Enterprise Architecture Development and Modelling

Combining TOGAF and ArchiMate

Marc Lankhorst, Hans van Drunen

In this article, we explore the possibilities of combining ArchiMate, a modelling language for enterprise architecture (EA), with TOGAF, The Open Group Architecture Framework, a design method for EA. We focus on the use of views and viewpoints and investigate how these two methods may complement each other.

Introduction

In current business practice, an integrated approach to business and IT is indispensable. Take for example a company that needs to assess the impact of introducing a new product in its portfolio. This may require defining additional business processes, hiring extra personnel, changing the supporting applications, and augmenting the technological infrastructure to support the additional load of these applications. Perhaps this may even require a change of the organizational structure.

Transferring new information technology to practice requires that a company has a clear, integrated vision on the relation between its business and IT. Without such a vision, the IT infrastructure will never adequately support the business, and vice versa, the business will not optimally profit from IT developments. A vast amount of literature has been written on the topic of strategic alignment, underlining the significance of both "soft" and "hard" components of an organization. Henderson and Venkatraman [1993], for example, distinguish between organizational strategy and organizational infrastructure on the one hand, and IT strategy and IT infrastructure on the other hand. Achieving alignment between business and IT requires an integrated approach to all aspects of the enterprise. Organizational effectiveness is not obtained by local optimizations, but is realised by well-orchestrated interaction of organizational components [Nadler et al., 1992].

Enterprise architecture is an important instrument to address this company-wide integration. It is a coherent whole of principles, methods and models that are used in the design and realization of the enterprise's organizational structure, business processes, information systems, and infrastructure [Bernus et al., 2003]. However, in practice these domains are mostly not approached in an integrated way. Every domain speaks its own language, draws its own models, and uses its own techniques and tools. Communication and decision making across domains is seriously impaired.

To create such an integrated perspective on enterprise architecture, one needs both a description technique for these architectures, and a method for architectural design in which this technique is employed. In this paper, we present the marriage between these two elements: the enteprise modelling language ArchiMate [Lankhorst et al., 2005] and The Open Group Ar-

chitecture Framework (TOGAF), and specifically its Architecture Development Method (ADM) [The Open Group, 2006].

Enterprise Architecture Methods

In order to define the field and determine the scope of enterprise architecture both researchers and practitioners have produced a number of "architecture frameworks". Frameworks provide structure to the architectural descriptions by identifying and sometimes relating different architectural domains and the modelling techniques associated with them. Well-known examples of architectural frameworks are:

- Zachman's "framework for enterprise architecture" [Zachman 1987; Sowa and Zachman, 1992]. The Zachman framework is widely known and used. The framework is a logical structure for classifying and organising the representations of an enterprise architecture that are significant to its stakeholders. It identifies 36 views on architecture ("cells"), based on six levels (scope, enterprise, logical system, technology, detailed representations and functioning enterprise) and six aspects (data, function, network, people, time, motivation).
- The Reference Model for Open Distributed Processing (RM-ODP) is an ISO/ITU Standard [ITU, 1996] which defines a framework for architecture specification of large distributed systems. It identifies five viewpoints on a system and its environment: enterprise, information, computation, engineering and technology.
- The architectural framework of The Open Group (TOGAF) is completely incorporated in the TOGAF methodology [The Open Group, 2006]. A main component of TOGAF is a high-level framework which defines three main views: Business Architecture, Information System Architecture and Technology Architecture.

Most of these architecture frameworks are quite precise in establishing what elements should be part of an enterprise architecture. Please beware that with any architecture framework, the objective is not to play "framework bingo" and fill all cells or parts of the framework! Rather, within a framework you decide which elements are relevant to your own architectural problem.

To keep the enterprise architecture coherent during its lifecycle, the adoption of a certain framework is not sufficient. The relations between the relevant types of domains, views or layers of the architecture must remain clear, and any change should methodically be carried through in all of them to ensure consistency. For this purpose, a number of methods are available, which assist architects through all phases of the lifecycle of architectures. This is where TOGAF distinguishes itself from the other frameworks mentioned. Moreover, TOGAF is supported by a large community of practioners and is an open standard, unlike the various vendor-specific methods for enterprise architecture.

TOGAF

The Open Group Architecture Framework (TOGAF) originated as a generic framework and methodology for development of technical architectures, but evolved into an enterprise architecture framework and method. Version 8 of TOGAF [The Open Group, 2006] is called the "Enterprise Edition" and is dedicated to enterprise architectures.

TOGAF has three main components (Figure 1):

- The Architecture Development Method (ADM), to derive an organisation-specific Enterpise Architecture
- The TOGAF Enterprise Continuum, which illustrates how architectures are developed across a continuum ranging from foundational architectures, through common systems architectures, and industry-specific architectures, to an enterprise's own individual architectures, specific for the enterprise models & generic models in the IT industry
- The TOGAF Resource Base, techniques available for use in applying TOGAF and the TOGAF ADM (architecture views, business scenarios, case studies, other architecture

frameworks, guidelines, templates, a mapping of TOGAF to the Zachman framework, etc.).

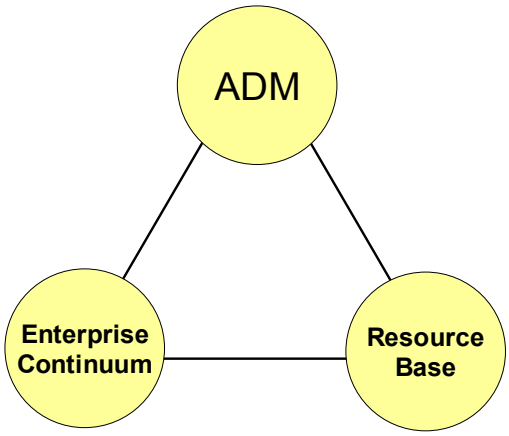

Figure 1. TOGAF [The Open Group, 2006].

