An Approach for Transforming IT-Network Diagrams into Enterprise Architecture Models

Daniel Maciejewski and Nicolas Mayer
Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg
maciejewski.dani@gmail.com, nicolas.mayer@list.lu

Keywords: Network Diagram, Enterprise Architecture Model, ArchiMate, Model Transformation, CISCO.

Abstract: Enterprise Architecture (EA) models propose to capture the activities of an organization, going from its business aspects to its IT infrastructure. Such an approach is promising to support reasoning on specific concerns, especially those relying on the business-to-IT stack, such as enterprise transformation, security, IT investment, etc. However, most of the organizations do not have existing EA models, and are reluctant to establish them, especially from scratch, mainly due to the length and complexity to do so. Our insight is that we can leverage on network diagrams, one of the most common kind of models available in organizations, to generate part of EA models. In this paper, we propose an approach to transform network diagrams into EA models. In this context, we focus on CISCO as the reference for network concepts, and ArchiMate as the standard modelling language for EA.

1 INTRODUCTION

Enterprise Architecture Management (EAM) has shown to be a valuable and engaging instrument to face enterprise complexity and the necessary enterprise transformation (Saha, 2013; Zachman, 1987). It offers means to govern enterprises and make informed decisions: description of an existing situation, investigation and expression of strategic direction, analysis of gaps, planning at the tactical and operational level, selection of solutions, and architecture design (Op't Land et al., 2008). As part of a global EAM adoption, describing Enterprise Architecture (EA) with a suited language (i.e. EA modelling) is considered as a key activity (Lankhorst, 2005) and the scope of our paper is focused on this concern.

However, the current problem is that most organisations pay few attention to the modelling of their structure. Among the current limitations to a broader adoption of EA modelling, we particularly noticed (Lankhorst, 2005):

- EA modelling is a complex task and requires specific skills
- EA modelling is a time consuming task, especially when started from scratch
- There are numerous and disregarded modelling tools that can be used

To deal with these issues, our main assumption is that the reuse of existing material (i.e. existing models) would help in the development and adoption of EA models and also reduce their development length. In this context, IT network diagrams are usual existing material in the structure of an organization, and appear to be a good basis to EA modelling. Indeed, each structure needs network models in order to support infrastructure design and most of them have them available.

In this paper, our aim is to propose an approach to use IT network diagrams to generate (part of) EA models. Our insight is that it is a relevant and realistic beginning to develop EA models, allowing saving time by accelerating part of the EA modelling tasks. It is worth to note we consider that such an approach would never produce complete and satisfactory EA models, but we see it as an interesting trigger to start the design of such models. In order to limit the scope of this project, we will focus on ArchiMate (The Open Group, 2013) as the exploited EA Modelling Language (EAML), but we are aware that others could be relevant too (BPMN (Object Management Group, 2011), UML (Fowler, 2003), etc.).

This paper is structured as follows. In the subsequent section, we provide some background knowledge about the main literature we use: CISCO as a *de facto* standard for Network Architecture Design and the ArchiMate Language, especially its technology layer. Section 3 describes the first step of our research method that is the definition of a network concept classification, followed, in Section 4, by the integration of these concepts into ArchiMate. We finish with concluding remarks and future work.

2 BACKGROUND

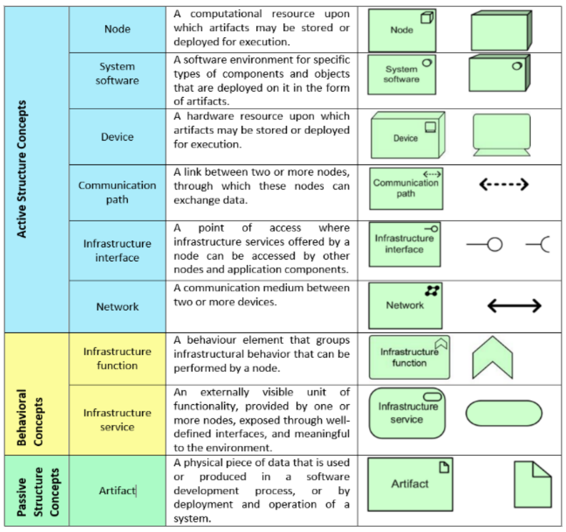
2.1 CISCO as a *de facto* Standard for Network Architecture Design

