Chapter 6
Applying Frameworks

Where are we at the moment with our journey through EAM? We started with an introduction to EA and EAM in Chap. 1. We were talking about its purpose and some basic tools and visualisations.

Then, in Chap. 2 *Understanding Business Architecture* we rather focused on those concepts that are required for describing the business with processes, business capabilities, and also business objects.

Further on, we used the business capabilities as a starting point for developing the application architecture in Chap. 3. It was a simple methodology for just making a decision. How should the ideal landscape look like? But we already had a look at data that is required for analysing applications and also the whole application landscape.

The analysis part was continued in Chap. 4 by looking at visualisations that are used for analysing and improving the EA, like the business support matrix. This tool is aiming at describing and understanding the relationship between software applications and business-related concepts.

Afterwards, we are not that much focusing on describing architecture but looking at how we can manage EA in a corporation. Most of it is about managing changes. This requires an adequate organisation for also sustaining the work of the enterprise architect. At the end of Chap. 5, we looked at criticisms to classical approaches for EAM. We presented some modern approaches, which are suggesting new organisational aspects or new organisational setups for managing EA.

This Chap. 6 will now introduce the notion of *frameworks* which is commonly used within the EA community. A framework basically provides methods and tools that are required for setting up and implementing EA in a large-scale organisation.

J. Jung, B. Fraunholz, *Masterclass Enterprise Architecture Management*,
https://doi.org/10.1007/978-3-030-78495-9_6

Learning Objectives

After completing this chapter, you will be able to . . .

- . . . summarise existing EA frameworks
- . . . explain elements of a framework
- . . . discuss TOGAF compared to concepts presented in this textbook
- . . . infer future EA projects

The learning objective for the current sections are as follows. After finishing the section, you should be capable of summarising a few existing EA frameworks. We will not discuss all of them but have a look at some of the popular ones in the subsequent Sect. 6.1. Section 6.3 provides an overview on TOGAF®, The Open Group Architecture Framework. It is an open standard, hence, documentation is widely available and TOGAF has been adopted in many organisations. After completing this section, you should be capable of explaining artefacts as well as methods and tools that are part of TOGAF.

Frameworks are based on best practices in the EA community and this also holds true for the textbook at hand. The book's purpose is not to publish a new framework but teach EA in a practical way. It also shares some commonalities with frameworks available today. These similarities are summarised in Sect. 6.4, showing how the contents of the book relates to existing frameworks. This will also serve as a summary of the previous sections. After completing this section, you should be capable of relating the contents to any framework—perhaps even develop a simple framework on your own.

6.1 Frameworks Overview

This section provides an overview on some frameworks and standards:

- *The Zachman Framework for Enterprise Architecture*™is supposed to be the origin for today's frameworks.
- The *Integrated Architecture Framework* provides more than two dimensions.
- *ArchiMate* is a modelling language specifically for EA.

The section will finish by providing a high-level overview on past and present frameworks.

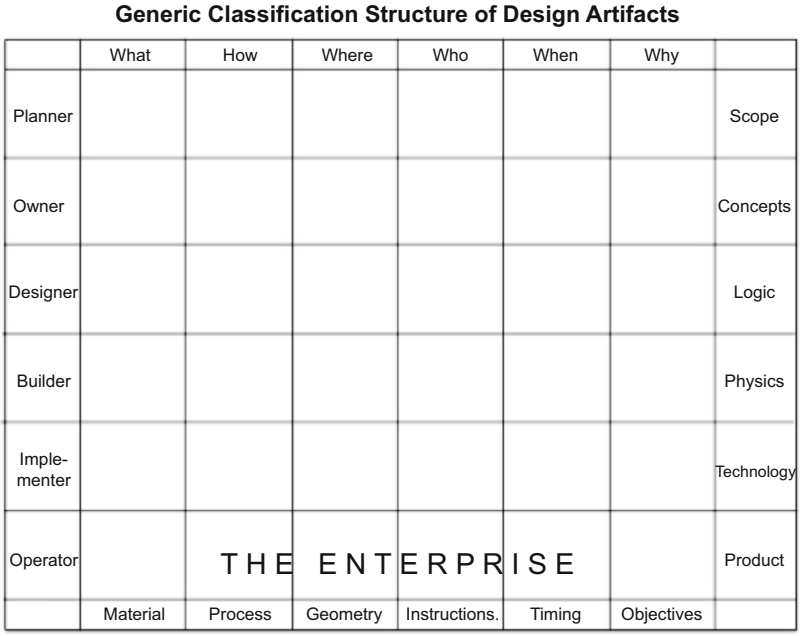

Fig. 6.1 *The Zachman Framework for Enterprise Architecture*™(Published with the permission of John A. Zachman and Zachman International®, Inc.—www.zachman.com)

6.1.1 Zachman Framework

When talking about frameworks, most people are starting with a very famous person called *John Zachman*. He is quite popular because he is supposed to be the first to develop an EA framework.[1] He first published his approach in the IBM Systems Journal in the 1980s. In this article, Zachman described his idea on a structure for providing various views on IT of a company[1]. *The Zachman Framework for Enterprise Architecture*™consists of several layers and views as shown in Fig. 6.1.

The top level of the framework suggests that we should understand the context of the company as well as its scope or markets (Fig. 6.1). We can then define something Zachman calls the enterprise model. It is a description of the company—including business processes and resources. A system model is then derived from the enterprise model. It is similar to what we introduced as application architecture in Sect. 3.1. It encompasses software applications and tools which are then further mapped to concepts in a technology model. Examples for those concepts networks, computer hardware or peripheral devices. The bottom layer refers to any details that are required for implementing information technology within the company.

[1]Zachman is still treated like a superstar when appearing at an EA conference.

This kind of model is not that different from what we already discussed so far. The business context and the enterprise model are quite similar to the business layer we already introduced in Chap. 2. In a similar way, the system model refers to the application layer and the technology model to the technology layer. The top four layers represent the view like the town planner, as it is introduced in Sect. 1.1.1.

Zachman suggests to focus on the top layers, providing the high level view on to the organisation. These layers are then further decomposed into a static view containing *data objects* on various levels of abstraction. You can have business objects in the scope layer (like market, competition, customer service). In the enterprise model, you can derive a data model for specific applications from the business objects. They can then be implemented in a specific technology (like a relational database).

A similar principle holds true for *functions*. We can start on a high level, similar to business capabilities on level 1. Functions on that level address how we want to establish our company on the market. They can then be decomposed into specific function (i.e. business capabilities) that are supposed to be implemented by the company. Functions are then implemented by functionality in software applications. Any application requires a certain technology. The implementation itself is then subject to the lowest level, which defines any details required for software development.