TOGAF Architecture Development Method

Central to the discussion in this paper is TOGAF's Architecture Development Method (ADM). The framework considers an overall Enterprise Architecture as composed of a set of closely interrelated Architectures: Business Architecture, Information Systems Architecture (comprising Data Architecture and Application Architecture), and Technology (IT) Architecture. ADM is considered to be the core of TOGAF, and consists of a stepwise cyclic iterative approach for the development of the overall enterprise architecture (Figure 2).

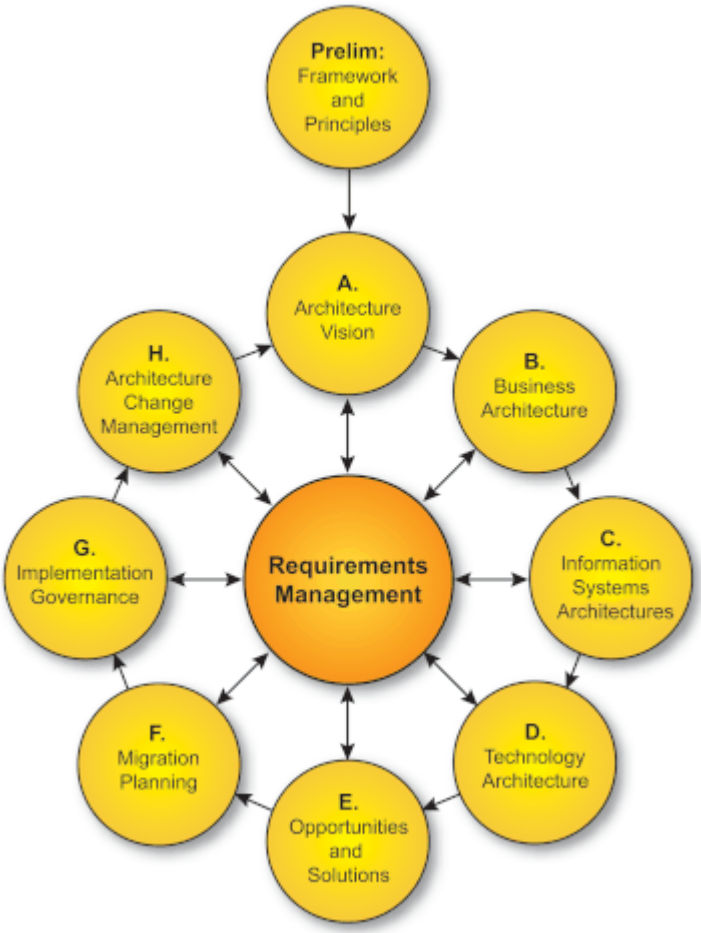

Figure 2. TOGAF ADM development process [The Open Group, 2006].

The ADM is iterative, over the whole process, between phases, and within phases. For each iteration of the ADM, a fresh decision must be taken as to:

- The breadth of coverage of the enterprise to be defined
- The level of detail to be defined
- The extent of the time horizon aimed at, including the number and extent of any intermediate time horizons
- The architectural assets to be leveraged in the organization's Enterprise Continuum, including assets created in previous iterations of the ADM cycle within the enterprise and assets available elsewhere in the industry

These decisions need to be made on the basis of a practical assessment of resource and competence availability, and the value that can realistically be expected to accrue to the enterprise from the chosen scope of the architecture work.

As a generic method, the ADM is intended to be used by enterprises in a wide variety of different geographies and applied in different vertical sectors/industry types. As such, it may be, but does not necessarily have to be, tailored to specific needs. For example:

- It may be used in conjunction with the set of deliverables of another framework, where these have been deemed to be more appropriate for a specific organization. (For example, many US federal agencies have developed individual frameworks that define the deliverables specific to their particular departmental needs)
- It may be used in conjunction with the well-known Zachman Framework, which is an excellent classification scheme, but lacks an openly available, well-defined methodology

Architecture Views in TOGAF

Apart from these components, TOGAF identifies a number of views, which are to be modelled in an architecture development process. The architecture views, and corresponding viewpoints fall into the following categories (the TOGAF taxonomy of views is compliant with the IEEE Std 1471-2000 [IEEE, 2000]):

- Business Architecture Views, which address the concerns of the users of the system, and describe the flows of business information between people and business processes (e.g. People View, Process View, Function View, Business Information View, Usability View, Performance View).
- **Information Systems Architecture views**, comprising **Data Architecture views** and **Applications Architecture views**, address the concerns of the database designers and administrators, and the system and software engineers of the system. They focus on how the system is implemented from the perspective of different types of engineers (security, software, data, computing components, communications), and how that affects its properties. Systems and software engineers are typically concerned with modifiability, re-usability, and availability of other services.
- **Technology Architecture views** address the concerns of the acquirers, operators, communications engineers, administrators, and managers of the system.
- **Composite views**, such as the Enterprise Manageability Views, addressing the concerns of systems administrators, operators and managers, and Enterprise security view

Figure 3. Views in the TOGAF ADM development process [The Open Group, 2006].

Enterprise Modelling

Modelling languages are an essential instrument for the description and communication of architectures, and languages and tools have evolved more or less "hand in hand". In some cases methodologies and frameworks have grown around and are supplied together with architecture support tools, for instance in the case of UML and Rational, EPCs and ARIS [Scheer, 1994], and Testbed [Eertink et al., 1999]. In other cases, tool vendors have strived to endow their tools with new functionality in order to support frameworks or other modelling notations such as UML [Object Management Group, 2003] or the IDEF family [IDEF, 1993], besides their own proprietary notations (e.g., ARIS, System Architect). Languages and modelling notations are at the core of all these architecture support packages.

Most languages mentioned provide concepts to model specific domains, e.g., business processes or software architectures, but rarely do they model the high-level relationships between these different domains. In current practice, architectural descriptions are made for the different domains. Although, to a certain extent, modelling support within each of these domains is available, well-described concepts to describe the relationships *between* the domains are almost completely missing. Such concepts are essential to tackle the problems of business–IT alignment and architecture optimization in a systematic way.

