An Enterprise Engineering based Examination of TOGAF[1]

Jan L.G. Dietz, Jan A.P. Hoogervorst

Delft University of Technology,
j.l.g.dietz@tudelft.nl, j.a.p.hoogervorst@tudelft.nl,

Abstract. TOGAF (The Open Group Architecture Framework) is growingly considered to be the de facto standard way of working for the development and deployment of modern IT systems in enterprises. A major characteristic of modern IT systems, as opposed to the ones in the past, is that they are an integral part of the total enterprise, and effectively support some part of the enterprise's activities. Consequently, they must be developed with unity and integration in mind. Business development, organization development, and IT systems development cannot be addressed anymore as unrelated subjects. In this paper, the authors report on an investigation of TOGAF (version 9) regarding the extent to which it satisfies the indispensable requirement of unity and integration. The conclusion is that TOGAF fails to do this, as it fails to achieve several other ambitions. The main cause of these failures seems to be the lack of a sound and appropriate theory. In carrying out the investigation, the authors have based themselves on the enterprise engineering theory, as well as on extensive practical experiences.

Key words: TOGAF, Enterprise Architecture, Architecture Framework, Enterprise Engineering

1 Introduction

The track record regarding successfully implementing strategic initiatives is rather poor. Some publications speak about less than 10% success rate [Mintzberg 1994]. This rather low figure compares with other sources. According to Kaplan and Norton, many studies prove that between 70% and 90% of strategic initiatives fail, meaning that the expected result is not achieved [2004]. Studies concerning a specific strategic domain, such as total quality management, business process reengineering, customer

[1] This paper was accepted for publication in 2010 by the editor-in-chief of the Journal of Enterprise Architecture. Under pressure of the Open Group, it was rejected a few months later. A condensed version of the paper, titled "A critical investigation of TOGAF", is published in Lecture Notes on Business Information Processing (LNBIP) no. 79, Springer 2011

relationship management, or mergers and acquisitions, similarly report high failure rates. All too often, failures are conveniently attributed to unforeseen or uncontrollable external events. However, strategic failure is seldom the unavoidable result of an inadequate strategy, but mostly the avoidable consequence of inadequate operationalization of the strategic intent. A plethora of literature indicates that a (if not the) core reason for strategic failures is the lack of coherence and consistency among the various enterprise aspects, which precludes it to operate in a unified and integrated manner [Kotter 1995, Nadler and Tushman 1997, Pettigrew 1998, Doucet et al. 2009, Leinwand and Mainardi 2010]. We contend that a unified and integrated enterprise does not occur incidentally, but must be intentionally *designed*. A McKinsey publication confirmed this observation: rather than the traditional management focus on ad-hoc structural changes, acquisitions or competition, "they would be better off focusing on organizational design" [Bryan and Joyce 2007].

To ensure enterprise unity and integration many mutually related aspects have to be taken into account. In order to master this enormous task, many authors argue that the system approach is the only way to address the core problem effectively, hence to study and develop enterprises as systems [Bertalanffy 1969, Bunge 1979, Gharajedaghi 1999, Rechtin 2000]. Ackoff therefore argues that the high rate of failing strategic initiatives mentioned previously, is the consequence of the initiatives being fundamentally anti-systemic [Ackoff 1999].

The Open Group Architecture Framework (TOGAF, version 9) is positioned as an approach for developing enterprises and/or enterprise IT systems [The Open Group 2009a]. TOGAF correctly acknowledges that developing entails a design perspective. Therefore, we consider TOGAF to be a candidate methodology in developing enterprises. In this article we will investigate TOGAF in this respect, specifically assessing how the design perspective is operationalized in a systemic approach for ensuring enterprise unity and integration. The authors have based their analysis on the current enterprise engineering theory [Albani & Dietz 2010, Albani & Terlouw 2010, Aveiro et.al. 2010a, Aveiro et.al. 2010b, Barjis et. al. 2009, Dietz 2006, Dietz 2008, Hoogervorst 2009, Jong & Dietz 2010, Nuffel et. al. 2009, Ven & Verelst 2009], as well as on extensive practical experiences[2].

The article is structured as follows. In section 2 we summarize our observations and conclusions after having read the TOGAF book [The Open Group 2009a] pertinent to the clarity, coherence, and consistency of the concepts used, as well as the system kind that TOGAF aims to address. In section 3, the foundations are presented and discussed that are considered appropriate and imperative for a sensible investigation of the claims that TOGAF apparently makes. On these foundations we formulate, in section 4, seven assessment criteria and we assess TOGAF on each of these criteria. In section 5, we summarize the conclusions of our research.

---

[2] The reader is referred to www.ee-institute.com for further information.

2 Initial observations and analyses

Our first step in reflecting on TOGAF is considering the clarity, coherence and consistency of the concepts used. There is much more to say about this issue than is possible within the limited space of this article. We will restrict ourselves to reflecting on TOGAF's notions of architecture, framework and governance, as well as TOGAF's scope and the theory and methodology on which its method is based.

