiMate relationships, an analysis of the meaning of each one is necessary. The main challenge on this phase is the mapping between relationships that don't have a direct mapping and which need to be defined. Following the work done in [32], the result we want to achieve in this phase is the materialization of the mapping between ArchiMate and TOGAF relationships.

4. *XSLT Tranformation Approach*

   Using the XSLT approach to create a metamodel transformation mechanism, it is necessary to define three files to be run in a batch and inserted in the Atlas repository. Two of the files contain the definition of the metamodels, a file with the definition of the entities that are in the TOGAF 9.2 specification and a file with the definition of the entities that are in the ArchiMate 3.1 specification need to be defined. The third file contains the mapping rules between entities and relationships from the origin model to target model. The mapping rules defined in this XSLT file are not general to all TOGAF to ArchiMate transformations or vice versa. Each model to be transformed must have a specific XSLT file with its mapping rules defined as the applied rules will be different from model to model. Since the ArchiMate 3.1 specification is already available in the Atlas repository, the objectives for this

phase are the definition of the XSLT file with the TOGAF 9.2 specification to be inserted in the Atlas repository and the understanding of the mapping rules that the Atlas tool offers and which ones will be used in order to achieve the model transformation without losing meaning nor coherence.

5. *Model Transformation - Example*

   Using the work and results acquired from the previous objectives. The purpose of this phase is to test the model transformation process. An example of an ArchiMate model will be used to test the transformation process into a TOGAF model and vice-versa.
   In order to achieve the transformation between models is necessary the specific definition of the XSLT file with the mapping rules between entities and relationships for the model example that we will use to demonstrate the transformation process. The objective of this phase is the definition of the XSLT file with the mapping rules for the model example used to be executed in a batch in the Atlas tool, and achieve the expected transformation thus testing the implemented solution.

1.2 Organization of the Document

In the next section the related work is described. In Section 3, our solution is proposed describing in depth the concept and relationship mapping between TOGAF and ArchiMate specializations which will be used in the model transformation process to achieve the metamodels integration. It is also explained which XSLT files need to be defined and processed by the Atlas tool in order to materialize the model transformation and create a way to consolidate models based in different metamodels into a single and enterprise-wide view. Following in Section 4 with the presentation of a model example, which will be used to evaluate the solution, and the definition of the XSLT file with the specific mapping rules for the model transformation. Finishing with a conclusion and future work section.

2 Related Work

The next section starts by analyzing Enterprise Architecture as a discipline, the Enterprise Architecture Framework TOGAF [37] and the ArchiMate language that will support the understanding of its metamodels and EA development methodology. The benefits and constraints that arise from an Enterprise Architecture solution implementation will be addressed. Such constraints often lead to the total failure of an EA project, so it is important to analyze why such limitations arise and how the adoption of Enterprise Cartography as a supporting tool and method can mitigate these limitations. To this end, the concept and essence of Enterprise Cartography is explained and how it can be implemented, followed by a brief discussion and summary about the two domains (EA and

EC) and how one complements the other. In the related work section, it is also covered the EC supporting tool that will be used in our solution, Atlas, research on Model-based Integration Methodology, focusing on the XSLT approach, and also previous work done on the subject of TOGAF and Archimate Language Harmonization.

2.1 Enterprise Architecture

EA discipline and benefits

The term architecture has been known for a long time in the context of building and construction and has been viewed as an integrated view of the system being designed or studied. Architecture at the level of an entire organisation has a different connotation and is referred to as 'Enterprise Architecture'. Adopting the definition from [38] EA can be viewed as a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure. The purpose of EA is to optimize across the enterprise the often fragmented legacy of processes (both manual and automated) into an integrated environment that is responsive to change and supportive of the delivery of the business strategy. [37] From EA models, we can make EA artifacts to represent viewpoints that address concerns of specific groups of organization's stakeholders. The benefits of EA stem from having the holistic view of the organisation, for example:

- Provides a means for rapid communication and change propagation [6]
- Turns a complex environment into a manageable one [7]
- Enables Business and IT alignment [7]
- Provides a means to manage integration of old and new systems [8]
- Maintains the link between the IS design world and the operational world [9]
- Allows for flexibility and responsiveness to change [10]
- Provides a means for business resource optimization [11]

Modeling architectures in detail requires some guidance, which is offered by architecture frameworks. A framework is a structure within which the key components of the architecture and the relationships between these components are defined. To guide the organization in the EA domain several frameworks were developed to be used for the developing of a broad range of different architectures. They describe a method for designing a target state of the enterprise in terms of a set of building blocks, and show how the building blocks fit together. In general, an architecture framework defines a set of views. Each view focus on a particular aspect of the architecture, presenting various details to different target audiences. An architecture framework also has a defined terminology that is applied to all the architectures defined by the framework. This ontology enables the standardization of terms and is widely used as the basis of a metamodel. A

metamodel captures the syntax and semantics of the different views. The metamodel defines the valid and necessary elements for the creation of the different views and also the relations existing between them. To ensure development and maintenance of architectures, a corresponding methodology may be part of the framework. The methodology also provides procedures to determine and organize the data that is the basis for the architecture. It contributes to ensuring the consistency, accuracy, and completeness of the acquired data. [30] Some of them also contain a set of tools and provide a common vocabulary and recommend a list of standards and compliant products that can be used to implement the building blocks. [37]

TOGAF Framework

The Open Group Architecture Framework (TOGAF) is a conceptual model of enterprise architecture conceived in 1995 by The Open Group Architecture Forum, whose goal is to provide a global approach to the design, planning, implementation and governance of architectures, thus establishing a common language of communication among architects.
TOGAF is based on an iterative, reusable, cyclical process and supported by the best modeling practices involved in the activities of an organization, comprising four types of architecture that are commonly accepted as subsets of an enterprise architecture, namely: business, data, applications, and technology.
The content of TOGAF is structured in five parts: [37]

1. Architecture Development Methodology (ADM) – Defines the process of creating an architecture. Is a full life-cycle process for planning, designing, realizing and governing EA. This cycle has a preliminary part and eight core parts arranged in a sequential cycle order: A - Architecture Vision; B – Business Architecture; C – Information System Architectures; D – Technology Architecture; E – Opportunities and Solutions; F – Migration Planning; G – Implementation Governance; H – Architecture Change Management.
2. Architecture Content Framework – The Architecture Content Framework within the TOGAF framework identifies the main types of architecture building blocks that are relevant in the context of the ADM. Provides a structured model of types, relationships and attributes of building blocks that used informally or as the basis for configuration of an Enterprise Architecture modelling tool represent the basic elements of architecture in TOGAF. The Content Framework features a core and extension concept. The Core concept provides an introduction to the way in which the TOGAF framework employs a basic core metamodel and the extension modules address specific architectual issues in more detail, tailoring the model and making it more robust to answer specific parts of the Enterprise Architecture.
3. Architecture Capability Framework – Defines the processes, the skills, the roles and the responsibilities that an organization need to have to construct and develop an EA.

4. Enterprise Continuum and Tools – Defines taxonomies and tools to categorize and store the outputs of an EA.
5. Reference Models – Technical Reference Model (TRM) and the Integrated Information Infrastructure Reference Model (III-RM).

ArchiMate Language

ArchiMate is an open and independent modelling language for enterprise architectures, owned and developed by The Open Group [37] . ArchiMate provides instruments to describe, analyze and visualize the relationships among different Enterprise architecture domains in a simple and unambiguous way.
Clear concepts and relationships between architecture domains are presented in the ArchiMate language that offers a simple and uniform structure for describing the contents of these domains.
The ArchiMate language provides a core framework that allows for the modeling of the enterprise from different viewpoints. The aspects (passive structure, behavior and active structure) and the layers (business, application and technology) create the structure of the framework where the position within the cells highlights the concerns of the stakeholder. There is also some layers and an aspect that can be added to the Core Framework. The physical elements are added to the Technology layer for modeling physical facilities and equipment, distribution networks, and materials. Besides, an additional motivation aspect and the implementation and migration elements are added.

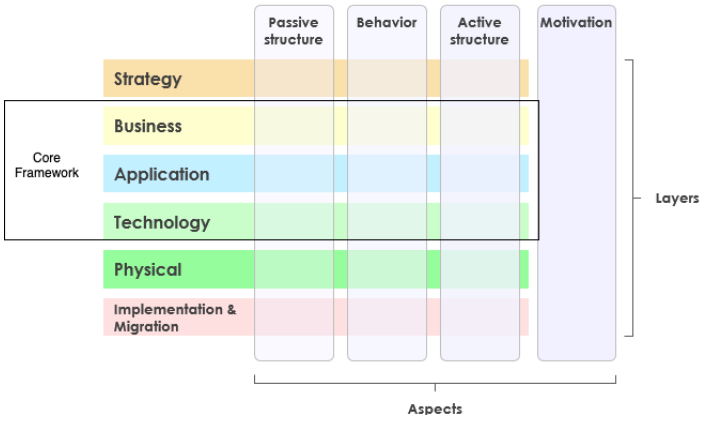


Fig. 1: ArchiMate Full Framework - retrieved from [34]

ArchiMate Language and TOGAF Framework

ArchiMate is a widely use visual language for modeling EA, but it is not a method which provides steps and techniques for guiding through the entire EA

development process. Although TOGAF is publicized as an architecture framework, it can be seen as a body of knowledge describing some aspects of EA practice. TOGAF strongest and most detailed content describes the process for developing architectures – the ADM. While neither provides a "complete" approach to Enterprise Architecture, TOGAF and ArchiMate complement each other. TOGAF describes a process for developing architectures which requires us to document the baseline, transition and target architectures and express stakeholders concerns through views that show pertinent aspects of the architecture, while ArchiMate gives us a modelling language, with a standard set of shapes and colours to graphically present these various views.[33] The TOGAF Content Framework and Metamodel also provides some concepts that help in the documentation of the enterprise architecture, there is quite a lot of overlap and similarity in the components described in TOGAF and those included in ArchiMate. At the high level, both divide EA into domains or layers: in TOGAF we have the business, information systems (data and application) and technology architectures, while in ArchiMate we have the Business, Application and Technology Layers (the application layer includes the data and application domain). Nonetheless the ArchiMate concepts and relationships are more powerful in the description process of the architecture where TOGAF is more powerful on helping in the process of developing architectures.

These two specifications are widely used in the development of Enterprise Architecture and both, being owned and developed by The Open Group [37], are designed in a way that their metamodels can be harmonized. For this reason these are the specifications that we will address in this research.

2.2 Enterprise Cartography

Keeping design and representation together is fine for a centralized coordinated design process, but organizations are actually designed in an asynchronous, distributed process, involving many actors and many stakeholders, without formal mechanisms of communication between the different designers. The current EA patterns are not capable of handling the organization´s changing pace, neither for design purposes nor to keep the architectural representations up to date. The emerging of the Enterprise Cartography keeps apart design from representation. [14]

EC discipline
Enterprise cartography denotes the discipline dealing with the conception, production, dissemination and study of enterprise maps to support the collective understanding of a dynamically changing organization. [12] Enterprise cartography doesn't focus on the enterprise design, but rather targets the abstraction and representation of the "enterprise reality".

enterprise reality - refers to the present state of the enterprise, sustained on relevant facts captured in logs and represented through artifacts based on previously defined and agreed upon models. [5]

With Enterprise Cartography we can consolidate each partial architecture into a single global set of architectural maps and represent the enterprise architecture in different points in time, namely: [5]

- *AS-IS* - Set of all alive artifacts, their relations and states in the present time.
- *TO-BE* - Set of alive artifacts, their relations and state at a given time in the future.
  - *emerging AS-IS* - set of all alive artifacts, their relations and states as observed after the successful completion of ongoing transformation initiatives.

From [5] we have managed to extract the main concepts that we need to apprehend in order to understand what is Enterprise Cartography:

Concept	**Definition**
Enterprise metamodel	Set of artifact types and the relations between used in the definition and visualization of the organization architecture
Architectural Statement	Well-formed proposition regarding the organization architecture.It must necessarily be expressed using the artifact types and relations defined in the metamodel
Enterprise Observation	Formulated architectural statements from the observation of the enterprise reality
Architectural Map	Graphical representation of a set of architectural statements over a period of time
Transformation Initiative	Is a set of planned and purposeful activities that might yield changes in the enterprise artifacts, such as projects
Alive Artifact	Enterprise artifact that can play roles in the organizational transformations and process to create value. Aliveness of an artifact is not an intrinsic property
Knowledge Base	Repository holding the metamodel, the architectural statements and the definition of the conceptual maps
Enterprise Architect	Makes architectural statements regarding some point in time in the future
Enterprise Cartographer	Collects architectural statements from observations of the enterprise reality and produces architectural maps

Table 1: EC Concepts - retrieved from [5]

But how does the adoption of an Enterprise Cartography project help in any way the EA process?

From the practical cases discussed in [5] we can see that an EC approach can be the solution to a wide range of challenges encountered in the development, management and maintenance of an EA, such as a way to consolidate models produced in different projects into a single and enterprise-wide view, as a way to create a central knowledge base to allow a descriptive and perspective approach for EA, as a way to integrate important information spread on different tools, having all the challenges essentially a common goal: deploy processes and tools to produce up-to-date architectural maps of the enterprise architecture with near zero effort.

2.3 EA and EC Discussion

From the work reviewed in Section 2.1, it is seen that, having an Enterprise Architecture solution in organizations is quite important, since having an holistic view can help to identify and rectify existing problems, reduce duplication and inefficiency and plan for the future by being able to effectively model the impact of change. Since the development of an EA project is a complex and time-

consuming process if not well implemented, maintained and managed, most of the time it will not return any value to the organization.

From the related work in Section 2.2 it can be concluded that EC can be viewed as a process to maintain and manage EA views, being its ultimate goal to minimize the effort required to provide updated and trustful EA views in enterprises where enterprise's architecture is a distributed process, performed by architects from different domains (from business to technology). [5]
The adoption of EC as a process for the management and maintenance of an EA project eradicates some of the pitfalls that many times cause the failure of an EA project, e.g. the need for cooperation between the various domains of the organization/enterprise to develop a conscious design to achieve an holistic view.

Having been used in practical cases [5] as solution for similar problems as the one we want address in this research, achieve the transformation between ArchiMate models and TOGAF models as a way to consolidate models from the different specifications in the same EA repository, the adoption of EC will be the approach to consider in the scope of this research.

2.4 EA and EC Tools

Enterprise Architecture management seeks to align both business and IT in order to increase flexibility in today's constantly changing environment. Information about an EA is often regarded complex such that specialized EA tools facilitate the respective documentation. To meet this challenge several EA tools have emerged in the market [15] [16] over the past decade. [17] The EA and EC tools used in the scope of this research are next introduced.

Archi
The Archi modeller is targeted toward all levels of Enterprise Architects and Enterprise Modellers. Easily and intuitively lets architects create all ArchiMate elements and relations in all of the ArchiMate views. Archi modeller fulfils the needs of most Enterprise Architects and associated stakeholders, and has been designed to elegantly provide the main features required for ArchiMate modelling and is used globally. [18] Archi supports The Open Group ArchiMate Model Exchange File Format standard, the file format uses XML, which is backed by a validating XSD Schema.The aim of The Open Group ArchiMate Model Exchange File Format is to provide a pragmatic and useful mechanism for exchanging ArchiMate models and visual representations between compliant tool sets.
An advantage of using XML/XSD for the file format is that it is possible to use XSLT to transform the XML ArchiMate model instances into HTML documents, another XML file, reports, as input for a database, and so on. [19]

Atlas
Atlas is a web based EA tool developed to reduce the effort needed to keep EA

views updated in enterprises designed in a decentralized, distributed and asynchronous process using multiple design tools.
Atlas acts as a central repository that allows users to configure in and out data both in format as well in content for integration with other tools and systems. It supports the configuration of behavior associated with EA views, validation rules, state propagation between objects in the repository, among others. Such behavior can be stated both in queries and rules that run directly on the selected repository as well as in jobs and batch's that run on a dedicated sandboxes. [20]
Atlas was developed to be able to give its users the possibility to have different models coming from different EA tools being consistent with each other in a single place, using the transformation model based on XSLT [21] technology, it allows users to integrate these models, with different metamodels, in a single one, accomplishing the desired integration in a simpler way.

2.5 TOGAF and ArchiMate Harmonization

From Estrem and Gonzalez work developed in the TOGAF® Framework and ArchiMate® Modeling Language Harmonization white paper [32], the Content Metamodel and relationships harmonization between TOGAF and ArchiMate language is exposed.
The paper has the purpose of comparing and contrast the Content Metamodels of the TOGAF 9.1 and ArchiMate 2.1 standards. It is also intended to provide guidance to EA practitioners to better utilize both standards in the development of models for EA projects. The two standards can be joined together but should remain independent of each other, they work well and are compatible and complementary for EA development.
They have been developed independently of each other and address different though related purposes. The TOGAF metamodel is an abstract, implementation-independent model used to define the elements that are used within the artifacts used in the framework. The ArchiMate language metamodel is more specific and defines entities and relationships used for architecture modeling.
The document is structured in two parts. The first part provides the examination of the entities defined in the Content Metamodel of the TOGAF 9.1 and ArchiMate 2.1 standards . The second part examines the relationships that are defined between the entities in the respective standards.
Estrem and Gonzalez give a detailed explanation of how the entities and relationships of each of the standards are related, in cases where there is mapping and how strong this mapping is. In cases where entities that are present in the ArchiMate metamodels do not exist in the TOGAF metamodel and vice versa, the mapping of entities between the standards sometimes can be performed by either defining attributes (profiling) or through specialization and some specific examples of these mappings are provided.
In Annex A is presented a table with the mapping between the TOGAF and Archimate entities and its strength (Weak, Fair, Strong).
In Annex B is presented a table with the mapping between the TOGAF and Archimate relationships and its strength (Weak, Fair, Strong).

In Annex C is presented the Content Metamodel Harmonization. It displays the representation of the TOGAF 9.1 Full Content Metamodel, modeled using the ArchiMate 2.1 language.

The work exposed in the article is focused on harmonizing the specifications of both metamodels, but does not present a concrete solution of how we can materialize the transformation between a TOGAF model and an ArchiMate model or vice versa, which will be addressed in this research. The work also focus on earlier versions of the TOGAF and ArchiMate specifications. This research will focus on the recent versions of the two specifications and the main changes of both standards are analysed to understand if the alignment between ArchiMate and TOGAF has increased in some aspects.