Network architecture design is the science to design good networks, making them safe and available (Stewart et al., 2008). IT Network concepts are numerous, especially because of the number of different technologies, products, companies, etc. We found a lot of information in uncertified articles or unprofessional tutorial to design a network diagram. However, to the best of our knowledge, there is a lack of standard or reference model depicting the concepts at stake, and in our context, it is a major issue. The current state-of-practice is actually the use of technology-specific terms and concepts. By reviewing major modelling software proposing network architecture modelling features (VISIO, Gliffy…), our conclusion is that CISCO is considered as a *de facto* standard. Hence, we decided to focus on CISCO concepts, expecting to cover a major proportion of existing concepts. The survey of literature for network architecture is consequently focused on CISCO. We distinguish three main sources of reference material related to CISCO:

- **CISCO documentation**: a set of documents listing main CISCO network concepts (e.g. Technology, protocol, or product). In this category, we identified the following documents: CISCO Iconography (Cisco Systems Inc., 2014), CISCO Product Quick Reference Guide (Cisco Systems Inc., 2013).
- **CISCO Website**: the CISCO website is a field-based source. It is the first interface between CISCO up-to-date products and clients.
- **Modelling Software**: most of modelling software uses modelling set of objects called "stencils", one among the available ones being generally based on CISCO concepts. We have only used those for which the CISCO stencil was sufficiently developed and specified, and we have set apart the others (e.g. Microsoft VISIO). The analysed modelling software are:
  - **Gliffy** (Gliffy, 2017): Gliffy is a cloud-based diagramming web application founded in 2005, allowing collaborative work. In our work, we focus specifically on the library concerning CISCO objects.
  - **CISCO Packet Tracer** (Cisco Systems Inc., 2017): Packet Tracer is a simulation program designed by CISCO Systems. The software allows users to create network topologies and reproduce nowadays computer networks.

2.2 ArchiMate Technological Layer

Enterprise Architecture (EA) is defined 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 (Lankhorst, 2005). To provide a uniform representation for diagrams that describe EA, the ArchiMate modelling language (The Open Group, 2013) has been produced by The Open Group, an industry consortium developing standards. It offers an integrated architectural approach to describe and visualize the different architecture domains and their underlying relations and dependencies. The role of the ArchiMate standard is to provide a graphical language for the representation of EA over time, as well as their motivation and rationale. It is today a widely accepted open standard for modelling EA (Vernadat, 2014), with a large user base and a variety of modelling tools that support it.



Active Structure Concepts	Node	A computational resource upon which artifacts may be stored or deployed for execution.	Node	
	System software	A software environment for specific types of components and objects that are deployed on it in the form of artifacts.	System software	
	Device	A hardware resource upon which artifacts may be stored or deployed for execution.	Device	
	Communication path	A link between two or more nodes, through which these nodes can exchange data.	Communication path	<---->
	Infrastructure interface	A point of access where infrastructure services offered by a node can be accessed by other nodes and application components.	Infrastructure interface	—○ —(
	Network	A communication medium between two or more devices.	Network	⟷
Behavioral Concepts	Infrastructure function	A behaviour element that groups infrastructural behavior that can be performed by a node.	Infrastructure function	
	Infrastructure service	An externally visible unit of functionality, provided by one or more nodes, exposed through well-defined interfaces, and meaningful to the environment.	Infrastructure service	
Passive Structure Concepts	Artifact	A physical piece of data that is used or produced in a software development process, or by deployment and operation of a system.	Artifact	

Figure 1: Definitions and visual representation of concepts of the ArchiMate technology layer.