2.1 Architecture

Since TOGAF is positioned as an architecture framework, it is essential to understand what notion of architecture is adopted. This cannot easily be discerned. Initially, it is stated that "Architecture has two meanings depending on the context: (1) a formal description of a system, or a detailed plan of the system at component level to guide its implementation, and (2) the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time" [The Open Group 2009a, p.9]. This characterization is ambiguous, since two principally different views on architecture are contained in one definition, a situation not conducive to clarity since in various discussions about architecture it remains unclear which of the different views is intended. Moreover, when subsequently introducing or discussing specific aspect architectures, descriptions are used that do not satisfy either of the two views. For example, Business Architecture "defines the business strategy, governance, organization and key business processes" [op. cit. p.10]. However, when discussing the Business Architecture Phase, business architecture is seen as "describing the product and/or service strategy, and the organizational, functional, process, information, and geographical aspects of the business environment, based on the business principles, business goals, and strategy drivers" [op. cit. p.94]. Yet another definition considers business architecture as "the business strategy, governance, organization, and key business processes information, as well as the interaction between these concepts" [op. cit. p.27]. Incidentally, how are we to understand the idea of 'interaction between these concepts' as part of architecture?

These obvious differences regarding the notion of (business) architecture do not contribute to clarity. Furthermore, by including fundamentally different subjects within the scope of architecture this concept becomes so broad that it virtually looses its meaning. Similarly, vagueness about the notion of architecture is apparent when the creation of an architecture vision is discussed: the enterprise mission, vision, strategy, and goals are seen as key elements of architecture [op. cit. p.83]. Other examples of the sloppiness in trying to clarify the meaning of architecture can be easily found. As an illustration, Foundation Architecture is seen as "a set of re-usable common models, policy and governance definitions, or even as specific as overriding technology selections" [op. cit. p.52], while the same concept is elsewhere defined as "An

architecture of generic services and functions that provides a foundation on which more specific architectures and architectural components can be built. The TOGAF Foundation Architecture includes a Technical Reference Model" [op. cit. p.30].

The general definition of architecture given at the beginning of this section also speaks of architecture principles and guidelines governing system design. Elsewhere these two concepts are considered synonymous: "principles are general rules and guidelines" that provide the basis for decision making [op. cit. p.265]. IT principles thus regard decision making about the use and deployment of IT. Confusion arises when it is stated that "architecture principles are a subset of IT principles" [ibid.]. So, is TOGAF merely concerned with IT and IT architecture? How are we then to understand the already confusing notion of business architecture? Further vagueness is created by considering two categories of architecture principles: (1) "principles that govern the architecture process, affecting the development, maintenance, and use of enterprise architecture", and (2) "principles that govern the implementation of architecture, establishing the first tenets and related guidance for designing and developing information systems" [ibid.]. This is rather confusing. The first category of principles concerns the 'architecture process'. In view of TOGAF's ambiguity regarding the notion of architecture it is hard to understand what this process is about. Probably architecture is meant to be a design. So, the architecture process is an IT design process? The implementation of architecture (second category of principles) is then physically realizing a design. However, rather confusingly, the second category of principles is seen as a guidance for designing and developing information systems. Yet, in the section on architecture principles, principles are considered as general rules and guidelines that offer limited design guidance.

Next, the notion of enterprise architecture seems to be central in TOGAF, since the term is frequently used. Nonetheless, it is nowhere properly defined. At one point it is stated that "enterprise architecture provides a strategic, top-down view of an organization to enable executives, planners, architects and engineers to coherently coordinate, integrate, and conduct their activities" [op. cit. p.69]. The central importance of enterprise architecture is further emphasized by stating that "enterprise architecture potentially provides the context for all enterprise activities" [op. cit. p.73]. Also, there is a view whereby "enterprise architecture structures the business planning into an integrated framework that regards the enterprise as a system or system of systems" [ibid.]. Finally, yet another view on enterprise architecture holds that "Enterprise architecture defines principles, constraints, frameworks, patterns, and standards that form the basis of design governance, ensuring aligned services, interoperability and reuse" [op. cit. p.235]. Notably, the vagueness and ambiguity in the notion of architecture encountered earlier is similarly present in the notion of enterprise architecture. Many aspects are included in this concept. Since, as we saw above, architecture principles are a subset of IT principles, are we thus to understand enterprise architecture as IT architecture for the whole enterprise? This might be an obvious interpretation given the origin of TOGAF as an information technology framework. This conclusion

is corroborated by our discussion of the scope of TOGAF in Section 2.4. But then, TOGAF's ambition should be far more modest.

2.2 Framework

TOGAF defines a framework as "a detailed method and a set of supporting tools for developing enterprise architecture" [op. cit. p.3]. However, in Chapter 3 (Definitions) a framework is defined as "a structure for content or process that can be used as a tool to structure thinking, ensuring consistency and completeness" [op. cit. p.30]. Also, an architecture framework is considered as "A foundational structure, or a set of structures, which can be used for developing a broad range of architectures. It should describe a method for designing a target state of the enterprise in terms of a set of building blocks, and for showing how the building blocks fit together. It should contain a set of tools and provide a common vocabulary. It should also include a list of standards and compliant products that can be used to implement the building blocks" [op. cit. p.7]. Is "developing a broad range of architectures" similar to "designing a target state of the enterprise"? Are we to understand an architecture framework as a design method, as well as a method for implementation?

When discussing the Architecture Development Method (ADM), it appears that the ADM is, rather surprisingly, also concerned with developing an architecture framework [op. cit. p.10]. The confusing idea that an architecture framework itself is concerned with the development of a framework is expressed at multiple occasions. For example, "enterprise architecture structures the business planning into an integrated framework that regards the enterprise as a system or system of systems" [op. cit. p.73]. In view of the elusive enterprise architecture concept, it is difficult to envision what "an integrated framework" is supposed to be.

