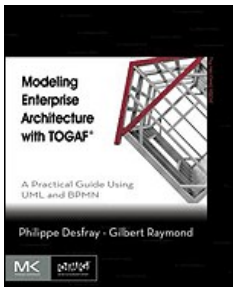

Modeling Enterprise Architecture with TOGAF: A Practical Guide Using UML and BPMN

by Philippe Desfray and Gilbert Raymond

Chapter 5: Key Modeling Techniques

TOGAF places particular emphasis on models and on the construction of a repository. Modeling languages help to better formalize knowledge, analyze problems, and design solutions. However, they constitute a toolbox whose uses, potential benefits, and limitations must be clearly understood. Architects need to understand how to use and benefit from models.

5.1 MODELS: BENEFITS, USES, AND CHARACTERISTICS

5.1.1 Definition

What is a Model?

According to TOGAF, a *model* is a representation of a particular subject. The model provides this representation on a reduced scale, in a simplified or more abstract manner depending on the subject in question. In the context of enterprise architecture, the subject is the enterprise or some of its parts. The finality of the model is the elaboration of views that address stakeholders' concerns; in other words, their viewpoints on the enterprise.

The model concept can be considered in a restrictive way, where the model is made up of and limited to what has been formalized in the modeling tool repository. Alternatively, it can also be looked at in a more extensive manner, where the model includes all the informal elements gathered during enterprise architecture work (texts, images, etc.).

A Universal Need

The need to build models is universal, reaching far beyond enterprise organizations and information systems. It would be impossible to imagine the absence of models in the construction business, where plans are needed to define what needs to be built, to coordinate the problems of different building trades, and to define who needs to do what. In this domain, models have a legal dimension, for the authorization of building permits, declarations of taxable surface areas, or contractual aspects between the client, the business owner, and the project manager. Plans and maps constitute another example of essential models, ones we could not imagine living without. In most domains, dedicated models have thus been defined and become widely used (mechanics, building architecture, CAD, avionics, and electronics are just a few examples).

Although the need for models is universal, enterprise architecture and information technology present particular difficulties that have delayed and reduced their implementation. Both fields are immaterial and theoretical, making them more difficult to represent than more concrete domains. Thus, while blueprints for a building pose no interpretation problems (everyone understands what a wall is, and that a wall is represented by a solid line), the same is not true in our field: How do you represent a concept, a state, an event, an application, a function, and so on? Conventions have to be fixed in the field, conventions whose technicity will prevent them from being as universally accepted as those of building plans. It is only during the last decade[a] that modeling standards have stabilized, finally paving the way to standardized support for the construction of enterprise architecture models.

History

In the field of information technology, models have been around since the very beginning, with significant growth during the 1980s. Essential modeling techniques have been known since the start of the 1990s. However, there was great heterogeneity among models, with significant differences between countries and IT fields, as well as unsuccessful cooperation between these models (Merise was a method and model specific to the IT management field in France, the Yourdon method was popular in the Anglo-Saxon world, use of the IDEF0 model was widespread in technical systems, the first object-oriented models were highly specialized and few and far between, and so on). Modeling standards were born in the 1990s/2000s UML and then BPMN), enabling modeling techniques to be harmonized.

Standard architectures have also evolved, resulting in the emergence of SOA architectures in the field of information systems. Since 2000, the UML and BPMN standards have matured and stabilized, together with their adaptations to specific applications (such as SysML for large-scale systems). The field of enterprise architecture, which has gradually emerged since the 1990s, can use these standards to model the entire enterprise. TOGAF thus recommends the use of UML and BPMN. However, TOGAF has its own metamodel. An architect who has to use models must first decide how to use UML for TOGAF and how to map UML concepts to TOGAF concepts. The goal of Chapter 6 is to provide an answer to this question.

5.1.2 Usefulness of a Model

Understanding and Thinking about a Problem

Models are used to meet several types of needs. By formalizing knowledge, they enable a problem to be understood and clarified. Using models, the different components of a field of study are represented, with different types of links used to position them. These components are further developed by allocating properties to them. Thus, models help participants think, and are then enriched by the results of this thought process. By materializing the understanding of a problem, models describe both the context and the target domain and reflect the intention, in other words, the construction project envisaged.

Models thus support two essential activities, namely, analysis and design. Analysis defines the description of the problem, and details the areas where intervention is needed. Design focuses on the solution, describing how the problem will be solved and detailing which techniques and activities will be used.

Communicating, Sharing, and Collaborating

Communication is essential within enterprises, and models provide important support in this area. Enterprises use models to represent their organization, and these models are used both to map the elements of the enterprise (such as roles within the enterprise, sites, business processes, material resources, applications, etc.) and to provide details on its components and its functioning (such as the progress of a business process). Mapping consists of listing, classifying, and positioning what already exists in order to share knowledge and enable everyone to situate the different components of the map.

In the same way as a map of the London tube helps passengers get around and supports designers in their study of its future development, recipients of data schemas know, for example, how to use or add to the data.[b]

Models also help establish dialog between the different experts in an enterprise, typically between business owners and project managers, or between users and business analysts. Thus, models can be used to represent the business needs of users, which are then precisely codified in order to prepare the work of architects and designers.

Models help share knowledge and contribute to its construction. Collaborative work can be carried out on models, which are enriched by each person's contributions and shared by all team members. This is typically the case for engineering work, which consists in building and consolidating models.

Modeling knowledge of the business, organization, processes, and IS enables the enterprise to constitute a legacy that can be used and reused in many different ways. The usefulness of this knowledge sharing goes beyond the walls of the enterprise, since it allows improved formalization of cooperation and exchanges between enterprises, partners, and clients. Last but not least, models allow preexisting business solutions to be reused by aligning the enterprise architecture model with shared business models, such as those that exist in the domain of insurance or certain banking fields. This alignment also makes it easier to guarantee that imposed business norms and standards are respected. For example, this can concern information that has to be kept or procedures that must be respected.

Planning and Simulating

Frequently used in all sorts of engineering projects, models are also used to *plan and simulate* the behavior of the system to develop, as well as the work involved in the actual construction of the system. By providing a view of the whole, the model enables changes to be applied more easily, in accordance with a particular strategy. Hypotheses can also be tested and variations thought up. These activities are not possible when the project is actually being carried out, as they are too costly in terms of both time and money. Moreover, models are often used to identify work packages within a project in order to delegate and monitor work.

Producing

In some domains, such as CAD or IT, models are used to guide, check, and automate production. Thus in mechanical CAD, the modeling of parts enables complex systems to be defined, precise specifications to be provided to subcontractors, and workshop construction of parts to be automated. IT also uses models to automatically generate more detailed IT productions (for example, code, database schemas, or documents).

As they evolve over time from vision to realization, models become more and more precise and formal, guiding or even automating the work of architects, designers, and then developers.

The drawback to this approach is that once models become precise enough to be executable or compiled into code, they become much less useful for understanding and reasoning on complex systems. Enterprise architecture is more about planning and less about building, and therefore requires less precise, less formal models.