Thus, in order to facilitate a service-oriented and model-driven approach to enterprise architecture, a high-level modelling language is needed in which the different conceptual domains and their relations can be described at a sufficiently abstract level. ArchiMate [Lankhorst et al., 2005] is such a language, in which the service concept plays a central role. The objective of the ArchiMate language is to provide well-defined relationships between concepts in different architectures, the detailed modelling of which may be done using other, standard or proprietary modelling languages. Concepts in the ArchiMate language currently cover the business, application, and technology layers of an enterprise. Services offered by one layer to another play an important role in relating the layers.

A similar movement towards integrated models and tools can be recognised in the Model Driven Architecture (MDA) approach to software development [Frankel, 2003]. MDA is a collection of standards of the Object Management Group (OMG) that raise the level of abstraction at which software solutions are specified. Typically, MDA results in software development tools that support specification of software in UML instead of in a programming language like Java.

The ArchiMate Language

Within many of the different domains of expertise that are present in an enterprise, some sort of architectural practice exists, with varying degrees of maturity. As we have previously described, all kinds of frameworks try to map these architecture domains. However, due to the heterogeneity of the methods and techniques used to document the architectures, it is very difficult to determine how the different domains are interrelated. Still, it is clear that there are strong dependencies between the domains. For example: the goal of the (primary) business processes of an organization is to realise their products; software applications support business processes, while the technical infrastructure is needed to run the applications; information is used in the business processes and processed by the applications. For optimal communication between domain architects, needed to align designs in the different domains, a clear picture of the domain interdependencies is indispensable.

With these observations in mind, we conclude that a language for modelling *enterprise architectures* should focus on inter-domain relations. With such a language, we should be able to model both the global structure *within* each domain, showing the main elements and their dependencies, and the relations *between* the domains, in a way that is easy to understand for non-experts.

To this end, we have defined the ArchiMate language [Lankhorst et al., 2005], an enterprise architecture modelling language that is gaining rapid acceptance in the Netherlands and abroad.

To create a language that is easy to learn and understand, we have limited its set of concepts and have created a number of basic elements that you will see throughout the various layers of the language. First, we distinguish between the *structural* or *static* aspect and the *behavioural* or *dynamic* aspect. Behavioural concepts are assigned to structural concepts, to show who or what displays the behaviour. In addition to *active* structural elements (the business actors, application components and devices that display actual behaviour, i.e., the 'subjects' of activity), we also recognize *passive* structural elements, i.e., the *objects* on which behaviour is performed.

Second, we make a distinction between an *external view* and an *internal* view on systems. When looking at the behavioural aspect, these views reflect the principles of service orientation as introduced in the previous section. The *service* concept represents a unit of essential functionality that a system exposes to its environment. For the external users, only this external functionality, together with non-functional aspects such as the quality of service, costs etc., are relevant. Services are accessible through *interfaces*, which constitute the external view on the structural aspect.

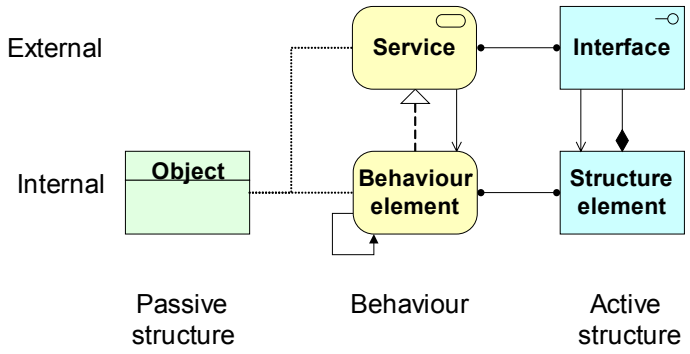

Figure 4. Core concepts of the ArchiMate language.

Although, at an abstract level, the concepts that are used throughout EA models in ArchiMate are similar (Figure 4), ArchiMate defines more concrete concepts that are specific for a certain layer of the architecture. In this context, we distinguish three main layers:

1. The **Business layer** offers products and services to external customers, which are realised in the organization by business processes (performed by business actors or roles).
2. The **Application layer** supports the business layer with application services which are realised by (software) application components.
3. The **Technology layer** offers infrastructural services (e.g., processing, storage and communication services) needed to run applications, realised by computer and communication devices and system software.

This results in the language framework shown below. In this framework, we have projected commonly occurring architectural domains.

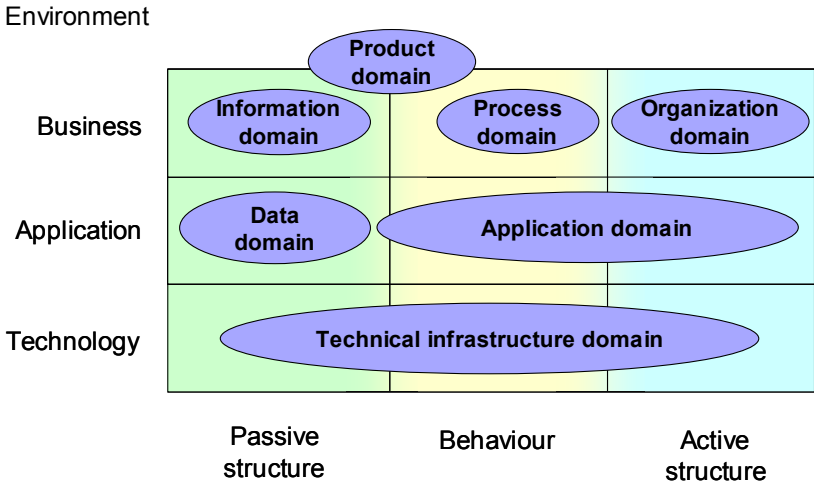

Figure 5. ArchiMate language framework.

The most important of the modelling concepts at the different layers of the framework are explained below. For a more detailed description please refer to [Lankhorst et al., 2005].

