



universität
wien

**Modeling Digital Enterprise Ecosystems with ArchiMate: A
Mobility Provision Case Study**

Benedikt Pittl and Dominik Bork

Accepted for:

5th International Conference on Serviceology, July 12 - 14, 2017,
Vienna, Austria

OMiLAB[®]
www.omilab.org

Modeling Digital Enterprise Ecosystems with ArchiMate: A Mobility Provision Case Study

Benedikt Pittl and Dominik Bork

Research Group Knowledge Engineering
Faculty of Computer Science
University of Vienna, Austria
`{firstname.lastname}@univie.ac.at`

Abstract. Currently there is a shift from product centered enterprises to product-service centered enterprises which rely on a network of customers, suppliers and partners called enterprise ecosystems. This trend also affects the underlying IT architecture which has to integrate and provide software components (e.g. services) as well as hardware components (e.g. sensors) leading to a digital enterprise ecosystem. This digital ecosystem is complex so that modeling approaches which aim on simplifying complexity are eligible for their design and management. In the paper at hand, we show that existing enterprise modeling approaches are inappropriate for modeling digital enterprise ecosystems comprehensively. By using a case-based analysis we sketch extension points for a digital enterprise ecosystem modeling method based on ArchiMate.

Keywords: Domain-Specific Modeling, Enterprise Ecosystem, Enterprise Modeling

1 Introduction

Currently, there is a shift from a *product* oriented to a *product-service* oriented economy [10]. Thereby enterprises face challenges in creating new value streams, business models, and IT architectures. Hilti's *Tools on Demand* business model as well as new enterprises trading services such as Uber or Airbnb are results of this trend. According to [32], the service trend leads to enterprises which rely on a tight network of stakeholders such as customers, suppliers and cooperation partners. This network is referred to as enterprise ecosystem. A vital part of such ecosystems is the tight integration of the underlying IT infrastructure (e.g. servers, cloud services, third party applications). To reflect this distinguishing character, the term *digital enterprise ecosystem* is used. This paper elaborates the importance of models for designing digital enterprise ecosystems which encompass the integration and provisioning of software components as well as hardware components. The design as well as the coordination of such ecosystems is complex and can be considered as a key challenge of enterprises in the future. Modeling approaches, with the aim on simplifying complex systems, are an appropriate instrument for describing them.

Enterprise modeling approaches integrate multiple perspectives or views to derive a coherent and comprehensive description (cf. [6]). See for example 4EM [33], ArchiMate [18], Semantic Object Model [12], or Integrated Enterprise Balancing [15]. All these enterprise modeling approaches were designed with the aim on generality: any enterprise can be described with them. However, the description of digital enterprise ecosystems with enterprise modeling approaches is unfeasible. This is because they currently lack in expressiveness when describing digital enterprise ecosystems. Existing enterprise modeling approaches need to evolve in order to comprehend current developments. In this paper, the ArchiMate modeling approaches serves as a starting point to identify current weaknesses and propose extensions for modeling digital enterprise ecosystems.

Our research endeavour is to create domain-specific conceptual modeling methods for digital enterprise ecosystems. Thereby we use the term conceptual model as defined in [29], which describes them as models used for communication and understanding by human beings. In the paper at hand, we analyze the appropriateness of ArchiMate to express the specifics of digital enterprise ecosystems and propose first ideas of how an extension of ArchiMate could be achieved to meet the aforementioned requirements.

The remainder of the paper is structured as follows: Section 2 describes foundations of modeling methods in context of the service-product trend and some related work. A mobility provision case study is described in section 3. Along this case study we develop requirements as well as a sketch of a domain-specific language for digital enterprise ecosystems. The paper is closed with a conclusion in section 4.

2 Foundations

2.1 Conceptual Modeling Methods

The authors of [20] define a modeling method as consisting of a modeling language, modeling procedure, and modeling algorithms. An overview of a modeling method is depicted in Figure 1. The modeling language consists of syntax, semantics, and notation. The modeling algorithms are algorithms which are executed on the modeling language. They range from generic algorithms (e.g. shortest path algorithms) to specialized algorithms (e.g. process and capacity simulations). The modeling procedure describes concrete steps which the modeler has to follow in order to create valid models.