2.3 Governance

From a general perspective, TOGAF defines governance as "the discipline of monitoring, managing, and steering a business (or IS/IT landscape) to deliver the business outcome required" [op. cit. p.31]. The term "business outcome" is elusive but seems to give governance a somewhat operational flavour. Rather remarkably, in some cases governance appears to be the result of establishing architecture. For example, among other things, business architecture defines governance [op. cit. p.10]. In the ADM, the implementation of governance is nonetheless addressed in a separate phase after having defined various architecture types [op. cit. p.185]. This phase of the ADM seems more concerned with program/project management than with governance. Furthermore, one would expect that governance is already required for establishing the various architectures types. Indeed, elsewhere a notion of governance is introduced that seems to suggest that governance must precede the creation of architectures: "Archi-

tecture governance is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level. It is concerned with change processes (design governance) and operation of product systems (operational governance)" [op. cit. p. 25]. Also here, operational aspects are included in the governance perspective. Indeed, the notion of governance is made ambiguous by including operational aspects such as SLA's and OLA's in the governance perspective [op. cit. p.676].

TOGAF mentions four governance types: Corporate Governance, Technology Governance, IT Governance and Architecture Governance [op. cit. p.671]. Corporate governance is considered "a board topic, beyond the scope of an enterprise architecture framework such as TOGAF" [ibid.]. One might argue nonetheless that corporate governance has much to do with the transparency and quality of financial/ administrative data and the compliance with pertinent rules and regulations. All these aspects are largely based on proper enterprise-wide design, including business processes and information systems, and hence should be in the scope of an enterprise-wide governance perspective. However, TOGAF offers no overall governance perspective to address these aspects in a unified and integrated manner.

Notably, the ADM also considers the development of technology architecture. Since TOGAF's notion of architecture is rather broad, it is difficult to grasp how the difference between architecture governance on the one hand, and IT governance on the other hand must be understood.

2.4 Scope

As was indicated previously, TOGAF's view on architecture speaks of "(1) a formal description of a system, or detailed plan of the system at component level to guide its implementation, and (2) the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time" [op. cit. p.9]. This begs the question of the system kind that TOGAF is concerned with. That question cannot easily be answered. It is stated that "the enterprise architect applying TOGAF cannot narrowly focus on the IT implementation, but must be aware of the impact that the architecture has on the entire enterprise" [op. cit. p.72]. This statement expresses an IT-driven focus, be it with awareness of the IT impact. It suggests that the systems of concern are information technology systems. Somewhat later we learn that "enterprise architecture potentially provides the context for all enterprise activities" and "enterprise architecture structures the business planning into an integrated framework that regards the enterprise as a system or system of systems" [op. cit. p.73]. Does that suggest a focus on enterprise-wide design, so the system type – hence the scope of TOGAF – concerns the enterprise as a whole? This appears however not to be the case when looking at the role and skills of the enterprise architect, which basically centers around IT: "An enterprise architect should possess an extensive

technical breadth through experience in the IT industry. This breadth should be in areas of application development and deployment, and in the areas of creation and maintenance of the infrastructure to support the complex application environment" [op. cit. p.702]. This would suggest that TOGAF's scope only concerns information systems. Again, what message is the adjective "enterprise" supposed to convey?

2.5 Foundations

The architecture framework of TOGAF aims at providing a method for developing enterprise architecture [op. cit. p.3], or a method for defining the deliverables of the architecturing activity [op. cit. p.18]. Well then, a method is a way of working to accomplish a task. Ideally, it is part of a methodology: a collection of methods based on a philosophically coherent collection of theories related to a particular discipline. Also in view of our earlier comments, it remains totally unclear on which theory and methodology the TOGAF method is based.

Based on our reflection in the above, we cannot escape the impression that TOGAF is a conceptual wilderness. Often definitions of the concepts used are vague, imprecise, and ambiguous, or so broad that they loose meaning. Aspects that should be formally distinguished are mixed up, thereby contributing to their vagueness and ambiguity. Definitions are also not consistent throughout the book. Precise definitions of key notions like architecture, framework and governance are lacking. The central notion of enterprise architecture is nowhere properly defined. TOGAF's scope and underlying theory and methodology remain elusive.

3 The theoretical basis for enterprise design

As the basis for assessing TOGAF in a profound and thorough way we will present and discuss in this section a theoretical outline regarding the development of systems, including enterprises. The basic paradigm we adopt is that enterprises are purposefully designed systems, in accordance with the Enterprise Engineering Manifesto[3]. We consider this outline suitable for assessing TOGAF since it covers all issues that are dealt with by TOGAF.

3.1 The notion of system

Various system definitions exist. Jackson sees a system as "a complex whole the functioning of which depends on its parts and the interaction between these parts" [2003,

[3] http://www.ciaonetwork.org/publications/EEManifesto.pdf

p.3]. Maier and Rechtin define a system as “a set of different elements so connected or related as to perform a unique function not performable by the elements alone” [2002]. Von Bertalanffy speaks of “a set of elements standing in interrelation among themselves and with the environment” [1969, p.252]. It appears that there are basically two kinds of system notions: teleological and ontological [Dietz 2006]. The *teleological* system notion is concerned with the function or purpose of the system. This notion is suited and perfectly adequate for using or controlling a system (whereas the ontological notion is not). The associated kind of model is the *black-box model*. Actually, the teleological system notion is equal to a black-box model of the system. This can easily be understood if one recognizes that purpose (or function) is not a system property but a relationship between a system and a stakeholder. For example, every car driver has a black-box model of the behavior of the car he or she drives. Black-box models can be decomposed by means of functional decomposition. Usually, a car driver also has some functional decomposition in his or her mind. It is important to note that black-box models are fundamentally subjective.