From ArchiMate 2.1 to 3.1 and TOGAF 9.1 to 9.2
The previous work on ArchiMate and TOGAF Harmonization was made following Version 2.1 of the ArchiMate Specification and version 9.1 of the TOGAF Specification and their current versions stand on ArchiMate 3.1 and TOGAF 9.2. An overview of the main changes of both standards were made to understand if the alignment between ArchiMate and TOGAF increased in some aspects. In this subsection, the differences found between Version 2.1 and Version 3.1 of the ArchiMate specification will be presented. Then the differences found between Version 9.1 and Version 9.2 of the TOGAF specification will be addressed. Complementing this analysis with the presentation of new focuses of harmonization between the two specifications.

1. ArchiMate Specification 2.1 to 3.1

   The main changes encountered between version 2.1 and 3.1 of the ArchiMate Specification were the following [34]:
   - Changed various definitions and introduced new elements to increase alignment with the TOGAF framework
   - Restructured the set of relationships into structural, dynamic, dependency, and other relationships
   - Allowed relationships to other relationships in some cases;
   - Improved the derivation of relationships
   - Relaxed the constraints on relationships between layers in the ArchiMate core language
   - Renamed the “used by” relationship to “serving”, in line with the other active names of relationships
   - Changed the notation of the representation and contract elements, to distinguish these from deliverable and business object, respectively
   - Moved the value and meaning concepts from the Business Layer of the ArchiMate core language to the motivation elements
   - Moved the location element to the generic metamodel
   - Extended the Technology Layer with elements for modeling the physical world: facility, equipment, material, and distribution network

- Renamed the “communication path” element to “path” and extended its meaning, to integrate with the physical elements
- Created new tables of relationships based on the changes in the metamodel and derivation properties

The following elements were introduced to the ArchiMate metamodel:

Outcome	An end result that has been achieved	Outcome
Course of action	An approach or plan for configuring some capabilities and resources of the enterprise, undertaken to achieve a goal.	Course of Action
Capability	An ability that an active structure element, such as an organization, person, or system, possesses.	Capability
Resource	An asset owned or controlled by an individual or organization.	Resource
Value Stream	A value stream represents a sequence of activities that create an overall result for a customer, stakeholder, or end user.	Value stream
Application process	A sequence of application behaviors that achieves a specific outcome.	Process Process
Technology process	A sequence of technology behaviors that achieves a specific outcome.	Process Process
Application event	An application behavior element that denotes a state change.	Event Event
Technology event	A technology behavior element that denotes a state change.	Event Event
Technology interaction	A unit of collective technology behavior performed by (a collaboration of) two or more nodes.	Collaboration Collaboration
Implementation event	A behavior element that denotes a state change related to implementation or migration.	Event Event

Fig. 2: ArchiMate 3.1 - New Elements [34]

2. TOGAF Specification 9.1 to 9.2

From TOGAF 9.1 to 9.2 Specification we will focus on the changes made in the Content Metamodel section [35]:

- The following entities are added to the core metamodel: Business Capability, Course of Action, Value Stream
- Platform Services are renamed Technology Services.
- The metamodel diagrams and the metamodel relationships tables are revised.
- New relationships are added, and others have been changed. Changes have been made for consistency across the diagrams and relationships table.
- The Location entity is now a global entity, so text about it being added as part of the Infrastructure Consolidation extension is removed.

The following elements were introduced to the TOGAF Content Metamodel:

Business Capability	A particular ability that a business may possess or exchange to achieve a particular purpose.
Value Stream	A representation of an end-to-end collection of value-adding activities that create an overall result for a customer, stakeholder, or end-user.
Course of Action	Direction and focus provided by strategic goals and objectives, often to deliver the value proposition characterized in the business model.

Fig. 3: TOGAF 9.2 - New Elements [35]

From the analysis of the changes in both specifications (ArchiMate 3.1 and TOGAF 9.2) we encountered new entity mappings between ArchiMate Capability element and TOGAF Business Capability element, ArchiMate Value Stream element and TOGAF Value Stream element and ArchiMate Course of Action element and TOGAF Course of Action element.

2.6 Tool Integration Methodology

A rich variety of modeling languages and tools are available, not only to support the development of EA, but to support other domains, but there is no single language that can model all the specific concepts of each organization, so it is quite common that an organization resorts to several EA modeling tools each with its own language. The exchange of models among different tools and thus the integration of the respective languages becomes an important prerequisite to achieve an holistic view of the enterprise. Due to the lack of interoperability, however, it is often difficult to use tools in combination. [22] To achieve the necessary interoperability between modeling tools various methods were developed on model-based integration.

Model-Based Tool Integration
Model-based tool integration means that tools are integrated on basis of metamodels defining syntax and semantics of the modeling languages supported by the tools. [23] A metamodel is a special kind of model that represents the abstract syntax of a modeling language and conforms to a meta-metamodel.

According to [23] we can analyze the advances in model-based tool integration by two integration challenges. The integration of model syntax where model transformations occur (metamodel integration) and the field of semantic integration (concepts mapping). Within the context of Enterprise Architecture Model Integration, when there is a need to integrate architectural data specified in different models, there are two approaches that we can take: an holistic approach or a federated approach. [24]
In an federated approach, a single model composed with all the needed artifacts is used, and any information specified using other metamodel must be converted into it in order to be integrated (Fig.1).

In an holistic approach the existing models (that originate from specialized architectures) are used. These models are linked to a main model by metamodel integration (Fig.2).

Considering one or the other approach, the two integration challenges must be met because no matter what kind of integration orientation is considered, there is a need to overcome syntactical, structural and semantic discrepancy of metamodels, in order to join their concepts together. [26]

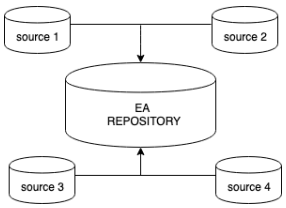


Fig. 4: EA Repository Federated Approach

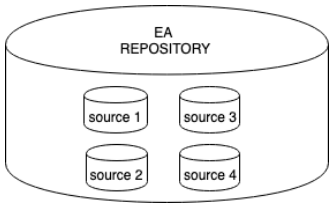


Fig. 5: EA Repository Holistic Approach

Model Transformation
Model transformation is one of the major building blocks in the context of model-based tool integration. [23] Many approaches have been developed, using model transformation languages, to tackle the problem of model-to-model transformations. [25] In model transformation the syntactical and structural discrepancy of metamodels are dealt with.

Syntactical heterogeneity represents the difference in formats intended for the serialization of metamodels. Two metamodelling platforms can base their serialization mechanisms on different formats (e.g. different XML based schemas). [26]
Structural heterogeneity concerns the representation of metamodels using different metamodelling languages. Even when agreed on the common metamodel, metamodels vary schematically when the same concepts being described are modelled in a different way. [26]
A range of model transformation approaches (considered as combinations of a transformation language and a tool for the language) have been developed. Some approaches focus on declarative specification, either logic-based or graph-theory based, others are imperative in nature, and others combine declarative and imperative aspects (hybrid approaches). [27]
In [31] a categorization and description of these approaches are made and some examples of each are given. In Fig. 3 a brief summary of each of them can be seen.

Model-to-Model Transformation Approaches		
Approach	**Description**	**Example**
Direct Manipulation	The most low-level approach. It offers the user little or no support or guidance in implementing transformations. Most of the work has to be done by the user. In the long run, this approach will become impractical.	Jamda
Structure Driven	Groups pragmatic approaches that were developed in the context of certain kinds of applications such as generating database schemas from UML models. These applications require a strong support for transforming models with a 1-to-1 and 1-to-n correspondence between source and target elements. It is unclear how well these approaches can support other kinds of applications.	OptimalJ, IOPT
Graph Transformation Based	Are inspired by heavily theoretical work in graph transformations. These approaches are powerful and declarative, but also the most complex ones. There is a large amount of theoretical work and some experience with research prototypes. However, experience with practical applications of these approaches is still limited.	BOTL, ATOM
Relational	Groups declarative approaches, based on mathematical relations. The main idea is ti state the source and target element type of a relation and specify it using constraints. Good balance between flexibility and declarative expression Implementable with logic programming	QVTP
Hybrid	Allow the user to mix and match different concepts and paradigms depending on the application. Practical approaches are very likely to have the hybrid character.	ATL, XDE
Others	These approaches can be used in a more varied range of model-to-model transformation contexts, on the other hand they show problems of scalability and lack of actual mechanism to implement model transformations.	XSLT, CWM

Fig. 6: Model-to-Model Transformations Categorization - retrieved from [31]

XSLT approach

Given that the Enterprise Architecture Modelling tools talked in the scope of this research can export (or import) models as XMI files, the research on XSLT declarative language model transformation approach it is the focus.
Because models developed with Enterprise Architecture Modelling tools can be serialized as Extensible Markup Language (XML) using the XML Metadata Interchange (XMI), implementing model transformations using XSLT, which is a standard technology for transforming XML, seems very attractive.
Extensible Stylesheet Language for Transformations (XSLT) is one of the W3C standards, it's a declarative rule-based programming language for transforming XML documents. An XSLT stylesheet consists of a set of rule templates, each rule template matches elements in source model, and produces output to the target model, transforming a XML file into another XML file. XSLT is the most common and powerful language for XML transformation and has strong support to complex pattern matching (Xpath). XSLT stylesheets can also be easily executed and integrated into different system environments and platforms, without additional packages and libraries. [28]

Semantic Integration

The mediation between semantic heterogeneities constitutes another crucial challenge for model-based tool integration. [23] *Semantic heterogeneity* represents the problem of concepts coming from different metamodels using the same linguistic terms to describe different concepts or use different terms to describe the same concept. [26]

To tackle the semantic integration challenge there have been several approaches already discussed, but in recent years ontologies became very popular. As model-based tool integration is able to perform tool metamodel integration on basis of

semantics covered by tool ontologies, these individual tool ontologies have to be integrated. [23] Based on a certain ontology integration architecture (direct mapping between ontologies, an indirect mapping via a common,shared ontology or a mapping based on a library of already mapped ontologies [29]), the mapping between ontologies has to be established, similar concepts have to be related to each other. This correspondence process between concepts of different ontologies is what is called *mapping discovery*. There are several mapping discovery techniques, it can be done in a fully manual way or by utilizing heuristic-based or machine learning techniques. [23]
In [26] the mapping technique described distinguishes different types of mappings, being the most relevant for the scope of this work the *1)class-to-class mappings*, which relate a concept from metamodel A to a concept of metamodel B and *2)relation-to-relation mappings*, which relate concept relationships from metamodel A to concept relationships of metamodel B.
These mappings can be represented in different forms. Following an approach similar to traditional data integration, views can be used to describe these mappings or with the use of transformation rules to specify how values should be changed and conditions and effects of such rules. [23]

3 Proposed Solution

In this section will be presented in detail the proposed solution for the integration between ArchiMate 3.1 and TOGAF 9.2 metamodels. The two specifications are widely used in the development of Enterprise Architecture and due to the fact that they are both developed by The Open Group, they were designed to be able to be used harmoniously. However, as previously discussed, using different specifications in the development of EA, having spread across different domains of the enterprise, EA models created with different metamodels, often leads to the lack of an enterprise-wide view. In order to be able to have all models in the same repository and achieve an enterprise-wide view, we present a solution that focuses on integrating these two specifications to be able to consolidate models developed with the two different metamodels, TOGAF and ArchiMate, in the same EA repository, through the Enterprise Cartography tool Atlas.
In order to achieve this objective the solution presented follows the following phases:

1. Concepts Mapping

   To understand which mappings of entities can be used in the transformation between metamodels, in this case between TOGAF entities and ArchiMate entities, an analysis of the meaning of each one is necessary. Following the work done in [32] and research on the new entities added in the TOGAF 9.2 specification and ArchiMate 3.1 specification already discussed in the relates work section, the result we want to achieve in this phase is the materialization of the mapping between ArchiMate entities and TOGAF entities and vice versa.

2. Relationships Mapping

   To understand which mappings of relationships can be used in the transformation between metamodels, in this case between TOGAF entities and ArchiMate relationships, an analysis of the meaning of each one is necessary. The main challenge on this phase is the mapping between relationships that don't have a direct mapping and which need to be defined. Following the work done in [32], the result we want to achieve in this phase is the materialization of the mapping between ArchiMate and TOGAF relationships.

3. XSLT Transformation Approach

   Using the XSLT approach to create a metamodel transformation mechanism, it is necessary to define three files to be run in a batch and inserted in the Atlas repository. Two of the files contain the definition of the metamodels, a file with the definition of the entities that are in the specification TOGAF 9.2 and a file with the definition of the entities that are in the specification ArchiMate 3.1 need to be defined. The third file contains the

mapping rules between entities and relationships from the origin model to the target model. The mapping rules defined in this XSLT file are not general to all TOGAF to ArchiMate transformations or vice versa. Each model to be transformed must have a specific XSLT file with its mapping rules defined as the applied rules will be different from model to model. Since the ArchiMate 3.1 specification is already available in the Atlas repository, the objectives for this phase are the definition of the XSLT file with the TOGAF 9.2 specification to be inserted in the Atlas repository and the understanding of the mapping rules that the Atlas tool offers and which ones will be used in order to achieve the model transformation without losing meaning nor coherence.

3.1 Concepts Mapping

Through the analysis of the work done in TOGAF® Framework and ArchiMate® Modeling Language Harmonization white paper [32] and taking into account the new additional entities added to the ArchiMate and TOGAF standards talked in the Related Work section **From ArchiMate 2.1 to 3.1 and TOGAF 9.1 to 9.2**, in this section it will be presented each concept of ArchiMate 3.1 and TOGAF 9.2 and the concept mapping that will be used in the solution presented for the ArchiMate and TOGAF metamodel integration will be exposed.

TOGAF 9.2 Content Metamodel Entities

The TOGAF Content Metamodel is expressed by a set of elements that represent the types of building blocks that can be found on a TOGAF architecture specification. The element types are related to one another by a set of relationships and belong to a specific architecture(Business, Data, Application and Technology architecture). Each entity type has its set of attributes. The TOGAF Content Metamodel is composed of two parts. The Core Content Metamodel describes a minimal set of entities needed to model the fundamental aspects of an enterprise from an EA perspective. In addition, the Content Metamodel also provides a set of metamodel extensions that support additional detailed modeling capabilities for concerns such as Motivation, Process, and Governance [32]. In Annex D we can find tables with the description of the elements that are included in the TOGAF 9.2 Full Content Metamodel as described in Section 30.5 of the TOGAF 9.2 standard [35]. The following image shows the elements that are included in the TOGAF 9.2 Full Content Metamodel as described in Section 30.5 of the TOGAF 9.2 standard [35].

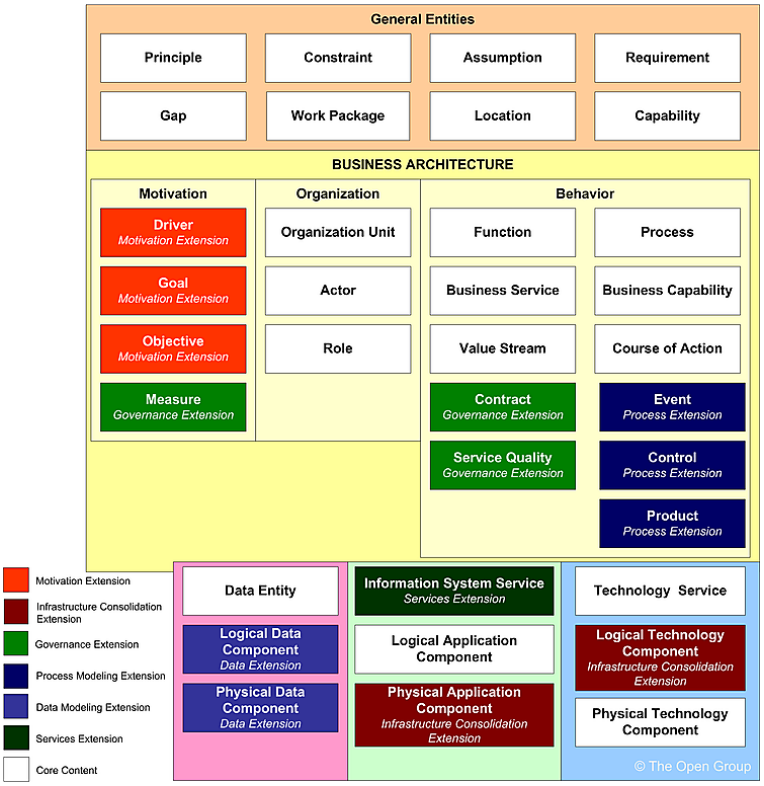


Fig. 7: TOGAF Content Metamodel - retrieved from [35]

ArchiMate 3.1 Entities

The ArchiMate modeling language defines a set of entities very similarly to the TOGAF standard. The ArchiMate core language is based on a set of generic elements and their relationships, which can be specialized in different layers - Business Layer, Application Layer, Technology Layer . The TOGAF Business, Application, and Technology architecture domains correspond with the ArchiMate Business, Application, and Technology layers.
Elements in the ArchiMate language are classified as Active Structure, Passive Structure or Behavior Elements. The entities that compose the TOGAF Data Architecture correspond with the ArchiMate Passive Structure elements and we can find them across the layers of the ArchiMate language.
In the ArchiMate language there is a differentiation between an internal and external view of a system. Services are described and seen in the ArchiMate language as units of functionality that a system exposes to its environment through interfaces to provide a certain value. An external view defines then the system as a set of services and interfaces that can be used by other systems to interact with the system in question. An internal view reveals the parts of the system that are hidden in the external view. The TOGAF standard does not provide internal and external views of the system, instead, the TOGAF metamodel defines a service as a governed way of exposing functions.
The ArchiMate language also differentiates between behaviors that are performed by an individual structure element and those that are performed by a collaboration of multiple structure elements. The TOGAF Content Metamodel does not classify elements in this way. [32]
In Annex E we can find tables with the description of the elements that are included in the ArchiMate 3.1 metamodel and its extensions, as described in Section 6 to 10 of the ArchiMate 3.1 standard [26]. The following image shows the elements that are included in the ArchiMate 3.1 metamodel and its extensions, as described in Section 6 to 10 of the ArchiMate 3.1 standard [26].