Business Layer Concepts

The main structural concepts at the business layer (Figure 7) are business roles and business actors, an entity that performs behaviour such as business processes or functions. A *business role* signifies responsibility for one or more *business processes* or *business functions*. A business function denotes the high-level capabilities of an organization, and offers functionality that may be used in business processes to realize the products and services of the organization. Business functions can be connected through *flows* that describe the information or goods exchanged.

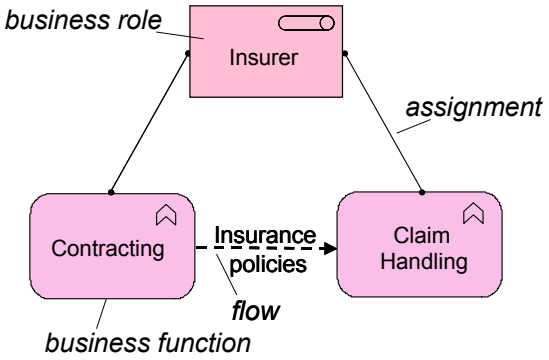

Figure 6. Business roles and business functions.

A business role is typically assigned to a business actor. *Business actors* may be individual persons (e.g. customers or employees), but also groups of people and resources that have a permanent (or at least long-term) status within the organizations. Business processes, which may be triggered by *events* and manipulate *business objects*, describe the business behaviour of a role. The externally visible behaviour of a business process is modelled by the concept of *business service*, which represents a unit of functionality that is meaningful from the point of view of the environment. Not shown in the example is that services can be grouped to form (financial or information) *products*, together with a *contract* that specifies the associated characteristics, rights and requirements.

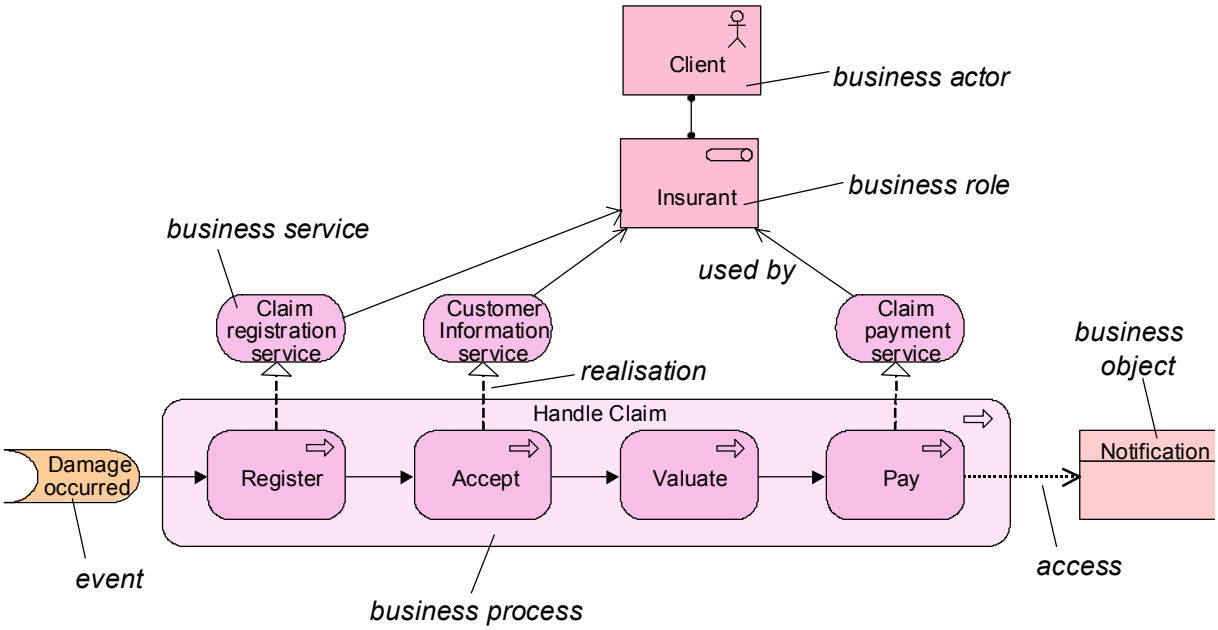

Figure 7. Example of a business process model.

Figure 8 illustrates how an *interaction* and *collaboration* can be used to model a business transaction and how the same situation can be modelled with the service and interface concepts. These two alternatives can be seen as two views, a symmetrical ('peer-to-peer') view and an asymmetrical ('client-server') view, on the same process. In the former view, the buyer and seller perform collaborative behaviour to settle a transaction, while in the latter view the selling of a product is considered to be a service that the seller offers to the buyer.

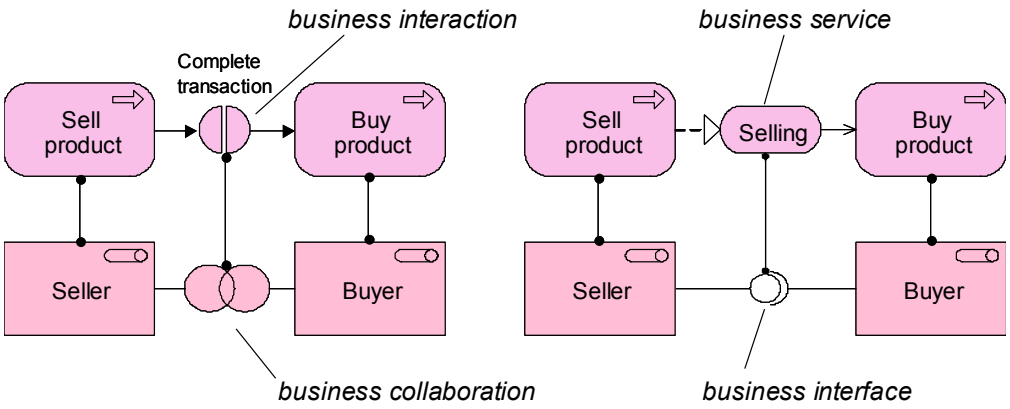

Figure 8. Interaction versus service use.