The *ontological* system notion regards the construction and operation of a system. It is independent of and therefore indifferent to the function or purpose that one assigns to the system. This notion is suited and perfectly adequate for building and changing systems (whereas the teleological notion is not). The associated kind of model is the *white-box model*. White-box models can be decomposed by means of constructional decomposition. Note that there is only one constructional decomposition of a system.

Based on the rigorous systemic ontology of Mario Bunge [Bunge 1979] we apply the next ontological notion of system [Dietz 2006]. A system is a tuple <C,E,P,S>, where:

C : *composition*: a set of system elements of one and the same system category.

E : *environment*: a set of elements of the same category as the elements of *C*.

P : *production*: the products or services that *C* delivers to *E*.

S : *structure*: the mutually influencing bonds among the elements of *C*, and between the elements of *C* and the elements of *E*.

The concept of system kind or category is crucial for deeply understanding the ontological system notion. Many system categories can be identified, for example biological, chemical, electrical and social. Note that the system definition provided above defines a homogeneous system. The most interesting systems, however, are heterogeneous systems, like cars and enterprises. They are constructions of homogenous systems. The unification and integration of a number of homogenous systems into a heterogeneous system is by no means trivial. That is why unity and integration are core concepts within the system approach [Gharajedaghi 1999].

3.2 The notion of architecture and architecture framework

So, the systems of our interest are designed systems, like IT systems and enterprises. The design of these systems is by its very nature not 'incidental'. Instead it is a goal-directed process in which the designer needs guidance. One explicit kind of guidance is provided by the system requirements, both the functional requirements and the constructional ones. However, in general these requirements keep an amount of design freedom left that the designer has to cope with somehow. He needs additional, normative, guidelines. Often such guidelines are kept implicit, even to the extent that designers are not aware of them. We fully agree in this respect with Ulrich's critical system heuristics, arguing that the normative aspects of system design must be made explicit [In: Jackson 2003]. Architecture provides the answer to this question, as it has done since time immemorial in constructional engineering.

Conceptually, architecture thus can be defined as the normative restriction of design freedom [Dietz 2009]. Practically, it consists of *a coherent and consistent set of principles and standards that guide system design* [Dietz 2009, Hoogervorst 2009]. Since architecture guides system design, devising architecture should take place pertinent to relevant system aspects as presented by an architecture framework. We conceive an architecture framework thus as a conceptual structure for architecturing, i.e. for devising architectures. Three dimensions play a role in the process of architecturing [Dietz 2009]:

System kinds, which we will identify by *S*
Design domains, labeled as *D*
Areas of concern, labeled as *A*

These three dimensions seem to be necessary and sufficient. Examples of system kinds are organizations, information systems, and IT systems. The second dimension is the *design domain* to which a design principle belongs. Examples of principles are: 'process control must be separated from process execution', or 'network access must be based on authentication and role-based authorization'. Building on the systemic foundations, as discussed in Section 3.1, we distinguish between two domains on the highest level of detail: function and construction. The third dimension is the *area of concern* that a design principle addresses. Whereas the first two dimensions (system kind and design domain) are timeless, the third one is not. It reflects the, generally time-dependent, focal interests of stakeholders. Examples of areas of concern are security, compliance, privacy, agility, and user-friendliness. They serve to organize the total set of design principles of an architecture into subsets that correspond to the interests of the distinct stakeholders. These subsets need not be disjoint. Put differently, areas of concern may overlap. It means that a particular design principle regards several interests and therefore addresses several areas of concern.

An architecture framework can thus be presented symbolically as a triplet <*S*,*D*,*A*>. It is a conceptual structure related to one or more system kinds, a number of areas of concern, and a necessary and sufficient set of design domains [Hoogervorst 2009]. As said, the highest level distinction between design domains concerns the distinction between the system *function* and the system *construction*. Within this distinction, further detailing is evidently required; it refers to the specialization of design domains associated with more detailed observations. Such specialization thus creates a certain order whereby a more detailed design domain is subordinated under the next higher design domain, in the way that, for example, the design domain 'engine' is subordinated under the overall design domain 'car', and 'piston' in turn is subordinated under the domain 'engine'. This is an important condition for safeguarding coherence and consistency, which has been emphasized previously as an important objective of defining architecture. Establishing the necessary and sufficient (complete) set of design domains can be a daunting task for complex systems, such as enterprises. Knowledge and experience of the architect concerning the system kind in question is obviously crucial.

3.3 The Generic System Development Process

Figure 1 exhibits the so-called Generic System Development Process (GSDP) [Dietz, 2008], which applies to the development of systems of any kind. Taking the function perspective, one `sees' the function and the (external) behaviour of a system; the corresponding type of model is the black-box model. Taking the construction perspective, one `sees' the construction and the operation of a system; the corresponding type of model is the white-box model. In developing a system both the function perspective and the construction perspective are relevant. The GSDP identifies the most basic steps in a development process.