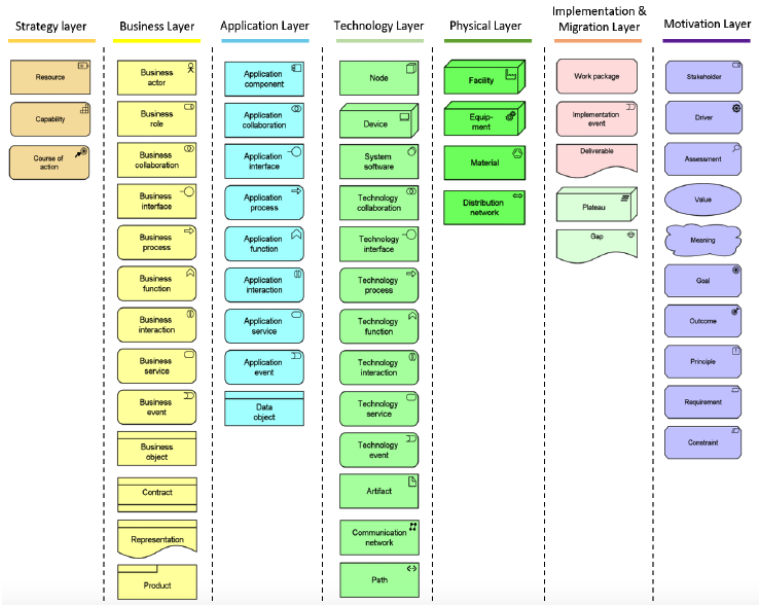


Fig. 8: ArchiMate Metamodel Entities - retrieved from [26]

Mapping TOGAF and ArchiMate Metamodel Entities

Some of the concepts found in the two metamodels share very similar names and analyzing their description we find that they also share the same meaning, that is, they have a very strong alignment. There are other concepts, which do not have a similar name but still maintain a good alignment, meaning that the concept describes a similar meaning. However, there are concepts that do not have this strong alignment, but that we can still find some kind of similarities. And then there are concepts that do not have any type of mapping, that is, the concept does not exist in one of the metamodels.
Only concepts with mapping, even if weak, will be mentioned. Next we will analyse for each ArchiMate concept the TOGAF concepts that can be aligned with it.

- *Business Actor*
  ArchiMate describes a business actor as a business entity that is capable of carrying out behavior. A business actor can include entities outside the real organization and can represent those business entities at different levels of detail. This definition is found in two entities in the TOGAF metamodel, it can correspond to an Actor or an Organizational Unit.

- *Business Function*
  ArchiMate uses the same definition for a Business Function as TOGAF for its entity Function , defines it as a collection of business capabilities closely aligned to an organization, but not necessarily explicitly governed by the organization.

- Business Object Both the Business Object concept in ArchiMate and the Data Entity concept in TOGAF represent a concept used within a specific business domain. Even though a Buiness Object can represent more than just data tied to applications and services, it is a more general concept than Data Entity, a fair mapping is defined between the two elements.
- *Business Process*
  Both the Business Process concept in ArchiMate and the Process concept in TOGAF represent a sequence of business behaviors that achieve a specific result.

- Business Role
  Both the Business Role concept in ArchiMate and the Role concept in TOGAF represent the performance of behavior or function assigned to an actor.

- Business Service
  Both the Business Service concept in ArchiMate and the Business Service concept in TOGAF represent behavior that provides specific functionality in response to requests from actors or other services and is explicitly defined / governed by a role, actor, or business collaboration.

- *Contract*
  Both the Contract concept in ArchiMate and the Contract concept in TOGAF represents an agreement between a provider and consumer that establishes functional and non-functional parameters for interaction.

- *Business Event*
  Both the Business Event concept in ArchiMate and the Event concept in TOGAF represents an organizational state change.

- *Location*
  Both the Location concept in ArchiMate and the Location concept in TOGAF represents a place, conceptual or physical, where business activity is located or performed.

- *Product*
  Both the Product concept in ArchiMate and the Product concept in TOGAF represents an output generated by the business. But it is important to consider that in ArchiMate the product is also defined by a coherent collection of services and / or passive structure elements, accompanied by a contract / set of agreements.

- *Application Service*
  The Application Service concept in ArchiMate has a fair mapping with the Information System Service concept in TOGAF. In TOGAF an Information System Service represents the automated elements of a business service. In ArchiMate the Application Service concept does not have this exact meaning but since an Application Service exposes the functionality of components to their environment and it may serve business processes, business functions, business interactions, or application functions and taking into account that these elements realize a business service we can find a fair mapping.

- *Data Object*
  The Data Object concept in ArchiMate has a strong mapping with the Data Entity and Logical Data Component concept in TOGAF. In TOGAF a Data Entity represents a data encapsulation that is recognized by a business domain expert as a thing. The logical data entites can be linked to applications, repositories and services. The concept of Logical Data Component in TOGAF represents the boundary zone that encapsulates related data entities to form a logical location to be maintained. In ArchiMate we can find a strong mapping between the Data Object concept and these two TOGAF entities since the Data Object entity represents a self-contained piece of information with a clear meaning to the business and can be accesed by an application function, application interaction, or application service.

- *Application Component*
  The Application Component concept in ArchiMate has a strong mapping

with the Physical Application Component and Logical Application Component concept in TOGAF. In TOGAF a Physical Application Component can represents a application, application module, application service, or other deployable component of functionality. The concept of Logical Application Component in TOGAF represents encapsulation of application functionality that is independent of a particular implementation.. In ArchiMate we can find a strong mapping between the Application Component concept and these two TOGAF entities since the Application Component represents a self-contained unit. As such, it is independently deployable, re-usable, and replaceable. An application component performs one or more application functions. It encapsulates its behavior and data, exposes services, and makes them available through interfaces.

– *Application Function*
The Application Function concept in ArchiMate has a strong mapping with the Logical Application Component concept in TOGAF. Maybe not immediately apparent from the definition of Application Function, which defines an automated behavior that can be performed by an application component ,but this concept is clearly intended to capture the same 'implementation-independent encapsulation of application functionality' as the Logical Application Component entity describes.

– *Artifact*
The Artifact concept in ArchiMate has a fair mapping with the Physical Data Component concept in TOGAF. In ArchiMate a Artifact represents a piece of data that is used or produced in a software development process, or by deployment and operation of an IT system describing a "physical" element in the IT world, and in TOGAF we find in the Physical Data Component concept the representation of a boundary zone that encapsulates related data entities to form a physical location to be held.

– *System Software*
The System Software concept in ArchiMate has a strong mapping with the Physical Technology Component concept in TOGAF, since the Physical Technology Component has broad representation. In ArchiMate a System Software represents a software that provides or contributes to an environment for storing, executing, and using software or data deployed within it and in TOGAF the Physical Technology Component has a broad representation that includes all that is specific technology infrastructure product or technology infrastructure product instance.

– *Device*
The Device concept in ArchiMate has a strong mapping with the Physical Technology Component concept in TOGAF, since the Physical Technology Component has broad representation. In ArchiMate a Device represents a physical IT resource upon which system software and artifacts may be stored

or deployed for execution and in TOGAF the Physical Technology Component has a broad representation that includes all that is specific technology infrastructure product or technology infrastructure product instance.

- *Node*
  The Node concept in ArchiMate has a strong mapping with the Logical Technology Component concept in TOGAF. In ArchiMate a Node represents a computational or physical resource that hosts, manipulates, or interacts with other computational or physical resources and a Logical Technology Component concept englobes this description since it represents an encapsulation of technology infrastructure that is independent of a particular product. It can be defined as a class of technology product.

- *Communication Network*
  The Communication Network concept in ArchiMate has a fair mapping with the Physical Technology Component concept in TOGAF, it can be seen as a special kind of PhysicalTechnology Component . In ArchiMate a Communication Network represents a set of structures that connects nodes for transmission, routing, and reception of data and a Physical Technology Component concept englobes this description since it represents a specific technology infrastructure product or technology infrastructure product instance.

- *Path*
  The Path concept in ArchiMate has a fair mapping with the Logical Technology Component concept in TOGAF, it can be seen as a special kind of Logical Technology Component. In ArchiMate a Path represents a set of structures that connects nodes for transmission, routing, and reception of data and a Logical Technology Component concept englobes this description since it represents an encapsulation of technology infrastructure that is independent of a particular product. It can be defined as a class of technology product.

- *Gap*
  Both the Gap concept in ArchiMate and the Gap concept in TOGAF represents a statement of difference between two states / plateaus.

- *Work Package*
  Both the Work Package concept in ArchiMate and the Work Package concept in TOGAF represents a set of actions identified and designed to achieve results specified by the business.

- *Stakeholder*
  The Stakeholder concept in ArchiMate has a fair mapping with the Role concept and Organization Unit concept in TOGAF. Since in ArchiMate a Stakeholder represents the role of an individual, team, or organization (or classes thereof) that represents their interests in the effects of the archi-

tecture in TOGAF it has a fair mapping to a Role and Organization Unit concepts.

- *Driver*
  Both the Driver concept in ArchiMate and the Driver concept in TOGAF represents an external or internal condition that motivates the organization to define its goals.

- *Assessment*
  The Assessment concept in ArchiMate has a fair mapping with the Measure concept in TOGAF. The TOGAF Measure entity is an indicator or factor that is linked to a goal or objective. The ArchiMate Assessment entity is the outcome of the analysis of some driver.

- *Goal*
  The Goal concept in ArchiMate has a strong mapping with the Goal concept in TOGAF and also the Objective concept in TOGAF. oth the Goal concept in ArchiMate and the Goal concept in TOGAF represents a high-level statement of intent, direction, or desired end state for an organization. And since TOGAF Objective entity is used in the attainment of a goal and the ArchiMate goal entity can be decomposed to represent an objective we can also find here a strong mapping.

- *Principle*
  Both the Principle concept in ArchiMate and the Principle concept in TOGAF represents a statement of intent defining a general property that should be met by the architecture.

- *Requirement*
  Both the Requirement concept in ArchiMate and the Requirement concept in TOGAF represents a statement of business need defining a property that applies to a specific system as described by the architecture.

- *Constraint*
  Both the Constraint concept in ArchiMate and the Constraint concept in TOGAF represents a factor that prevents an organization from pursuing particular approaches to meet its goals.

- *Capability*
  Both the Capability concept in ArchiMate and the Business Capability concept in TOGAF represents an ability thay a business possess.

- *Course of Action*
  Both the Course of Action concept in ArchiMate and the Course of Action concept in TOGAF represents an approach or plan based on goals and ob-

jectives to achieve/deliver a goal.

– *Value Stream*
Both the Value Stream concept in ArchiMate and the Value Stream concept in TOGAF represents a sequence of activities that create an overall result for a customer, stakeholder, or end user.

Through the detailed semantic and comparative analysis of the entities of the respective metamodels, and after analyzing the mapping covered and presented in [32], the mapping between the entities of the TOGAF 9.2 standard and the ArchiMate 3.1 standard that will be used for the transformation between metamodels is exposed in the following tables:

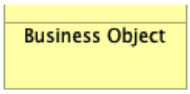
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Business Actor		Actor
Business Actor		Organization Unit
Business Function		Function
Business Object		Data Entity
Business Process		Process
Business Role		Role
Business Service		Business Service
Contract		Contract
Business Event		Event
Location		Location
Product		Product

Table 2: Concepts Mapping - Business Domain

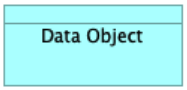
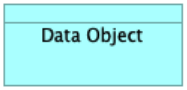
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Application Service		Information System Service
Data Object		Date Entity
Data Object		Logical Data Component
Application Component		Logical Application Component
Application Component		Physical Application Component
Application Function		Logical Application Component

Table 3: Concepts Mapping - Application Domain

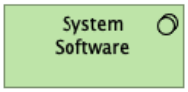
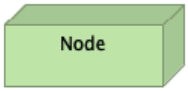
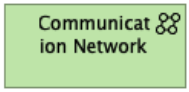
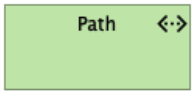
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Artifact		Physical Data Component
System Software		Physical Technology Component
Device		Physical Technology Component
Node		Logical Technology Component
Communication Network		Logical Technology Component
Path		Logical Technology Component

Table 4: Concepts Mapping - Technology Domain

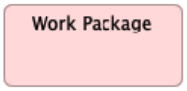
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Gap		Gap
Work Package		Work Package

Table 5: Concepts Mapping - Implementation and Migration Extension

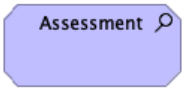
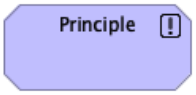
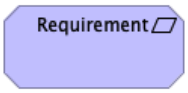
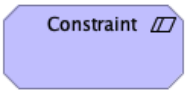
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Stakeholder		Organization Unit
Stakeholder		Role
Driver		Driver
Assessment		Measure
Goal		Goal
Goal		Objective
Principle		Principle
Requirement		Requirement
Constraint		Constraint

Table 6: Concepts Mapping - Motivation Extension

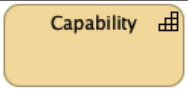
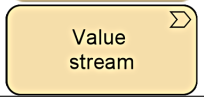
ArchiMate Concept	ArchiMate Figure	TOGAF Concept
Capability		Business Capability
Course of Action		Course of Action
Value Stream		Value Stream

Table 7: Concepts Mapping - Strategic Extension

3.2 Relationships Mapping

In TOGAF, relationships are represented by lines with labels that describe their meaning, indicating how the source entity relates to the target entity.
In ArchiMate there are explicit types of relationship symbols. The meaning of each symbol is dependent on what is the source entity and what is the target entity.
Although most TOGAF relationships have similar conceptual meanings when reading from the two directions, target to source and source to target, there are cases when this is not true. For example, an Actor interacts with a Function, and the Function supports or is performed by the Actor. In this example we can say that the relationship carries the same meaning in both directions, even though the words used are not in exactly the same way. But if we look to the example of the relationship between a Process and a Business Service the scenario is different. A process "orchestrates" a business service and the business service "supports" a process, in this cases there must be taken into account important considerations when making the harmonization between TOGAF standard and the ArchiMate standard.

Having different directional correspondence, leads to a larger number of different kinds of relationships. In TOGAF, unlike in ArchiMate, there are no relationship types classification nor relationship meaning classification, so the number of relationships between entities we encounter is much larger, even though some of the meaning are similar.
In mapping relationships it is necessary to take into account not only the meaning of the relationship itself but also its use between concepts, and if it is possible or not when the transformation between standards occur. In this section we will present the relationship mapping between the two standards.

Detailed TOGAF 9.2 and ArchiMate 3.1 Relationship Mapping

The TOGAF standard defines its metamodel with associations that are named to describe their meanings. The relationship names are different going from one entity type to another. To understand which mapping should be done to one of ArchiMate's relationships its meaning must be decoded
In TOGAF every element can have decomposition. This type of relationship can be seen in ArchiMate as the Composition relationship. It some ways the Aggregation relationship is also appropriate, even though the concept is not explicitly considered in the TOGAF standard, so the mapping would depend on the context.
For each TOGAF entity we will present all the other relationships mapping to ArchiMate.
In the solution proposed we categorized the relationship mapping in two groups:

- **Direct Relationships:** TOGAF relationships that have a direct mapping to one of ArchiMate's relationships.

– **Derived Relationships:** TOGAF relationships that don't have a direct mapping to one of ArchiMate's relationships and need a more complex transformation to achieve the same meaning.

Actor

– Actor - Organization Unit

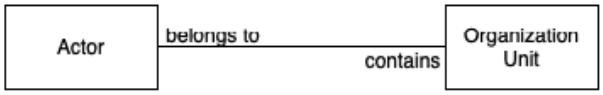


Fig. 9: TOGAF Relationship: Actor - Organization Unit

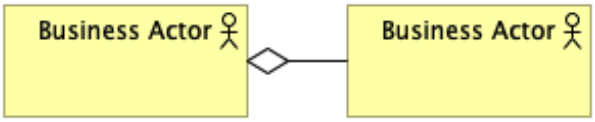


Fig. 10: ArchiMate Relationship Mapping: Actor - Organization Unit

– Actor - Actor

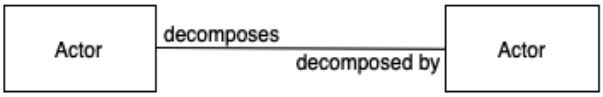


Fig. 11: TOGAF Relationship: Actor - Actor

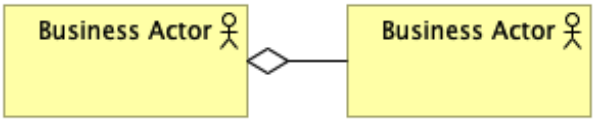


Fig. 12: ArchiMate Relationship Mapping: Actor - Actor

– Actor - Function

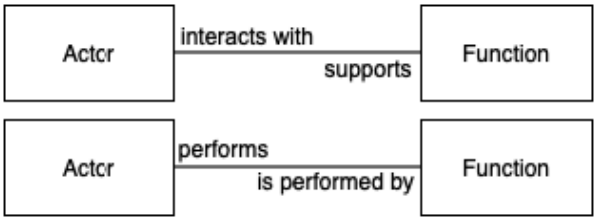


Fig. 13: TOGAF Relationship: Actor - Function

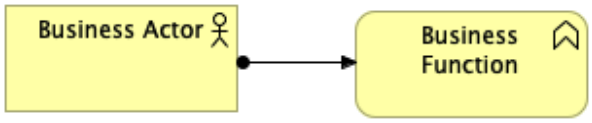


Fig. 14: ArchiMate Relationship Mapping: Actor - Function

– Actor - Role

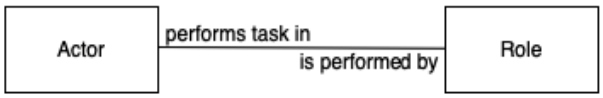


Fig. 15: TOGAF Relationship: Actor - Role

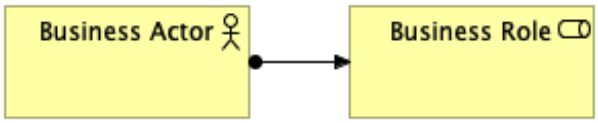


Fig. 16: ArchiMate Relationship Mapping: Actor - Role

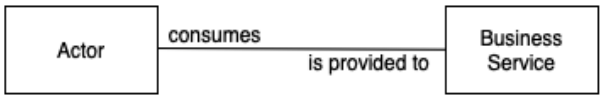


Fig. 17: TOGAF Relationship: Actor - Business Service

– Actor - Business Service

Fig. 18: ArchiMate Relationship Mapping: Actor - Business Service

– Actor - Event

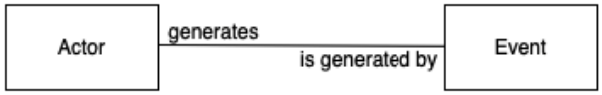


Fig. 19: TOGAF Relationship: Actor - Event (1)

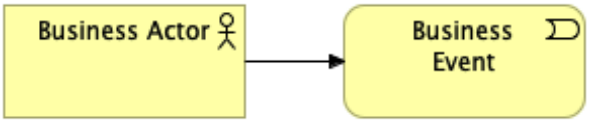


Fig. 20: ArchiMate Relationship Mapping: Actor - Event (1)

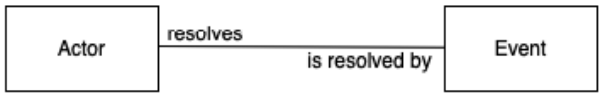


Fig. 21: TOGAF Relationship: Actor - Event (2)

The "resolves" relationship characterizes an action taken as a result of a process or the execution of a business service. To try to achieve this meaning in ArchiMate the following derived relationship must be created.