The same principle of various levels of abstraction is applied to each view of the Zachman framework. It starts with using high-level concepts the business context which will then be further specified in lower levels. Figure 6.1 shows examples for each remaining view: network, people, time motivation. The people view might start with strategic human resources which can then be further described by human resources, staff members, application users and roles, and then having detailed work description for each actor.

Even though Fig. 6.1 is showing some example concepts for filling in the framework, there is no extensive documentation of required modeling languages (cf. Fig. 6.2). Any of his publications, like the paper published in the IBM system journal

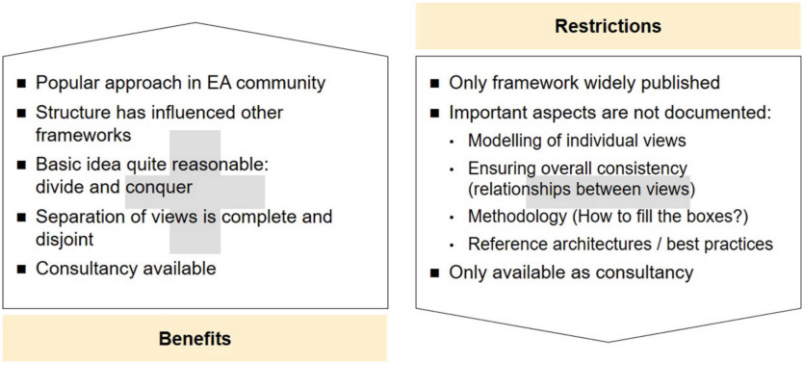

Fig. 6.2 Critical review of the Zachman framework

are basically describing the framework and its basic idea. Zachman is also owning a consultancy firm offering services for applying the framework in a client's company. You can hire a skilled consultant providing a specific solution for a company based on the framework. Nevertheless, Zachman is just the beginning. There are more frameworks available, including TOGAF. They can provide more specific concepts and tools relate to the cells defined by the Zachman framework.

6.1.2 *Integrated Architecture Framework (IAF)*

The Integrated Architecture Framework (short IAF) has been developed and published by Capgemini [2]. The IAF has its origins in preliminary work for defining a framework that can also support structuring large IT consultancy projects. An overview on the IAF is given in Fig. 6.3. It covers relevant views on an IT project together with its context (i.e. the motivation for the IT project). However, its shows several similarities with an EA framework.

It might not surprising that the IAF contains a building block labeled *Business*. It, in fact, refers to the business view on the IT project and uses similar concepts as the ones introduced with the business architecture in Chap. 2. It, furthermore, consists of building blocks for information systems (analog to the application layer) as well as a technology. The latter one covering technology components and networks.

The framework is complemented by a building block labeled *information*. It relates to any kind of data used within the company and managed by information systems. This is also nothing completely new for us. We already introduced the notion of *data architecture* in Sect. 3.4. In contrast to the IAF, we put it as an overarching layer across any of the other three layers representing business, applications and technology.

One of the obvious differences is that the IAF spans three dimensions. *Governance* and *security* represent relevant functions for managing compliance and risk

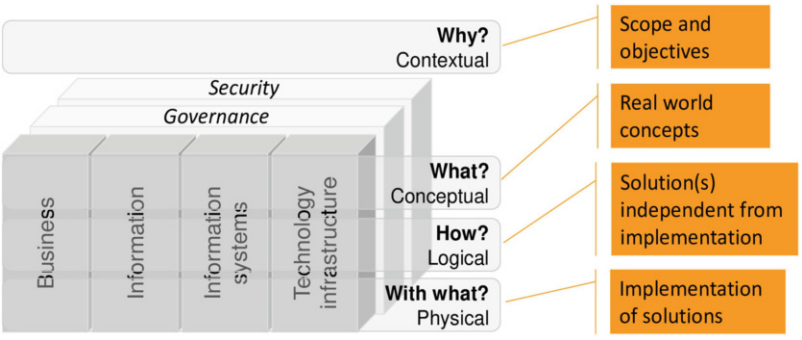

Fig. 6.3 Integrated architecture framework (based on a figure in [3, p. 67])

in an organisation. They are stretched across the previously discussed layers by providing controlling mechanisms. Governance ensures that an organisation follows existing laws, regulations and compliance rules for business and also IT. Security addresses any measure for protecting information and technology against theft and misuse.

The IAF is not the only framework adding dedicated layers for governance or security: *PEAF* (Pragmatic Enterprise Architecture Framework) includes a governance product [4], *FEAF* (Federal Enterprise Architecture Framework) a security reference model and security controls [5], *GERAM* (Generalised Enterprise Reference Architecture and Methodology) a governance reference model [6] or *E2AF* (Extended Enterprise Architecture Framework) includes both, governance and security viewpoints [7]. The last one provides it as cross-cutting views—similar to IAF. It is important having governance and security as overarching views instead of separate layers as they only work is corresponding processes and measures cannot be treated separately from business and application layer concepts.

Using the third dimension, IAF differentiates between three levels of abstraction together with the context (cf. Fig. 6.3). The top one (*Why?*) covers the context of an organisation or a project. It defines the scope and objectives, similar to the business motivation presented in Sect. 2.4. It does not specifically address architecture or solutions but rather the drivers for for the organisation.[2] It is further used for shaping the initiative by defining concrete boundaries (i.e. scoping) in order to have a focus for subsequent activities.

Each of the four contents layers—business, information, information systems and technology infrastructure—can be regarded on three levels of detail. The conceptual level (*What?*) covers concepts for documenting the application domain on a high level of abstraction. It rather refers to the perspective of the town planner as introduced in Sect. 1.1.1. It also does not comprise solutions but real world concepts that characterise the company or the project provided within the scope defined in the contextual layer. These concepts can be represent the business (i.e. business service), information (information object), information systems (information system service) and technology infrastructure (infrastructure service). Basically, this is the starting point for discussing corporate IT architecture with business stakeholders. We would also regard core concepts covered by this textbook (business capabilities, business objects and applications) on the conceptual layer.

The logical level (*How?*) serves as a blueprint for potential solutions implemented in the organisation. It is still independent from specific solutions—which can be found further down the line on the physical layer. Consequently, it does not contain deployable software solutions but logical components that represent the desired functionality. It also contains business processes, logical data objects (including information ownership) or types of technology infrastructure components. This kind of concepts can play the role of requirements that drive the implementation of information systems.

[2]Why are we doing business on the market? Why do we want to change our business?