ArchiMate proposes a 3-layered architecture for structuring its core concepts: the Business Layer, Application Layer, and Technology Layer. In our perspective focused on the modelling of network infrastructure, we will particularly analyse the Technology Layer that provides infrastructural services needed to support business. Figure 1, extracted from the ArchiMate 2.1 Specifications (The Open Group, 2013), gives an overview of the Technology Layer concepts, including definition and visual representation for each concept.

3 ALIGNMENT AND INTEGRATION BETWEEN CISCO AND NETWORK CONCEPTS

Alignment between CISCO and network concepts is the first phase of our contribution. First, for each analysed literature reference, we need to determine concept classes (i.e. a concept classification for each literature reference). Each class can be seen as a set of modelling objects (or instances of generic concepts) and the set of classes creates a taxonomy of network concepts of a given literature reference.

Then, we need to map the defined taxonomies together thanks to a classification methodology. The objective of this alignment is to propose a consolidated list of network generic classes. The classification methodology uses refinement cycles based on criteria to do the most relevant arrangement between network classes. Thus, each chosen classification pass through each criterion cycle, which correspond to different levels of details and analysis:

- **Syntax:** the first level is the semantic analysis of the class *i.e.* name's similarities or spelling syntax. For example, a grouping called "*Routing*", can easily be associated to another grouping called "*Routers*".
- **Network subdomains:** the second level of analysis aims at determining, for each instance of the class, to which network fields it belongs. For example, "*Connecting Safety and Security*" gather physical security concepts (doors, cameras, etc.) and IT security concepts (firewall, guards, etc.). It helps us to exclude concepts which are out of our scope (in that case *Physical Security* concepts).
- **Functionality:** To remove any ambiguity, a detailed analysis of each instance functionalities is required. In fact, an instance can belong at a time to different classes (or to different network subdomains). There is a need to analyse objectively to which class each instance belongs the most. For example, a wireless router can belong to the "*Wireless class*" and to the "*Router class*". Thanks to the predominant functionality, we can say that a wireless router is a router and effectively belong to the "*Router class*".

We mainly focus on the most relevant sources of our survey, in order to make these "instance's groups" the most significant towards the network engineering field. First, we focus on the CISCO Documentation (Cisco Systems Inc., 2014, 2013), also including the CISCO website. Then, we chose Gliffy Online Platform (Gliffy, 2017), and finally, the simulation software CISCO Packet Tracer (Cisco Systems Inc., 2017). Then, we applied the classification methodology described previous1y on each reference, to refine the network classification. The output of the process creates a classification for our instances. We're getting the so-called *"Network concepts classification",* presented below.

This consolidated list expresses main CISCO-based generic network concepts. By using this classification, we will be able for instance to classify other concept's instances coming from other brand models. We use a granularity generic enough, which allows us to have a viable classification on time. We give a description for each class of our network concepts classification and examples of what we find in each of them:

Collaboration: It regroups all instances that aim with the teamwork. We find conferencing devices, IP phones, communicating software, etc.

Security: This class regroups all security products including physical security and IT security. Thus, we have devices such as monitoring cameras, firewalls, etc. Also, we can find some security software products (e.g. IOS Firewall).

Switch: The class gather all different types of switches that exist in some various forms depending on the technology used or the functionalities incorporated (e.g. Multilayer Switch, etc.*).*

Router: The class regroups the various types of routers (e.g. Router with Firewall, NetFlow Router, etc.*).*

Wireless: Collection of wireless devices, or wireless modules (e.g. access points, WLAN controller, etc.)

Software: This is a class which gather different type of software solutions, mainly from CISCO. We especially find system software, like OS.

Transport/Telephony: This class gather all devices that concern the physical devices of telephony and transport. We find mainly devices that deals with the first layer of the OSI model like modems, DSLAM, etc.