According to this definition of a modeling method, most of the popular modeling approaches like BPMN or UML can be considered as modeling languages as they do neither define a modeling procedure nor modeling algorithms.

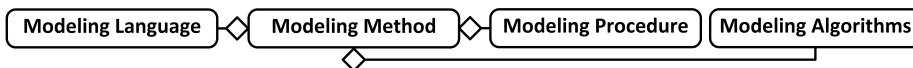


Fig. 1: Core components of modeling methods (excerpt from [20])

While enterprises and their environments require agility, modeling approaches seem to remain nearly unchanged. Prominent modeling methods like BPMN and ArchiMate are heavily used in science and industry, even if their update cycles are rather long: The update from BPMN 1.2 to the current version 2 took about two years¹, while ArchiMate 3.0 was released three years after its predecessor ArchiMate 2.1². Similarly, the latest UML specification was released approximately four years after the previous specification³. The stability of these so-called general purpose modeling languages is based on a rather high abstraction level implying limited semantic specificity. For example, an activity in BPMN can be used for describing processes using different levels of abstraction in almost all domains. Contrary to general purpose modeling languages, domain-specific modeling languages have a high expressiveness in a certain domain. Due to their strong domain focus such languages are usually not standardized and their awareness is low.

2.2 Related Work

The related work analysis is structured into two parts: the first part summarizes literature relevant for enterprise ecosystems while the second part summarizes existing enterprise modeling approaches which might be promising starting points for modeling digital enterprise ecosystems.

In [30], the authors elaborate the importance of currently emerging ecosystems which leads to a tight integration of different stakeholders. This integration requires a harmonization of software systems. Therefore, different approaches are summarized in [3]. Further enterprise integration aspects, beyond software system integration aspects, have been investigated in [32]. The author of [22] analyzes how the model quality framework Semiotic Quality Framework (SEQAL) can be used for evaluating models in the domain of digital ecosystems.

A generic reference architecture for modeling enterprises was described in [25], where the author introduces a business rule-, an activity-, a resource-, a business process- as well as an organisational view. All the enterprise modeling methods which we describe in the following have a comparable structure. The Multi-Perspective Enterprise Modeling (MEMO) method, introduced in [14] tool support described in [4], has a strategy, a organisational as well as an information system perspective whereby each perspective is organized along the four aspects structure, process, resources, goals. The 4EM modeling method [33] has the vision of a holistic description of enterprises. It encompasses a process model, a goal model, a rule model, a concept model, an actors and resource model as well as a technical component model. There are also research initiatives using the Unified Modeling Language (UML) for modeling enterprises [27]. The Open Group Architecture Framework (TOGAF) is a framework which aims on capturing enterprise architectures [16]. Thereby, TOGAF distinguishes between a

¹ <http://www.omg.org/spec/BPMN/>

² <http://opengroup.org/standards/ea>

³ <http://www.omg.org/spec/UML/>

business architecture, an application architecture, a data architecture as well as a technical architecture.

ArchiMate is a modeling method which distinguishes between a business layer, an application layer as well as a technology layer [18]. The business layer is about to describe business processes, the application layer describes software applications, and the technology layer allows to describe hardware as well as software infrastructures. It is designed for modeling enterprise architectures.

ArchiMate served as extension point for a couple of other modeling methods such as [11, 1]. In [15], the authors designed a modeling method for Integrated Enterprise Balancing with the aim of generating a common data structure. The Integrated Enterprise Modeling method was introduced in [28], it has a strong focus on processes of manufacturing enterprises.

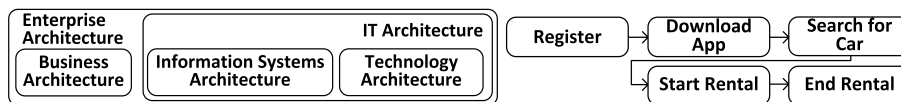
All of these modeling approaches focus on an isolated description of enterprises. What is missing is an approach for modeling digital enterprise ecosystem-specific concepts in a comprehensive and integrated manner.

3 Case Study: Digital Enterprise Ecosystem for Mobility Provision in the Automotive Industry

The trend towards product-services is a result of the digital transformation. For a better analysis of the effects, we follow the classification of [23], which describes that an enterprise architecture consist of a business architecture, an information systems architecture and a technology architecture as shown in Figure 2a. The information systems architecture forms together with the technology architecture the IT architecture.