Specific solutions and resources are then located on the physical level (*With what?*). We will match requirements with specific software products, databases and technology components. The logical level should be independent from those solutions so that we can evolve the organisation over time, without changing the conceptual or logical level. A similar principle is applied in database design. The conceptual data model captures real world concepts. They are then further detailed by a logical data model that can then be implemented by a specific database management system (DBMS). In case of changing the DBMS we can still use the same logical data model and implement it with a different DBMS product. the same leveling can be applied to business-related concepts (business capability → business process → automated workflow) or information systems (software service → software component → software implementation).

We did not discuss these kinds of levels in the course of our textbook. We rather provide a small amount of relevant concepts which can be the starting point for a more detailed architecture. However, real-world architectures will cover these or similar levels as the the view of the town planner needs to manifest in a live city. For further information on the IAF, we would like to refer to the references provided, especially [2].

6.1.3 ArchiMate

Technically, ArchiMate is not a framework, but a modeling language for EA. The name *ArchiMate* is short for *Architecture-Animate* and has its origins in a project funded by the Dutch government and executed by several Dutch research institutions [8]. It has been handed over to The Open Group which is maintaining it as a standard complementing TOGAF [9]. ArchiMate is a modeling language like the UML or BPMN and, thus, allows for describing specific architectures. It, therefore, provides typical elements like for example business functions, processes or software applications together with their relationships.

ArchiMate is using a similar layered approach as introduced in this text book. It distinguishes between business layer, the application layer, and technology layer (cf. Fig. 6.4). Each of them is further divided into views for the passive structure, behaviour, and an active structure. Can you imagine the concepts for each of them? Please, take some time and reflect what we have discussed so far with business and application architecture.

As you might have assumed, the passive structure contains static concepts like data or business objects. In contrast to this, the active structure encompasses concepts that can perform activities like for example software applications. The behavior part describes behavior that can be performed by elements from the active structure. Wierda describes the interplay of the by referring to the structure of a simple English sentence: subject, verb and object [10, p. 18]. An application (element from the active structure) performs an an operation (behavior) on a data object (passive structure).

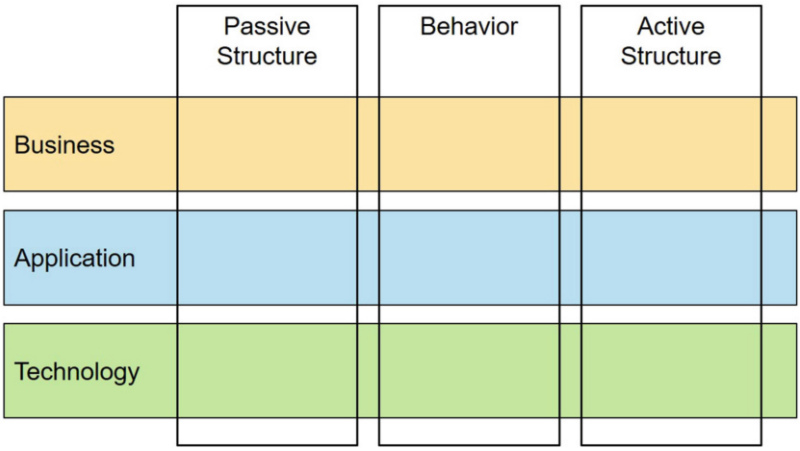

Fig. 6.4 ArchiMate layers and views (based on a figure in [3, p. 69])

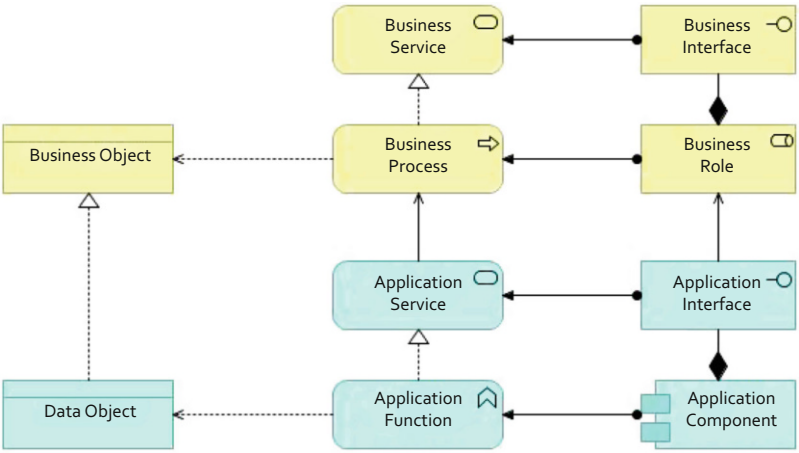

Fig. 6.5 ArchiMate language concepts (based on [10, p. 20])

The ArchiMate language structure follows a similar principle we already used for data architecture in Sect. 3.4. Elements of the static structure do not only relate to similar thing, but also provide views on different levels of detail. We can see this in Fig. 6.5 showing an example ArchiMate model consisting of business and application layer concepts. It contains the business object on the left hand side, which is *realised*[3] by a data object on the application layer (analog to Sect. 3.4). This data object might then be realised (i.e. implemented) by an artifact in a file or a database on the technology layer (not shown here).

[3]The meaning of the arrow (i.e. its semantics) between data object and business object is *realises*.

The same principle is also applied in the behaviour view and the activity structure using special relationships. A *business service* is is a service offering by an organisational unit. It is realised (i.e. performed) by a business process. A business process is *served* by an application service (service offering by an application) from the application layer which is, in turns, realised by an application function (i.e. functionality to be implemented in a software application). Please, note that application service and function do not refer to the actual application, but to the service or functionality that is expected to be provided by the application. They are part of the activity structure.

Some elements of the activity structure are shown on the right hand side of Fig. 6.5. They cover actors (i.e. business role) and software applications (application component). An application component realises an interface (i.e. a user interface) which can then be used by actors in a business process. There are no elements shown from the technology layer as we are focusing on the business and application layers within the text book at hand. However, you can imagine application components running on computers which are called *node* in ArchiMate.

There are also well defined relationships between the model elements. Realisation and service relationships have already been explained above. There are also relationships between the three views which allow for assigning active elements to behaviour and static objects. Actors are *assigned* to business processes and application components to application functions. In fact, an application function might be implemented by different applications. Behavioural elements *access* static elements shown be the arrows between business process and business object as well as application function and data object.

As already stated in the beginning, ArchiMate is a modelling language defined by syntax[4] and semantics.[5] It allows for a more detailed view on EA than the concepts provided in previous sections. It allows allows for combining EA models with other modeling languages, like BPMN[6] or the UML.[7] It also complements existing EA frameworks and has, thus, been adopted as an additional standard by the Open Group (see Sect. 6.3) [9].

[4]The syntax defines language elements and rules how to connect them properly (i.e. realisation relationship is allowed between a business object and a data object but not between two business objects).