The starting point is the need by some system, called the *using system* (US), for a supporting system, called the *object system* (OS). A clear distinction between the US and the OS is often neglected, leading to blurred discussions about the functionality of the OS. In the GSDP, the US is the stable starting point for the development process. One must have an appropriate understanding of the US in order to successfully design an OS. By nature, this understanding must be constructional understanding, since it is the construction of the US that is going to be supported by the function of the OS. So one starts with conceiving a white-box model of the US. Preferably, this model is an ontological model [Dietz, 2006]; otherwise one can easily become confused by irrelevant implementation issues of the US. From the white-box model of the US one determines the functional requirements for the OS (*function design*). These requirements are by nature formulated in terms of the construction and operation of the US. Moreover, and consequently, they need to be fully independent of the construction of

the OS. The next basic design step is to devise specifications for the construction and operation of the OS, in terms of a white-box model of the OS (*construction design*).

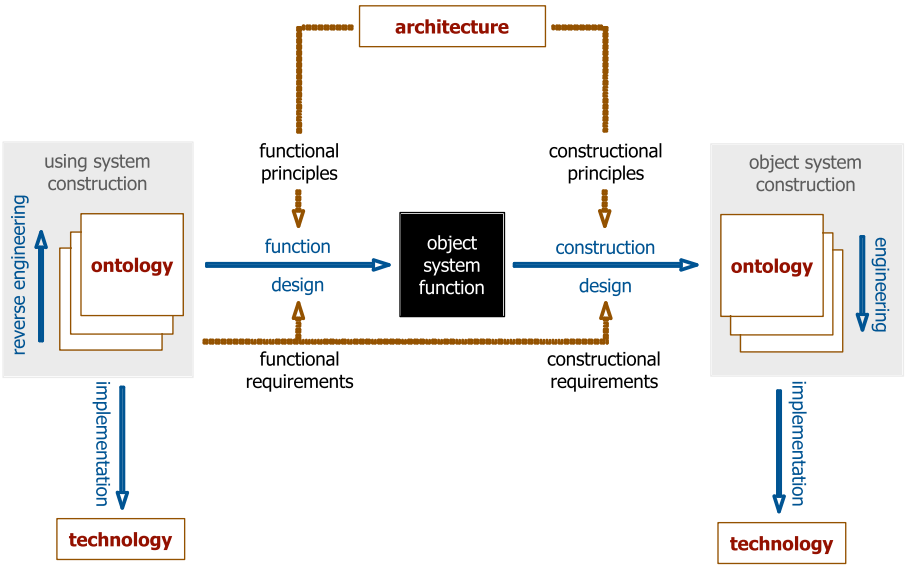

Figure 1 The Generic System Development Process

For this design phase, the US may provide constructional requirements, often also called non-functional requirements. They mostly regard performance and quality aspects. A thorough analysis of the white-box model of the OS must guarantee that building the OS is feasible, given the available technology. The GSDP also includes the important experience from practice that designing is an iterative process: the final result of every design process is (or should be) a balanced compromise between reasonable functional requirements and feasible constructional specifications [Dietz, Albani 2005]. For the sake of simplicity, however, we left it out from the figure.

In addition to the functional and constructional requirements, there may be functional and constructional principles respectively. These design principles are the operational shape of the notion of architecture, as discussed in Section 3.2. They generally hold for a class of systems. An example of a functional principle is that man-machine dialogs must comply with some standard. An example of a constructional principle is that the applications must be component-based. Ideally the construction design phase results first into an ontological model of the OS, i.e. a white-box model that is completely independent of its implementation. Gradually this ontological model is transformed into more detailed (and more implementation dependent) white-box models, the last one being the implementation model. This process is called implementation design or just engineering. If the OS is a software application, then the

implementation model would be the source code in some programming language. The act of implementing consists of assigning appropriate technological means to the implementation model, e.g. running the source code on an appropriate platform.

3.4 Enterprise and Enterprise Architecture

Building on the theoretical foundations that have been discussed in Sections 3.1 thru 3.3, we will now provide precise and consistent definitions of the notions of enterprise and of enterprise architecture. Enterprises are goal-directed social entities that are designed as deliberately structured activity systems linked to the external environment [Daft 2001]. All enterprises face the fundamental issue of indispensable *differentiation* (the creation of specific tasks) on the one hand, and establishing *unity and integration* (the realization of coherence and consistency in task execution) on the other hand [Lawrence Lorsch 1967]. Certain cooperative interaction patterns must therefore necessarily exist between human actors for collectively realizing the enterprise purpose and function. Enterprise performance is thus ultimately the result of social interaction between human actors who enter into, and comply with, commitments concerning the execution of tasks. Obviously, an enterprise design theory and methodology must address aforementioned issue effectively. For that we adopt the so-called Ψ(psi)-theory: Performance through Social Interaction [Dietz 2006], which is grounded in the view that the collaborative interaction patterns between human actors in enterprises are constituted by coordination activities, which are based on mutual communication. These notions form the basis for the so-called speech/act theory, or the language/action perspective on the design of cooperative work [Winograd and Flores 1987]. Thus language is seen "as the primary dimension of human cooperative activity" [Winograd 1988]. Within this perspective, the focus is on communicative patterns that constitute the mutual coordination, since people act through language.