5.1.3 Characteristics of Models

Abstraction

Models provide *abstraction* mechanisms, which enable users to consider the system at more macroscopic levels, by aggregating detailed elements, only showing significant parts or generalizing notions and mechanisms. Abstraction helps manage complexity, which is one of the main brakes within an enterprise, causing the lethargy and inertia that prevent many enterprises from being reactive. When there are thousands of applications in an enterprise, dozens of repositories, hundreds of processes, and consequently thousands of tasks, and when the volume of application code is counted in millions of lines of code, the problem of complexity linked to volume and diversity cannot be ignored. Abstraction is necessary for primary management and classification needs, as well as for more sophisticated needs, such as pooling, reconciliation, and rationalization. For pedagogical reasons, abstraction is also used to adapt the level of detail presented to the participants in question.

We will see that models for TOGAF will be separated into different viewpoints dedicated to specific stakeholders. The level of abstraction therefore needs to be carefully defined for a model's targeted purpose and stakeholders.

Standardization

Model *standardization* greatly increases the benefits they bring. Standardization ensures notation that is unique, used by everyone, and shared between all countries and all fields related to enterprise architecture (organization, business processes, data, applications, and IT). Standardization provides formally developed semantics; in other words, a formal definition of all its terms, mechanisms, and constructions, thereby limiting the number of possible interpretations of a model. Standardization also guarantees a market for a large number of associated tools (modeling tools, generation tools, etc.) and enables interoperability between modeling tools, allowing users to avoid being locked into

one particular tool. Models developed for the enterprise have greater value because they can be used by a large number of people and tools.

However, models such as UML and BPMN constitute a vast toolbox, and it is up to each organization to define its conventions, which parts of these models will be used and to what ends, and which extensions are provided. This book provides an example of conventions and extensions for TOGAF, whose implementation facilitates the work of participants, the sharing of information, and the building of a common repository. By defining these conventions and extensions of UML and BPMN, we naturally also introduce extensions and adaptations to TOGAF as described in Chapter 6.

Formal or Informal Models

Informal models do not respect rigorous formalism. They are often free images, media for ideas, or spontaneous means of communication. They are naturally used during meetings and facilitate spontaneous communication. During early analysis phases, where the problem has not yet been clearly established and consensus is needed to scope the area of intervention and the work to be carried out, or where there is a need to communicate with participants who have no experience of the models used, informal models can be used.

During phase A (Vision), these informal models can, for example, be used to produce solution concept diagrams or value chain diagrams. They will then be used as a basis for building more precise "formal" models, which will be managed by modeling tools.

However, as we progress toward the technical solution, models must be more precise and must provide a high level of detail regarding the functioning of the solution. In this way, we end up with more formal models, which describe in detail the structure of data or the logic behind activity sequencing in a process. Here, we are addressing more experienced participants, which means that more technical models can be used. As a result, the development phases of the ADM will make greater use of formal models. The viewpoint related to a model drives the domain, scope, time horizon, and level of detail of a model.

5.1.4 Limitations of Models

The model is only a component within TOGAF, the main point being the ADM cycle. The ADM provides the processes and activities that produce and consume the models, and motivate their change over time for some purpose.

Models are tools that must be pertinently used in the context of the enterprise. In Section 5.6 we look at the limitations of models (incompleteness, difficult to update, etc.), which necessitate adapted calibration and governance.

Moreover, models do not provide the contextual information that is necessary to their being understood correctly. Why was this model developed? Which problem is it to deal with? How will it be used? This last point will be addressed by specializing models using the different "diagram" type TOGAF artifacts, and by structuring them into viewpoints: every time a model is used in a TOGAF artifact (see Chapters 6–11), TOGAF and the *viewpoint* affected will indicate the main issues that are being targeted and the participants at which the models are aimed.

[a]Notably BPMN and UML, mentioned in this book.

[b]However, these models are too often oriented toward the description of the structure of data, and not its meaning. Conceptual models, oriented to defining the semantics of a domain, are of high value for enterprise architecture.

5.2 THE CONCEPT OF VIEWPOINTS

5.2.1 The Angle from Which a Problem is Looked at

Complex systems imply the involvement of a wide variety of expertise and stakeholders. Each type of expertise requires a specific view of the system, and will only be interested in a part of the model of the system, according to a particular representation. This angle of vision or these concerns being addressed, which target(s) certain categories of stakeholders, constitute(s) a *viewpoint* on the model.

Figure 5.1 successfully illustrates the need for different views, according to who the participants are and what problems are to be managed. When dealing with a client and future user, attractive and/or functional views will be chosen. However, when working with different building trades, views dealing with trade-specific problems will be developed (structural plans, electric networks, plumbing, etc.).

Figure 5.1: "Technical" and "marketing" viewpoints of a building

A similar problem exists for information systems. For example, users can attach importance to *ergonomics*, security managers to the *security* of the system, system administrators to *technical deployment*, data administrators and database designers to *data schemas*, business analysts to *business processes*, or service architects to *technology architecture*, and so on. Figure 5.2 shows an example of a business analyst contemplating a process model, and an application architect faced with an application architecture.

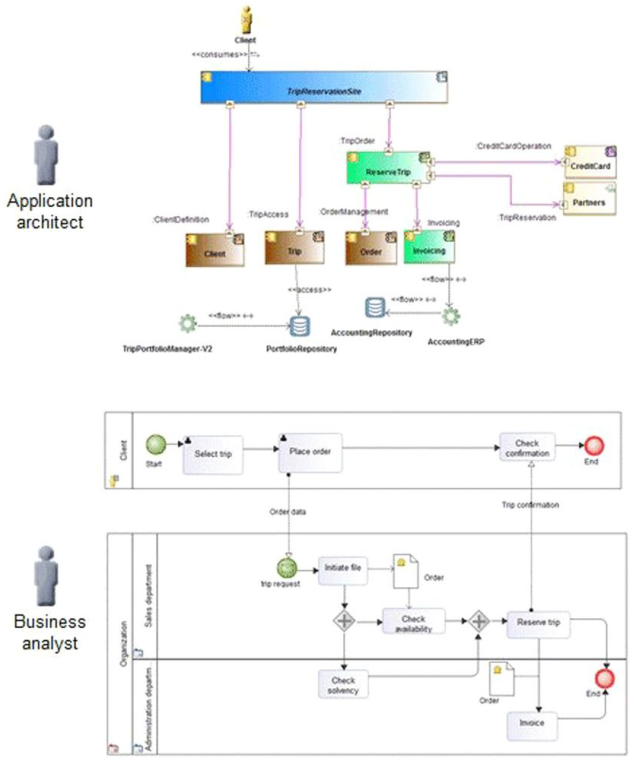

Figure 5.2: Different TOGAF models according to different viewpoints

This leads to the identification of a number of viewpoints in the enterprise, which materialize both the principal groups of issues that will have to be managed and also the participants concerned. Determining viewpoints provides real structure to the organization and the work to be carried out, by configuring the types of problems to be dealt with and the nature of the people who will be involved. This is why this concept has

become increasingly important since the 1990s in modern methodologies, such as RM-ODP (for network architectures), RUP, Zachman (the reference in terms of enterprise architecture), Praxeme[c] (enterprise method), MODAF,[d] DODAF,[e] and of course TOGAF. The concepts of viewpoint and view are standardized in the IEEE 1471-2000 norm.