[5]The meaning of elements of a modelling language is defined by its semantics.

[6]BPMN (Business Process Modelling and Notation) is a standardised modeling language for business processes.

[7]The UML (Unified Modelling Language) is defined for specifying software applications.

6.1.4 Historical Overview

It is quite impossible to introduce all EA frameworks as there are so many of them. The text book at hand rather aims at providing a practical introduction to relevant concepts for starting your EA journey. We only introduce some frameworks for highlighting how these concepts are applied in existing frameworks and also for showing the historical evolution of EA. Zachman has been chosen as it is often referred to as the very first framework providing an elaborate structure for managing large and complex information systems in a corporation. The IAF is one example for approaches having more than two dimensions and also incorporating typical IT responsibilities (i.e. governance and security). Many frameworks just cover the high-level perspective by providing a structure but do not go into detail. ArchiMate is one of the approaches that extends EA by providing a language for a more detailed description of an organisation ad its information systems.

An historical overview on prominent frameworks is shown in Fig. 6.6—and to be honest, it is only a small fraction of frameworks available. We find the Zachman framework starting in the '80s with the paper published in the IBM Systems Journal, then evolving into a more elaborate framework and still being used today by Zachman. Another representative of the early frameworks is the CIMOSA reference model. It provided a standard for describing information systems. The first version has been published in the 1980s and the latest version in the 1990s.

PERA (Purdue Enterprise Reference Architecture) has an academic background. It started in the 1980s and being available at its current version from 2001. DoDAF (Department of Defense Architecture Framework) is an example for frameworks being defined by an organisation for its own use. In fact, several large companies started to develop their own frameworks. the picture in Fig. 6.6 also contains some research initiatives. sebis (bottom right) is the Software Engineering and Business information Systems research group at the Technical University München, Germany. Members of this team do extensive research and provide best practices for EA including concepts for individual frameworks.

6.2 EA Frameworks

We started introducing the notion of an EA framework by just referring to some examples, providing an overview to existing frameworks. Let's try to cover it with some small definition.

Definition 6.1 (Enterprise Architecture Framework) An **Enterprise Architecture Framework** (or just framework) provides principles and tools for establishing and sustaining EAM in an organisation. It usually provides

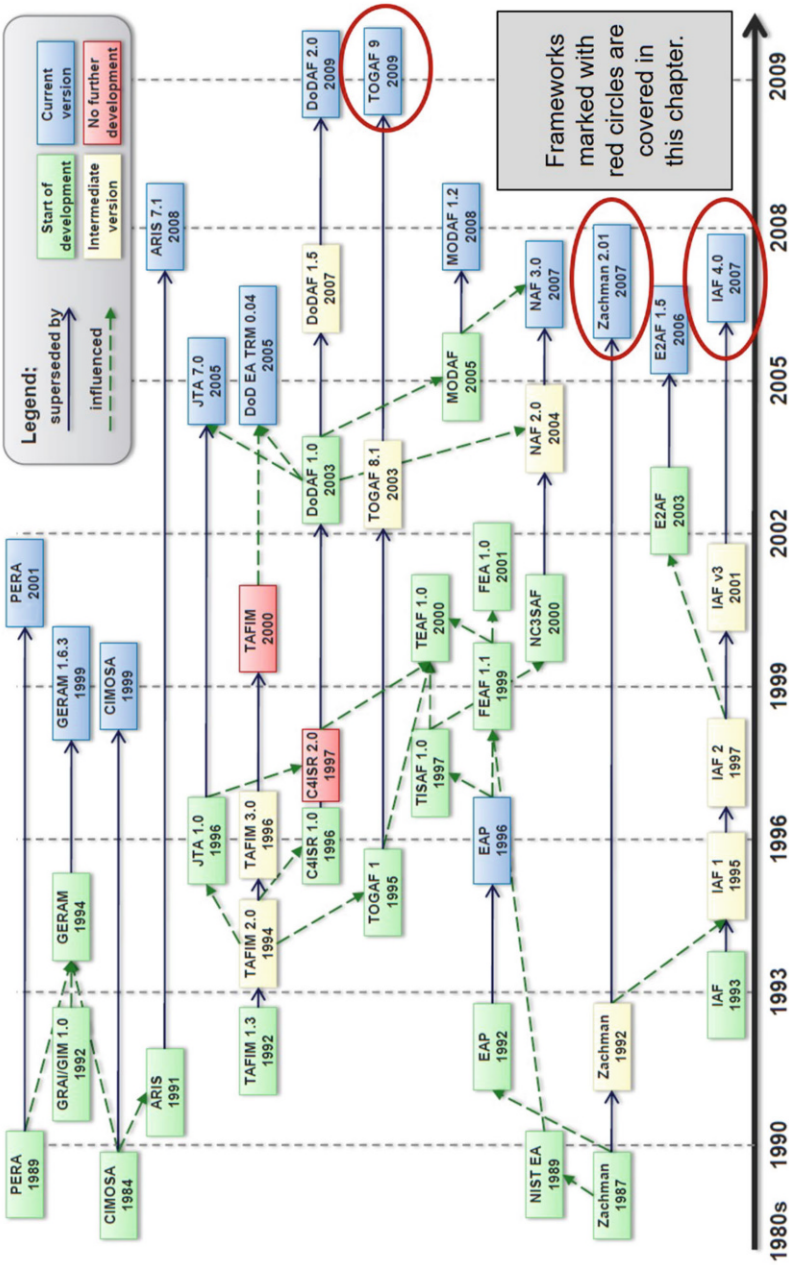

Fig. 6.6 EA frameworks overview (based on a figure in [11, p. 130])

- structure
- concepts (defined by a meta-model)
- visualisations
- methodology

The purpose of an EA framework is in providing some kind of toolbox—a toolbox full of principles, tools, methods—that we can use for implementing and maintaining EA in a company. Typically, an EA framework starts with defining an underlying structure, so the structure like the three layers we had in the beginning—business, application, technology layer—or some structure like the IAF we presented, or like the Zachman framework. And against this structure, the frameworks are usually providing concepts and also maps for describing EA.

Some of them address it in a very detailed way, others, like Zachman, rather on a very high level or even just by providing hints on how you could describe it. And they also usually define some kind of methods or some kind of procedures you need to implement for EAM, also some hints on skills required for enterprise architects, and a corresponding organisation in the organisation—in the company. This is the basic idea of a framework. And if we look at this one, with our lecture, we already started defining some kind of EA framework, so a framework which is not intended to be used outside of the university, but which has been developed for teaching EAM. It does not have a name. It will not be published. But we can see in the lecture, already, basic concepts that are required for an EA framework (Fig. 6.7).