Individuals within enterprises fulfill actor roles (manifesting the differentiation in task execution), whereby basically two types of activities are performed: (1) *production activities* and (2) *coordination activities* [Dietz 2006]. Production activities can yield a material or immaterial result. Material production has to do with manufacturing, storage or the transport of goods for example. Immaterial production concerns decision-making, granting something, sentencing a person by a judge, appointing a person in a function and so on. Coordination activities concern the communicative actions mentioned above pertinent to entering into and complying with commitments about production activities. Coordination activities are therefore always linked to production activities. The enterprise is then seen as a "network of commitments" [Winograd and Flores 1987, p.150]. Coordination and production activities come in universal patterns, called transactions. An enterprise process is thus a structure of causally related transactions.

Based on the general systems development process discussed before, the first step in designing an enterprise is establishing the implementation-independent (ontological) models, which are based on, and reflect, the essential transactions of the enterprise [Dietz 2006]. Discussing these models falls outside our current scope. For now we like to stress that the implementation-independent nature of the models substantially reduces the complexity, hence the effort to comprehend the enterprise. Moreover, the models precisely depict the essential communicative – hence collaborative – patterns between the different actors in the enterprise that collectively depict the whole essential enterprise operation. The issue of differentiation on the one hand, and unity and integration on the other, is thereby formally addressed at the essential, implementation-independent level. Additionally, the explicit modeling of the coordination activities of a transaction allows the precise definition of operational rules guiding these activities.

After having defined the implementation-independent models, further design of the enterprise system (*S*) needs to take place in order to devise construction models that can be implemented. Design takes place in various design domains (*D*), guided by architecture for addressing areas of concern (*A*) and further ensuring unity and integration. As indicated previously, there are basically two main categories of design domains: *functional* and *constructional* ones. Various labels are used in the literature to identify areas where (some form of) design should take place, such as business, information, data, application, infrastructure, or technology. The distinction between function and construction is virtually never made explicitly. With reference to aforementioned distinction, we will identify design domains comprehensively enterprise-wide. Subsequent design pertinent to these design domains (guided architecture) transforms or complements the implementation-independent ontological models in construction models that ultimately can be implemented. We propose four main enterprise design domains [Hoogervorst 2009]:

Business. This design domain concerns the enterprise *function*, having to do with: (1) the elements of the enterprise environment, such as customers, suppliers, business partners, or stakeholders, (2) the products and services the enterprise delivers to its environment, and (3) the relationships between them, like sales and communication channels. In fact, all these topics concern sub (functional) design domains within the main business design domain.

Organization. The organization design domain concerns the internal arrangement of the enterprise for delivering the enterprise's function. It includes sub design domains like processes, employee behavior, enterprise culture, management/leadership practices, and various structures and systems, such as concerning accounting, purchasing, payment, or employee remuneration and evaluation. These are all topics that must be formally brought within the enterprise design perspective [Luthans 1992, Daft 2001, Schein 2004]

With the label 'organization' the main enterprise *construction* design domain can be identified. In view of information as a crucial production factor, rather than seeing

the design domain ‘information’ as part of the design domain ‘organization’ we will consider the design domain ‘information’ as a separate main constructional design domain. Similar considerations hold for the design domain ‘(information) technology’.

Information. Many informational aspects play a role, such as the structure and quality of information, the management of information (gathering, storage, distribution), and the utilization of information. These topics are examples of sub information design domains. As indicated, also the information design domain has to do with the enterprise *construction*.

Technology. Appreciably, technology is essential for organizational and informational support, hence an import aspect of the enterprise *construction*. Within the scope of this paper we restrict ourselves to information technology. This design domain is concerned with sub domains like the IT network, application, storage, data communication, etc.

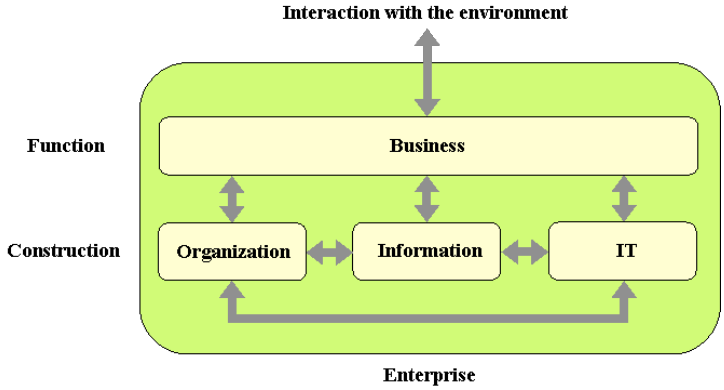

Figure 2 Main enterprise design domains

In view of the above, the enterprise construction is defined by three aspect systems: organization, information, and IT, hence three main construction design domains. Together with the business functional design domain, we have the four main enterprise design domains that are schematically depicted in figure 2. The arrows between the design domains express mutual relationships that must be addressed for establishing a unified and integrated enterprise. For all main enterprise design domains (*D*) (and their sub design domains) architecture must be defined that (1) adequately serves as design guidance for ensuring unity and integration, and (2) addresses one or more areas of concern (*A*). Ample examples are provided in [Hoogervorst 2009]. Restricting ourselves to the four main design domains, four main architectures can be identified:

Business architecture: the function architecture of the enterprise. The business architecture can formally be defined as a coherent and consistent set of principles and standards that guide the design of the enterprise's function. We might consider principles concerning the provisioning of products and services to customers, the market position relative to competitors or the relationship with stakeholders.