Product-service centered enterprises have to redesign their business architecture, requiring also a redesign of the underlying IT architecture. Thus, in the information systems architecture layer, enterprises face challenges in integrating and provisioning of software components. Further, flexible application architectures are required which allow modifications immediately after changes occur in the environment. The technological architecture is increasingly distributed due to the usage of clouds as well as of other hardware components such as sensors. Their integration and coordination becomes a key challenge in the future.

The described enterprise architecture does not only allow a top-down approach where changes of the business model are propagated down to the IT architecture level. Disruptive technologies like the IoT [2] which belong to the IT architecture layer may force enterprises to modify their business architecture



(a) Enterprise architecture and IT architecture (b) Simplified scenario for renting a car at Car2Go

Fig. 2: Overview over enterprise architecture and mobility provision scenario

which represents a bottom-up approach [26]. The digital photography is an example of such a disruptive technology where corporations like Kodak failed to adapt their business models.

In the case study we describe the digital enterprise ecosystem for a car-rental scenario on a very high level of abstraction using the example of Car2Go (www.car2go.com). The scenario of renting a car is visualized in Figure 2b:

1. **Register.** New costumers have to enter their personal data including payment relevant information.
2. **Download App.** Customers have to download the Car2Go app to access all Car2Go services.
3. **Search for Car.** Customers have to find an available car in proximity to their current location. Therefore the app can be used as a route planer to find the next car.
4. **Start Rental.** The rental process of a car is triggered after the consumer found a car, opened it, and started it with the app.
5. **End Rental.** The rental ends when the customer leaves the car. Therby the customer is charge based on the chosen pricing model.

For executing such a scenario, services and devices are required belonging to the digital enterprise ecosystem as summarized in Table 1: The registration step requires a web application while the app store is required so that consumers can download the app. The search of cars requires communication with geographical services as well as with a car sensor. For initiating the rental process a communication to car sensors e.g. for unlocking the car as well as for tracking the car trip is necessary. The end rental process step requires an integration with payment services. Overall, parts of the data are stored on an external cloud.

3.1 Modeling the Mobility Provision Scenario with existing Approaches

Modeling of digital enterprise ecosystems such as described in the scenario is not trivial: For the overall process we could use BPMN which fails to model the technical aspects. Process modeling with enterprise modeling methods like 4EM are approximately on the same level of abstraction as the BPMN. Hence, they face similar problems. Contrary, the technical modeling languages grouped into the UML might be more appropriate for modeling the technical aspects even if the semantics of the resulting models is limited. However, the UML is inappropriate for modeling the business aspects.

ArchiMate with its business layer, application layer and technology layer seems to be an appropriate modeling method for describing such digital ecosystems. However, ArchiMate is a standard modeling language working with abstract concepts [24]. The modeling class representing sensors in the car, which is required in the previously described scenario, can only be represented with ArchiMate's device modeling class. The (informal) semantics of the device class is described in the standard [31]: *A device models a physical computational resource, upon which artifacts may be deployed for execution.* It is obvious that

modeling sensors with the device class is generally possible, however, the expressiveness of such modeling constructs is limited. Even prefixing the names of modeling elements of the type device with '*sensor*' leads only to limited expressiveness improvements for humans while the expressiveness for machines remains unchanged.

Step	Service	Sensors
1 Register	WebApp-Cloud	-
2 Download App	WebApp-Cloud	-
3 Search for Car	Geo-Service	Car GPS Sensor
4 Start Rental	Tracking-Service	Car Lock Sensor
5 End Rental	Payment-Service	Car Lock Sensor

Table 1: Summary of the scenario with its required services and sensors

An excerpt of a model for this scenario created with ArchiMate is depicted in Figure 3. The process model (shown on the top of the figure) seems to be appropriate for our purpose. The application model (shown in the center of the figure) is on a very abstract level of detail: We used ArchiMate's application component class for modeling the Car2Go Registration Site, the AppStore as well as the Car2Go App. The car information system which belongs to ArchiMate's technical layer is depicted on the lower left corner: Here, we used the device class for modeling the sensors. Further, we used the generic technology interface modeling class to elaborate that there are well defined interfaces for accessing the sensors. On the lower right corner, we modeled an excerpt of the server-side system which is used by the Car2Go App. The Car2Go server system uses an external Data Cloud which was modeled by using the generic node class. The Car2Go server accesses the cloud. Thereby we connected the cloud with the server system via the network model element which we named Car2Go App Server System. The server offers a tracking service which is accessed by the app. The tracking service is modeled by using a generic technical service class.