When looking at existing frameworks, we will see they have even much more concepts to offer for EA work. Some of the aspects were already covered, like they provide some method or some kind of procedure that shows us, OK, if you want to implement EA in a company, then you should do it the following way. You should do the following steps in this order. Sometimes you need to have feedback and then start with some activities again. But they will provide some process that helps us with implementing EA and also making sure that we are maintaining EA over time.

They also provide, on a detailed or on an abstract level, information about the organisation that is required for executing the implementation and maintenance of EA. Beside of this, some frameworks also have a catalogue containing best practices, so best practices describing information or experiences made in previous

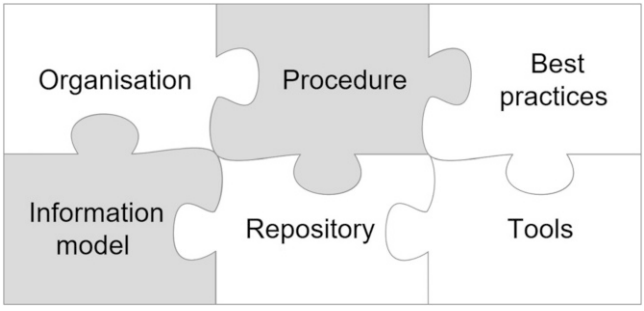

Fig. 6.7 Components of an EA framework

projects, that help people with making decisions within their own project, for setting up EA in their own company. The process and organisation in the best practise are usually accompanied by something they call an information model, which is often defined by something called a meta-model. The meta-model defines all concepts that are required for describing an EA. In our case, it would be defining that we need to collect information about business capabilities, applications, business objects, data objects. Any of the concepts we introduced would be subject to this kind of information model.

Beside of this, frameworks also offer some kind of tools. *Tool* can be a very broad term, here. It can refer to some document, some Excel spreadsheet, but even to a software tool that helps us with performing your EA work. And within this tool, very often you have a big repository with all the data. The data structures in this repository is defined by the meta-model. Remember that we already talked about the repository. In the beginning and also when talking about analysing EA, we had this working mode that bases on the fact we have a repository full of information for EA. It contains all our business capabilities and applications, which will then be used for generating views and maps for stakeholders. This is the same kind of repository in each and every EA framework.

6.3 TOGAF

The Open Group Architecture Framework (*TOGAF*®) is an open standard defined and published by *The Open Group*. The Open Group is a consortium of organisations including "customers, systems and solutions suppliers, tool vendors, integrators, academics, and consultants across multiple industries" (cf. [12]). They aim at defining and establishing various technology standards—including EA.

The TOGAF standard originated with its first version in 1995. It was based on an existing technical standard called *Technical Architecture Framework for Information Management (TAFIM)* (cf. [13][8]). The latest version 9.2 is available since 2018 and documented in [14]. The standard is also available for free as HTML after registration. Being a framework, TOGAF is providing a structure and concepts for describing EA in an organisation (cf. page 194). This supports structuring the architecture but also organising work around managing EA. Concepts establish a common terminology among people involved in EA. One of TOGAF's strengths is the method for describing and managing EA: Architecture Development Method (ADM). The ADM will be further described below (page 196). There are further aspects in TOGAF which are not covered by this textbook. The interested reader is referred to its specification in [14].

TOGAF is generic and is supposed to be applied in any kind of industry or organisation. Hence, we should have a clear understanding about its limitations how to apply it. Especially, TOGAF **is no**...

[8]This title is available in German only.

- . . . *solution*: Introducing TOGAF in an organisation is not a solution. You need to understand your problem (i.e. motivation for introducing EAM) first. The adoption of TOGAF will not automatically solve all your problems.
- . . . *cookbook*: reading the specification will help you understanding the concepts. However, it is not a step-by-step instruction for implementing EA in your organisation. People applying TOGAF still need a lot of experience.
- . . . *toolbox*: TOGAF might look like a tool set for EA. However, it is rather a common framework for a toolbox. It needs to be complemented by software tools and specific methods (e.g. business process modelling).
- . . . *organisational blueprint*: TOGAF does not specify how to establish an EA unit in your organisation.

In general, TOGAF is not *ready to use*. Adopting TOGAF still requires tailoring to an organisation's objective, organisation and processes.

6.3.1 Structure and Concepts

An overview on TOGAF's structure and concepts is shown in Fig. 6.8. It reflects similar layers as introduced starting from page 27 in Sect. 1.4. The middle part shows from left to right:

Business Architecture

The *Business Architecture* contains concepts for describing business-related aspects of an organisation. *Motivation* covers business drivers, objectives (goals) and measures (i.e. key performance indicators). These concepts correspond to those introduced as part of the *business motivation* in Sect. 2.4.

The *Organisation* part addresses the description of an organisation's structure consisting of organisational units, actors and roles. These (and some more) are covered by the *Business Execution* part in Sect. 2.4 in the textbook at hand.[9]

Behavior comprises any concept describing behavioural aspects of an organisation. *Business Services*, *Contracts* and *Service Qualities* specify the interaction with external partners via business services. Details on the usage of services are established via contracts and qualities attributes. *Processes*, *Events* and *Controls* are used for describing business processes anf the flow of control. They refer to classical business process models including business rules. Figure 6.8 lists some high-level behavioural concepts right beside the processes—including *Business Capabilities* as introduced in Sect. 2.2. Please note, that there is no concept corresponding to the notion of *business object* (cf. Sect. 2.3).

[9]The naming is slightly different but the purpose of the concepts is quite similar.

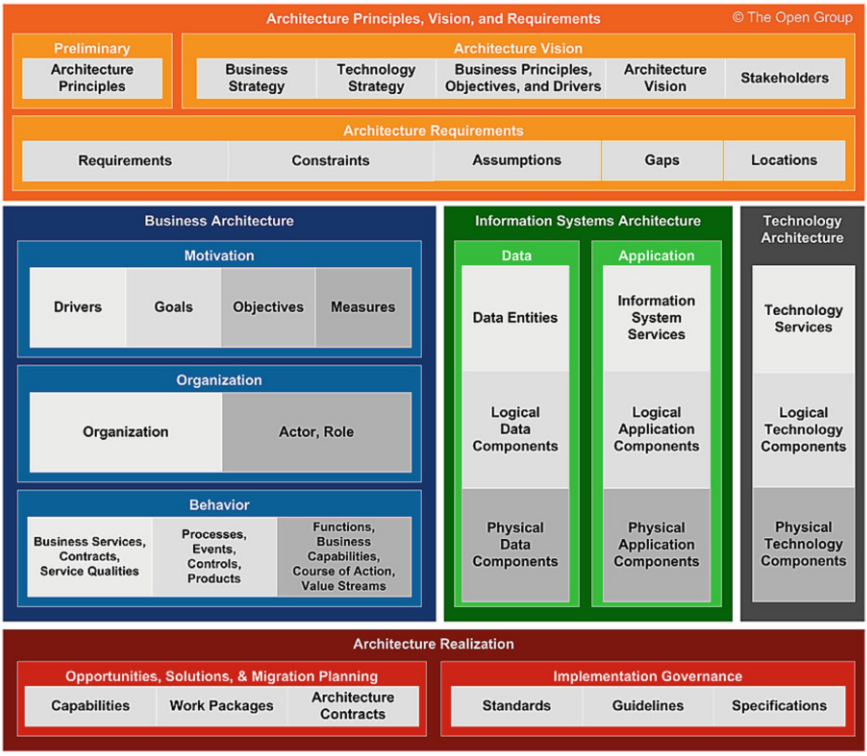

Fig. 6.8 Components of TOGAF (©The Open Group)