End Device: It concerns end devices like personal computers, TVs, printers, etc.

Physical Server: It regroups the different types of existing servers like Web servers, certificates servers, file servers, etc.

Data Storage: This class is derived from the physical servers' class. Indeed, because of the huge number of existing data storage types' instances (storage module, data monitoring software, etc.) it is essential to distinguish it from the physical devices.

Management: It is a field's class that gathers management devices, modules and software. In big networks, it is crucial to consider management devices, or management software to access, control, or monitor devices.

Hub: Similarly, to the *Router* class or the *Switch* class, it regroups hub concepts and instances. This type of devices is tending to disappear because of the technological advancement. However, we have to take into account old networks that could still have this type of devices.

Protocol: This class is particular because in general, protocols cannot be depicted in an architecture, apart from writing the name besides a device or a connections. However, CISCO has created protocols' concepts like IP protocols, or FDDI Ring that allow representing some kind of protocols.

Topology: Such as the protocol class, it is possible to represent the topology of a network.

Connections: In a network diagram, connections are represented in different manners. For example, in CISCO Packet Tracer, we are able to recognize optical link, wireless communication, etc. This class collects all type of connections that can be materialized in a network diagram.

Network Architecture: It represents different types of network architectures like Cloud, CISCO layered architecture, etc. In diagrams that use modularization, it is possible to represent a network architecture through a dedicated icon representing this concept.

4 CONSTRUCTION OF A NETWORK-ARCHIMATE METAMODEL

4.1 Conceptual Alignment

After having analysed CISCO concepts with their instances, we have created the network concepts classification following our methodology, as depicted in the previous section. As a reminder, the goal of our research work is to use network diagrams so as to generate a basis of EA models. Therefore, this second step aims at creating a relationship between our network concepts classification and ArchiMate Technology Layer concepts. Thus, in order to integrate these global concepts (*i.e.* classes*)* into the ArchiMate metamodel, we need to respect the ArchiMate structure (*i.e.* rules and concepts) and use the ArchiMate metamodel as an input. We called this relationship the "integration step".

In order to do this, it is necessary to add network generic classes to the ArchiMate Technology Layer metamodel. Consequently, we will be able to represent a network diagram with the ArchiMate Technology Layer concepts, thanks to the metamodel rules (*i.e.* links between concepts of the metamodel).

To allow this conceptual alignment, we use relationships inspired from the approach of semantic correspondence between concepts from Zivkovic et al. (Zivkovic et al., 2007). Themselves inspired by UML relationships (Fowler, 2003), we explain these relationships below:

- ***Equivalence***: *concept A* is semantically equivalent to *concept B*;
- ***Generalisation***: *concept A* is a generalisation of *concept B*, i.e. *concept B* is a specific class of *concept A*;
- ***Specialisation***: *concept A* is a specialisation of *concept B*, i.e. *concept B* is a generic class of *concept A*;
- ***Aggregation***: *concept A* is composed of *concept B*, i.e. *concept B* is a part of *concept A*;
- ***Composition***: *concept A* is composed of *concept B* (with strong ownership), i.e. *concept B* is a part of *concept A* and does only exist as part of *concept A*;
- ***Association***: *concept A* is linked to *concept B*

By confronting each instance (of network generic classes) with ArchiMate, we realize that we can affect to each instance one of the Active Structure concepts of the ArchiMate Technology Layer, i.e. Node, System software, Device, Communication path, Infrastructure interface and Network. Actually, the ArchiMate framework can represent its own network diagrams, as CISCO, that's the reason there are some similarities between our created network generic classes and Active Structure concepts. Table 1 show these equivalences we identified. Table 1 presents two types of equivalences:

- First, we have equivalences between some instances of (network generic classes) and leaf's concepts of the ArchiMate metamodel (Device, Node, System Software). These latter are the most used concepts for network diagram conception in the EA context.