5.2.2 View and Viewpoint: Definition

One of the objectives of enterprise architecture is to produce representations that cover all stakeholders' concerns. These specific representations must be linked, and must reflect all the compromises and adjustments that manage potential conflicts between concerns (for example, performance and security).

Viewpoints constitute different perspectives of a system, developed according to the main concerns of stakeholders. A viewpoint focuses on one or several issues, and therefore on the stakeholders concerned by these issues, and defines a set of conventions to develop adapted views.

A *view* is a representation of the system from the perspective of a set of issues (the viewpoint). A view is what we see of a particular system, while the viewpoint is the angle from which the system is looked at. Viewpoints are defined generically (independently of any particular system). Figure 5.3 shows two viewpoints and six views, each related to one of the viewpoints and each providing a particular representation of the same system.

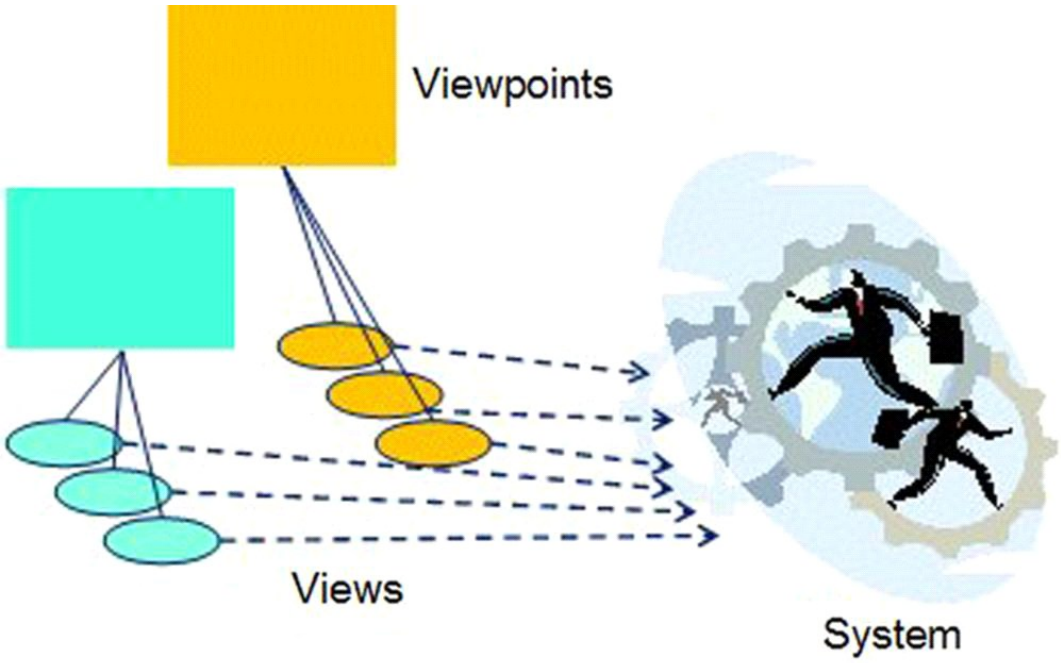

Figure 5.3: Viewpoints, views, and the system

The viewpoints (or concerns) we can have for an enterprise and its IS can be related to:

- Security (access rights, intrusions, etc.)
- Persistent data (data models, databases, etc.)
- The organization of the enterprise (actors, organizational units, etc.)
- Application architecture (applications, messages, etc.)

Generally speaking, several views exist for each viewpoint, and for this reason, Chapter 9 presents several views related to application architecture.

5.2.3 Usefulness of Views

Views help manage complexity by separating problems into different skill domains related to business, technology, or organization.

In contrast, when considering a given model, it is difficult to interpret it if we do not know the *viewpoint* to which it is related. Is it a business model? A technical solution? A design model? A technology architecture model? With regard to the concerns being addressed, does this address the structuring of the application, the combination of legacy applications and new evolutions? Does it describe the conceptual business domain, or does it specify how a business process should be optimized? Is it an example or an illustration? Does it reflect what exists or what is intended; in other words, the project that is going to be developed? In itself, the model determines neither the context in which it is defined nor the intentions that led to its development. Knowing the viewpoint from which a model should be looked at allows the spectator

to interpret it and find out how to use it. This is not only true for spectators but also for tools, which can apply different checks and usage services according to the viewpoint in question. For example, code generation, matrix or report generation, and consistency checks are features that are closely linked to the viewpoint.

Viewpoints are used to break complex models down into different aspects, and to present these aspects to the different participants according to their field of interest. In this way, they help provide different formalisms, which are adapted to different participants. They provide assistance in organizing teamwork within large-scale teams and in structuring models and deliverables, by allowing each contributor to participate according to his or her level of expertise.

Recent methods use viewpoints to organize work and skills, such as in the Zachman framework, the DODAF and MODAF architecture frameworks, or the Praxeme enterprise method.

5.2.4 TOGAF Viewpoints

The implementation of TOGAF implies the definition of several viewpoints. By default, we can consider TOGAF's four architectural domains as being predefined viewpoints. The examples of views provided in Chapters 10–15 are based on this structure.

However, TOGAF recommends that viewpoints relevant to the ADM cycle be identified during the preliminary phase (see Section 2.2.1). For this, we have to think about which architectural views and viewpoints have to be established if we are to satisfy the requirements of the different stakeholders. One of the essential uses of viewpoints is to allow architects to show how stakeholder concerns have been dealt with in the different TOGAF architectures (business, data, etc.). Table 7.2 (Section 7.2) shows an example of a stakeholder matrix consolidated in phase A, which helps determine which viewpoints to develop. For any given viewpoint, appropriate types of diagrams must then be defined, as well as adapted tools and methods (for example, by selecting diagrams from those presented in Chapters 6–11).

Table 7.2: Stakeholder Matrix (Extract)

Participant	Concern	Decision-Making Power	Level of Interest
CEO	Goal orientation, decisions	High	High
Organization unit director	Requirements orientation, decisions	Quite high	Medium
Business architect	Business, architecture	Medium	High
CIO	Project management, IS	Quite high	High
Data architect	Data architecture	Low	Quite high
System and network engineer	Hardware, systems, network	Low	Medium

The TOGAF architectural domains are already viewpoints that deal with the typical concerns of stakeholders. For example, if we use some of the capabilities that appear in Table 7.2:

- Business architecture deals with the needs of users, organizational unit directors, business analysts, and business managers.
- Data architecture deals with the needs of data architects and software designers.
- Application architecture addresses the needs of application architects, technology architects, and CIOs.
- Technology architecture targets the needs of operations managers, system and network engineers, and technical architects.

Table 5.1 gives a general overview of the expectations of each different category of stakeholder.

Table 5.1: Development of Viewpoints According to the Issues to Deal with Extract from the TOGAF Guide

Need	Stakeholder	Goal	Example
Design	Architects, software designers, BPM analysts	Design, explore, establish a basis for decisions, compare the alternatives	UML or BPMN diagrams
Decide	Managers, CIOs	Make decisions	Cross-referenced tables, mappings, lists, and reports
Inform	Users, clients	Explain, convince, obtain support	Animations, images, prototypes, model illustration

[c] www.praxeme.org, http://en.wikipedia.org/wiki/Praxeme.

[d] UK Army Enterprise Architecture Framework, http://www.modaf.org.uk.