Information Systems Architecture

The *Information Systems Architecture* covers typical aspects of corporate information systems divided into *Application* and *Data*. It looks slightly different from ours as explained in Sect. 1.4 as the *Application Architecture* is only a part of it. The main difference is the data architecture which we introduced across the other layers, having concepts on business, application and technology level. TOGAF only locates it in the information systems part.[10]

The data architecture includes similar abstractions as provided by classical data modelling:

1. *conceptual data model* defined by entities (data entities) from the application domain and their relationships

[10]This statement is rather neutral as each approach has its positive and negative aspects. At this point, it is just mentioned as a difference.

2. *logical data model* representing data (logical data components) for a database paradigm (e.g. relational, object-oriented or XML)
3. *physical data model* being implementation-specific with respect to a dedicated database management system

Technology Architecture

The *Technology Architecture* is not discussed in detail within the textbook at hand. It encompasses any kind of technology that is required for running information system in a corporate environment. This includes computer hardware, system software and computer networking technology. Previous versions of TOGAF provided a technical reference model defining typical technology components.[11] It is now available as a separate publication [16].

Further Views

Beside the three architectures for describing EA, TOGAF also provides two views for managing EA. *Architecture Principles, Vision, and Requirements* cover any aspect for guiding EA work. *Architecture Principles* define criteria for the architecture of a given organisation. The *Architecture Vision* puts architecture work into the organisation context. Corporate strategies and objectives need to drive how EA is done in a company. *Architecture Requirements* can be derived from the as-is or reflect assumptions/constraints on the future state.

Managing EA consists of planning, coordinating and executing many change initiatives. This is covered by *Architecture Realisation* in Fig. 6.8.

6.3.2 Method for Applying Enterprise Architecture

One of the essential concepts is an elaborate method that explains basic activities in order to manage EA. The TOGAF Architecture Development Method (ADM) is a process model based on best practices and fits to the meta-model presented in the previous Sect. 6.3.1. Figure 6.9 provides an overview on the ADM by showing major phases and the control flow.

The *Preliminary* phase deals with defining the need for EAM in an organisation and setting up the corresponding capability. The capability consists of an organisational model (e.g. dedicated EA team), tools (including an EA software tool) and guidelines for architecture work. Afterwards, the *Architecture Vision* can be established (see also Sect. 5.1.3 concerning the role of the vision in managed

[11] cf. Section 43 in [15].

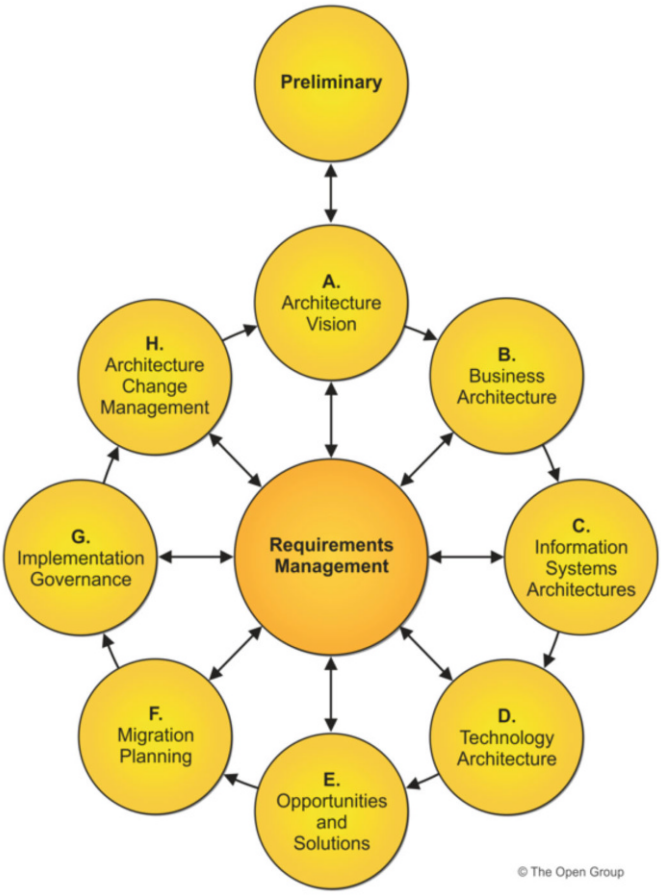

Fig. 6.9 TOGAF architecture development method (©The Open Group)

evolution). The subsequent phases *Business Architecture*, *Information Systems Architecture* and *Technology Architecture* then basically deal with documenting the three architectures as introduced in Sect. 6.3.1.

Phase E, *Opportunities and Solutions*, aims at identifying optimisation potential in the corporation. This can be done by tools like the analysis presented in Chap. 4 within this textbook. Architects need to identify opportunities (i. potential improvements) and define corresponding solutions (concrete improvements). Those solutions need to be implemented within initiatives that are planned in phase F *Migration Planning*. Tools like a roadmap (cf. Sect. 5.1.4) can be used here. The remaining phases *Implementation Governance* and *Architecture Change Management* will then enable steering the initiatives and guide their execution by a holistic change management (cf. Sect. 5.1).

The focal phase *Requirements Management* is needed for managing any architecture requirement. These are relevant throughout the other phases and even impact several initiatives. Furthermore, new architecture requirements may occur in any phase and even trigger a new cycle in ADM. The ADM is not meant to be executed once but rather iteratively. This allows for starting with small improvements and later on extend the use of EAM in the company. Especially if an organisation is new to EAM or resistance is to be expected, then EAM should not be implemented in the whole company in the beginning.