- Secondly, we noticed that some classes can be mapped as is to ArchiMate main concepts. For example, instances of our *Connections* class are in some way equivalent to the ArchiMate *Communication Path* class instances.

Table 1: Equivalence between type of instances and ArchiMate concepts.

Type of classes instances	ArchiMate	Semantic mapping
Devices instances	Device	*Equivalence*
Modules instances (Node)	Node	
System Software	System software	
Connections	Communication Path	

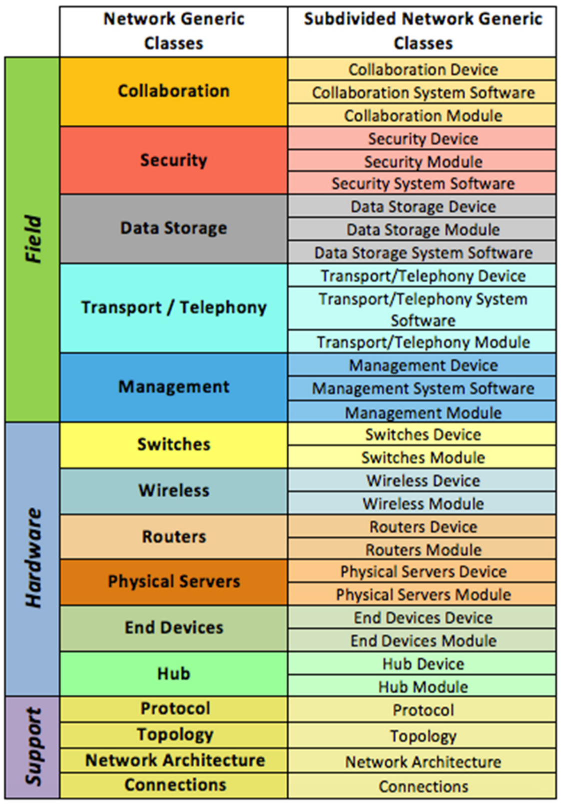
To use Network generic classes as concepts of ArchiMate, we have to add our concepts to the ArchiMate metamodel without modifying its structure. In fact, we can only improve the metamodel by specializing ArchiMate concepts or finding equivalence between both types of concepts. Thus, there is two relevant relations from Zivkovic et al. we can use: specializations and equivalences.

Specialization is a type of semantic mapping used for concepts that are a part of ArchiMate concepts (see Figure 1). For example, physical router devices of our *Router* class are specializations of the ArchiMate *Device* concept. Thus, there is a need to split our network generic classes in order to make possible the addition of instances to the ArchiMate metamodel, and thus to be compliant with ArchiMate.

Table 2 expresses the different partitions we had to make in order to be able to specialize the ArchiMate metamodel with our concepts. Table 2 presents three main type of classes:

- Network generic classes that are enough generic to propose Device instances, System Software instances and Module instances, which are the most used ArchiMate concepts for network diagramming in EA context. We called these classes "*Field*" because of their huge number of different type of instances.
- Some network generic classes are specifically concerning hardware instances. In the ArchiMate context, it concerns only physical type of instances (Physical router, End devices, etc.)
- Remaining network classes can't be divided, because of too specific instances, such as specific topologies, or protocol. In that context, it is irrelevant to subdivide these network generic classes. We called them the "*Support*" classes.

Table 2: Partition of network generic classes.



	Network Generic Classes	Subdivided Network Generic Classes
Field	Collaboration	Collaboration Device
		Collaboration System Software
		Collaboration Module
	Security	Security Device
		Security Module
		Security System Software
	Data Storage	Data Storage Device
		Data Storage Module
		Data Storage System Software
	Transport / Telephony	Transport/Telephony Device
		Transport/Telephony System Software
		Transport/Telephony Module
	Management	Management Device
		Management System Software
		Management Module
Hardware	Switches	Switches Device
		Switches Module
	Wireless	Wireless Device
		Wireless Module
	Routers	Routers Device
		Routers Module
	Physical Servers	Physical Servers Device
		Physical Servers Module
	End Devices	End Devices Device
		End Devices Module
	Hub	Hub Device
		Hub Module
Support	Protocol	Protocol
	Topology	Topology
	Network Architecture	Network Architecture
	Connections	Connections

Then, we create a detailed alignment table between previously subdivided network generic classes and ArchiMate concepts. Table 3 presents the semantic mapping:

Table 3: Detailed alignment table.