[e] DOD Enterprise Architecture Framework, http://dodcio.defense.gov/dodaf20.aspx.

5.3 SPECIAL ROLE PLAYED BY DIAGRAMS

5.3.1 Models and Diagrams

Enterprise architecture develops a model through different views, each related to one or more viewpoints. Some of these views are diagrams.

A *diagram* is thus a graphical view that represents a part of a model. The model of an enterprise can be considered to be a repository that includes all concepts, properties, processes, tasks, actors, and so on, and all the different types of links that associate them. Diagrams are only one of a number of representation forms that exist for a model, some of which take graphical form (for example, class diagrams or process diagrams), others textual or syntactic form (for example, business rules), and still others table form (for example, TOGAF matrices), as well as other forms, such as model element hierarchies. For this reason, Figure 5.4 shows the classic layout of a modeling tool. The explorer on the left is used to browse the entire model, while the diagram on the right-hand side graphically represents a small subset of the model.

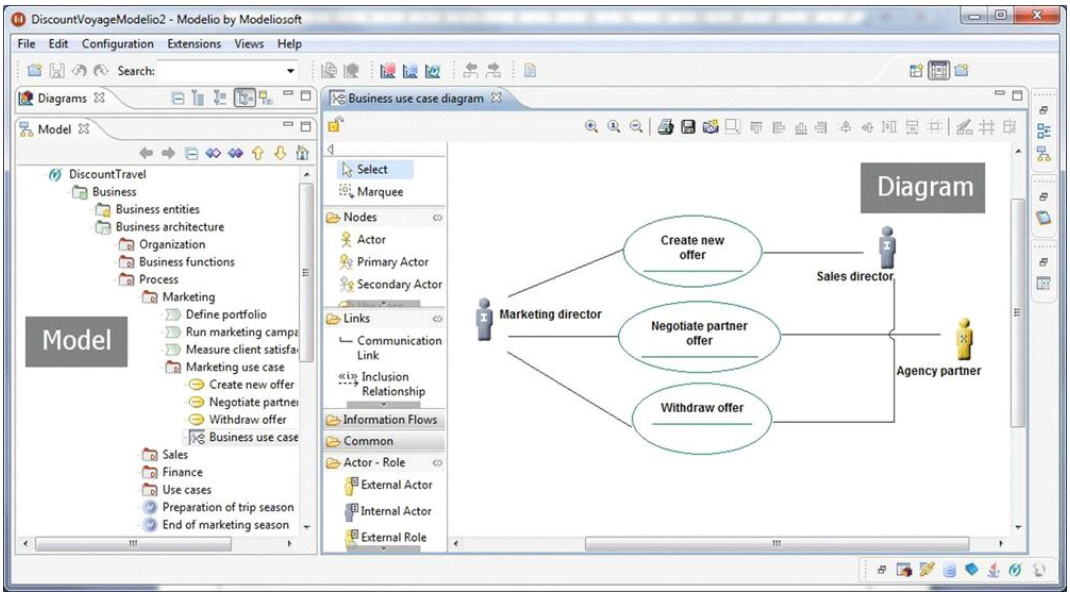

Figure 5.4: A model and a diagram in a modeling tool

5.3.2 Diagrams for Communicating

The main function of diagrams is communication between participants. Diagrams help position elements based on specific visuals (schemas or images) that would be hard to do without. Their aim is therefore not to present the entire model, but rather to illustrate it and to explain it. To this end, diagrams filter the elements to present, starting from models that are sometimes very large. Diagrams are dedicated to particular participants, and focus on a part of the model. UML and BPMN provide several types of diagrams, each of which represents a specific facet of a problem, with a different representation mode. For a given model, there exist several diagrams of the same type, each dealing with a different part of the model (for example, several process diagrams for several processes, several class diagrams for different parts of the model, different representation levels, etc.). One model element (for example, the "Order" class) can be represented in several different diagrams.

Table 5.2 shows which types of UML and BPMN diagrams can be used to answer essential questions (defined in the "Zachman framework"). As we can see, these standards do not answer the question "why," despite its fundamental importance. This question is handled notably using goal diagrams and requirement diagrams (see Sections 7.3 and 7.4), which are neither UML nor BPMN. The extensions to UML and BPMN presented in this book are used to better answer these questions in order to cover the entire scope of TOGAF by completing all the boxes of this table (Table 5.2).

Table 5.2: Questions Dealt with by Different Types of UML and BPMN Diagrams

The Question ...	Is Handled by ...
What?	Class diagrams, package diagrams, object diagrams, component diagrams, state diagrams
Who?	Use case diagrams, process diagrams
How?	Process diagrams, use case diagrams, sequence diagrams, collaboration diagrams
Where?	Deployment diagrams
Why?	
When?	Process diagrams, state diagrams

Table 5.3 Diagrams play an essential role in communication between the participants involved in handling a problem, as the table 5.3 shows for UML and BPMN. Their aim is therefore not to present all the information modeled. Due to their role of exchange and communication, they must be understandable by all stakeholders. Diagrams which are too "technical" will not be understood by management personnel, users or certain "business" participants. Organization diagrams, process diagrams or use case diagrams can, for example, be presented to users or directors, while class diagrams or state diagrams will be reserved for more "expert" participants (analysts, architects, and designers). These considerations help determine which types of diagram should be used with regard to the different viewpoints identified. We will see that the adaptations that we propose to cover TOGAF will address these issues, and that the technical parts of UML, in particular, will be filtered out.

Table 5.3: Use of Different Types of UML and BPMN Diagrams (Example)

Diagram	Example of Use
Use case diagram	Expression of requirements; functional requirements
Class diagram and package diagram	Conceptual model, data model, software models
Sequence diagram	Example of functioning
State diagram	Entity lifecycle; functioning dynamic
Collaboration/Object diagram	Cooperation between objects; architecture illustration
Process diagram	Business process and workflow modeling
Component diagram	Logical and physical architecture models
Deployment diagram	Hardware architecture; geographical distribution

5.4 CONSISTENCY AND TRACEABILITY

5.4.1 What is a "Good" Model?

There is no one right answer to this question, which is subject to lengthy discussion. However, there exist three important criteria that help qualify a model:

- Its consistency: A model is consistent when it satisfies the consistency rules imposed by the modeling language used. The model must be consistent, in other words structurally sound, although this does not in itself guarantee that the model is good. For example, the blueprints of a building may respect construction rules (roofs are supported by walls, there must be an entrance and an emergency exit, etc.), but this in no way determines whether or not the building is appropriate.
- Its relevance: A model is relevant when it successfully represents the problem that is to be dealt with. The model must describe the problem using the correct concepts, with an appropriate level of detail. All necessary elements must be present, and all the elements present must be necessary. This criterion is both the most important and the most difficult to judge.
- Its justification: Justifying the existence of a model's elements helps ensure its relevance. A model element is justified by the fact that it meets a defined need, or a goal, or that it represents one of the aspects of the domain to be dealt with. Model elements are essentially justified through traceability. In this way, the elements necessary to the completion of the model are built based on known and desired elements.

When modeling languages are used, their consistency rules must be respected. This consistency can be naturally imposed by modeling tools, or else checked later by model checks. Furthermore, each "viewpoint" predefined for enterprise architecture will add specific consistency rules. These rules act as a guide for the model designer and as a guarantee for readers.