Fig. 22: ArchiMate Relationship Mapping: Actor - Event (2)

– Actor - Location

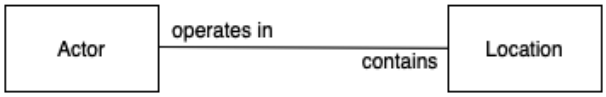


Fig. 23: TOGAF Relationship: Actor - Location

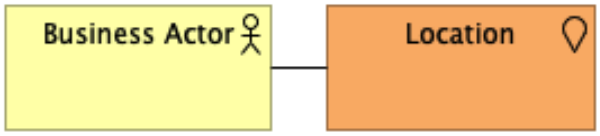


Fig. 24: ArchiMate Relationship Mapping: Actor - Location

– Actor - Data Entity The "supplies/consumes" relationship characterizes an

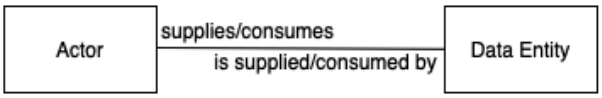


Fig. 25: TOGAF Relationship: Actor - Data Entity

information concept that is given or used by an Actor. To achieve this meaning in ArchiMate and considering the context the following relationships can be created.

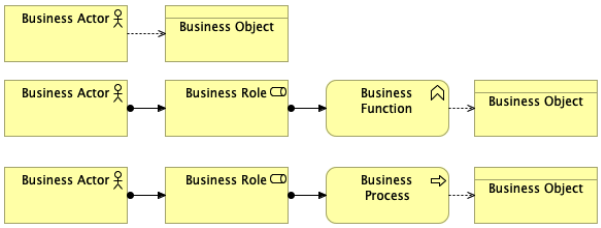


Fig. 26: ArchiMate Relationship Mapping: Actor - Data Entity

Business Capability

– Business Capability - Course of Action

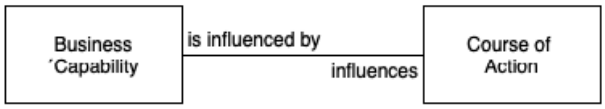


Fig. 27: TOGAF Relationship: Business Capability - Course of Action

Fig. 28: ArchiMate Relationship Mapping: Business Capability - Course of Action

– Business Capability - Function

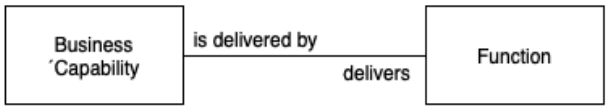


Fig. 29: TOGAF Relationship: Business Capability - Function

The "is delivered by" relationship characterizes a deliverable generated by a specific activity, in this case a Function that delivers value through a Capability. To achieve this meaning in ArchiMate and considering the context,

the mapping of this relationship can be made by the association or the realization relationship in ArchiMate.

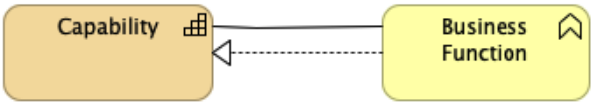


Fig. 30: ArchiMate Relationship Mapping: Business Capability - Function

– Business Capability - Organization Unit

Fig. 31: TOGAF Relationship: Business Capability - Organization Unit

The "is used by" relationship between a Business Capability and Organization Unit characterizes the Organization Unit's use of a specific skill that a company may possess or exchange to achieve a specific objective. This ability is generated many times by the Organization Unit itself, so, following the context, the mapping of this relationship can be made by the association or the realization relationship in ArchiMate.

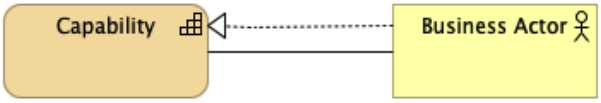


Fig. 32: ArchiMate Relationship Mapping: Business Capability - Organization Unit

– Business Capability - Value Stream

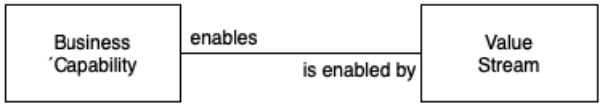


Fig. 33: TOGAF Relationship: Business Capability - Value Stream

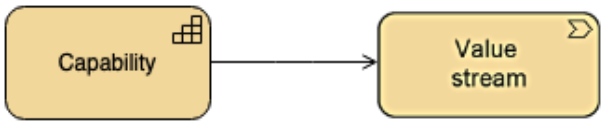


Fig. 34: ArchiMate Relationship Mapping: Business Capability - Value Stream

– Business Capability - Process

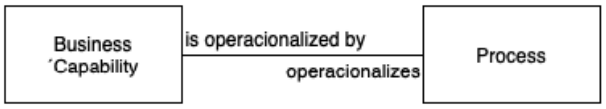


Fig. 35: TOGAF Relationship: Business Capability - Process

Taking into account the context, the mapping of this relationship can be made by the association or the realization relationship in ArchiMate.

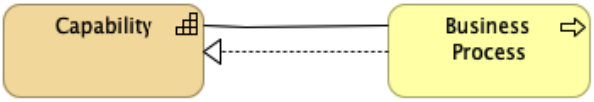


Fig. 36: ArchiMate Relationship Mapping: Business Capability - Process

Business Service

– Business Service - Data Entity

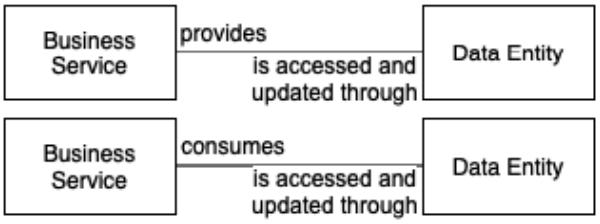


Fig. 37: TOGAF Relationship: Business Service - Data Entity

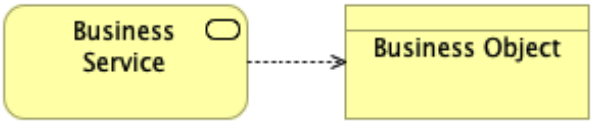


Fig. 38: ArchiMate Relationship Mapping: Business Service - Data Entity

– Business Service - Business Service

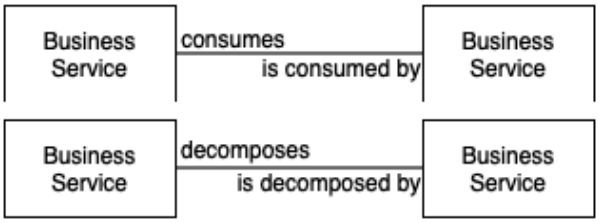


Fig. 39: TOGAF Relationship: Business Service - Business Service

Taking into account the context, the mapping of this relationship can be made by the aggregation or the composition relationship in ArchiMate.

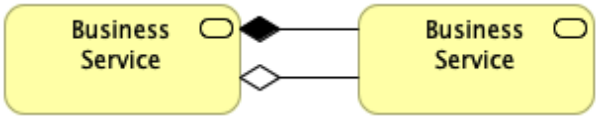


Fig. 40: ArchiMate Relationship Mapping: Business Service - Business Service

– Business Service - Process

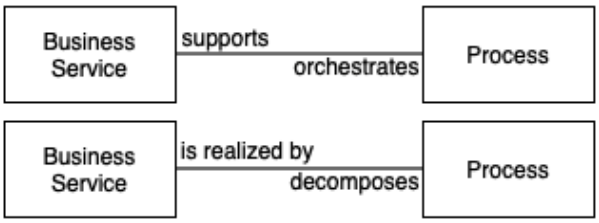


Fig. 41: TOGAF Relationship: Business Service - Process

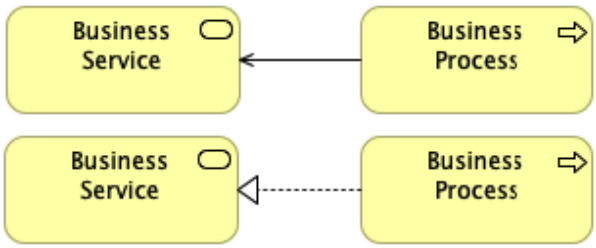


Fig. 42: ArchiMate Relationship Mapping: Business Service - Process

– Business Service - Contract

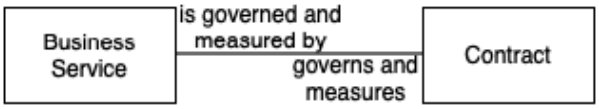


Fig. 43: TOGAF Relationship: Business Service - Contract

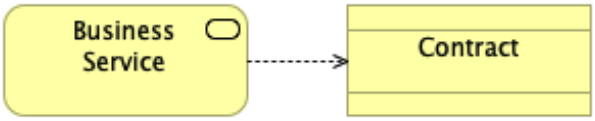


Fig. 44: ArchiMate Relationship Mapping: Business Service - Contract

– Business Service - Logical Technology Component

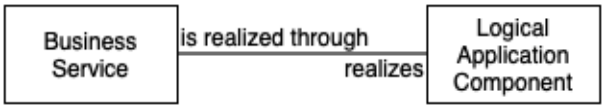


Fig. 45: TOGAF Relationship: Business Service - Logical Technology Component

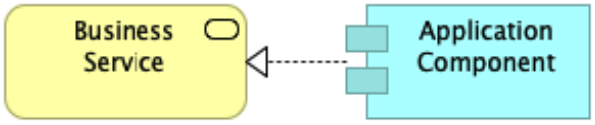


Fig. 46: ArchiMate Relationship Mapping: Business Service - Logical Technology Component

– Business Service - Organization Unit

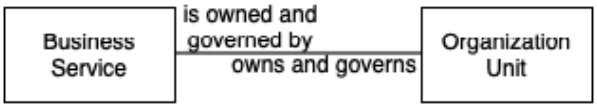


Fig. 47: TOGAF Relationship: Business Service - Organization Unit

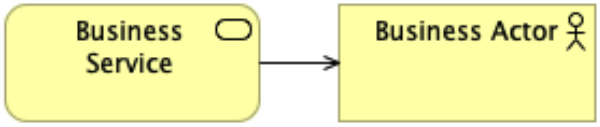


Fig. 48: ArchiMate Relationship Mapping: Business Service - Organization Unit

– Business Service - Information System Service

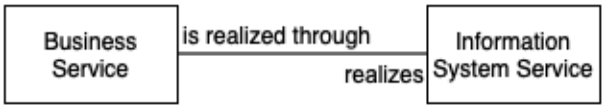


Fig. 49: TOGAF Relationship: Business Service - Information System Service

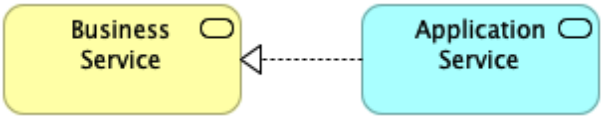


Fig. 50: ArchiMate Relationship Mapping: Business Service - Information System Service

– Business Service - Function

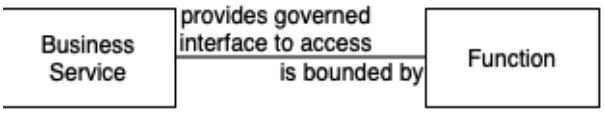


Fig. 51: TOGAF Relationship: Business Service - Function

Fig. 52: ArchiMate Relationship Mapping: Business Service - Function

– Business Service - Event

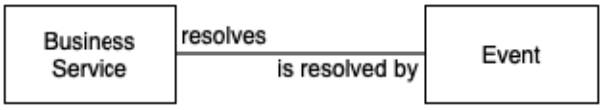


Fig. 53: TOGAF Relationship: Business Service - Event

The "resolves" relationship characterizes an action taken as a result of a process or the execution of a business service. Taking into account the context, this relationship can be mapped by a direct or derived relationship as followed

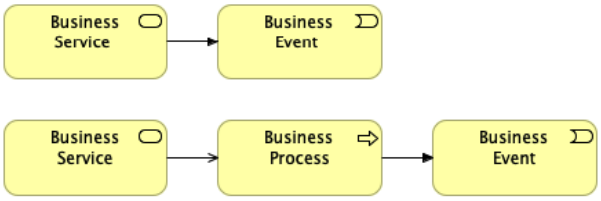


Fig. 54: ArchiMate Relationship Mapping: Business Service - Event

Course of Action

– Course of Action - Function

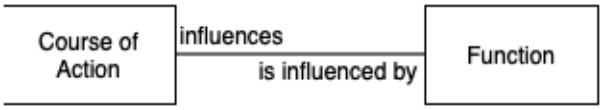


Fig. 55: TOGAF Relationship: Business Service - Function

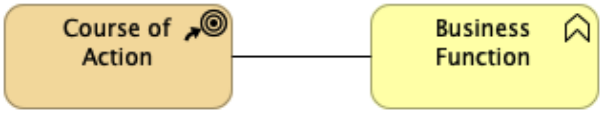


Fig. 56: ArchiMate Relationship Mapping: Business Service - Function

– Course of Action - Value Stream

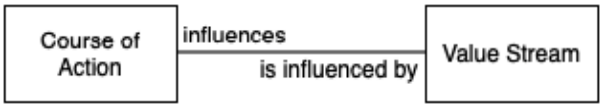


Fig. 57: TOGAF Relationship: Business Service - Value Stream

Taking into account the context, the mapping of this relationship can be made by the serving, association or the realization relationship in ArchiMate.

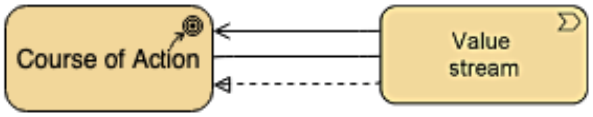


Fig. 58: ArchiMate Relationship Mapping: Business Service - Value Stream

– Course of Action - Goal

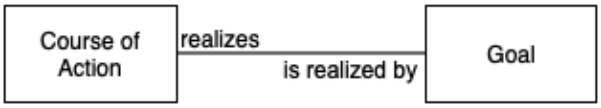


Fig. 59: TOGAF Relationship: Business Service - Goal

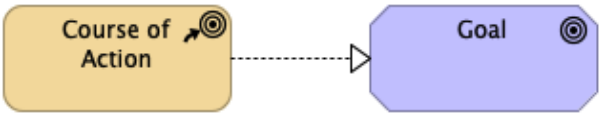


Fig. 60: ArchiMate Relationship Mapping: Business Service - Goal

Data Entity

– Data Entity - Information System Service

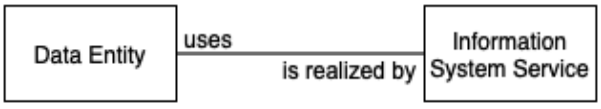


Fig. 61: TOGAF Relationship: Business Service - Information System Service

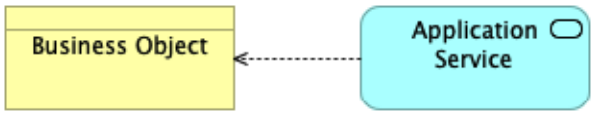


Fig. 62: ArchiMate Relationship Mapping: Business Service - Information System Service

Event

– Event - Process

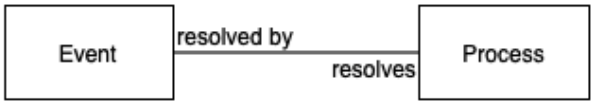


Fig. 63: TOGAF Relationship: Event - Process

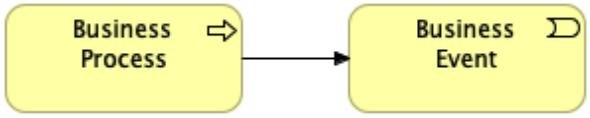


Fig. 64: ArchiMate Relationship Mapping: Event - Process

Function

– Function - Organization Unit

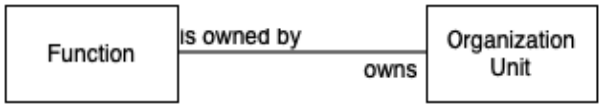


Fig. 65: TOGAF Relationship: Function - Organization Unit

The 'is owned by' relationship characterizes Governance meaning and stewardship. In ArchiMate there is no relationship that passes this meaning . The following derived relationships can achieve a similar connotation but

the ownership might not be well represented. This could be complemented using additional attributes through profiles that could represent ownership as a management and governance concept.

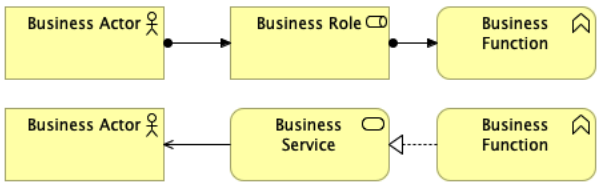


Fig. 66: ArchiMate Relationship Mapping: Function - Organization Unit

– Function - Process

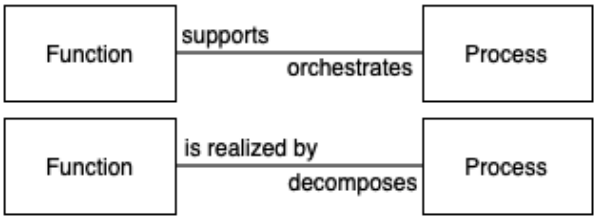


Fig. 67: TOGAF Relationship: Function - Process

Taking into account the context, the mapping of this relationship can be made by the aggregation, triggering or the flow relationship in ArchiMate.

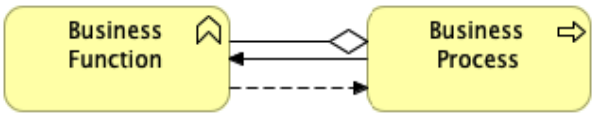


Fig. 68: ArchiMate Relationship Mapping: Function - Process

– Function - Function

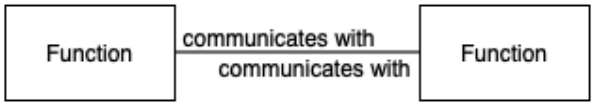


Fig. 69: TOGAF Relationship: Function - Function

Taking into account the context, the mapping of this relationship can be made by the aggregation, composition or the triggering relationship in ArchiMate.