The described example elaborates that we have to improvise for modeling such a simple scenario using ArchiMate. ArchiMate's limitations in modeling precisely use cases is a well known fact [24]. Looking at the model without showing the name labels reveals that semantics of the model itself is very low due to the high level of abstraction of the ArchiMate modeling language. The names of the model elements such as 'External Data Cloud' contain the majority of the semantics. Simply placing information into the name labels works only for a limited amount of information. For example the name label of the 'External Data Cloud' is insufficient to describe the service level agreement which the Car2Go system has with the external cloud. Therefore, further classes or class attributes are necessary.

3.2 Requirements for Modeling a Digital Enterprise Ecosystem with ArchiMate

For an adequate modeling of the product-service scenario additional domain-specific concepts have to be introduced to ArchiMate. For the scenario model, classes such as 'Lock Sensor' and 'GPS Sensor' are necessary which have inter

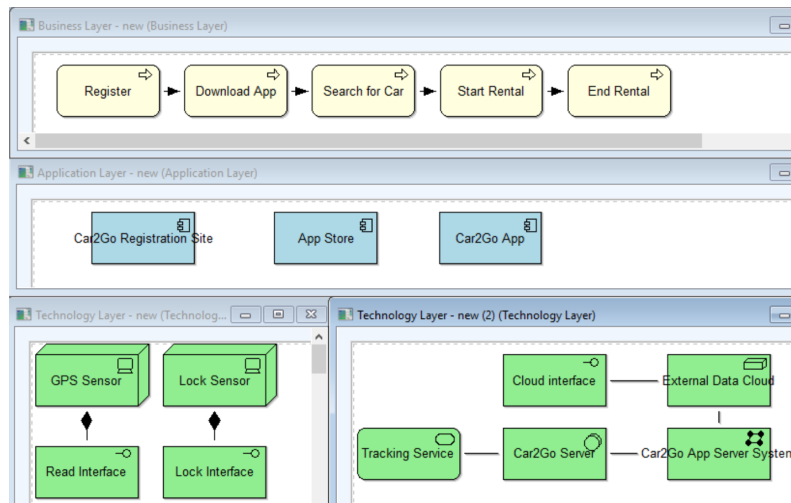


Fig. 3: Excerpt of the scenario model created with ArchiMate

alia the attributes 'communication protocol', 'status' as well as 'interface'. The expressiveness of the ArchiMate interface class is limited. Due to the high level of abstraction it is not clear if the interface is a hardware interface or a software interface. For the purpose of our scenario, it would be necessary to describe interfaces in more detail, e.g. by introducing a modeling class for expressing REST interfaces. ArchiMate is continuously extended by introducing new elements - in the current version 3.0, 56 different elements are existing⁴. However, due to ArchiMate's broad domain it is unfeasible to reach a semantic richness compared to particularly designed domain-specific languages.

A major drawback of ArchiMate is that it does not foresee attributes of modeling classes (except the name). So it is not possible to distinguish between external and internal nodes to e.g. model the external cloud. From ArchiMate's point of view the ignorance of attributes is clear: attributes describe details of classes. However, ArchiMate's classes are generic which simply do not have any details. Hence, all kinds of class-related information which would be appropriate to be a class attribute have to be expressed as a separate modeling classes.

An exemplary list of domain-specific classes and attributes required for modeling our scenario is shown in Table 2. This table is not a prime solution which allows to model the described scenario adequately. Instead it should elaborate that further modeling classes are necessary to model the scenario adequately. For creating these additional modeling classes ArchiMate's profiling and specification extension mechanisms can be used [31].