Subdivided Network Generic Classes	ArchiMate	Semantic mapping type
Collaboration Device	Device	Specialization
Collaboration System Software	System Software	
Collaboration Module	Module	
Security Device	Device	
Security Module	Module	
Security System Software	System Software	
Data Storage Device	Device	
Data Storage Module	Module	
Data Storage System Software	System Software	
Transport/Telephony Device	Device	
Transport/Telephony System Software	System Software	
Transport/Telephony Module	Module	
Management Device	Device	

Table 3: Detailed alignment table. (cont.)

Subdivided Network Generic Classes	**ArchiMate**	**Semantic mapping type**
Management System Software	System Software	Specialization
Management Module	Module	
Switches Device	Device	
Switches Module	Module	
Wireless Device	Device	
Wireless Module	Module	
Routers Device	Device	
Routers Module	Module	
Physical Servers Device	Device	
Physical Servers Module	Module	
End Devices Device	Device	
End Devices Module	Module	
Hub Device	Device	
Hub Module	Module	
Protocol	Interface	
Topology	Network	
Network Architecture	Network	
Connections	Communication Path	**Equi.**

4.2 Proposed Network-ArchiMate Metamodel

Based on the alignment performed, we need to integrate the classes of our network classification into the existing ArchiMate metamodel, following and complying with the existing structure and associated rules. The mapping table permits us to integrate the concepts into the ArchiMate metamodel. Thanks to the semantic rules, we are able to propose the integrated Network-ArchiMate metamodel, as represented in Figure 2. We give also the example of Firewall instances, which we scattered into corresponding type of concept (*i.e.* Device, System Software, Module).

5 CONCLUDING REMARKS

The approach presented in this paper aims at proposing a way to transform network diagrams into EA models. First, we had to suggest a network concepts taxonomy that allows us to classify each network concept and its instances in these generic classes. Then, by aligning these classes with ArchiMate concepts, we proposed a metamodel which allows us to integrate network-specific concepts with ArchiMate.
The approach presented here, and especially the integrated metamodel, have been used in a fictitious case to experiment the transformation from network diagram to an ArchiMate model. This experiment, not described here for sake of brevity, has shown that our