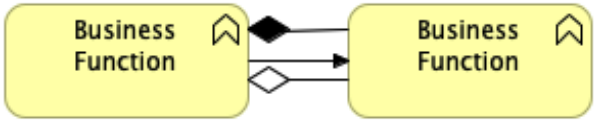


Fig. 70: ArchiMate Relationship Mapping: Function - Function

Goal

– Goal - Goal

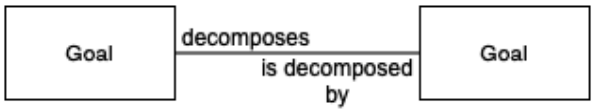


Fig. 71: TOGAF Relationship: Goal - Goal

Fig. 72: ArchiMate Relationship Mapping: Goal - Goal

– Goal - Driver

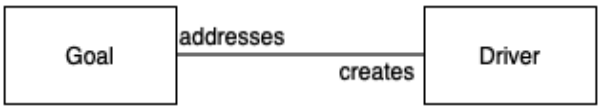


Fig. 73: TOGAF Relationship: Goal - Driver

Taking into account the context, the mapping of this relationship can be made by the association or the influence relationship in ArchiMate.

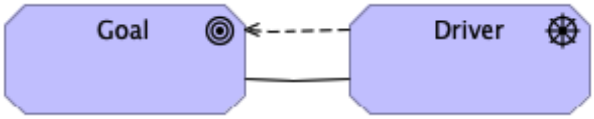


Fig. 74: ArchiMate Relationship Mapping: Goal - Driver

Information System Service

– Information System Service - Logical Application Component

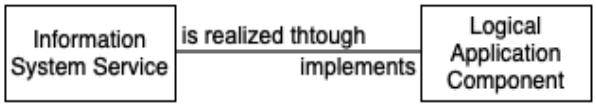


Fig. 75: TOGAF Relationship: Information System Service - Logical Application Component

Taking into account the context, the mapping of this relationship can be made by the realization or the serving relationship in ArchiMate.

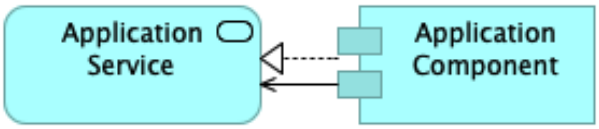


Fig. 76: ArchiMate Relationship Mapping: Information System Service - Logical Application Component

Location

– Location - Organization Unit

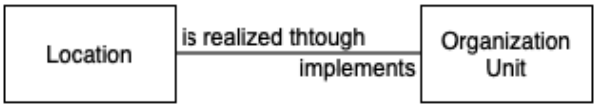


Fig. 77: TOGAF Relationship: Location - Organization Unit

Fig. 78: ArchiMate Relationship Mapping: Location - Organization Unit

– Location - Physical Data Component

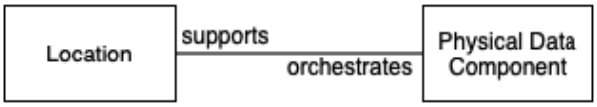


Fig. 79: TOGAF Relationship: Location - Physical Data Component

Fig. 80: ArchiMate Relationship Mapping: Location - Physical Data Component

– Location - Physical Technology Component

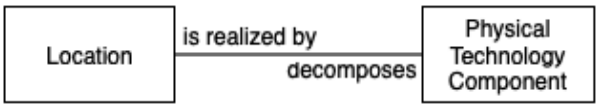


Fig. 81: TOGAF Relationship: Location - Physical Technology Component

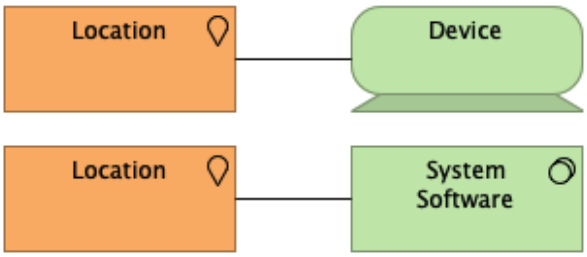


Fig. 82: ArchiMate Relationship Mapping: Location - Physical Technology Component

Logical Application Component

– Logical Application Component - Logical Application Component

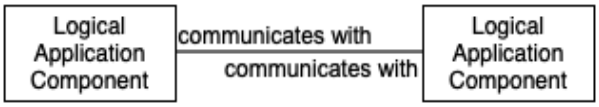


Fig. 83: TOGAF Relationship: Logical Application Component - Logical Application Component

Taking into account the context, the mapping of this relationship can be made by the aggregation, composition or the triggering relationship in ArchiMate.

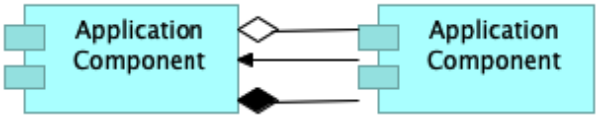


Fig. 84: ArchiMate Relationship Mapping: Logical Application Component - Logical Application Component

– Logical Application Component - Logical Technology Component

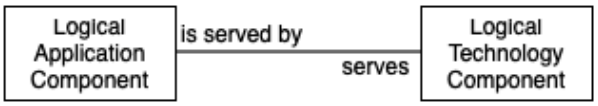


Fig. 85: TOGAF Relationship: Logical Application Component - Logical Technology Component

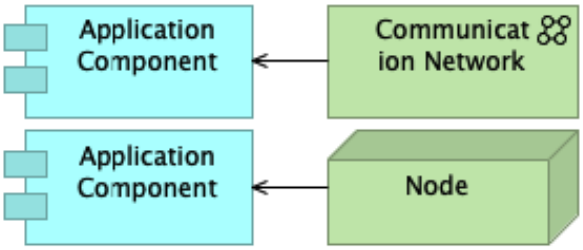


Fig. 86: ArchiMate Relationship Mapping: Logical Application Component - Logical Technology Component

Logical Technology Component

– Logical Technology Component - Physical Technology Component

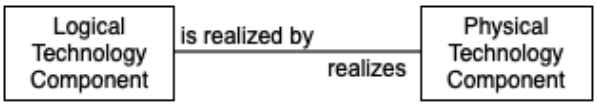


Fig. 87: TOGAF Relationship: Logical Technology Component - Physical Technology Component

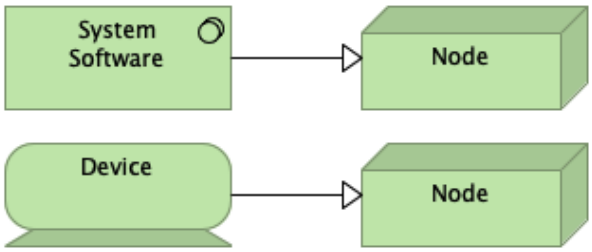


Fig. 88: ArchiMate Relationship Mapping: Logical Technology Component - Physical Technology Component

– Logical Technology Component - Logical Technology Component

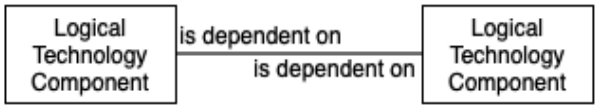


Fig. 89: TOGAF Relationship: Logical Technology Component - Logical Technology Component

Taking into account the context, the mapping of this relationship can be made by the aggregation or the composition relationship in ArchiMate.

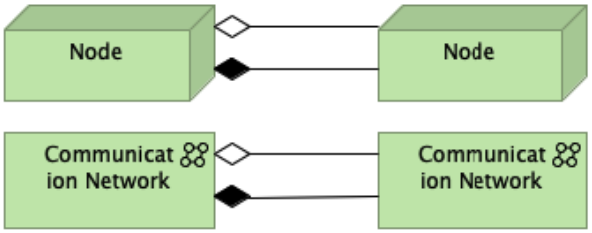


Fig. 90: ArchiMate Relationship Mapping: Logical Technology Component - Logical Technology Component

Organization Unit

– Organization Unit - Product

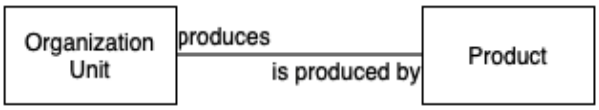


Fig. 91: TOGAF Relationship: Organization Unit - Product

The 'produces' relationship characterizes a product made by an organizational unit through a process. A product in the TOGAF framework is the output generated by a business process. In the ArchiMate language there is no relationship that can match exactly with the concept of "produce". A product is seen as a collection of services and a contract that is offered to the customers, so there are no direct relationship between a Business Actor and a Product. The following derived relationship can be created to deliver a similar meaning

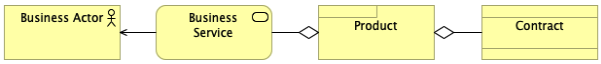


Fig. 92: ArchiMate Relationship Mapping: Organization Unit - Product

– Organization Unit - Process

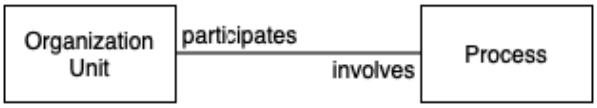


Fig. 93: TOGAF Relationship: Organization Unit - Process

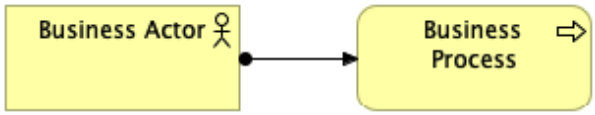


Fig. 94: ArchiMate Relationship Mapping: Organization Unit - Process

– Organization Unit - Driver

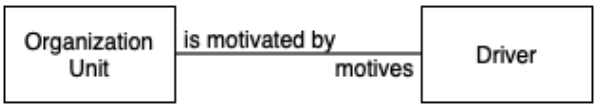


Fig. 95: TOGAF Relationship: Organization Unit - Driver

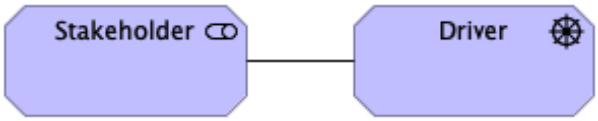


Fig. 96: ArchiMate Relationship Mapping: Organization Unit - Driver

Process

– Process - Product

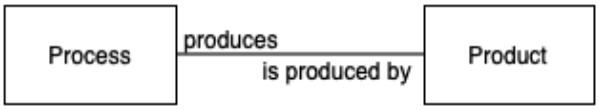


Fig. 97: TOGAF Relationship: Process - Product

The 'produces' relationship characterizes a product made by an organizational unit through a process. A product in the TOGAF framework is the output generated by a business process. In the ArchiMate language there is no relationship that can match exactly with the concept of "produce". A product is seen as a collection of services and a contract that is offered to the customers, so there are no direct relationship between a Business Actor and a Product. The following derived relationship can be created to deliver a similar meaning

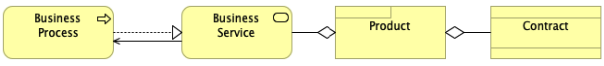


Fig. 98: ArchiMate Relationship Mapping: Process - Product

– Process - Role

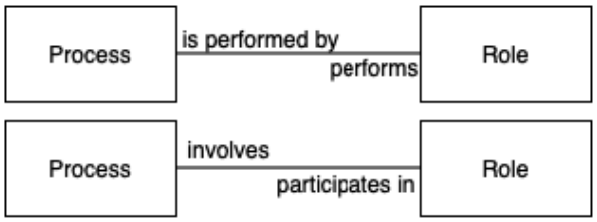


Fig. 99: TOGAF Relationship: Process - Role

The "is performed by" and "involves" relationships characterize an active element that is participating or taking a specific role in certain activity (behavior). The following derived relationship can be created to deliver a similar meaning

Fig. 100: ArchiMate Relationship Mapping: Process - Role

– Process - Process

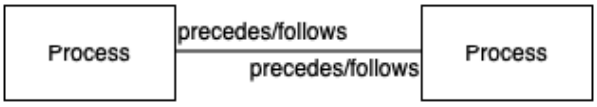


Fig. 101: TOGAF Relationship: Process - Process

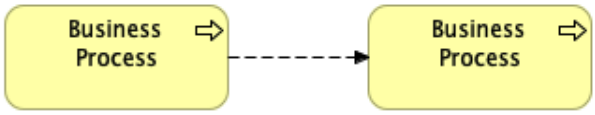


Fig. 102: ArchiMate Relationship Mapping: Process - Process

The mappings presented in this section are based on a set of assumptions made by the authors of the TOGAF® Framework and ArchiMate® Modeling Language Harmonization White Paper [32]
From the summary on the General Recommendations and Comments Regarding Relationship Harmonization retrieved from [32] we should take the following considerations when harmonizing between TOGAF and ArchiMate standards:

- TOGAF relationships that are related with governance, such as governs, measures, is governed, and measured by or tracked against, do not have exact matches to relationships in the ArchiMate standard; however, specialization and profiling can be applied for the contract entity. Even though contract is an entity from the Business layer, it can be applied in other ArchiMate domains and extensions.

- The relationships between the TOGAF logical technology component and physical technology component have some slightly different meanings than in the ArchiMate Technology layer, but can be mapped in order to deliver the architecture descriptions. For example, node could represent the logical technology component and device or system software could represent the physical technology component.

- The ArchiMate aggregation relationship, conceived as a "temporary grouped" concept, cannot be mapped exactly in the TOGAF standard, because it only considers the decompose relationship which could not have the same meaning. In this case, using aggregation or composition for making the mapping with the decompose relationship could be done depending on the context for any particular modeling situation.

- The specialization concept does not exist explicitly in the TOGAF standard; however, the framework does state that additional taxonomies, classifications, and extensions can be applied to the core metamodel to depict more complex real-life situations.

- The ArchiMate uses/used by relationship is not present in the TOGAF standard. The "consumes/supplies" relationship has similar meaning, but it may not be applicable to all situations. For example, a business service can use a data entity. Similarly, the support relationship between a process and a business service can be mapped with uses/used by or the realization relationship in the ArchiMate language. As previously noted, the mapping would depend on the particular modeling context. This can also be true for the assignment relationship. It can only be mapped using perform task in or operates in or participates in even though this example is between an actor and a process rather than a role and a process.

- In the TOGAF standard, there is no realization relationship between a process and a service. The closest mapping might be orchestrates; however, there

is a realized by relationship between the service and the process. These details have to be taken into consideration in modeling practical situations.

3.3 XSLT File Definition Approach

In this section we will present in detail the files definition and process necessary for the model transformation.
As mentioned in the Research section, *2.4 EA and EC Tools*, with the Atlas tool it is possible to have different models coming from different EA tools being consistent with each other in a single place, using the transformation model based on XSLT technology. [21]
The model transformation mechanism is based on two steps.
The first step deals with the configuration of the TOGAF and ArchiMate metamodels in the Atlas repository. For this, a specific file must be defined with the entities and attributes of each standard. The ArchiMate 3.1 metamodel was already published in the Atlas repository , so for the solution we only had to defined the file for the TOGAF 9.2 metamodel entities.
In the TOGAF 9.2 standard, Section 30.6 [35] describes the typical attributes that are associated with the entities defined in the core and extension metamodel entities. In general, all metamodel entities should have basic attributes such as:

- ID
- Name
- Description
- Category
- Source
- Owner

Although the TOGAF standard does not specify the data types for these attributes and does not provide information regarding the creation of user-defined attributes so it is essential to have the ability to define attributes for entities as required for the context in which they are being used. [32]
The transformation process treats the relationships as attributes of the entity so in the file we also need to define them for each entity. In Annex F, an example of the file and entity definition is provided.
After the file is defined a batch is run in Atlas and the metamodel entities are populated in the repository. In Annex G, is presented a print of the repository after the introduction of the entities that form the TOGAF metamodel.
Having the basis defined, the following step is to determine the model we want to transform, and which direction TOGAF to ArchiMate or ArchiMate to TOGAF, and define the relationships mapping file accordingly.
Atlas uses an high level type-based rules that operates on types, instances and relationships.To configure such rules, a XSLT file is defined with the transformation rules, that transforms the source model and objects into target model and objects. To understand how to define the transformation rules a thorough read of the Atlas Documentation Section 8, Transformation Configuration - XML Rule,

was necessary. This section explains how each of the rules should be defined and what are their purpose. Realizing how this set of rules works and how to apply them to the solution is essential for the transformation of the model to be successful.
Retrieved from the Atlas documentation [36] some of the rules that are likely to be used are presented next:

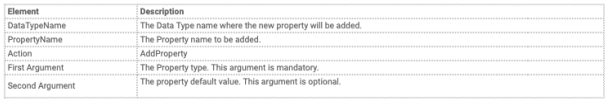
– Add Property

  Add Property Transformation Rule adds a property with a default value to a Data Type. The result of this transformation rule is to add to Data

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	ApplicationComponent	isCritical		AddProperty	Bool true	1

Fig. 103: Add Property Rule

  Type “ApplicationComponent” the property “isCritical” of type “Boolean” with a default value set to “true”. This means that, after this transformation, all data instances of “ApplicationComponent” Data Type will have this property set to “true”.



Element	Description
DataTypeName	The Data Type name where the new property will be added.
PropertyName	The Property name to be added.
Action	AddProperty
First Argument	The Property type. This argument is mandatory.
Second Argument	The property default value. This argument is optional.

Fig. 104: Add Property Rule Arguments

– Copy Class

  Copy DataType Transformation Rule allows the creation of a duplication of a DataType, including all its data instances. After the above Transfor-

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Node			CopyDataType	Environment	1

Fig. 105: Copy Class Rule

  mation Rule is applied, the DataType “Environment” will exists, as well as a duplication of all data instances of “Node”, as instances of “Environment” Data Type.

Element	Description
DataTypeName	The DataType name to be copied.
Action	CopyDataType
Argument	The name of the new Data Type

Fig. 106: Copy Class Rule Arguments

– Copy Property

CopyTo Transformation Rule copies the value of a property from one data instance to the value of a property of another data instance, where the first data instance holds a reference to the second data instance. The result of

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Application	Name	AccessRelationship	CopyTo	Node Applications	1

Fig. 107: Copy Property Rule

the above Transformation Rule is the setting of the value of property Applications in all Node instances that are referenced by an Application instance with the value of property Name of the referring instances.

Element	Description
DataTypeName	The Data Type name of the instances holding the property to be copied from.
PropertyName	The name of the Property whose value if to be copied to another property
RelationName	The name of the reference type the referring instances refer to the referred instances (it can also have the Data Type name of the relational objects in case the relation was created using the AddProperty rule)
Action	CopyTo
First Argument	The name of the Data Type holding the property whose value if to be set with the new value
Second Argument	And the name of the property whose value if to be set with the new value

Fig. 108: Copy Property Rule Arguments

– Copy Property From

CopyFrom Transformation Rule copies the value of a property from a referred data instance to the value of a property of the referring data instance.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Application	Productive Date	Releases	CopyFrom	Application Min[Ec.Productive Date]	1

Fig. 109: Copy Property From Rule

The result of above Transformation Rule is the setting of the Productive Date property of applications with the Minimum Productive Date of all theirs´s releases, referred by the Releases property.