Model consistency rules on models essentially concern models located at the same level of representation, or the same level of TOGAF architecture (or inside the same viewpoint). If several levels coexist, for example, when building a model based on higher level models, such as an application architecture model based on a business model, then intermodel links will essentially be "traceability" links.

5.4.2 Traceability Links

The term "*traceability*" designates the ability to link artifacts (see Section 4.1) produced by enterprise architecture and subsequent technical realization activities to other artifacts from which they originate or to which they refer. This practice is widely used, notably in requirements management, either to check that all requirements are traced to at least one artifact describing their satisfaction (completion of the artifacts realized), or conversely to find out which requirement is linked to an artifact. Traceability links are also used to carry out impact analyses, which indicate what will be affected if a requirement is modified and calculate the consequences. Traceability links form a network between artifacts and/or model elements, which constitutes a graph. More precise types of traceability links can be defined, for example, to express that a model element "refines," "satisfies," or "verifies" a requirement (see Chapter 11 and the "Requirement analysis diagram" artifact for the definition of these different types of links). By extension, we use the term "traceability" link to designate any dependency that is not predefined by modeling languages and that a designer creates between two different levels of representation or architecture. Thus, an "assign" link from a goal to a process or an actor, or an "implements" link between an application component and a process, will be classified in the very generic category of traceability links. Figure 5.5 uses the "trace" link, which is very vague about the meaning of the dependency in question. We recommend the use of more precise links ("refine," "implements," "satisfy," etc.) to express the nature of the dependency or relationship between a model element and its reference.

Figure 5.5: Traceability between several model elements

A traceability link is always oriented toward the reference element. So in Figure 5.5, the "model" element refers to the definition of the "reference" element. For example, a business process may have been built based on one of the enterprise's goals, or an application

component may have been defined based on a business process. Note that this does not mean that the reference element has to be defined *before* the model element. The identification of a model element often leads to the subsequent identification of a reference element.

In the example shown in Figure 5.6, the "TripReservationSite" application component has been defined to partially realize the "reserve trip" process, and to satisfy the "IS access via web site" requirement. The "reserve trip" process has been realized to meet the "Increase number of trips reserved per day" and "Reduce file processing times" goals. This process supports one of the enterprise's functions, namely, the "Sales" function.

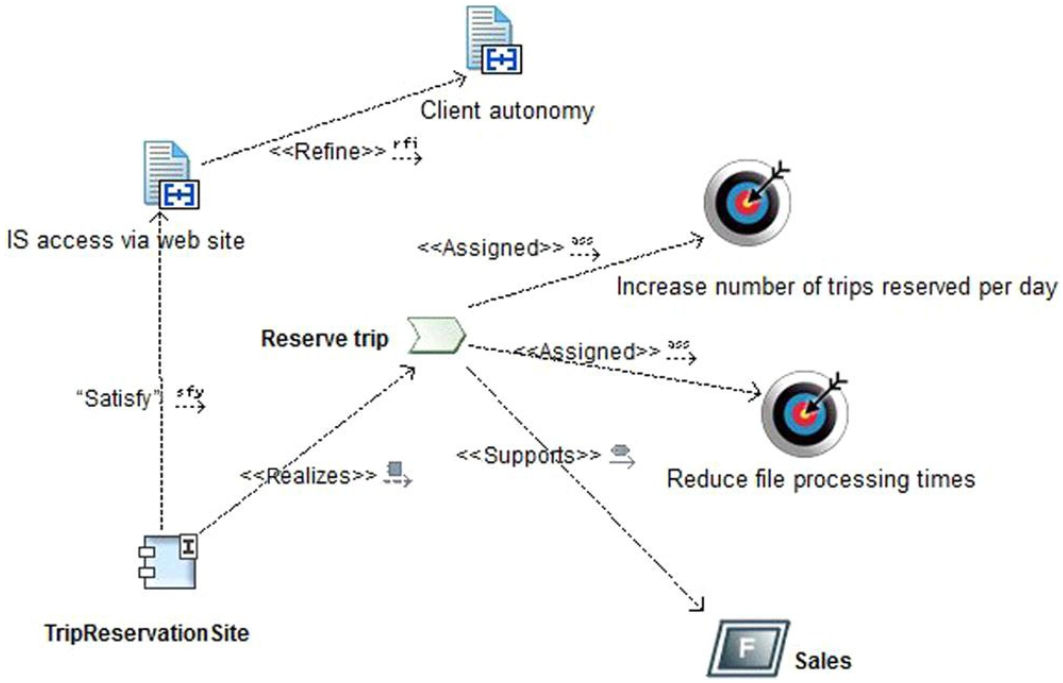

Figure 5.6: Example of traceability

5.4.3 Using Traceability in Enterprise Architecture

The origin of traceability links is generally established on the basis of knowledge obtained prior to the development of the model, and by elements that have justified the launch of an ADM cycle. Thus, the fundamental elements of the business (business terms, business entities) can constitute preliminary knowledge that initiates traceability links. The enterprise's goals (see Section 11.3) and drivers are elements that are essential to the motivation of the ADM cycle. Enterprise organization and business processes are therefore traced to these initial elements, and then the application model traced to business architecture elements. We obtain a sort of traceability graph, which clearly shows where all intermediate model elements come from, right up to the solution. The description of what already exists is also an essential origin of traceability; here the model is simply justified by the retranscription of elements resulting from the description of what already exists.

Traceability enables us to see what justified the construction of a model element (justification), and conversely to find out which model elements are based on a given element. In the latter case, starting with a model at a certain level, we can determine its coverage by lower-level models, and thus get an idea of how exhaustive a model is with regard to a reference. Another use of traceability is impact analysis, for example, to find out the cost of changing a requirement or altering a technical component.

We have already seen that a model constitutes a database of model elements and links. Traceability links are part of the model, and can therefore be systematically used from this database. Generally speaking, the term "traceability" designates the use of any links in this model database, to clarify if and how several elements are connected.

In practice, traceability links can appear in diagrams, but their systematic use is more often presented in the form of matrices. Most of the matrices defined by TOGAF result from the use of these links (Table 5.4).

Table 5.4: Retranscription of Figure 5.6 in the Form of a Matrix

	IS Access via Web site	Reserve Trip	Client Autonomy	Reduce File Processing Times	Increase Number of Trips Reserved Per Day	Sales
TripReservationSite	Satisfy	Realize				
IS access via web site			Refine			
Reserve trip				Assigned	Assigned	Supports

5.5 ARCHITECTURE REPOSITORY

During the activities defined by the TOGAF ADM, a large number of elements will be produced as a result of the work carried out. For this reason, TOGAF defines *deliverables, artifacts*, and *architecture building blocks*. The definition of these elements is provided in Section 3.1.1.

These elements are part of the assets of the enterprise, which will be able to reuse them during future evolutions, as well as in the current project. It is, therefore, essential to set up an enterprise repository for all these elements. The model is a fundamental part that must be taken into account by the repository, since a very large proportion of the types of elements that must be stored in the repository are model elements.