6.3.3 Complementary Standards by The Open Group

We only covered a fraction of the TOGAF standard so far. It also offers additional guidelines and techniques, typical visualisations (cf. Sect. 4.2) as well as a structure for the EA repository.[12] But still, TOGAF is generic as it addresses any kind of organisation and needs to be complemented by additional methods and tools. This includes standard methods for business process modelling (or management), change management or project portfolio management.

TOGAF does not include a language for modelling EA. This is provided by an additional standard, ArchiMate (cf. [9]). ArchiMate is a modelling language for describing architectures across all architectural layers. It implements most of the concepts presented in the meta-model of TOGAF (cf. Sect. 6.3.1). A preliminary standard is available focussing on business architecture (cf. [17]). The Open Group is also working on a standard for agile EAM (cf. citeTheOpenGroup.2019a). This standard aims at supporting the *Digital Transformation* by a more agile approach (compared to TOGAF).

6.4 The Framework Provided by This Book

Even though this textbook does not aim at defining a new framework, it covers concepts, tools and methods that are also common to an EA framework. This section reflects individual topics and relates them to elements of a framework. At the same time, it will also provide a summary of the textbook from a slightly different angle.

[12]The Open Group uses the term *Enterprise Continuum* for referring to any kind of documentation and knowledge that is relevant for managing EA. It covers architecture artefacts and solutions from any source. Sources include TOGAF but also industry standards or organisation-specific information.

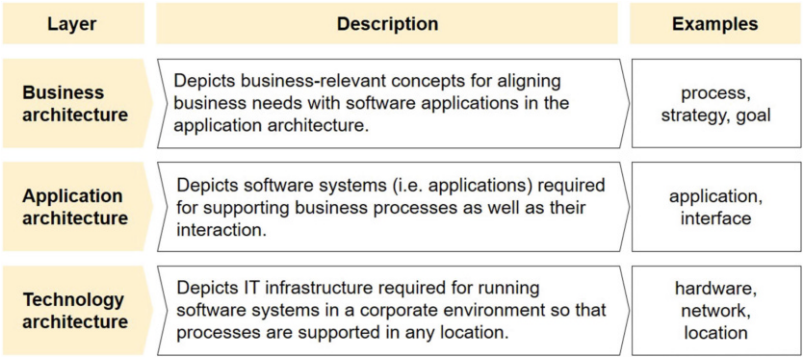

Fig. 6.10 Enterprise architecture layers

6.4.1 Structure

Each EA framework provides a basic structure. Such a structure usually defines certain layers or specific architectures. We introduced a very simple structure in Sect. 1.4.4. It divides EA into a business focused perspective (*Business Architecture*, the application landscape (*Application Architecture*) and the underlying technology (*Technology Architecture*) as depicted in Fig. 6.10. Each of them contains descriptive elements for each perspective and there can be relationships across the architectural layers.

EA frameworks usually come with their own layering scheme. IAF and TOGAF differ from the structure provided here (cf. Sects. 6.1 and 6.3). The structure of this book is the same as the one in ArchiMate (page 187). An overview on different frameworks and their layered structures is provided in [18].

6.4.2 Meta-model

Frameworks provide concepts for describing EA. These concepts are defined by an *information model*. This model is sometimes also referred to as *meta-model* as it emphasizes the fact that an EA is a model of the organisation. ArchiMate is a dedicated modelling language, therefore, requiring a meta-model for defining language concepts.

The meta-model describing the concepts used in this textbook is provided in in Fig. 6.11. I contains business architecture concepts introduced in Chap. 2 and application architecture concepts discussed in Chaps. 3 and 4. The technical architecture is indicated as a background but still empty as we did not cover it in this textbook.

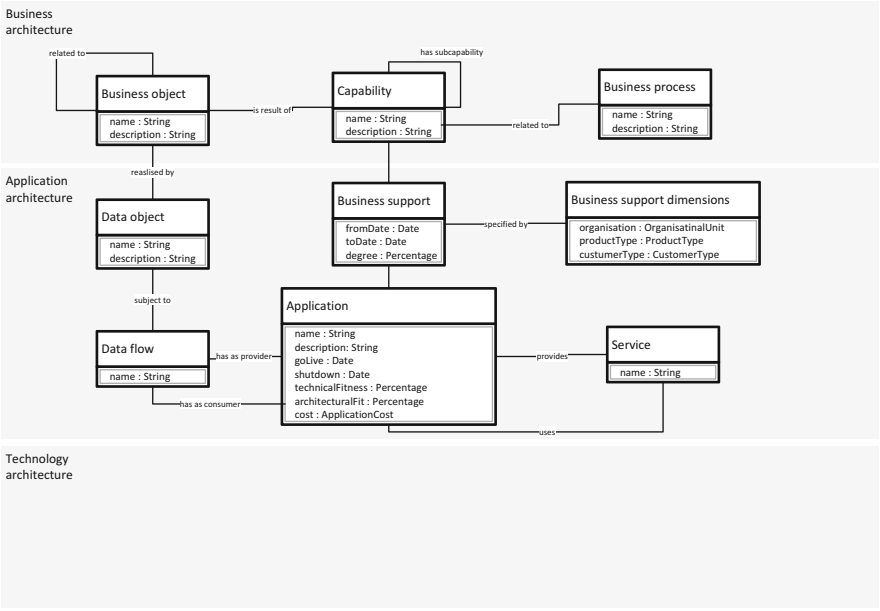

Fig. 6.11 Meta-model

The business architecture has business capabilities in its centre, because it is one of the focal elements in this book. A capability has a name and can be further described by text. It can be decomposed into sub-capabilities (relationship *has subcapability* in Fig. 6.11) and might have business objects as a result. Business objects can have relationships to each other. Business processes can provide further details on how capabilities are implemented in an organisation. Most of the business architecture concepts in Sect. 2.4 (e.g. *Strategy*, *Driver*, *Operating Model* or *Constraint*) are not listed in the meta-model. They are relevant for describing a business architecture but not discussed here in very detail. A more elaborate meta-model should include them.

Application architecture has a focus on software applications, their relationships (*Data flow* and *Service*) and their connection to the business architecture (*Business support*). Data flows relate to data objects being exchanged. A data object relates to business objects form the business architecture (cf. Sect. 3.4). A (software) service is provided by one application and can be used by others (pp. 90). Analysing and improving an application landscape require additional information (pp. 93). Such information can consist of an applications lifetime (provided by go-live date and sut-down) as well as quality attributes explained in Sect. 3.3.

Applications are linked to business capabilities by a concept called *business support* as explained in Sect. 4.3. Business support can have a temporal validity if an existing application is only used for a limited amount of time for a certain business capability. An Enterprise Architect can also estimate for a certain degree, if

the capability is only partially supported. Further dimensions allow a more detailed view on application support for different product, organisational units or customer segments (pp. 127).