Element	Description
DataTypeName	The Data Type name of the instances holding the property set
PropertyName	The name of the Property to be set with the be copied value
RelationName	The name of the reference type the referring instances refer to the referred instances (it can also have the Data Type name of the relational objects in case the relation was created using the AddProperty rule)
Action	CopyFrom
First Argument	The name of the Data Type holding the property whose value if to be copied from
Second Argument	An expression using properties of the child and parent objects The name of the property whose value if to be copied from

Fig. 110: Copy Property From Rule Arguments

– Delete Class

Delete DataType Transformation Rule allows to delete a Data Type from the initial model. Deleting a Data Type also deletes corresponding Data Instances, and therefore also deletes references to deleted instances.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	ComunicationPath			DeleteDataType		1

Fig. 111: Delete Class Rule

The result of above Transformation Rule is the removal of Data Type “CommunicationPath” from the resulting model.

Element	Description
DataTypeName	The DataType name to be removed.
Action	DeleteDataType

Fig. 112: Delete Class Rule Arguments

– Filter Property

The Filter Property Transformation Rule allows us to filter out from a property the references to a given Data Type and stores the result in another property (existing or new). Thus, if a property holds references to data instances of multiple Data Types, the receiving property will receive only the references to the desired Data Type. This Transformation Rule is applied to all data instances of a given Data Type.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Business Process	TriggeringRelationship		Filter Property	Triggers Business Event	1

Fig. 113: Filter Property Rule

The Transformation Rule above applies to each instance of “Business Process” Data Type. For each instance, it will add to the “Triggers” property

the references existing in “TriggeringRelationship” property that refer to instances of Data Type “Business Event”.

Element	Description
DataTypeName	The DataType name of the instances to with the rule should be applied
PropertyName	The Property Name of the origin values to be filtered out.
Action	FilterProperty
First Argument	The destiny property to which the filtered-out values should be added to. This argument is mandatory.
Second Argument	The Data Type filter. Only the references for instances of this DataType will be added. This argument is mandatory.

Fig. 114: Filter Property Rule Arguments

– Filter by Property Value

The Filter by Property Value Transformation Rule allows to filter out data instances (of the specified Data Type) with a given value in a given property. Data Instances of the specified Data Type with non-matching values will be discarded.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Application Component	External Provider		FilterByPropertyValue	== true	1

Fig. 115: Filter Property by Value Rule

The Transformation Rule above applies to each instance of “Application Component” Data Type. Instances with a value of property “External Provider” matching “true” it will remain, whereas the others will be discarded. If the property value is null or if it is not defined it will match as non-matching values.

Element	Description
DataTypeName	The DataType name of the instances to with the rule should be applied
PropertyName	The Property Name of the origin values to be filtered out.
Action	FilterByPropertyValue
First Argument	Either: <, <=,==, >=, >, !=.
Second Argument	The property value

Fig. 116: Filter Property by Value Rule Arguments

– Rename Class

The rename DataType Transformation Rule changes the name of a DataType.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Business Process			RenameDataType	Business Process	1

Fig. 117: Rename Class Rule

The result of the above Transformation Rule is the change of the name of the "BusinessProcess" DataType to "Business Process".

Element	Description
DataTypeName	The original data type name.
Action	RenameDataType
First Argument	The new data type name. This argument is mandatory.

Fig. 118: Rename Class Rule Arguments

– Rename Property

The Rename Property Transformation Rule changes the name of a Property in a given DataType. The values of the renamed property will remain unchanged in corresponding data instances.

Id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Args	ExecutionOrder
1			Archi	Business Process	Owner		RenameProperty	Process Owner	1

Fig. 119: Rename Property Rule

The result of this Transformation Rule is that property "Owner" of "Business Process" DataType will be renamed to name "Process Owner".

Element	Description
DataTypeName	Defines Data Type name.
PropertyName	Defines the original property name.
Action	RenameProperty
First Argument	Defines the new property name. This argument is mandatory.

Fig. 120: Rename Property Rule Arguments

The structure of the relationships transformation file is based on a matrix where the Origin Metamodel and Target Metamodel entities are defined as well as the rules that must be applied to them.
In Annex H, an example of the file and the definition of rules is provided.

4 Model Transformation - Solution Demonstration

Next, an ArchiMate model example will be presented that will be used to demonstrate the transformation between the metamodels of the TOGAF and ArchiMate standards. The example used represents a very simple view of the enterprise architecture of an insurance company that offers claims services. This insurance company offers its customers a registration service, a service for attaching claim information and a payment service. These services are supported by a process of handling claims and its distinct functions as well as by an application and technological layer.

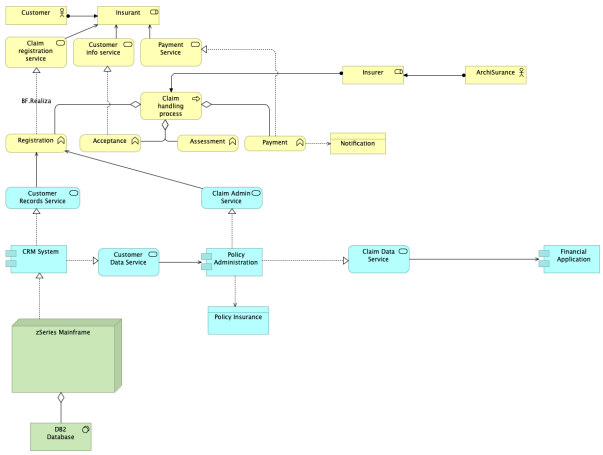


Fig. 121: ArchiMate - Insurance Company Example

When transforming a model between TOGAF and ArchiMate standards, and vice-versa, many considerations must be taken into account, as seen in the Solution Section. The context influences immensely what type of relationships will be used when transforming the model. For the simple ArchiMate model example shown above, our goal is only to show that the transformation can be done through the Atlas tool, so the context and the need to add new properties to relationships and objects so that the model does not lose coherence and transport the exact same meaning was not taken fully into account. Following the concepts and relationships mapping presented in the Solution Section it is necessary understand how can we, using the Atlas rules, transform the example model into a TOGAF model.
As previously seen, it is known that in the TOGAF standard metamodel there is no realization relationship between a process and a function. The closest mapping might be orchestrates. However, there is a "realized by" relationship between the service and the process. In this specific example, are the business functions aggre-

gated to the process that realize the Business Services so in a way the Business Services are realized by the Process . This will be the assumption that will be implemented in the transformation.
Below the TOGAF model we want to achieve after the transformation of the ArchiMate model presented.

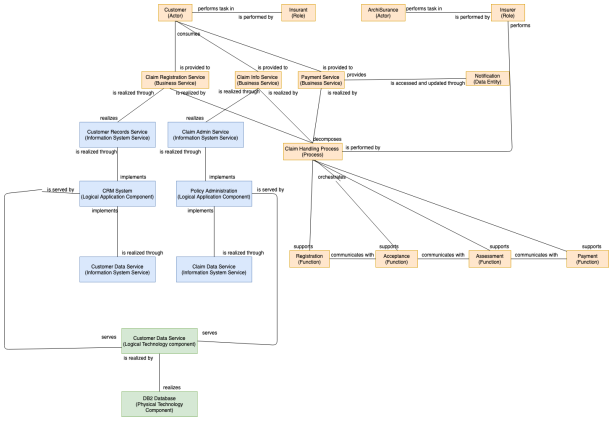


Fig. 122: TOGAF - Insurance Company Example

Following the Atlas transformation rules presented in the Solution Section, below the description of the transformation rules created to transform the model will be described. In the XSLT file we can define mappings based on 3 types of transformation rules:
Firstly the Delete Data Type Rule is applied to all the ArchiMate entities that do not incorporate the model

C Rule: Ignore Capability	Capability		DeleteDataType
C Rule: Ignore CommunicationNetwork	CommunicationNetwork		DeleteDataType
C Rule: Ignore Constraint	Constraint		DeleteDataType
C Rule: Ignore Contract	Contract		DeleteDataType
C Rule: Ignore CourseOfAction	CourseOfAction		DeleteDataType

Fig. 123: Model Transformation - Delete Data Type

Also the Rename Data Type is applied to the ArchiMate entities that appear in the model so that the transformed model will have the description of the entities of the TOGAF metamodel.

C Rule: Rename BusinessObject As Data Entity	BusinessObject		RenameDataType Data Entity
C Rule: Rename BusinessProcess As Process	BusinessProcess		RenameDataType Process
C Rule: Rename BusinessRole As Role	BusinessRole		RenameDataType Role
C Rule: Rename BusinessService As Business Service	BusinessService		RenameDataType Business Service

Fig. 124: Model Transformation - Rename Data Type

These types of transformation rules only use the source class as an argument to convert the reference.
Finalizing step one of processing class rules, the rules applied next are rules based on source class, relationship type and target class. In the example model we apply this type of rules in the following relationships:

- Business Actor and Business Role
- Business Actor and Business Role
- Business Role and Business Process
- Business Process and Business Function
- Business Function and Business Function
- Business Function and Business Service
- Application Component and Application Service
- Business Service and Business Role
- Node and Application Component
- Node and System Software
- Business Function and Business Object
- Application Component and Data Object
- Application Service and Business Function

For each relationship a specific rule must be defined. Below the rules definitions are presented.

- Business Actor - Business Role

CRC Rule(BA>BR): Set (Actor.performs task in) = [Role] (Actor.AssignementRelationship)	Actor	AssignementRelationship		FilterProperty	performs task in Role

Fig. 125: Business Actor - Business Role Relationship Transformation - Filter Property

- Business Role - Business Process

CRC Rule(BR>BP): Set (Role.performs) = [Process] (Role.AssignementRelationship)	Role	AssignementRelationship		FilterProperty	performs Process

Fig. 126: Business Role - Business Process Relationship Transformation - Filter Property

- Business Process - Business Function

CRC Rule(BP>BF): Set (Process.orchestrates) = [Function] (Process.AggregationRelationship)	Process	AggregationRelationship		FilterProperty	orchestrates Function

Fig. 127: Business Process - Business Function Relationship Transformation - Filter Property

– Business Function - Business Function

Fig. 128: Business Function - Business Function Relationship Transformation - Filter Property

– Business Function - Business Service

Fig. 129: Business Function - Business Service Relationship Transformation - Filter Property

– Application Component and Application Service

Fig. 130: Application Component - Application Service Relationship Transformation - Filter Property

– Business Service - Business Role

Fig. 131: Business Service - Business Role Relationship Transformation - Filter Property

– Node - Application Component

Fig. 132: Node - Application Component Relationship Transformation - Filter Property

– Node - System Software

Fig. 133: Node - System Software Relationship Transformation - Filter Property

– Business Function - Business Object

Fig. 134: Business Function - Business Object Relationship Transformation- Filter Property

– Application Component - Data Object

Fig. 135: Application Component - Data Object Relationship Transformation - Filter Property

– Application Service - Business Function

Fig. 136: Application Service - Business Function Relationship Transformation - Filter Property

Due to the fact that there are some class-to-class relationships in ArchiMate that do not have direct mapping in the TOGAF metamodel, it is necessary to make additional rules to be able to recreate the same meaning. In the ArchiMate model example, it is the Business Role that has a relationship with the Business Service, since this relationship it is not known in the TOGAF metamodel a rule must be define to create a derived relationship that will propagate the same meaning. For this a Copy To rule is defined so that in the TOGAF model transformation is the Actor that will have this property.

Fig. 137: Business Service - Business Actor Relationship Transformation - Copy To

Some others derived relationship were needed for the transformation of this ArchiMate model example. Below the additional rules defined to create the derived relationships are presented.

CRC Rule(BF>BS): Set (Function.realizes) = [Business Service] (Function.BF.Realiza)	Function	BF.Realiza		FilterProperty	realizes Business Service

Fig. 138: Business Function - Business Service Relationship Transformation - Copy To

Application Component	AccessRelationship		CopyTo	Information System Service used by

Fig. 139: Application Component - Information System Service Relationship Transformation - Copy To

Application Service	ServingRelationship		CopyTo	Business Service provides governed interface to access

Fig. 140: Application Service - Business Service Relationship Transformation - Copy To

Business Function	BF.Realiza		CopyTo	Information System Service realizes

Fig. 141: Business Function - Information System Service Relationship Transformation - Copy To

There is also the possibility to define in the file another type of rules that, converts the reference, based on the source class and relationship type. This type of rules are applied to classes of origin that regardless of the class of destination, for a specific type of relationship, always have the same mapping. In this specific example we did not defined any rules for this type of transformation.
Having all the rules defined in the XSLT file, all that remains is to run the batch in the Atlas tool, which is responsible for integrating the new model transformed into the Atlas repository.

5 Conclusion

5.1 Conclusion

Through this document we examine the importance of an EA solution in organizations, as a form of adaptation to the constant and rapid change within the organization and the need to establish a defined process to manage and maintain the EA repository. To be able to benefit from the EA, we verified the need to have a consolidated EA repository, which is certainly a challenge within enterprises where thee planning process originates in many of the enterprise's communities without a consistent, coherent and complete systemic view of the enterprise to support them, individually and as a whole as well as the struggle to align the different elements designed by different architects resulting into an unconscious design. Entering Enterprise Cartography discipline, it presents itself as a discipline that aims to abstract the "enterprise reality", keeping design apart from representation, meaning it eliminates the need for design coherence while modeling the enterprise.
We look to EC discipline, what characterizes it and how it can be used as a tool to assist the process of managing, maintaining and integrating tools with the EA repository, as suggested in practical cases [5].
Having studied the framework and language, widely used in the development of EA, the TOGAF framework and the ArchiMate language, we found that there is space to create a solution for the integration of models based on the two specifications. This solution goes through the analysis of the work already developed in the harmonization of the two metamodels in question and in the creation, using the EC Atlas tool, of a semi-automatic mechanism for the transformation between ArchiMate and TOGAF models and vice-versa.

From the research done in this work we can conclude that, through the Atlas tool, it is possible to achieve the model transformation using a well defined XSLT file where are presented the mappings based on 3 types of transformation functions whose arguments are:

- Convert Reference based on source class only
- Convert Reference based on source class and type of reference
- Convert Reference based on source class, reference type and target class.

As seen in the Model Transformation - Solution Demonstration Section, for the example presented, there are some cases where derived relationships are needed for the transformation and additional rules must be defined.
As for the reversibility of the conversions, the XSLT file must be modified so that the source and destination model are exchanged, that is, the transformation rules applied will have to be implemented according to the direction of the transformation, but the mapping between the two specifications, both entities and relationships, presented in the Solution Section remains the same.
As seen in the Proposed Solution Section there are entities between both specifications that do not have any type of mapping. In these cases where the model

to transform has entities that do not have any mapping in the target model, the entities would not appear in the model transformed.
As seen in the Model Transformation - Solution Demonstration Section in the rules created to transform the model example, the entities that do not incorporate the model to be transform will suffer a Delete Data Type Transformation Rule allowing to delete a Data Type from the initial model. Deleting a Data Type also deletes corresponding Data Instances, and therefore also deletes references to deleted instances. Being the result presented, a simple example of a way to consolidate models based on the two metamodels in a single repository.

5.2 Future Work

Given the capacity found in the Atlas tool for integrating models from different tools, there is a lot of work that can be done in this field and in the integration between EA tools metamodels.
Within the work carried out for the elaboration of this research it would be a good practice to have several models and views to convert that encompass all entities with mapping between the two metamodels (TOGAF and ArchiMate), to analyze and test the transformation and the use of the solution in a specific use case to verify its usability and versatility.
It would also be interesting to study the creation of an XSLT transformation file, for each of the transformation directions, TOGAF to ArchiMate and ArchiMate to TOGAF, with defined set of generalized rules that could be used as a basis for the transformation of several models, instead of having to create a transform file from scratch every time there is the need to transform a specific model / view.

References

1. Zachman, A. J. (1997). Enterprise architecture: The issue of the century. Database Programming and Design magazine
2. Shanks G., Gloet M., Someh A. I., Frampton K., Tamm T. (2018). Achieving benefits with enterprise architecture, The Journal of Strategic Information Systems, 27(2),139-156.doi.org/10.1016/j.jsis.2018.03.001.
3. Kaisler H. S., Armour F., Valivullah M.(2005). Enterprise Architecting: Critical Problems. HICSS 2005 - 38th Hawaii International Conference on System Sciences, 3-6.doi.ieeecomputersociety.org/10.1109/HICSS.2005.241
4. Roeleven, S. (2010). Why Two Thirds of Enterprise Architecture Projects Fail. Business White Paper
5. Sousa P.,Tribolet J., Guerreiro S., Enterprise Cartography: From Theory to Practice
6. Carter H., 1999, Information Architecture, Work Study, Vol. 48, No. 5, pp. 182-185.
7. Perdeck, M., Soetendal, J., Raadt, B., Van Vliet, H. (2004). Polyphony in Architecure.Proceedings of the 26th International Conference on Software Engineering.
8. James G., (2003). High-Level Architecture Models Give Enterprise Perspective, Gartner Research Notes.
9. Zachman J. (1997). Concepts of Framework for Enterprise Architecture.
10. Aerts, A.T.M, Goossenaerts, J.B.M., Hammer, D.K., and Wortmann, J.C.(2004). Architectures in Context: On the Evolution of Business, Application Software and ICT Platform Architectures. Information and Management, Vol. 41, Iss. 6, pp 781-794.
11. Roberts J.(2002). What Makes I.T. Enterprise Architecture Successful. Gartner Research Notes.
12. Tribolet, J., Sousa, P., Caetano, A. (2014). The Role of Enterprise Governance and Cartography in Enterprise Engineering. Enterprise Modelling and Information Systems Architectures, 9(1), 38-49.
13. Lankhorst, M. (2009). Introduction to Enterprise Architecture. Enterprise Architecture at Work.
14. Sousa, P., Carvalho, M. (2018). Dynamic Organizations Representation. Linking Project Management with Enterprise Architecture. 2018 IEEE 20th Conference on Business Informatics (CBI).
15. Berneaud, M., Buckl, S., Diaz-Fuentes, A., Matthes, F., Monahov, I., Nowobliska, A., Roth, S., Schweda, C., Weber, U. Zeiner, M. (2012). Trends for enterprise architecture management tools survey. Technical report, Technical University Munich
16. Matthes, F., Buckl, S., Leitel, J., Schweda, CM. (2008) Enterprise Architecture Management Tool Survey 2008. Technical report, Technical University Munich
17. Roth, S. (2014). Federated enterprise architecture model management conceptual foundations, collaborative model integration, and software support. Technical University Munich, Diss.
18. Archi – Open Source ArchiMate Modelling. (n.d.). Retrieved from https://www.archimatetool.com/
19. The Open Group ArchiMate® Model Exchange File Format and Archi 3.3. (2015, September 25). Retrieved from https://blog.opengroup.org/2015/09/25/the-open-group-archimate-model-exchange-file-format-and-archi-3-3/
20. Sousa, P., Leal R., Caetano, A. (2018). Atlas: the Enterprise Cartography Tool
21. W3C.(1999). XSL Transformations (XSLT).
22. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., . . . Wimmer, M. (2006). Lifting Metamodels to Ontologies: A Step to the

Semantic Integration of Modeling Languages. Model Driven Engineering Languages and Systems Lecture Notes in Computer Science.
23. Kapsammer, E., Reiter, T., Schwinger W. (2006). Model-Based Tool Integration-State of the Art and Future Perspectives.
24. Antunes, G., Bakhshandeh, M., Mayer, R., Borbinha, J., Caetano, A. (2014). Using Ontologies for Enterprise Architecture Integration and Analysis. Complex Systems Informatics and Modeling Quarterly. 1. 10.7250/csimq.2014-1.01.
25. Czarnecki k., Helsen S. (2003). Classification of Model Transformation Approaches. OOPSLA Workshop on Generative techniques in the context of MDA.
26. Zivkovic, S., Kühn, H., Karagiannis, D. (2007). Facilitate Modelling Using Method Integration: An Approach Using Mappings and Integration Rules. ECIS.
27. Kolahdouz-Rahimi, S., Lano, K., Pillay, S., Troya, J., Gorp, P. V. (2014). Evaluation of model transformation approaches for model refactoring. Science of Computer Programming, 85, 5-40. doi:10.1016/j.scico.2013.07.013
28. Li, D., Li, X., Stolz, V. (2011). QVT-based model transformation using XSLT. ACM SIGSOFT Software Engineering Notes, 36(1).
29. Uschold, M., Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. ACM SIGMOD Record, 33(4), 58. doi:10.1145/1041410.1041420
30. Ota, Daniel Gerz, Michael. (2011). Benefits and Challenges of Architecture Frameworks. 40.
31. Czarnecki, Krzysztof Helsen, Simon. (2003). Classification of Model Transformation Approaches.
32. Estrem, W., Gonzalez, S. (2014, December). TOGAF® Framework and ArchiMate® Modeling Language Harmonization Content Metamodel Harmonization: Entitles and Relationships
33. Evernden, R. (2020, February 19). Combining TOGAF and ArchiMate. Retrieved from http://blog.goodelearning.com/subject-areas/togaf/togaf-and-archimate-come-from-the-same-stable/
34. (n.d.). Retrieved from https://pubs.opengroup.org/architecture/archimate3-doc/apdxe.html
35. (n.d.). Retrieved from https://pubs.opengroup.org/architecture/togaf9-doc/arch/
36. (n.d.). Retrieved from https://atlas-docs.linkconsulting.com/login
37. The Open Group. (2011). TOGAF Version 9.1
38. Lankhorst, M. (2013). Enterprise architecture at work: Modelling, communication and analysis. Berlin: Springer.