The *repository* saves and manages all model elements and their links, in particular, traceability links between model elements. It also saves diagrams and manages the connection between diagrams and the model elements represented. The "content *metamodel*" recommended by TOGAF defines which types of elements are stored in the repository. For example, all the model element types used in this book (Actor, Business Service, Business Entity, etc.) must be defined by the metamodel. The repository (content framework) is the medium for the activities of the TOGAF ADM, with each phase using repository elements as input, and producing other elements as output.

As an architectural framework, TOGAF can be extended. It must be adapted to each enterprise, which means that the metamodel must be able to adapt to each context.

In TOGAF the concept of the repository includes all elements created or modified during the ADM, which means that the scope of the repository covers a set of elements far beyond models. TOGAF indicates some of its components: elements linked to governance, the definition of business and application services, the modeling of business processes, the modeling of data, elements linked to application or technology architecture, and elements linked to vision, for example, goals. These examples should all be part of the model.

The repository allows viewpoints to be developed on the enterprise. Viewpoints do not organize models into separate, disjointed parts, but rather act as filter for models which are too large and too complex to be handled in their entirety. If we use the example of a building again, general plans interact with specialized plans concerning structure, plumbing, electricity, and so on. Overall consistency is essential. The repository and modeling tools guarantee this overall consistency, notably consistency between viewpoints, so as to ensure that the interactions between model elements handled by the different participants complement one another harmoniously. In Section 6.1 we provide an example of how to structure the model repository.

Typical uses of a repository include recording the results of enterprise architecture work, providing access to these results to each participant according to his rights, ensuring overall consistency, and enabling requests and extractions (matrices, reports) to be carried out.

5.6 RISKS AND MAIN DIFFICULTIES

5.6.1 Limitations Inherent to Any Model

Intrinsic Limitations of Models

Modeling is all about choices, and is largely down to analysts and architects. The famous painting "The Treachery of Images" by Magritte (1929) representing a pipe reminds us that all models are false and can never get very close to reality. For example, geographical maps are marred by approximations, despite being meticulously realized.

Modeling therefore consists in building a theory that we want to conform to reality. This task is easier in domains that are themselves man-made, such as insurance contracts or bank accounts, rather than in real-world phenomena. For example, it is extremely tricky to model human cooperation processes, since it is so difficult to take into account all possible interaction modes between people and to define a generalization that everyone in the enterprise must apply.

A Partial Description of the Problem

Models do not enable all the problems and all the knowledge of an enterprise to be represented. In general, they cannot do without related explanations and precisions, which can only be provided through associated documents, but also through direct communication between participants, in order to guarantee a good level of mutual understanding. Models are only a medium.

Figure 5.7 illustrates this point. The model shown highlights the concepts of "Holiday" and "Trip" in the field of a travel agency. Associations determine the possible links between the concepts of this domain (participants, range, and order for a holiday, for example). Without textual additions, this model lacks detail. For example, the following information, which is important to the business, is absent from the model:

- There is client uniqueness.
- A single client cannot take the same holiday several times.
- If a client participates in several holidays, there can be no conflict of dates.
- A holiday is only confirmed if an order is placed.
- A holiday only exists if the trip is still available.
- The departure and arrival dates of a holiday must correspond to the duration of its offer (trip).

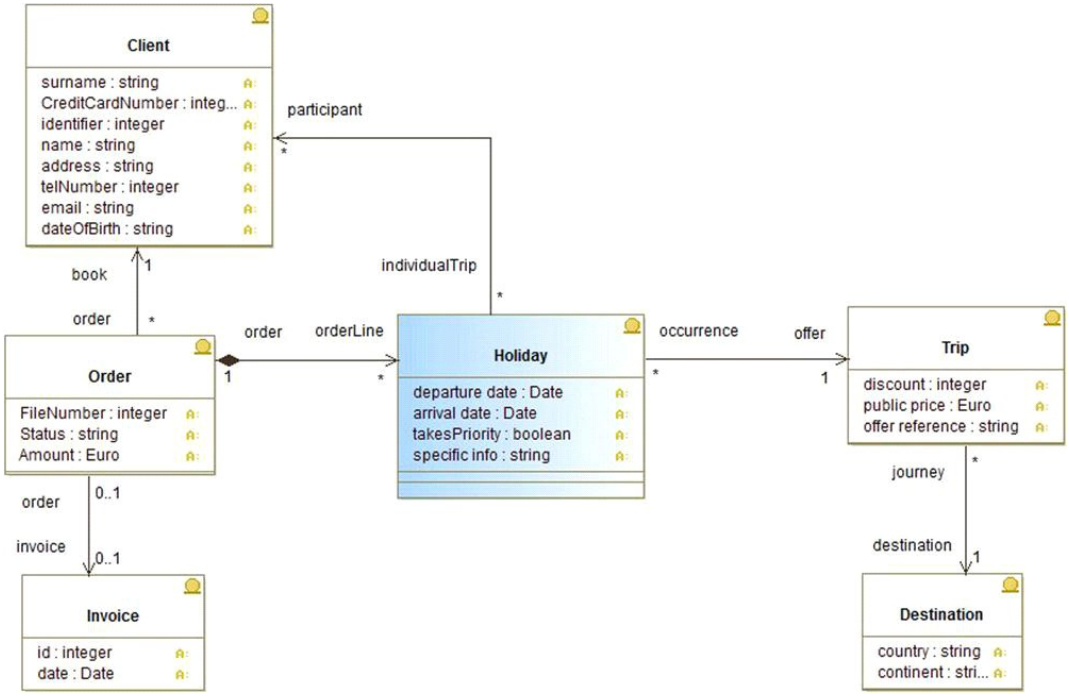

Figure 5.7: "Concept of holiday in a travel agency" class model

In this example, this information can be expressed in the form of business rules, and defined using adapted rule languages, but it is never possible to provide all the information. A model, like a document, cannot describe everything.

Confining Modeling Work

The benefits of high levels of detail must be evaluated. A detailed model supposes a larger volume of information, which has required a greater amount of work in its development. The effort involved in reviewing, understanding, and updating evolutions increases, thereby limiting the number of possible participants. Thus, investment in modeling increases significantly, compared with the benefits obtained. This forces us to define a practical limit regarding the degree of detail that should be provided by models. This limit is guided by considering the return on investment (ROI). For example, for parts of systems where security issues exist (nuclear power plants, transport systems, etc.), a higher level of detail is required. When generation services are applied to models (documentation, code, DBMS schemas, etc.), model designers will tend naturally to provide the level of detail necessary for the generation in question, and the model is then productive. However, where the goals are a general understanding of and a consensus on architecture, the level of detail will be reduced to what is strictly necessary pedagogically speaking.

Making a Model a Reference

The main difficulty of an enterprise architecture model is its constant evolution, and consequently its permanent update. Unlike building plans, which we can assume will remain stable over a very long period of time, enterprise architecture models quickly become obsolete if they are not updated. The organization, IS, procedures, business, and goals are constantly evolving, making it essential to rework models in order to reflect changes.

The situation can deteriorate very quickly. If analysts or architects do not trust in the relevance of existing models, the pressure of deadlines may force them into rebuilding models ad hoc, focusing on the requirements of the moment, which are even more short-lived. Worse still, they may decide to do without models altogether, in order to continue on to the actual realization of projects. If this happens, the enterprise is back at square one, where knowledge is neither controlled nor recorded, but rather scattered across detailed realizations.