Application Layer Concepts

The main structural concept for the application layer (Figure 9) is the *application component*. This concept can be used to model any structural entity in the application layer: not just (re-usable) software components that can be part of one or more applications, but also complete software applications or information systems. Behaviour in the application layer can be described in a way that is very similar to business layer behaviour. Again, we make a distinction between the externally visible behaviour of application components in terms of *application services*, and the internal behaviour, *application functions,* that realise these services[1]. Services are offered through the *application interfaces* of an application*. Data objects* are used in the same way as data objects (or object types) in well-known data modelling approaches, most notably the 'class' concept in UML class diagrams.

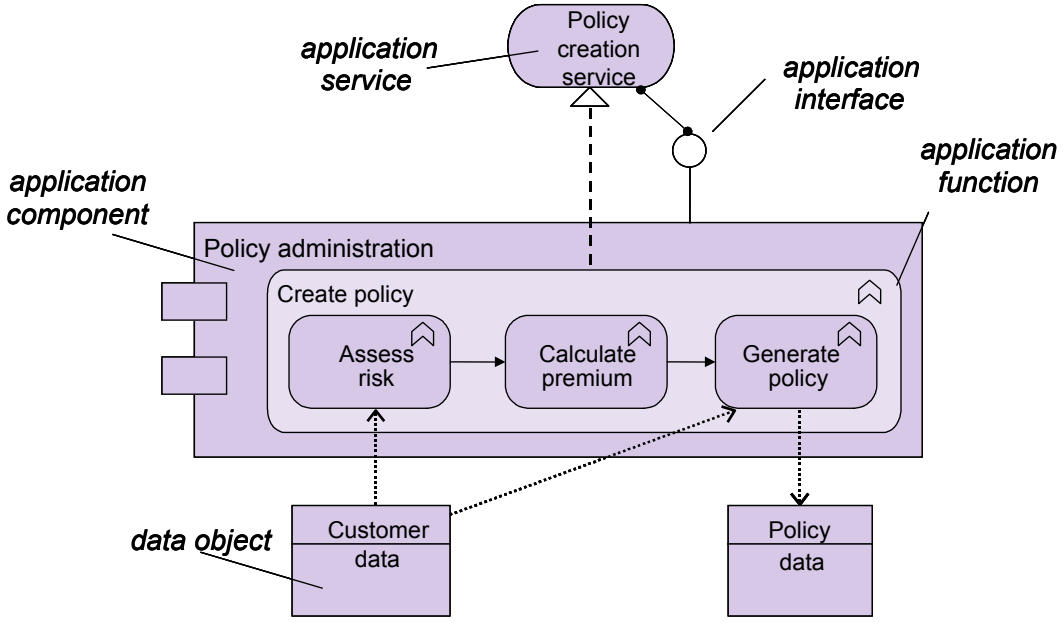

Figure 9. Example of an application model.

Information models (Figure 10) are very similar to a stripped down version of UML class diagrams. The *composition*, *aggregation*, *specialization*, and *realization* relations have been taken more or less directly from UML, with a slightly simplified semantics.

[1] Note that in the figure, we have used nesting to denote both the assignment relation between application functions and components and the composition relation of the application function "Create policy" with its subfunctions.

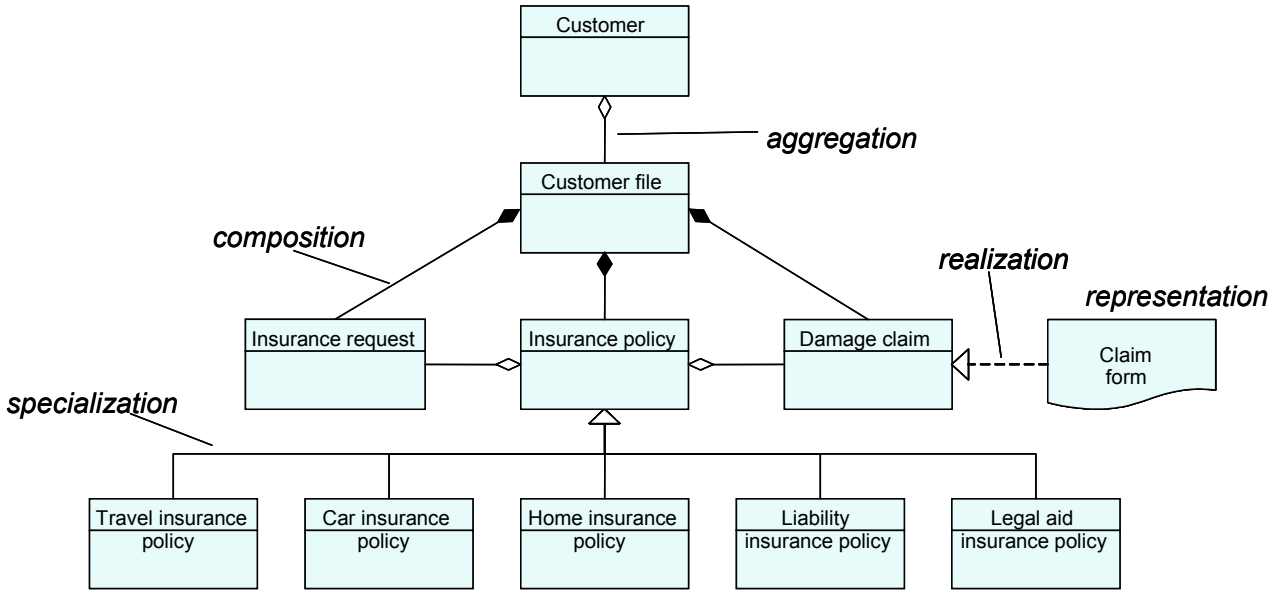

Figure 10. Example of an information model.

Technology Layer Concepts

The main structural concept for the technology layer (Figure 11) is the *node*. This concept is used to model structural entities in the technology layer. Nodes come in two flavours: *device* and *system software*, both inspired by UML 2.0 (the latter is called *execution environment* in UML). A device models a physical computational resource; system software represents the software environment for specific types of components and data objects. An *infrastructure interface* (not shown in the example) is the (logical) location where the infrastructural services offered by a node can be accessed by other nodes or by application components from the application layer.