List of Figures

List of Tables

A TOGAF and ArchiMate Entities Mapping Strength

ArchiMate Element	Fit	TOGAF Element	Notes
Service	Fair Fit	- Business Service - IS Service - Platform Service	The TOGAF standard models specific architecture domain services.
Business Actor	Strong Fit	Actor	Business Actor can Actor or an Organizational Unit in TOGAF
Business Actor	Strong Fit	Organization Unit	Business Actor can Actor or an Organizational Unit in TOGAF
Business Function	Strong Fit	Function	
Business Object	Fair Fit	Data Entiity	Business Object is more general than Data Entity, can be used to model any business-relevant passive element
Business Process	Strong Fit	Process	
Business Role	Strong Fit	Role	
Business Service	Strong Fit	Service	
Contract	Strong Fit	Contract	
Business Event	Strong Fit	Event	
Business Interaction	Weak Fit	Process or Function	The TOGAF standard does not explicitly define interactions . The best mapping may be a process because it is the behavior made by a collection of roles.
Location	Strong Fit	Location	
Product	Strong Fit	Product	
Application Service	Fair Fit	Information System Service	

Data Object	Strong Fit	Data Entity	
Data Object	Strong Fit	Logical Data Component	
Application Component	Strong Fit	- Logical Application Component - Physical Application component	
Application Function	Strong Fit	Logical Application Component	This concept is intended to capture the same 'implementation-independent encapsulation of application functionality.
Infrastructure Service	Strong Fit	Platform Service	
Infrastructure Function	Strong Fit	Logical Technology Component	
Artifact	Fair Fit	Physical Data Component	Some artifacts should be mapped to Physical Data Components, some to Physical Application Components; in both cases, the mapping itself is Fair.
System Software	Strong Fit	Physical Technology Component	
Device	Strong Fit	Physical Technology Component	
Node	Strong Fit	Logical Technology Component	
Infrastructure Interface	Fair Fit	- Logical Technology Component - Physical Technology Component	If the mapping is made at a physical or logical level, would define the mapping.

Network	Fair Fit	Physical Technology Component	Special kind of Physical Technology Component.
Communication Path	Fair Fit	Logical Technology Component	Special kind of Logical Technology Component.
Gap	Strong Fit	Gap	
Work Package	Strong Fit	Work Package	
Stakeholder	Fair Fit	Role	
Stakeholder	Fair Fit	Organization Unit	
Driver	Strong Fit	Driver	
Assessment	Fair Fit	Measure	The TOGAF measure entity is an indicator or factor that is linked to a goal or objective. The ArchiMate assessment entity is the outcome of the analysis of some driver.
Goal	Strong Fit	Goal	
Goal	Strong Fit	Objective	The TOGAF objective entity is used in the attainment of a goal. The ArchiMate goal entity can be decomposed to represent an objective.
Principle	Strong Fit	Principle	
Requirement	Strong Fit	Requirement	
Constraint	Strong Fit	Constraint	

B TOGAF and ArchiMate Relationships Mapping Strength

TOGAF Name	Description	ArchiMate Mapping	Fit
Accesses	A role access a function.	Assigned to/assigned from	Strong Fit
Addresses	Goal and a driver.	Association, influence	Strong Fit
Association	Architecture principles, assumptions, requirements, gaps, and work package can have association with the rest of the Content Metamodel objects	Any core concept in the ArchiMate language realizes a requirement. A principle and requirement realize a goal. Gaps are associated with work packages that realize any core element.	Weak Fit
Applies to	Used to relate a service quality, has governance meaning	The service quality concept does not exist in the ArchiMate standard so the relationship could be represented by an attribute for a contract or business service or through a contract related to a business services	Weak Fit
Automates some of all of	Implementation or realization meaning	Indirect – uses/used by	Fair Fit
Belongs to	Governance or formal structure meaning	Assigned to	Strong Fit
Can be accessed by	Similar meaning to used by or assigned to	Assigned to	Strong Fit
Communicates with	Interaction, integration from one element with itself	Composition, aggregation, or specialization depending on the case.	Fair Fit
Consumes	Similar to used by or access (for informational concepts)	Uses and used by, access (business and data entity) and aggregation, composition, and specialization (service to service)	Fair Fit
Contains	Similar meaning to assignment – organizational assets contained in locations	Assign to	Strong Fit
Creates	Motivational meaning a driver that is realized by a goal; a goal created to achieve the goal	Association/influence	Fair Fit
Decomposes	Composition or aggregation. The meaning would depend on the context.	Composition The aggregation concept is not explicitly considered in the TOGAF standard, so the mapping would depend on the context.	Fair Fit
Delivers	Similar to realization or provides	Realize assigned, association The relationship used would depend on the elements being related.	Fair Fit
Encapsulates, resides within	A logic concept that is implemented or realized by a more concrete concept or that is contained in a more concrete or physical element.	Realization between a business object and a data object and a specialization or aggregation between artifacts	Fair Fit

Extends	The meaning is associated with a logical data component being extended or represented or realized by a physical data component.	Realization An artifact realizes a data object. It can also be represented by a representation that realizes a business object that is realized by a data object.	Strong Fit
Generates	It has a dynamic connotation, one concept that has a cause effect in other, like triggering.	Triggers/is triggered by Derived relationship, two active concepts that inherit the dynamic triggering relationship for the actor – event.	Strong Fit
Governs and Measures	Has a governance meaning.	Is accessed by/accesses or Derived relationship;	Weak Fit
Implements	Implementation, provides platform for deployment, realization.	Is realized by/realizes or Derived relationship;	Strong Fit
Interacts with	An active element that performs an activity through a dynamic concept; assignation.	Assigned to/assigned from (relationship through the role).	Strong Fit
Involves	An active element that is participating or taking a specific role in certain activity	Assigned to (through business role).	Strong Fit
Is accessed and updated through	An active element that performs an action on an informational passive concept.	Access/accessed by	Strong Fit
Is bounded by	Functionality that is encapsulated into a service through a governed interface.	Realizes/is realized by, uses/used by .	Weak Fit
Is delivered by	Deliverable generated by a specific activity, in this case a work package that delivers value through a capability.	Realizes/is realized by	Fair Fit
Is dependent on	Elements that are related and one of them cannot exist without the other.	Composition	Strong Fit
Is extended by	Implementation or deployment meaning. A logical concept realized by a physical concept.	Realization	Strong Fit
Is generated by	A behavior that is realized or executed by any active element; triggering.	Triggering	Fair Fit

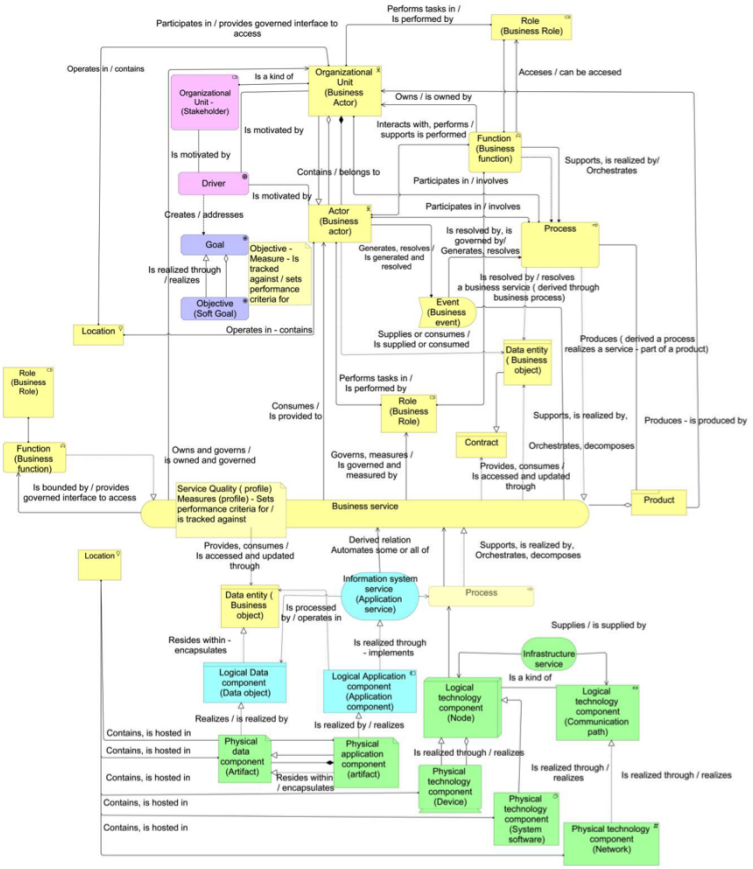
Is governed and measured by	Business service and contract	Is accessed by/accesses Indirect relationship, a contract can be seen as a specialization of a business object.	Weak Fit
Is hosted in	Physical location for concepts	Assigned to/assigned from	Strong Fit
Is implemented on	A component that is deployed in another or realized by another	There are no relationships that could go through the node to the business service. A uses/used by derived relationship can be used going from the business service to the node	Fair Fit
Is motivated by	Strategic view	Association	Strong Fit
Is performed by	Functional or role assignation meaning	Assigned to	Strong Fit
Is processed by	Information concept that is being updated or created by an application component	Is accessed by/accesses or Derived relationship;	Fair Fit
Is produced by	A product that is generated or made by an organizational unit through a process	Derived relationship;	Weak Fit
Is provided to	A service that is delivered by an actor	Uses or used by; also can be assigned to through a role.	Strong Fit
Is realized by	Deployment or implementation meaning	Realization (for Application layer) or for the technology concept could be through a system software or device that specialize the node. For the function and the process the relationship can be aggregation, flows, or triggering.	Strong Fit
Is realized through	A more concrete element that is representing a more conceptual definition;	For the motivational concept an objective can be seen as a special kind of goal, so the relationship is aggregation or specialization. For the implementation concept in the Application layer the realized by/realizes or used by/uses both derived relationships through a behavior element	Fair Fit

Is resolved by	Interaction between an actor and a process, service or event	Derived relationship;	Fair Fit
Is supplied by	A service that is being implemented by a platform	Uses or realizes Derived relationship through an infrastructure function that realizes the service and that is assigned to a node.	Fair Fit
Motivates	Motivational strategic concept	A business actor is a specialization of a stakeholder which is associated with a driver.	Strong Fit
Operates in	Location assigned to a core element	Assigned to	Strong Fit
Orchestrates	A process that is delivering a control flow over a function or business service	Realizes/is realized by, uses/used by, aggregates/is aggregated by, a process and a service	Fair Fit
Owns	Governance and stewardship meaning, functional decomposition	Assigned to (derived through role) or used by through a business service (business actor uses a business service realized by a business function).	Weak Fit
Owns and Governs	Governance and stewardship	Uses, even realization through an indirect relationship (actor assigned to a role, assigned to a business function or business process that realizes a business service).	Weak Fit

Participates in	Performing action made by an actor into a process	Assigned to (through business role)	Strong Fit
Performs	Performing action made by an actor into a function	Assigned to/assigned from, Indirect relationship through the role	Strong Fit
Performs task in	Role assigned to an actor	Assigned to	Strong Fit
Precedes/Follows	Flow between processes	Flows to (dynamic relation) or triggering	Strong Fit
Produces	Product made by an organizational unit through a process	A business actor, assigned to a behavior element, accesses a business object (specialization of a contract) that is aggregated into a product, so the business actor accesses the product though the contract. A business process realizes a product through a derived relationship (business process realizes the business service aggregated into the product).	Weak Fit
Provides governed interface to access	Governance meaning	Realizes/is realized by, uses/used by	Weak Fit
Provides platform for	Implementation meaning	An application component can be realized by an artifact deployed into a node, so the node indirectly realizes the application component that can be assigned to a business process or function which realizes the business service.	Fair Fit
Realizes	Deployment; a more concrete concept that defines or implements a less concrete concept	Realization	Fair Fit

Relates to	Relation between information concepts of the same kind	Aggregation or composition	Strong Fit
Resides within	An informational concept that can reside or be implemented by a data logical element or a physical data component that resides or is encapsulated in a physical application component.	Realization (business object to a data object) Realization, association, aggregation, or composition between artifacts,one artifact mapping the physical data component and the other mapping the physical application component.	Strong Fit
Resolves	An action performed as a consequence of a process or business service execution.	Triggering from a process and indirectly from a service used by a business process or an actor assigned to a business process through a role.	Fair Fit
Supplies	Implementation or deployment meaning	Uses or realizes	Fair Fit
Supplies/Consumes	An information concept that is given or used by an actor.	A business actor assigned to a process or function (through a role) can indirectly access the data entity represented by a business object.	Weak Fit
Supports	An action performed by one concept that is realizing or helping the achievement of another concept.	An actor is assigned to a function through a role, a function flows to or from a process or can be triggered, and can also be aggregated. A business service is used by or realized by a process	Fair Fit

C Content Metamodel Harmonization



D TOGAF Metamodel Entites

Metamodel Entity	Description
Actor	A person, organization, or system that has a role that initiates or interacts with activities; for example, a sales representative who travels to visit customers. Actors may be internal or external to an organization. In the automotive industry, an original equipment manufacturer would be considered an actor by an automotive dealership that interacts with its supply chain activities.
Assumption	A statement of probable fact that has not been fully validated at this stage, due to external constraints. For example, it may be assumed that an existing application will support a certain set of functional requirements, although those requirements may not yet have been individually validated.
Business Capability	A particular ability that a business may possess or exchange to achieve a particular purpose.
Business Service	Supports business capabilities through an explicitly defined interface and is explicitly governed by an organization.
Capability	A business-focused outcome that is delivered by the completion of one or more work packages. Using a capability-based planning approach, change activities can be sequenced and grouped in order to provide continuous and incremental business value.
Constraint	An external factor that prevents an organization from pursuing particular approaches to meet its goals. For example, customer data is not harmonized within the organization, regionally or nationally, constraining the organization's ability to offer effective customer service.
Course of Action	Direction and focus provided by strategic goals and objectives, often to deliver the value proposition characterized in the business model.
Data Entity	An encapsulation of data that is recognized by a business domain expert as a thing. Logical data entities can be tied to applications, repositories, and services and may be structured according to implementation considerations.
Gap	A statement of difference between two states. Used in the context of gap analysis, where the difference between the Baseline and Target Architecture is identified.
Function	Delivers business capabilities closely aligned to an organization, but not necessarily explicitly governed by the organization. Also referred to as "business function".
Role	The usual or expected function of an actor, or the part somebody or something plays in a particular action or event. An actor may have a number of roles.
Requirement	A quantitative statement of business need that must be met by a particular architecture or work package.
Location	A place where business activity takes place and can be hierarchically decomposed.
Logical Application Component	An encapsulation of application functionality that is independent of a particular implementation. For example, the classification of all purchase request processing applications implemented in an enterprise.
Organization Unit	A self-contained unit of resources with goals, objectives, and measures. Organization units may include external parties and business partner organizations.
Physical Technology Component	A specific technology infrastructure product or technology infrastructure product instance. For example, a particular product version of a Commercial Off-The-Shelf (COTS) solution, or a specific brand and version of server.
Principle	A qualitative statement of intent that should be met by the architecture. Has at least a supporting rationale and a measure of importance.
Process	A process represents flow of control between or within functions and/or services (depends on the granularity of definition). Processes represent a sequence of activities that together achieve a specified outcome, can be decomposed into sub-processes, and can show operation of a function or service (at next level of detail). Processes may also be used to link or compose organizations, functions, services, and processes.
Technology Service	A technical capability required to provide enabling infrastructure that supports the delivery of applications.
Value Stream	A representation of an end-to-end collection of value-adding activities that create an overall result for a customer, stakeholder, or end-user.
Work Package	A set of actions identified to achieve one or more objectives for the business. A work package can be a part of a project, a complete project, or a program.

Table 8: TOGAF 9.2 Content Metamodel Entities - Core Content [35]

Metamodel Entity	Description
Logical Technology Component	An encapsulation of technology infrastructure that is independent of a particular product. A class of technology product; for example, supply chain management software as part of an Enterprise Resource Planning (ERP) suite, or a Commercial Off-The-Shelf (COTS) purchase request processing enterprise service.
Physical Application Component	An application, application module, application service, or other deployable component of functionality. For example, a configured and deployed instance of a Commercial Off-The-Shelf (COTS) Enterprise Resource Planning (ERP) supply chain management application.