This difficulty requires vigilant governance (see Section 4.2) and significant involvement of the architecture board.

5.6.2 Usefulness and Support: Major Criteria

Modeling therefore demands great modesty on the part of designers, who must realize that their models will never be perfect. They will simply aim to build models that are useful with regard to identified needs. Thus, the London tube map is a model that is geographically false, but that is very useful in practical terms. It is recognized by everyone (designers and users) as being a reference, which makes it extremely valuable.

Besides the criteria identified in Section 5.4, the value of a model essentially resides in how well it is accepted by all the stakeholders. A business process model, which is understood and accepted by users, business analysts, and, where necessary, IT engineers, is of tremendous value. The same is true of an organization model shared by decision-makers and business analysts. A model designer must therefore avoid working alone on an individual task, and must use all possible means to obtain the support of stakeholders, using a pedagogical approach when presenting a model. The organization of reviews also contributes to the quality of models. The earlier the model is produced, the more these exercises are necessary. This means that the models produced must be understood by everyone. Later on, models become more technical and more detailed, meaning they cannot easily be presented to users, for example. These models will have to be based on earlier models, notably using traceability links, in order to justify choices and the reasoning behind them.

As we have already mentioned, models cannot do everything. They cannot replace the willingness, discipline, organization, and skill of human beings. However, they are a powerful tool that facilitates the analysis, design, and communication of solutions.

Designing a model requires significant pragmatism. Model design is not concerned with covering a whole array of model types, nor with aiming for maximum detail, which will prove to be useless. For every problem to be dealt with, the models built must bring added value, and we must only model what is absolutely necessary. Models must clarify the problem. For example, modeling a business process clarifies its functioning and procedures. If the process is realized differently depending on the participant, then producing a generalization that satisfies all participants becomes a complex activity. Similarly, realizing a conceptual class diagram clarifies a business domain. Models must make sure that construction of the solution is secure. For example, a conceptual class diagram brings significant added value to the definition of a repository or software application. Models must clarify intention, share knowledge, and encourage consensus. For example, application and technology architectures constitute an area of negotiation and sharing between business owners and project managers, enabling them to contractualize what has to be done and apply decisions to what already exists.

5.6.3 "*Bottom-up*" or "*Top-Down*": Two Limited Techniques

Two major approaches to modeling exist. The "*top-down*" approach starts with general models, and progressively builds models that are more and more detailed, right up to the complete definition of the solution. This is a highly analytical approach, which has the advantage of covering the entire scope of the problem and of positioning all components. The risk of this approach is that it can be too general, too theoretical, and too disconnected from reality. By arriving at the solution relatively late, we then realize that overall solutions cannot be applied to real situations, which sometimes calls the entire theoretical construction into question. Conversely, a "*bottom-up*" approach has the advantage of focusing on a specific part of the problem by providing a solution that will be a tried and tested brick in the construction of the entire edifice. This approach can start with a prototype of the solution, which will be used to validate its viability. The disadvantage is that this approach provides no overview, and does not guarantee the consistent integration of bricks that is necessary to a pertinent view.

A combination of the *bottom-up*/*top-down* approaches should be used. For any given problem, we have to define what should be carried out in *top-down* mode and what should be realized in *bottom-up* mode. A specific strategy must be established each time. Risk analysis is a guide when defining this strategy: insufficiently known parts, subject to functional or technical risks, are good candidates for a *bottom-up* approach.

5.7 REPOSITORY GOVERNANCE

The goal of the *repository* is to constitute an asset of knowledge for the enterprise, which can be used as a basis for reusing information about the business, process, organization, application architecture, database schemas, and so on. A properly managed repository is of tremendous value, as it provides an immediate framework for defining evolutions, building specifications, and evaluating the feasibility and consistency of new projects, among other things.

However, the correct use and durability of the repository depend greatly on its quality. If close attention is not paid to what is going into the repository, an abundance of produced models will find their way in, without their relevance, completeness, and overall consistency having been established. In other words, the value of the model is thus reduced, and participants cannot use existing models as a basis for their development. Any reasoning on existing models is marred by the errors and imprecisions they include. Enterprises frequently have a large number of nonmanaged models, obsolete process models, and application mappings that do not correspond to reality and do not meet the expectations of a repository. The absence of investment in governance leads to the depressing impression of a heap of models that must constantly be reworked depending on current needs and emergencies.

Thus, construction of a repository must be organized and its *governance* ensured, if its quality is to be guaranteed. Repository evolutions must be managed, as well as evolutions of the enterprise and its IS over time, which must be properly reflected in the repository. Participant rights on identified parts must be identified. The quality of elements must be checked before they are added to the repository. The data located in the repository must also be audited, and corrected or updated where necessary, in order to align the repository with enterprise evolutions. Everything that is in the repository must be managed in the long term, which means significant management costs.

This update presents the challenge of having a large amount of information updated by a dedicated team that does not always have all the business knowledge. When architects are in pure production mode, similar to project mode, they systematically give priority to their own emergencies, thereby neglecting update operations. This is the core difficulty of governance: managing the conflict between the needs of individual projects, delivering work products that meet specific project requirements within a specific timeframe, and the needs of longer-term enterprise asset management. This is generally handled by having different streams represent changes to model elements over time for a given purpose. Governance then controls the flow of changes between these streams.

As such, a pragmatic approach must be taken. Only essential information should be put in the repository, information in which we are willing to invest (checking, monitoring, and updating). This confirms the fact that the level of detail required in a model must be adjusted to include only what is absolutely necessary. Having the necessary and sufficient level of information in the repository is one of the key points in repository governance.

The repository enables enterprise knowledge to be managed. In order for this knowledge to be procured, all stakeholders must contribute; knowledge must be extracted, collected, and federated from all participants. Participants must be encouraged to open up and fight their natural tendency to keep their individual knowledge to themselves and to appropriate skills for themselves alone. This is a knowledge management and collective intelligence project in itself, whose accomplishment is at the heart of the success of any enterprise architecture project.

We have seen that the ADM is correlated to the repository, as its activities withdraw and deposit information. The constitution of the repository and the nature of the stored elements have an influence on the definition of the ADM.

The TOGAF architectural capability framework handles these organizational aspects, notably the need to define an organization, roles, responsibilities, and skills when setting up enterprise architecture. Managing the repository is not a project but rather a continual, long-term operation. The governance of the repository is a cross-organizational function in the enterprise under the responsibility of the architecture board, which brings together a limited number of representatives of the stakeholders.

As shown in Figure 5.8, governance has a much wider spectrum than the constitution of the repository, the chosen field of architects. It must also check that projects respect enterprise architecture, as well as deployment and operational system management operations.