For fully leveraging the expressive power of ArchiMate, a formal description of how the model elements of different layers e.g. the application and technical layer can be connected is necessary [5]. This is for example necessary to describe that a certain sensor is used for an application which is executed for running

⁴ <https://masteringarchimate.com/2016/06/26/archimate-3-0-the-good-the-bad-and-the-ugly/> for a discussion

a certain process step. Currently, the standard is very vague regarding such inter-model connections. It describes for example that the interface model class which belongs to the technology layer [...] can be accessed by other nodes or by application components from the Application Layer.

Layer	Class	Attributes
Technology Layer	Lock Sensor	Name, Communication protocol, Status, Interface
Technology Layer	GPS Sensor	Name, Communication protocol, Precision, Interface
Technology Layer	WebService Interface	Name, Webservice type, Parameter, Functional description
Technology Layer	Cloud component	Name, Service type
Application Layer	Application Component	Name, Application type, External/Internal component

Table 2: Additional/extended model classes including attributes

3.3 Design of a Domain-Specific Modeling Language

In this section an excerpt of a modeling language for the description of the use case shown on the lower left corner of Figure 3 is introduced. The model elements, its notation as well as its semantics are described in Table 3. In total the domain-specific language contains three classes and one connector. For each of these elements we described the semantics informally using natural language (following the formalization framework as proposed by [5]). Additionally, we added the corresponding ArchiMate modeling element - which we used for modeling the use case in Figure 3 - as well as its semantics to the table. We took the semantic description of the ArchiMate elements from the standard [31].


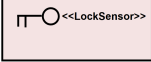

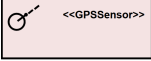

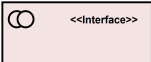

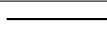
Concept	ArchiMate Element	Domain-Specific Element
Lock Sensor	 Device A device is a physical IT resource upon which system software and artifacts may be stored or deployed for execution.	 <<LockSensor>> A lock sensor is a physical device in a car for locking an unlocking it. It might offer digital interfaces for controlling it.
GPS Sensor	 Device A device is a physical IT resource upon which system software and artifacts may be stored or deployed for execution.	 <<GPSSensor>> A GPS sensor is a physical device which tracks the position. It might offer digital interfaces for accessing the current position.
Interface	 Technology interface A technology interface represents a point of access where technology services offered by a node can be accessed.	 <<Interface>> An interface represents a technical interface (REST or SOAP) through which other applications can communicate, guide or control the system which offers the interface.
OffersInterface	 OffersInterface A path represents a link between two or more nodes, through which these nodes can exchange data or material.	 The offersInterfaces connector is used by sensors to describe that they offer an interface.

Table 3: Comparison of the elements of the domain-specific language with the corresponding elements in ArchiMate

To emphasize that the domain-specific modeling language is not only the introduction of a user-defined notation, we created an example which is depicted in Figure 4. It shows the model layer as well as the metamodel layer from the classical metamodeling stack (see e.g. [17] for more information). In the paper at hand we analyzed ArchiMate and the domain-specific language. Excerpts of their metamodels are shown on layer 2, the metamodel layer. On layer 1 there are two identical models which we created for a better readability of the figure. With both metamodels we are able to create the model elements used in the two models. From a syntactical point of view the introduced domain-specific modeling language is similar to the excerpt created with ArchiMate. The most obvious syntactical difference is that the domain-specific modeling language distinguishes between two sensor classes - in ArchiMate we used the device class for representing both sensors. Furthermore, as shown in Table 2 we enriched e.g. the interface class with attributes.

While the syntax of the two languages is similar, a comparison of the semantic description of the domain specific language and ArchiMate - see Table 3 - reveals that their semantics is different. In the right model (referenced with 1 in Figure 4) the sensors are represented by instances of the device class and the interfaces were created by instancing the technology interface class. Thereby, the instances like the GPS Sensor and the Lock Sensor inherit the semantics of the class *Device*. The semantics gained through this kind of inheritance is known as the *type semantics* [17]. As already described, the semantics of the classes in ArchiMate is limited and consequently, the inherited semantics of the corresponding instances is limited, too. Contrary, by using domain-specific languages the type semantics is richer. This semantic richness comes at the price of limited applicability for other domains. For example, the appliance of the previously introduced lock sensor class is limited to cars.