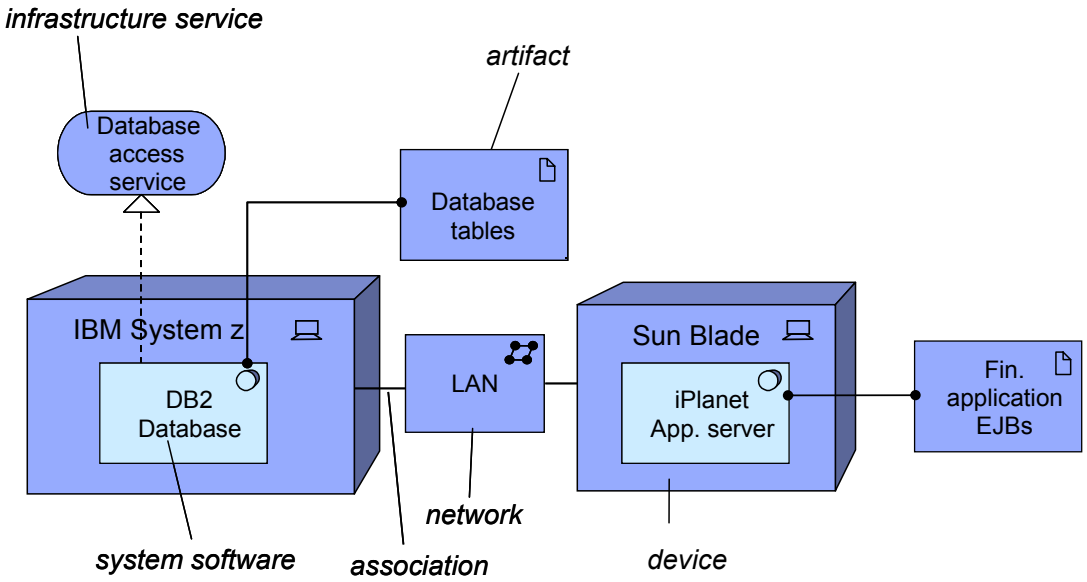

Figure 11. Example of a technology model.

An *artifact* is a physical piece of information that is used or produced in a software development process, or by deployment and operation of a system. A *network* models a physical communication medium between two or more devices. In the technology layer, the central behavioural concept is the *infrastructure service*. We do not model the internal behaviour of infrastructure components such as routers or database servers; that would add an amount of detail that is not useful at the enterprise level of abstraction.

Relations Between Layers

Figure 12 shows a small example of an integrated and service-oriented enterprise architecture model. This was constructed by connecting models from different layers, such as those shown in the previous sections, by means of services and realization relations.

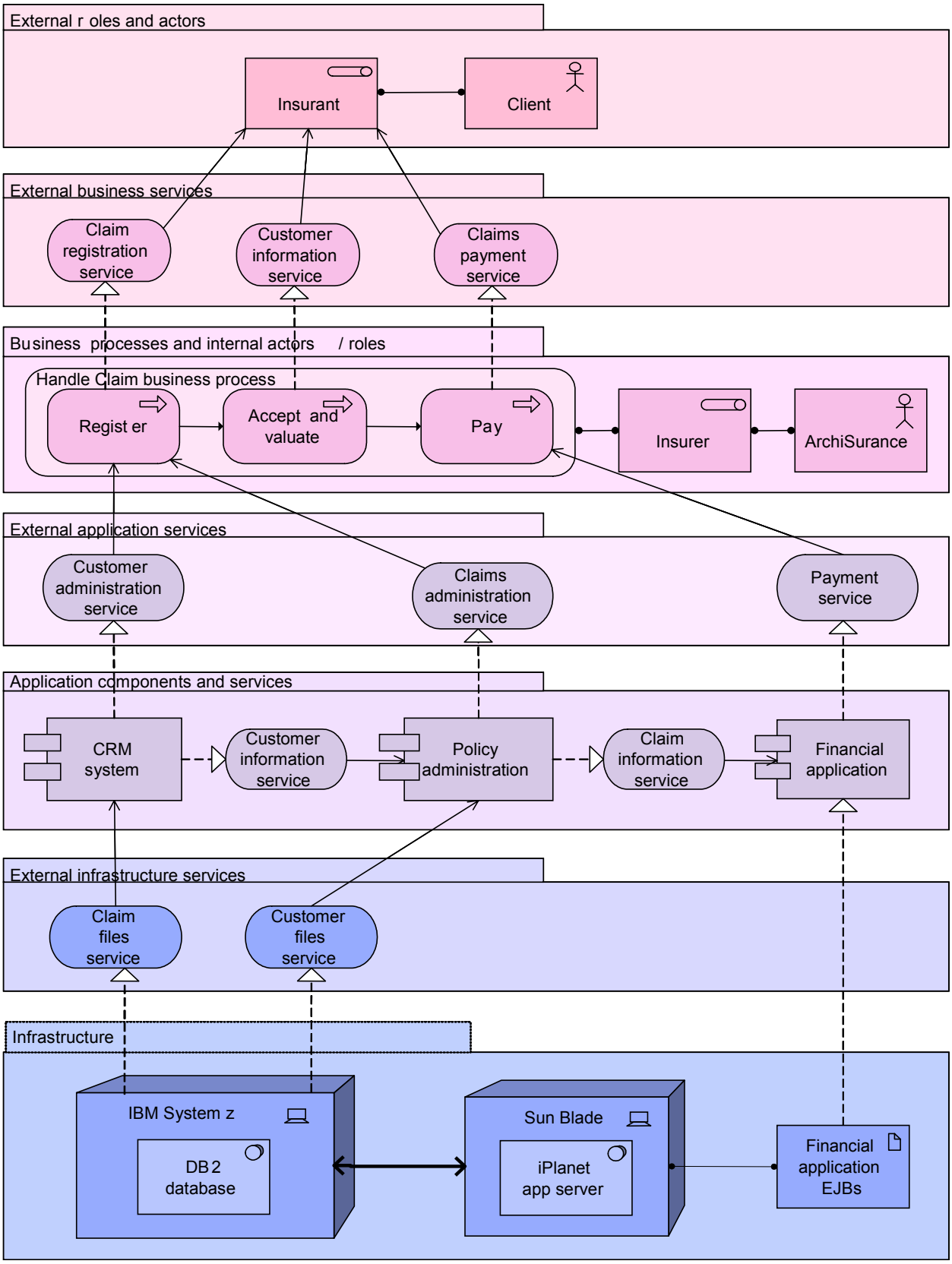

Figure 12. Example of an integrated enterprise architecture.