Organization architecture: an aspect of the enterprise construction architecture. The organization architecture is defined as a coherent and consistent set of principles and standards that guide the design of the enterprise's organization for providing the enterprise products and services. Organization architecture for example concerns the design of processes, but also the design of accounting, purchasing, payment, or employee remuneration and evaluation systems. This design perspective is thus necessarily broad. So, establishing desired employee and management behavior, or certain norms and values – all have to do with organizational design.

Information architecture: viewed as a coherent and consistent set of principles and standards that guide the design of an enterprise's information systems. Principles and standards, for example, concern the structure and quality of information, the management of information (gathering, storage, distribution), and the utilization of information. Information architecture is also a sub-architecture of the enterprise construction architecture. Notably, information architecture differs from IT architecture because it is technology independent.

IT architecture: the third sub-architecture of the enterprise construction architecture, defined as a coherent and consistent set of principles and standards that guide the design of IT systems. Principles and standards in this sub-architecture concern all technology dependent IT aspects like network, application, storage, middleware, and so on.

Notably, more detail is required to carry out design activities within the main design domains. More specific sub design domains thus need to be considered, like the sub design domain 'processes' within the main design domain 'organization', as mentioned before. So, process architecture can be seen as a sub-architecture of organization architecture. As indicated before, this points to a hierarchy of design domains and associated architectures: the enterprise at large, the four main design domains, and within these main domains further sub design domains. The latter we have discussed elsewhere [Hoogervorst 2009]. The totality of these architectures is enterprise architecture, which is defined as a coherent and consistent set of principles and standards for the design of the enterprise as a whole. Within the sub design domains, the function/construction perspective likewise holds for the relevant subsystems.

4 Assessing TOGAF

In this section, we will assess TOGAF on the theoretical basis as presented in Section 3, taking into consideration the initial observations and analysis in Section 2. All aspects of TOGAF will be addressed except governance. In our view, governance is not part of an architecture framework or a development methodology. We view enterprise (IT) governance as an organizational competence for continuously exercising guiding authority over enterprise (IT) strategy and architecture development and the subsequent design and implementation of the enterprise (IT system). This organizational competence must be established for successfully applying the enterprise (IT) development methodology. Moreover, addressing governance in depth would exceed the limited space of this article. From the theoretical basis as presented in Section 3 we derive the seven concrete assessment criteria listed below. After the formulation of each criterion (in italics) we discuss the extent to which it is satisfied by TOGAF.

In order to successfully operationalize strategic changes and ensure enterprise performance, an enterprise must operate as a unified and integrated whole. Unity and integration are not achieved 'incidentally' but must be intentionally created through enterprise design.

Integrated and unified enterprise design seems to be the only way towards mastering the 'organized complexity' of an enterprise [Weinberg 2001]. Assessing TOGAF on this criterion is complicated by the fact that it remains unclear what the objective of TOGAF precisely is. In our discussion in Section 2 we argued that TOGAF's notion of enterprise architecture is ambiguous. The descriptive view on architecture (meaning (1) in [The Open Group 2009a, p.9]) is predominant. Indicative of this conclusion is the frequent use of terms like "architecture design" and "architecture description". Also 'The Open Group Business Architect Certification Program' discusses architecture in the descriptive sense: in terms of the result of design activities [The Open Group 2009b]. Thus we consider the TOGAF notion of enterprise architecture as some form of a design. Hence, by applying TOGAF, unified and integrated enterprise design can be established, in principle. However, no formal theory and associated methodology is offered for addressing the organized complexity of enterprises and for guaranteeing unified and integrated enterprise design. Although TOGAF purports to be holistic [The Open Group 2009a, p.43] it remains unclear how the holistic view is effectuated. Moreover, effectively establishing enterprise unity and integration requires a scope far exceeding mere IT and process integration [Peters and Waterman 1982, Scott Morton 1991, Doz and Thanheiser 1993, Miles et al. 1995, Martin 1995, Hoogervorst 1998, Daft 2001].

The essential nature of enterprises is their being social systems. Enterprises consist of human beings (actors) who enter into and comply with commitments. In doing so, they collectively realize the enterprise function, and determine enterprise performance.

Given the technology-focused origin of TOGAF, we feel the essential nature of enterprises is not captured by merely extending the IT focus to the 'business domain' in view of the fact that IT supports business processes. Human beings constitute the core of enterprise success and this important aspect appears not to be addressed by TOGAF. As Drucker has pointed out, social aspects of organizing should be the central focus of organizational and management science [Drucker 1985]. Ultimately, enterprise performance is determined by the enterprise's social system [McGregor 1960, Hoogervorst 1998]. This essential aspect must be acknowledged and addressed in any enterprise design methodology.

An important first step in designing an enterprise is capturing its implementation independent essence, also known as enterprise ontology. The ontological model of an enterprise comprises the identified transactions and the corresponding actor roles (chunks of authority and responsibility).

The notion of an implementation independent (ontological) model is totally absent in TOGAF. At the same time, in order to master the organized complexity of enterprises, one needs a perspective that allows one to initially reduce the complexity of enterprises substantially by focusing on the implementation independent essence of the enterprise, i.e. on the enterprise ontology. In doing so, the previously mentioned notion that enterprises are social systems in which human beings enter into, and comply with, commitments is formally operationalized.

Two distinct perspectives are paramount in any design-oriented study of enterprises: function and construction. The function perspective concerns the function and behavior of the enterprise vis-à-vis its environment. The construction perspective concerns the construction and operation of the enterprise.