We will further develop the introduced language within the Open Models Initiative Laboratory (OMiLAB). OMiLAB is a research initiative for *conceptualization, development, and deployment of modeling methods and the models designed with them*⁵. OMiLAB hosts already projects in domains such as semantic alignment of models [13], multi-view modeling [7] and industry related domain-specific modeling [8]. For more information about the OMiLAB please see [19]. An overview of existing open modeling tools realized within the OMiLAB is given in [21].

4 Conclusion and Further Research

The trend towards a product-service driven economy leads to ecosystems where enterprises are tightly connected with other stakeholders and service providers. These connections lead to complex IT infrastructures that need to be integrated and managed in a digital enterprise ecosystem. Existing enterprise modeling methods have the problem that their expressiveness is limited. In this paper

⁵ <http://www.omilab.org/psm/about>

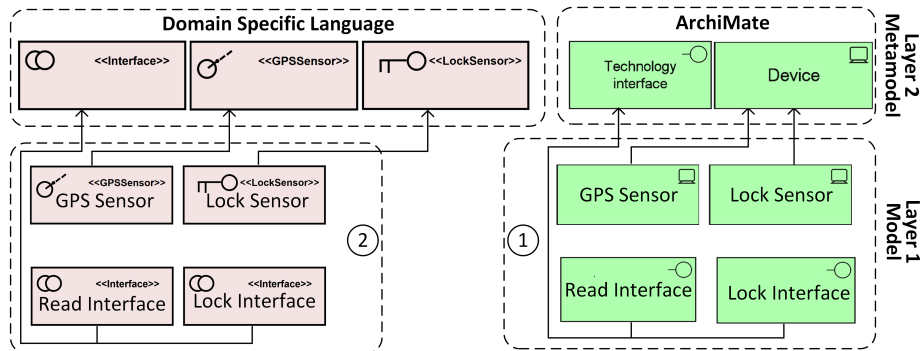


Fig. 4: Excerpt of the scenario model created with ArchiMate

we elaborate this problem by developing a scenario in the mobility provision domain. Based on the Car2Go scenario, we proposed domain-specific extensions for ArchiMate in order to increase its expressiveness and adequacy. In our future work we will develop a comprehensive digital enterprise ecosystem modeling method within the Open Models Initiative Laboratory (OMiLAB, www.omilab.org). moreover, we plan to design and develop an open modeling tool to support design and management of complex digital enterprise ecosystems. In this regard, we plan to investigate how we can use the created models as a diagrammatic source for knowledge engineering (cf. [9]).

Acknowledgement

Part of this research has been funded through the South Africa / Austria Joint Scientific and Technological Cooperation program with the project number ZA 11/2017.

References

1. Al-Fedaghi, S.: Enterprise architecture: An alternative to archimate conceptualization. In: Silhavy, R., Silhavy, P., Prokopova, Z., Senkerik, R., Kominkova Oplatkova, Z. (eds.) Software Engineering Trends and Techniques in Intelligent Systems: Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 3, pp. 68–77. Springer International Publishing, Cham (2017)
2. Asghar, M.H., Negi, A., Mohammadzadeh, N.: Principle application and vision in internet of things (iot). In: Computing, Communication & Automation (ICCCA), 2015 International Conference on. pp. 427–431. IEEE (2015)
3. Barbosa, O., Alves, C.: A systematic mapping study on software ecosystems. In: Proceedings of the Third International Workshop on Software Ecosystems, Brussels, Belgium, June 7th, 2011. pp. 15–26 (2011)
4. Bock, A., Frank, U.: Multi-perspective enterprise modeling - conceptual foundation and implementation with *adoxx*. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, pp. 241–267. Springer (2016)