The meta-model used in this textbook is kept very small as the book aims at providing a practical journey through EA. more detailed view is given by, for example, ArchiMate's meta-model (cf. [8, 9]) or dedicated software tools for EA. An evaluation of existing tools is given in [19, 20].

6.4.3 Procedure

The textbook at hand is not explicitly defining an EA method like for example TOGAF's ADM (cf. pp. 196). However, the structure of the book indicates already a sequence of activities (Fig. 6.12).

1. There needs to be a clear understanding on why EA is introduced in an organisation. This is why the book started with a discussion on the purpose of EA in Chap. 1. This also relates to the *Preliminary* and *Architecture Vision* phases in TOGAF.
2. Describing EA start s with the business architecture as business drives IT. There are many concepts available, but is usually a good idea to only start with a minimal set (e.g. business capabilities and objects).
3. An optimal application landscape can be derived from the business capabilities. There are usually some factors that will drive detailed decisions for individual capabilities. A dedicated method for deriving an application landscape is presented in Sect. 3.2. Hence, it is recommended to describe the application architecture after having a solid view on business architecture. The same order of phases is specified in TOGAF and other frameworks.
4. EA analysis requires the availability of relevant information–expressed in business and application architecture (Chap. 4). This section correlates with the *Opportunities and Solutions* phase in TOGAF.
5. All changes implied by EA work need to be managed as described in Chap. 5. We discussed roadmaps and application portfolio management as related tools. TOGAF covers this aspect by phases *Migration Planning* and *Architecture Change Management*

However, the table of contents of this text books is not a method. Chapter 6 is out of sequence, as it summarises the whole topic by presenting frameworks. Also Chap. 5 does not represent the fifth step, as we need to consider organisational aspects already in the beginning of an EA initiative. We also skipped the Technology Architecture which needs to be incorporated for more elaborate analysis.

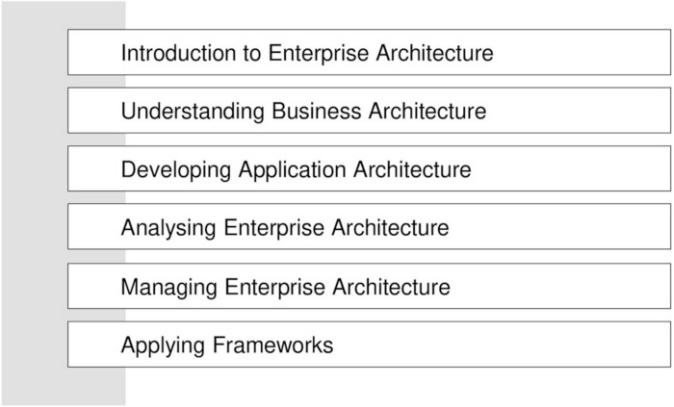

Fig. 6.12 Structure of this textbook

6.4.4 Organisation

Setting up an organisational unit for EAM is covered by Chap. 5. The chapter is not a detailed explanation on EA organisation, but provides an overview on relevant aspects. We covered

- Performing change management
- Typical activities of an Enterprise Architect
- How to set up an EA organisation
- Approaches for a collaborative EA

6.4.5 Tools

The term *tool* can be interpreted n a broad sense. It might be a software tool, a specific methodology or any artefact used in EAM. Using a software tool for maintaining EA information is crucial for any EA initiative. However, the purpose of the book is to teach the concepts. Software tools usually implement these concepts. This is why we did not use any software tool throughout the whole text book. Only Excel was mentioned as a pragmatic tool for generating and analysing the business support matrix (page 4.3.1). The interested reader can find a tool survey in [19, 20].

Beside of this, the following tools were used in the textbook:[13]

[13]The list is not complete and is only meant to provide an overview on some tools.

- **Capability map**: The capability map is a document (i.e. model, artefact or viewpoint) showing the business capabilities (cf. Sect. 2.2).
- **Viewpoints**: Viewpoints are artefacts, models or documents for visualising EA. Viewpoints are available in any EA software tool. Furthermore, TOGAF suggests a couple of common viewpoints [14, section 31].
- **Business Support Matrix**: The business support matrix is a tool for visualising relationships between business artefacts and IT systems. It can be used for identifying redundancies or gaps (cf. Sect. 4.3).
- **Deriving optimal application landscape**: The method for deriving an optimal application landscape from business capabilities is a also a tool. It provides some step-by-step instructions so that an Enterprise Architect can decide on which applications should exist under ideal circumstances.

6.4.6 Best Practices

The text book at hand also includes many examples showing best practices. They have been created based on experience and document how certain artefacts can look like. The example capability maps in Figs. 2.12 and 2.13 (pp. 53) are not reflecting real companies but show elements that can be seen in many maps. The same holds true for many examples and recommendations in this textbook.

This does not make it a best practices handbook (which never has been the intention). However, the examples can be reused and adjusted for similar companies or contexts.

6.5 Further Reading

We would like to close the sixth chapter of this book by also referring to some further readings, so books or papers that are recommended for having a look at for future purposes, but also, for the current activities we are about to have during the active seminars.

There is one paper which is quite old, but still is not irrelevant. It is a paper providing an overview on frameworks for EA [21]. A more up-to-date overview can be found in [22]. The authors categorise frameworks by four domains: military, company, government and manufacturing. They also draw links to historical standards that are usually not listed as typical EA approaches.

The second publication is The TOGAF Standard—the specification describing how TOGAF looks like [14]. It is published by The Open Group as a book, and you can buy it in the bookshop, but it is also available as a free resource in the internet. After registering at the home page of The Open Group, you can access the HTML version of TOGAF.

For those who are interested in having a closer look at ArchiMate, I would like to recommend a book which is some kind of textbook for teaching and learning ArchiMate published by Gerben Wierda, a name you might remember [10]. He is the same guy that also was talking about Chess and the Art of EAM. Instead of being the governance part, rather act like a chess player, adjusting decisions as the company evolves over time. He also published a good book on teaching ArchiMate.

6.6 Summary

That's almost the end of this textbook on *EAM*. Let us reconsider the topics we covered as shown in Fig. 6.13. After introducing the topic EA and EAM, we had a look at details for describing business perspectives and then describing applications and how they relate to the business perspective. We then learned some basic tools for analysing an EA, and then also talked about the skills of an enterprise architect and how should we set up an organisation for EAM in a company.

We were closing the book by a look at frameworks. There are many frameworks available and, at the end of the day, it is not important to know all frameworks or to choose the best framework you can find on the market. The most important thing is to understand basic concepts of EAM—which we did in the course of the first five chapters.

Fig. 6.13 That's all folks!