Basic design viewpoints in ArchiMate

The basic type of viewpoint in ArchiMate is a selection of a relevant subset of the ArchiMate concepts and the representation of that part of an architecture that is expressed in different diagrams. A set of such viewpoints was developed based on practical experience, and each of these is targeted to a specific design problem. An overview of these viewpoints is given in Table 1. For more information about the individual viewpoints, see [Doest et al.,, 2004].

Early design	*Cooperation*
Introductory	Actor cooperation Business process cooperation Application cooperation
Composition	*Realisation*
Organisation Business function Business process Information structure Application behaviour Application structure Infrastructure	Service realisation Implementation and deployment
	Support
	Product Application usage Infrastructure usage

Table 1. ArchiMate viewpoints.

ArchiMate and TOGAF ADM

The structure of the ArchiMate language neatly corresponds with the three main architectural domains of TOGAF. This is illustrated in the figure below. This correspondence would suggest a fairly easy mapping between TOGAF's views and the ArchiMate viewpoints.

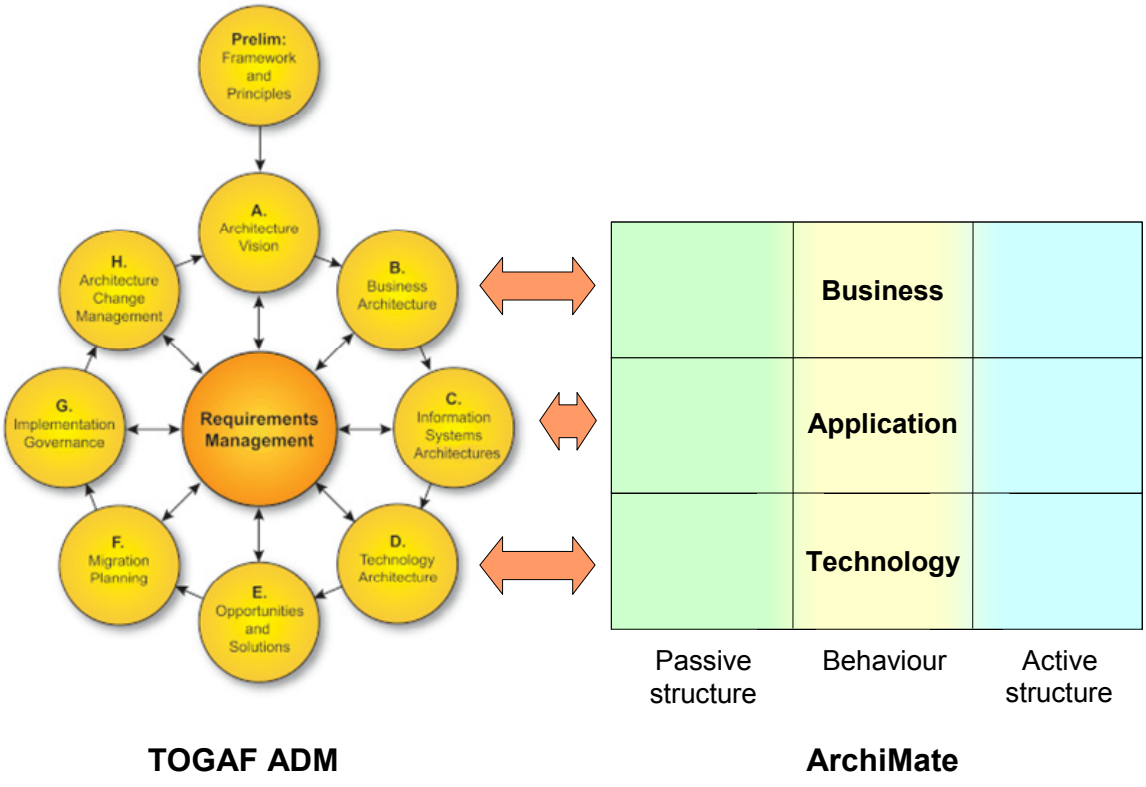

Figure 13. Correspondence between ArchiMate and TOGAF.

We have mapped the TOGAF views and ArchiMate viewpoinsts onto each other, to evaluate whether ArchiMate would be a suitable language for expressing TOGAF's views. This yielded Table 2.

TOGAF	*ArchiMate*
Business strategy	
Business objectives	
Business rules	
Business function view	Business function
	Product
Business service view	Service realisation
Business process view	Business process Business process cooperation
Business information view	Information structure
Business location view	Organisation
Business logistics view	
People view	Organisation, actor cooperation
Workflow view	Business process
Usability view	
Business events	Business process
Business performance	
Data entity view	Information structure
Data flow view	Business function
Logical data view	Information structure
Software engineering view	Application structure
	Application usage
Application interoperability view	Application cooperation
Software distribution view	Implementation & deployment
Network computing / hardware view	Infrastructure usage
Communications engineering view	Infrastructure
Processing view	Infrastructure
Cost view	
Standards view	
System engineering view	
Enterprise security view	
Enterprise manageability view	
Enterprise quality of service view	
Enterprise mobility view	

Table 2. Correspondence between TOGAF views and ArchiMate viewpoints.