5. Bork, D., Fill, H.G.: Formal aspects of enterprise modeling methods: a comparison framework. In: System Sciences (HICSS), 2014 47th Hawaii International Conference on. pp. 3400–3409. IEEE (2014)
6. Bork, D.: A Development Method for the Conceptual Design of Multi-View Modeling Tools with an Emphasis on Consistency Requirements. Ph.D. thesis, University of Bamberg (2015)
7. Bork, D.: Using conceptual modeling for designing multi-view modeling tools. In: 21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13-15, 2015. Association for Information Systems (2015)
8. Buchmann, R.A.: Modeling Product-Service Systems for the Internet of Things: The ComVantage Method. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 417–437. Springer (2016)
9. Buchmann, R.A., Karagiannis, D.: Domain-specific diagrammatic modelling: a source of machine-readable semantics for the internet of things. *Cluster Computing* 20(1), 895–908 (2017)
10. Chew, E.K.: Service innovation for the digital world. *Enterprise Modelling and Information Systems Architectures* 9(1), 70–89 (2015)
11. Engelsman, W., Jonkers, H., Quartel, D.: Archimate® extension for modeling and managing motivation, principles, and requirements in togaf®. White paper, The Open Group (2011)
12. Ferstl, O.K., Sinz, E.J., Bork, D.: Tool support for the semantic object model. In: Dimitris Karagiannis, Heinrich C. Mayr, J.M. (ed.) Domain-Specific Conceptual Modeling, pp. 291–310. Springer (2016)
13. Fill, H.G.: Semantic evaluation of business processes using semfis. In: Dimitris Karagiannis, Heinrich C. Mayr, J.M. (ed.) Domain-Specific Conceptual Modeling, pp. 149–170. Springer (2016)
14. Frank, U.: Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages. In: System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on. pp. 1258–1267. IEEE (2002)
15. Gericke, A., Fill, H.G., Karagiannis, D., Winter, R.: Situational method engineering for governance, risk and compliance information systems. In: Proceedings of the 4th international conference on design science research in information systems and technology. p. 24. ACM (2009)
16. Haren, V.: TOGAF Version 9.1. Van Haren Publishing (2011)
17. Höfferer, P.: Achieving business process model interoperability using metamodels and ontologies. In: ECIS. pp. 1620–1631 (2007)
18. Iacob, M.E., Jonkers, H., Lankhorst, M.M., Proper, H.A.: ArchiMate 1.0 Specification. Zaltbommel: Van Haren Publishing (2009)
19. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M.: Fundamental conceptual modeling languages in omilab. In: Dimitris Karagiannis, Heinrich C. Mayr, J.M. (ed.) Domain-Specific Conceptual Modeling, pp. 3–30. Springer (2016)
20. Karagiannis, D., Kühn, H.: Metamodeling Platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, Proceedings. Lecture Notes in Computer Science, vol. 2455, p. 182. Springer (2002)
21. Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.): Domain-Specific Conceptual Modeling, Concepts, Methods and Tools. Springer (2016), <http://dx.doi.org/10.1007/978-3-319-39417-6>
22. Krogstie, J.: Modeling of digital ecosystems: Challenges and opportunities. In: Working Conference on Virtual Enterprises. pp. 137–145. Springer (2012)

23. Kurbel, K.E.: *Developing Information Systems*. Springer (2008)
24. Lankhorst, M.: Introduction to enterprise architecture. In: *Enterprise Architecture at Work*, pp. 1–10. Springer (2013)
25. Liles, D.H., Presley, A.R.: Enterprise modeling within an enterprise engineering framework. In: *Proceedings of the 28th conference on Winter simulation*. pp. 993–999. IEEE Computer Society (1996)
26. Lyytinen, K., Rose, G.M.: The disruptive nature of information technology innovations: the case of internet computing in systems development organizations. *MIS quarterly* pp. 557–596 (2003)
27. Marshall, C.: *Enterprise modeling with UML: designing successful software through business analysis*. Addison-Wesley Professional (2000)
28. Mertins, K., Jochem, R.: Integrated enterprise modeling: method and tool. *ACM SIGGROUP Bulletin* 18(2), 63–66 (1997)
29. Mylopoulos, J.: *Conceptual modelling and Telos. Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development*, New York: John Wiley & Sons pp. 49–68 (1992)
30. Nachira, F., Dini, P., Nicolai, A.: *A network of digital business ecosystems for europe: roots, processes and perspectives*. European Commission, Bruxelles, Introductory Paper (2007)
31. OpenGroup: Archimate standard (2016), <http://pubs.opengroup.org/architecture/archimate3-doc/toc.html>, accessed: 2017-02-23
32. Panetto, H., Jardim-Goncalves, R., Molina, A.: Enterprise integration and networking: theory and practice. *Annual Reviews in Control* 36(2), 284–290 (2012)
33. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: *Enterprise modeling. Tackling Business Challenges with the 4EM Method*. Springer 309 (2014)