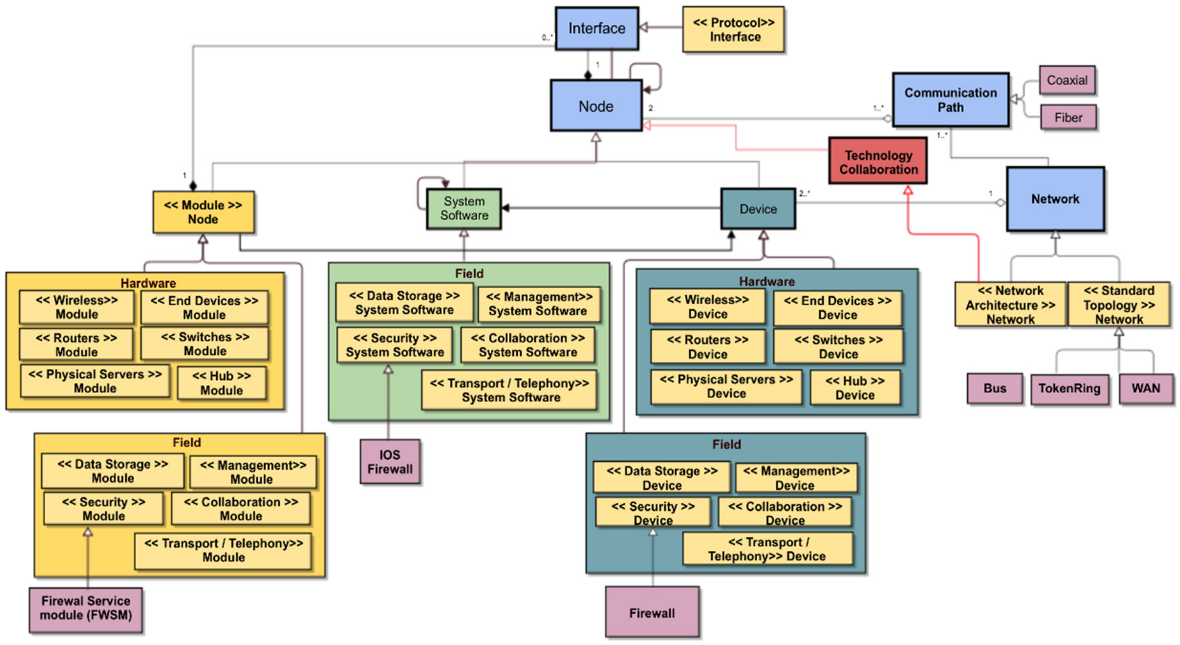


Figure 2: Proposed Network-ArchiMate metamodel.

approach is applicable. It has also shown some ways of improvement, such as the introduction of additional specifications (e.g. a color code) in the ArchiMate language in order to keep trace of the original network class, this information being lost in the transformation proposed. However, no conclusion can be drawn from this experiment at the level of soundness and usefulness of our approach in a real-world context and further validation work is deemed as necessary to consider our approach as valid.

Regarding future work, we first need to experiment our approach on a wider and real world case. Then, it is necessary to assess with users the relevance of the approach especially at the level of the soundness of the EA models obtained compared to the initial network models, as well as their usefulness as a starting point to design complete EA models.

ACKNOWLEDGEMENTS

Supported by the National Research Fund, Luxembourg, and financed by the ENTRI project (C14/IS/8329158).

REFERENCES

Cisco Systems Inc., 2013. Cisco Product Quick Reference Guide.

Cisco Systems Inc., 2014. Cisco Iconography.

Cisco Systems Inc., 2017. Packet Tracer.

Fowler, M., 2003. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gliffy, 2017. Gliffy Online User Manual.

Lankhorst, M. (Ed.), 2005. Enterprise Architecture at Work: Modelling, Communication And Analysis. Springer.

Object Management Group, 2011. Business Process Model and Notation (BPMN) 2.0.

Op't Land, M., Proper, H.A., Waage, M., Cloo, J., Steghuis, C., 2008. Enterprise Architecture - Creating Value by Informed Governance, Enterprise Engineering. Springer Berlin Heidelberg.

Saha, P. (Ed.), 2013. A Systemic Perspective to Managing Complexity with Enterprise Architecture: IGI Global.

Stewart, K., Adams, A., Reid, A., Lorenz, J., 2008. Designing and Supporting Computer Networks, CCNA Discovery Learning Guide, 1st ed. Cisco Press, Indianapolis, Ind.

The Open Group, 2013. ArchiMate® 2.1 Specification (No. Open Group Standard (C13L)).

Vernadat, F., 2014. Enterprise Modeling in the context of Enterprise Engineering: State of the art and outlook. International Journal of Production Management and Engineering 2, 57.

Zachman, J.A., 1987. A framework for information systems architecture. IBM Systems Journal 26, 276–292.

Zivkovic, S., Kuhn, H., Karagiannis, D., 2007. Facilitate Modelling Using Method Integration: An Approach Using Mappings and Integration Rules. In: Proceedings of the 15th European Conference on Information Systems (ECIS 2007).