Although there is no one-to-one mapping between ArchiMate viewpoints and TOGAF views[2], the table above shows a fair amount of correspondence. Although corresponding viewpoints from ArchiMate and TOGAF do not necessarily have identical coverage, we can see that many viewpoints from both methods address approximately the same issues.

Some viewpoints are not matched, however. Partially, this is because TOGAF's scope is broader and in particular addresses more of the high-level strategic issues and the lower-level engineering aspects of system development, whereas ArchiMate is limited to the enterprise architecture level of abstraction and refers to other techniques both for strategies, principles, and objectives, and for more detailed, implementation-oriented aspects. Secondly, although some of the TOGAF views cannot easily be mapped onto ArchiMate viewpoints, the ArchiMate language and its analysis techniques do support many of these. For example, performance or cost views could be created using ArchiMate's quantitative analysis techniques [Iacob & Jonkers, 2005].

Conclusions

From the previous sections, it is clear that TOGAF and ArchiMate can easily be used in conjunction and appear to cover much of the same ground, although with some differences in scope and approach. The most important disparity we observe between TOGAF and ArchiMate is that it appears that the ArchiMate viewpoints that deal with the relationships between architectural layers, such as the product and application usage viewpoints, are difficult to map onto TOGAF's structure, in which views are confined to a single architectural layer. This points to an important limitation of TOGAF, that of the integration (or lack thereof) between the different architectural domains. TOGAF itself provides no guidance on creating a consistent overall model of the architecture, but refers to tools that should provide this support [TOGAF, 2006]:

> "In order to achieve the goals of completeness and integrity in an architecture, architecture views are usually developed, visualized, communicated, and managed using a tool.
>
> In the current state of the market, different tools normally have to be used to develop and analyze different views of the architecture. It is highly desirable that an architecture description be encoded in a standard language, to enable a standard approach to the description of architecture semantics and their re-use among different tools."

This is where ArchiMate could nicely complement TOGAF: it provides a vendor-independent set of concepts that would help to create a consistent, integrated model "below the waterline", which can be depicted in the form of TOGAF's views. Thus, these two complementary open standards would reinforce each other and help to advance the enterprise architecture discipline in general.

Marc Lankhorst

Telematica Instituut

marc.lankhorst@telin.nl

Hans van Drunen

Atos Origin

hans.vandrunen@atosorigin.com

[2] Note that TOGAF's use of the term "view" differs from the IEEE Std 1471-2000 terminology, although this is acknowledged in the documentation: "Since in ANSI/IEEE Std 1471-2000 every view has an associated viewpoint that defines it, this taxonomy may also be regarded as a taxonomy of viewpoints by those organizations that have adopted ANSI/IEEE Std 1471-2000." (The Open Group, 2006].

References

[Bernus et al., 2003] P. Bernus, L. Nemes, G. Schmidt. *Handbook on Enterprise Architecture*, Springer, 2003.

[Doest et al., 2004] H. ter Doest, M.-E. Iacob, M.M. Lankhorst (Ed.) & D. van Leeuwen, Viewpoints Functionality and Examples. TI/RS/2003/091, Enschede: Telematica Instituut, 2004. https://doc.telin.nl/dscgi/ds.py/Get/File-35434/

[Eertink et al., 1999] H. Eertink, W..Janssen, P. Oude Luttighuis, W. Teeuw & C. Vissers. A Business Process Design Language, in *Proc. of the 1st World Congress on Formal Methods*, Toulouse, France, 1999.

[Frankel, 2003] D.S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley, 2003.

[Henderson & Venkatraman, 1993] J.C. Henderson, N. Venkatraman. Strategic Alignment: Leveraging Information Technology for Transforming Organizations, *IBM Systems Journal*, 32(1), 1993.

[Herzog, 2001] U. Herzog, Formal methods for performance evaluation, In *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science* (LNCS, 2090], pp. 1–37. Springer, Berlin, 2001.

[Iacob & Jonkers, 2005] M.-E. Iacob & H. Jonkers, Quantitative Analysis of Enterprise Architectures In: *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'2005)*, Geneva, Switzerland, 2005.

[IDEF, 1993] IDEF, *Integration Definition for Function Modeling (IDEF0) Draft*, Federal Information Processing Standards Publication FIPSPUB 183, Springfield, VA, USA: US Department of Commerce, 1993

[IEEE, 2000] IEEE Computer Society, IEEE Std 1471-2000: IEEE Recommended Practice for Architecture description of Software-Intensive Systems. IEEE, New York, 2000.

[ITU-T, 2002] ITU-T, *Information Technology – Open Distributed Processing – Reference Model – Enterprise Language*, ITU–T Recommendation X911 ISO/IEC 15414. International Telecommunication Union, 2002.

[Lankhorst et al., 2005] M.M. Lankhorst et al., *Enterprise Architecture at Work – Modelling, Communication, and Analysis*. Springer, 2005.

[Nadler et al., 1992] D.A. Nadler, M.S. Gerstein & R.B. Shaw. *Organizational Architecture: Designs for Changing Organizations*. San Francisco. Jossey-Bass Publishers, 1992.

[OMG, 2003] Object Management Group, *Unified Modeling Language: Superstructure. Version 2.0.* Final Adopted Specification ptc/03-08-02, Object Management Group, 2003.

[Scheer, 1994] A.-W. Scheer, *Business Process Engineering: Reference Models for Industrial Enterprises*, Springer, 1994.

[Sowa & Zachman, 1992] J.F. Sowa & J.A. Zachman. Extending and Formalizing the Framework for Information Systems Architecture, *IBM Systems Journal*, 31(3):590-616, 1992.

[The Open Group, 2006] The Open Group, *The Open Group Architectural Framework (TOGAF) Version 8.1.1 'Enterprise Edition'*. The Open Group, Reading, UK, 2006. http://www.opengroup.org/togaf/.

[Zachman, 1987] Zachman J.A. (1987), A Framework for Information Systems Architecture, IBM Systems Journal, 26(3):276–292, 1987.