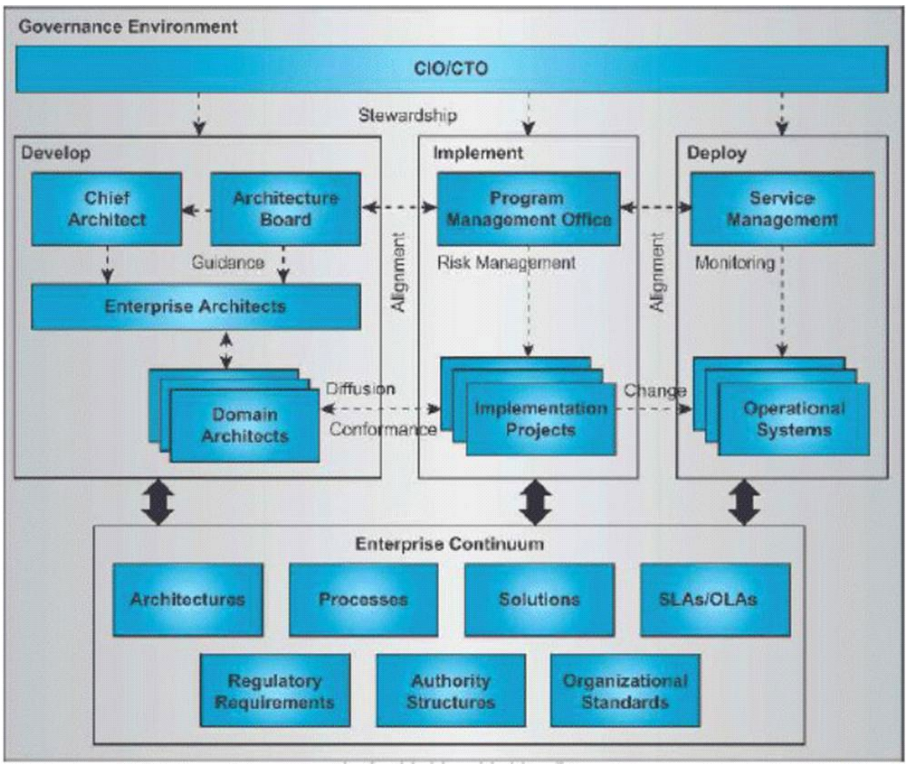

Figure 5.8: TOGAF architectural governance framework

5.8 TOOLS AND LANGUAGES

5.8.1 Modeling Tools: A Necessary Medium

Modeling, tooling, and the repository cannot be dissociated. The main function of enterprise architecture modeling tools is to:

- Provide graphical editors that support TOGAF models
- Guarantee the correct use of TOGAF concepts
- Manage the part of the model repository
- Support and coordinate teamwork between participants
- Generate useful products for artifacts or deliverables (for example, matrices or documents)
- Provide a set of services associated with models (such as model requests, impact analysis, redundancy searches, searches for unused elements, statistical reports, service and application component reuse rate analysis, and consistency and quality analysis).

In short, the purpose of the model is to capture, communicate, validate, reason, and act upon shared information, and tools make this possible.

Tools thus provide a "content *metamodel*," which includes TOGAF elements. All these tools must be adaptable, so as to take into account adaptations of the generic TOGAF framework. It must also be possible to customize the assistance they provide in the production of other TOGAF artifacts and deliverables, typically catalogs and matrices.

These tools generally structure models according to TOGAF viewpoints, thereby presenting diagrams adapted to the people concerned. They guarantee the overall consistency of models by managing element uniqueness, the update of elements in all the diagrams in which they appear, and the overall consistency of the model. By managing cooperation between the participants involved in building models, they help organize teams, coordinate work on the model by avoiding access conflicts, and manage model versions and configurations.

When an enterprise architecture approach is set up within an enterprise, everything must be taken into account: the architectural framework, the tooling, the adaptation to the enterprise's approach, and the adaptation to the participants' work mode.

Successful implementation of tooling will help improve communication between participants and record the work carried out. It will also help manage the enterprise continuum on a carefully controlled enterprise architecture repository, as well as facilitating overall governance.

Modeling tools do not do everything. They should be included as part of a wider toolset, in order to cover all services linked to a complete repository in the TOGAF sense of the term. For example, listings, deployment configurations, library versions, and production or tooling statistics producing matrices or architectural structures are frequently managed by integrated, third-party tools. The definition of tools is carried out during the TOGAF preliminary phase (see Section 2.2).

5.8.2 Tools Available in the Marketplace

There exist several tools capable of modeling enterprise architecture, all of which manage the model in a database and provide graphical editors.

Purely graphical tools such as PowerPoint or Visio should be ruled out. These tools can be used to build isolated diagrams, but provide no assistance when constituting a model repository. Simply using two or three diagrams that reference the same model elements is enough to be sure of ending up with a set of inconsistencies between these diagrams.

Two families of tools can be distinguished in the marketplace: tools dedicated to enterprise architecture, and UML and BPMN modeling tools providing extensions for enterprise architecture.

Tools dedicated to enterprise architecture, such as IBM's Rational System Architect, Mega, ARIS, or Case Wise, provide a general solution to enterprise architecture, and are customized for each enterprise context. They provide BPM (business process modeling) solutions and enterprise architecture solutions. The modeling language supported is often proprietary, but frequently includes BPMN. All propose predefined TOGAF customization, which means each provides a specific solution for TOGAF. However, it is possible to customize these tools to support the approach presented in this book.

This book presents an enterprise architecture modeling approach that uses TOGAF, UML, and BPMN modeling standards, and UML extensions dedicated to TOGAF in the form of "UML *profiles*" (see Chapter 10). The advantage of this extension technique is that it can be applied to all UML and BPMN tools.

Tools dedicated to enterprise architecture successfully cover functional needs through their focus, while UML and BPMN tools bring the benefit of their support of very widely used standards. TOGAF 9.1 recommends:

> "It is highly desirable that the description of the architecture be encoded in a standard language in order to enable a standard approach for the description of architecture semantics and to facilitate its reuse by different tools."

This addresses the use of TOGAF as a standard approach, but can also be extended to the modeling techniques used for TOGAF.

Additional elements online

The examples in this book were developed using the Modelio modeling tool, which provides the following useful features to support TOGAF modeling:

- UML and BPMN support
- Support of the "UML profile" extension mechanism
- Catalog and matrix generation
- Support of goal analysis and requirement analysis
- Traceability management

An open source version of the Modelio tool can be downloaded from the www.modelio.org web site. This version enables users to access the model database containing the examples presented in this book.[f]

5.8.3 Summary of the Appropriate Use of Modeling Techniques

Modeling is an investment, but to what end? Keep in mind the reasons for which you are investing in change.

- Goals, requirements, and domain

A model must have the agreement and consent of all those working on it.

- However, formalism can be an obstacle.

Modeling work includes establishing a consensus.

- Agreeing on goals and terminology sets the foundations for consensus on models.

Relevance is a model's most important quality.

- A model must correspond to the reality of the business and the company.
- Does a given model serve its purpose?
- What is the purpose of the model?

[f]The model database can be downloaded from www.togaf-modeling.org/togaf-en-pratique/.

5.9 FUNDAMENTAL CONCEPTS

The following fundamental concepts were introduced in this chapter:

- Model: Representation of a particular subject.
- Abstraction: Mechanisms that enable users to consider the system at more macroscopic levels.
- View: Representation of the system from the perspective of a set of issues (the viewpoint).
- Traceability: Ability to link artifacts produced by enterprise architecture and subsequent technical realization activities to other artifacts from which they originate or to which they refer.
- Modeling tool: Enables the realization of the goal of the model—to capture, communicate, validate, reason, and act upon shared information.