Table 9: TOGAF 9.2 Content Metamodel Entities - Infrastructure Consolidation Extension [35]

Metamodel Entity	Description
Driver	An external or internal condition that motivates the organization to define its goals. An example of an external driver is a change in regulation or compliance rules which, for example, require changes to the way an organization operates; i.e., Sarbanes-Oxley in the US.
Goal	A high-level statement of intent or direction for an organization. Typically used to measure success of an organization.
Objective	A time-bounded milestone for an organization used to demonstrate progress towards a goal; for example, "Increase capacity utilization by 30% by the end of 2019 to support the planned increase in market share".

Table 10: TOGAF 9.2 Content Metamodel Entities - Motivation Extension [35]

Metamodel Entity	Description
Contract	An agreement between a service consumer and a service provider that establishes functional and non-functional parameters for interaction.
Measure	An indicator or factor that can be tracked, usually on an ongoing basis, to determine success or alignment with objectives and goals.
Service Quality	A preset configuration of non-functional attributes that may be assigned to a service or service contract.

Table 11: TOGAF 9.2 Content Metamodel Entities - Governace Extension [35]

Metamodel Entity	Description
Control	A decision-making step with accompanying decision logic used to determine execution approach for a process or to ensure that a process complies with governance criteria. For example, a sign-off control on the purchase request processing process that checks whether the total value of the request is within the sign-off limits of the requester, or whether it needs escalating to higher authority.
Event	An organizational state change that triggers processing events; may originate from inside or outside the organization and may be resolved inside or outside the organization.
Product	Output generated by the business. The business product of the execution of a process.

Table 12: TOGAF 9.2 Content Metamodel Entities - Process Modeling Extension [35]

Metamodel Entity	Description
Logical Data Component	A boundary zone that encapsulates related data entities to form a logical location to be held; for example, external procurement information.
Physical Data Component	A boundary zone that encapsulates related data entities to form a physical location to be held. For example, a purchase order business object, comprising purchase order header and item business object nodes.

Table 13: TOGAF 9.2 Content Metamodel Entities - Data Modelling Extension [35]

Metamodel Entity	Description
Information System Service	The automated elements of a business service. An information system service may deliver or support part or all of one or more business services.

Table 14: TOGAF 9.2 Content Metamodel Entities - Services Extension [35]

E ArchiMate Metamodel Entites

Metamodel Entity (Business Layer)	Description
Business Actor	A business actor represents a business entity that is capable of performing behavior.
Business Role	A business role represents the responsibility for performing specific behavior, to which an actor can be assigned, or the part an actor plays in a particular action or event.
Business Collaboration	A business collaborationrepresents an aggregate of two or more business internal active structure elements that work together to perform collective behavior.
Business Interface	A business interface represents a point of access where a business service is made available to the environment.
Business Process	A business process represents a sequence of business behaviors that achieves a specific result such as a defined set of products or business services.
Business Function	A business function represents a collection of business behavior based on a chosen set of criteria (typically required business resources and/or competencies), closely aligned to an organization, but not necessarily explicitly governed by the organization.
Business Interaction	A business interaction represents a unit of collective business behavior performed by (a collaboration of) two or more business actors, business roles, or business collaborations.
Business Event	A business event represents an organizational state change.
Business Service	A business service represents explicitly defined behavior that a business role, business actor, or business collaboration exposes to its environment.
Business Object	A business object represents a concept used within a particular business domain.
Contract	A contract represents a formal or informal specification of an agreement between a provider and a consumer that specifies the rights and obligations associated with a product and establishes functional and non-functional parameters for interaction.
Representation	A representation represents a perceptible form of the information carried by a business object.
Product	A product represents a coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers.

Table 15: Archimate 3.1 Business Layer Entities

Metamodel Entity (Application Layer)	Description
Application component	Represents an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable.
Application collaboration	Represents an aggregate of two or more application internal active structure elements that work together to perform collective application behavior.
Application interface	Represents a point of access where application services are made available to a user, another application component, or a node.
Application function	Represents automated behavior that can be performed by an application component.
Application interaction	Represents a unit of collective application behavior performed by (a collaboration of) two or more application components.
Application process	Represents a sequence of application behaviors that achieves a specific result.
Application event	Represents an application state change.
Application service	Represents an explicitly defined exposed application behavior.
Data object	Represents data structured for automated processing.

Table 16: Archimate 3.1 Application Layer Entities

Metamodel Entity (Technology Layer)	Description
Node	Represents a computational or physical resource that hosts, manipulates, or interacts with other computational or physical resources.
Device	Represents a physical IT resource upon which system software and artifacts may be stored or deployed for execution.
System software	Represents software that provides or contributes to an environment for storing, executing, and using software or data deployed within it.
Technology collaboration	Represents an aggregate of two or more technology internal active structure elements that work together to perform collective technology behavior.
Technology interface	Represents a point of access where technology services offered by a node can be accessed.
Path	Represents a link between two or more nodes, through which these nodes can exchange data, energy, or material.
Communication network	Represents a set of structures that connects nodes for transmission, routing, and reception of data.
Technology function	Represents a collection of technology behavior that can be performed by a node.
Technology process	Represents a sequence of technology behaviors that achieves a specific result.
Technology interaction	Represents a unit of collective technology behavior performed by (a collaboration of) two or more nodes.
Technology event	Represents a technology state change.
Technology service	Represents an explicitly defined exposed technology behavior.
Artifact	Represents a piece of data that is used or produced in a software development process, or by deployment and operation of an IT system.

Table 17: Archimate 3.1 Technology Layer Entities

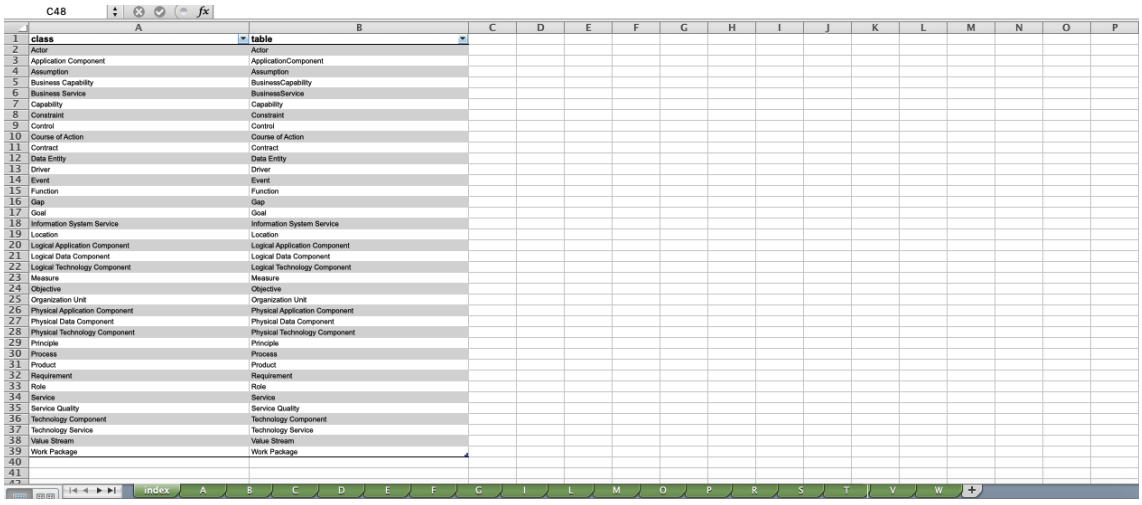
Metamodel Entity (Motivational Elements)	Description
Stakeholder	Represents the role of an individual, team, or organization (or classes thereof) that represents their interests in the effects of the architecture.
Driver	Represents an external or internal condition that motivates an organization to define its goals and implement the changes necessary to achieve them.
Assessment	Represents the result of an analysis of the state of affairs of the enterprise with respect to some driver.
Goal	Represents a high-level statement of intent, direction, or desired end state for an organization and its stakeholders.
Outcome	Represents an end result.
Principle	Represents a statement of intent defining a general property that applies to any system in a certain context in the architecture.
Requirement	Represents a statement of need defining a property that applies to a specific system as described by the architecture.
Constraint	Represents a factor that limits the realization of goals.
Meaning	Represents the knowledge or expertise present in, or the interpretation given to, a concept in a particular context.
Value	Represents the relative worth, utility, or importance of a concept.

Table 18: Archimate 3.1 Motivational Entities

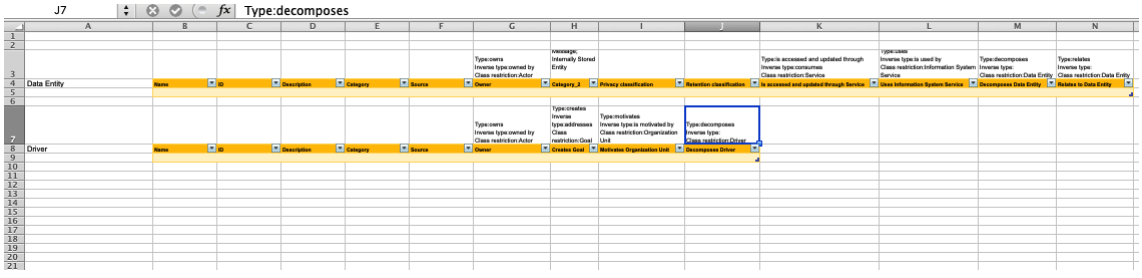
Metamodel Entity (Strategy Elements)	Description
Resource	Represents an asset owned or controlled by an individual or organization.
Capability	Represents an ability that an active structure element, such as an organization, person, or system, possesses.
Value stream	Represents a sequence of activities that create an overall result for a customer, stakeholder, or end user.
Course of action	Represents an approach or plan for configuring some capabilities and resources of the enterprise, undertaken to achieve a goal.

Table 19: Archimate 3.1 Strategy Entities

F XSLT TOGAF Entities Definition File - Example

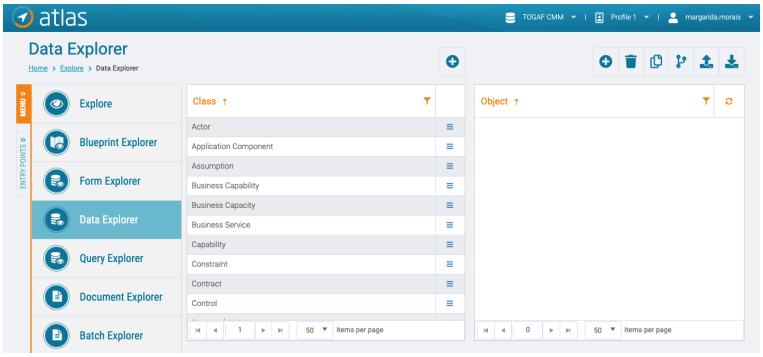


	A	B
1	class	table
2	Actor	Actor
3	Application Component	ApplicationComponent
4	Assumption	Assumption
5	Business Capability	BusinessCapability
6	Business Service	BusinessService
7	Capability	Capability
8	Constraint	Constraint
9	Control	Control
10	Course of Action	Course of Action
11	Contract	Contract
12	Data Entity	Data Entity
13	Driver	Driver
14	Event	Event
15	Function	Function
16	Gap	Gap
17	Goal	Goal
18	Information System Service	Information System Service
19	Location	Location
20	Logical Application Component	Logical Application Component
21	Logical Data Component	Logical Data Component
22	Logical Technology Component	Logical Technology Component
23	Measure	Measure
24	Objective	Objective
25	Organization Unit	Organization Unit
26	Physical Application Component	Physical Application Component
27	Physical Data Component	Physical Data Component
28	Physical Technology Component	Physical Technology Component
29	Principle	Principle
30	Process	Process
31	Product	Product
32	Requirement	Requirement
33	Role	Role
34	Service	Service
35	Service Quality	Service Quality
36	Technology Component	Technology Component
37	Technology Service	Technology Service
38	Value Stream	Value Stream
39	Work Package	Work Package

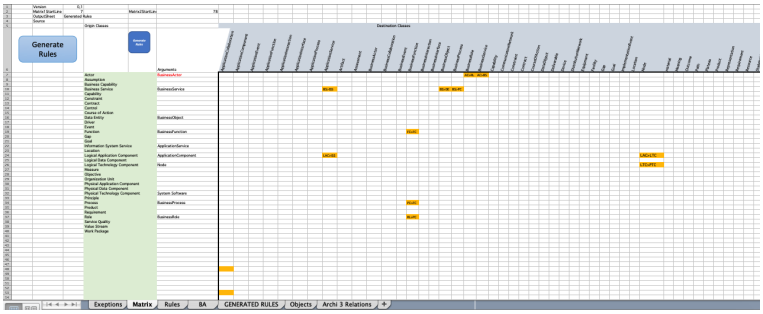


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
4	Data Entity	Name	ID	Description	Category	Source	Owner	Category_3	Privacy classification	Retention classification	Is accessed and updated through Service	Uses Information System Service	Decomposes Data Entity	Relates to Data Entity
8	Driver	Name	ID	Description	Category	Source	Owner	Creates Goal	Motivates Organization Unit	Decomposes Driver				

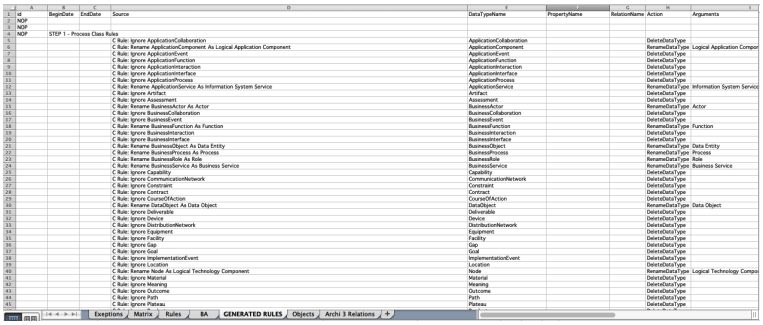
G Atlas - TOGAF Content Metamodel Repository



H XSLT Relationships Definition File - Example



Rule	Relation	Map To::Property	Mirror	Check	Rule	MapTo:: Property	Reverse	Check
AC>RL	performs task in	AssignmentRelationship	D	1				
AC>BS	consumes	ServingRelationship	D	1				
BS>PC	is realized by	RelaizationRelationship	C	1				
PC>FC	decomposes	AggregationRelationship	D	1				
FC>FC	communicates with	TriggeringRelationship	D	1				
BS>ISS	is realized through	RelaizationRelationship	C	1				
LAC>ISS	implements	RealizationRelationship	D	1				
LAC>LTC	is served by	ServingRelationship	C	1				
LTC>PTC	is realized by	RealizationRelationship	C	1				
BS>DE	consumes	AccessRelationship	D	1				
RL>PC	performs	AssignmentRelationship	D	1				



	A	B	C	D	E	F	G	H	I
1	id	BeginDate	EndDate	Source	DataTypeName	PropertyName	RelationName	Action	Arguments
2	NDP								
3	NDP								
4	NDP	STEP 1 - Process Class Rules							
5				C Rule: Ignore ApplicationCollaboration	ApplicationCollaboration			DeleteDataType	
6				C Rule: Rename ApplicationComponent As Logical Application Component	ApplicationComponent			RenameDataType	Logical Application Compor
7				C Rule: Ignore ApplicationEvent	ApplicationEvent			DeleteDataType	
8				C Rule: Ignore ApplicationFunction	ApplicationFunction			DeleteDataType	
9				C Rule: Ignore ApplicationInteraction	ApplicationInteraction			DeleteDataType	
10				C Rule: Ignore ApplicationInterface	ApplicationInterface			DeleteDataType	
11				C Rule: Ignore ApplicationProcess	ApplicationProcess			DeleteDataType	
12				C Rule: Rename ApplicationService As Information System Service	ApplicationService			RenameDataType	Information System Service
13				C Rule: Ignore Artifact	Artifact			DeleteDataType	
14				C Rule: Ignore Assessment	Assessment			DeleteDataType	
15				C Rule: Rename BusinessActor As Actor	BusinessActor			RenameDataType	Actor
16				C Rule: Ignore BusinessCollaboration	BusinessCollaboration			DeleteDataType	
17				C Rule: Ignore BusinessEvent	BusinessEvent			DeleteDataType	
18				C Rule: Rename BusinessFunction As Function	BusinessFunction			RenameDataType	Function
19				C Rule: Ignore BusinessInteraction	BusinessInteraction			DeleteDataType	
20				C Rule: Ignore BusinessInterface	BusinessInterface			DeleteDataType	
21				C Rule: Rename BusinessObject As Data Entity	BusinessObject			RenameDataType	Data Entity
22				C Rule: Rename BusinessProcess As Process	BusinessProcess			RenameDataType	Process
23				C Rule: Rename BusinessRole As Role	BusinessRole			RenameDataType	Role
24				C Rule: Rename BusinessService As Business Service	BusinessService			RenameDataType	Business Service
25				C Rule: Ignore Capability	Capability			DeleteDataType	
26				C Rule: Ignore CommunicationNetwork	CommunicationNetwork			DeleteDataType	
27				C Rule: Ignore Constraint	Constraint			DeleteDataType	
28				C Rule: Ignore Contract	Contract			DeleteDataType	
29				C Rule: Ignore CourseOfAction	CourseOfAction			DeleteDataType	
30				C Rule: Rename DataObject As Data Object	DataObject			RenameDataType	Data Object
31				C Rule: Ignore Deliverable	Deliverable			DeleteDataType	
32				C Rule: Ignore Device	Device			DeleteDataType	
33				C Rule: Ignore DistributionNetwork	DistributionNetwork			DeleteDataType	
34				C Rule: Ignore Equipment	Equipment			DeleteDataType	
35				C Rule: Ignore Facility	Facility			DeleteDataType	
36				C Rule: Ignore Gap	Gap			DeleteDataType	
37				C Rule: Ignore Goal	Goal			DeleteDataType	
38				C Rule: Ignore ImplementationEvent	ImplementationEvent			DeleteDataType	
39				C Rule: Ignore Location	Location			DeleteDataType	
40				C Rule: Rename Node As Logical Technology Component	Node			RenameDataType	Logical Technology Compo
41				C Rule: Ignore Material	Material			DeleteDataType	
42				C Rule: Ignore Meaning	Meaning			DeleteDataType	
43				C Rule: Ignore Outcome	Outcome			DeleteDataType	
44				C Rule: Ignore Path	Path			DeleteDataType	
45				C Rule: Ignore Plateau	Plateau			DeleteDataType	

Exeptions | Matrix | Rules | BA | GENERATED RULES | Objects | Archi 3 Relations