TOGAF does not seem to offer these perspectives because it lacks an appropriate system notion. Consequently, the system kind remains unclear. As indicated earlier, we consider the TOGAF notion of enterprise architecture as some form of a design. This suggests the enterprise as the system kind of observation. Nonetheless, this perspective is not thoroughly operationalized through functional and constructional enterprise design. It might however very well be that TOGAF's notion of enterprise architecture is not about comprehensive enterprise-wide design. Instead, given the origin of TOGAF, the focus might be on the design of the enterprise 'IT system', which we consider to be the totality of IT subsystems for delivering IT services to the enterprise. But then, the construction perspective on the enterprise as the using system be-

comes imperative in order to meaningfully determine the IT systems' functions. Only within this perspective the issue of 'business and IT alignment' can be addressed effectively. That brings us back to the enterprise-wide design perspective, which, as pointed out before, seems not to be fully explored by TOGAF.

Enterprise design comprises both function design and construction design. The main enterprise design domains are: business (function), and organization, information, and IT (construction). Through subsequent design, the implementation independent enterprise ontological models are complemented by a sequence of ever more detailed construction models of which the 'lowest' one is implementable.

Creating enterprise unity and integration necessitates comprehensive enterprise-wide functional and constructional design. Despite TOGAF's frequent reference to the notions of 'enterprise' and 'design' no approach for enterprise-wide functional and constructional design is provided. Comprehensive design domains that cover the enterprise in its totality are not indicated. The multiple aspects, as known from the traditional organizational literature, that determine enterprise performance, are thus not or inadequately addressed.

Design pertinent to these domains is respectively guided by business architecture, organization architecture, information architecture, and IT architecture. Collectively they constitute enterprise architecture, where architecture is conceptually defined as the normative restriction of design freedom, and practically as a coherent and consistent set of standards and design principles.

Comprehensive enterprise-wide design entails design guidance, such that unity and integration is achieved and areas of concern are adequately addressed. As indicated in Section 2, TOGAF uses a vague and ambiguous notion of architecture. Based on the actual usage of the term "architecture", this concept must in fact be understood as a result of design activities, rather than a concept providing design guidance. TOGAF's section on architecture principles rightly so addresses their importance. However, the focus seems to be limited to IT: "Architecture principles define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise" [The Open Group 2009a, p.266]. The general, not design-specific character is reflected in the examples given [ibid.]. Enterprise-wide design guidance in the form of normative, prescriptive architecture is arguably not addressed as a central point of attention.

Even if we take into account TOGAF's focus on the enterprise IT system, a comprehensive set of IT system (functional and constructional) design domains must be defined. For these domains architectures must be devised – collectively referred to as IT architecture – that addresses areas of concern and ensure a unified and integrated enterprise IT system design. Pertinent to the enterprise IT system, TOGAF speaks about data, application, and technology architecture. However, not in the normative,

prescriptive sense. According to TOGAF, application architecture "provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization" [op. cit. p.10]. This viewpoint shows TOGAF's lack of a formal distinction between system kinds (cf. Section 3.2) since business processes cannot be addressed within the IT system perspective, but only within the enterprise-wide system perspective. By the same reasoning, the functional IT design domains cannot be addressed within the IT system scope but only within the white-box scope of the enterprise as the using system.

Indeed, TOGAF does introduce the notion of 'business architecture', but in an inappropriate way, as we have seen.

Devising an architecture (called architecturing) must be performed within an architecture framework. Consequently, it regards particular system kinds, design domains, and areas of concern.

As outlined in Section 3, devising an architecture must take place with reference to (1) the system kind *S* of observation, and for which architecture provides design guidance, (2) the design domains *D* were the architecture is used, and (3) the areas of concern *A* that the architecture addresses. We feel that TOGAF lacks a formal distinction between architecturing (devising design principles and standards) and designing, as well as between designing and implementing. TOGAF refers to design activities in phases that should concern implementation only. Implementation and project planning aspects are obviously important topics but, in our view, they should be clearly distinguished from architecturing and designing, hence, should not be part of an architecture framework.

5 Conclusions

Enterprises are (1) goal directed, (2) purposefully designed social systems, that (3) interact with their environments, and for which (4) their unified and integrated operation is conditional for enterprise performance and the successful operationalization of strategic intentions. These four aspects must be central to any design approach for enterprises. Over the years, The Open Group has contributed considerable efforts to devising approaches – collectively identified under the TOGAF label – for enterprise and/or IT system design. Although these efforts can obviously be appreciated, our analysis forces us to conclude that the TOGAF approach is not grounded in formal, theory-based concepts such that the essential aspects of enterprises are effectively and comprehensively addressed. Moreover, the concepts used are largely ambiguous and ill-defined, thereby making it difficult to master the organized complexity of enterprises. We fail to see how conceptual incoherence and inconsistency can methodically bring forward a coherent and consistent whole. Obvious improvement areas are the

clarity, coherence and consistency of the concepts used, and addressing the perspective outlined in section 4.

A major objection one could have is that we have applied our own evaluation framework for assessing TOGAF. To our defense, we can only argue that we have not found another framework that suits our objectives. An often suggested candidate is the GERAM framework [GERAM 1999], since it is intended to facilitate the unification of methods for enterprise integration. However, GERAM offers no theory and associated methodology to accomplish the task. Admittedly, TOGAF acknowledges the importance of unity and integration for proper enterprise and IT systems performance, but it fails to make this operational. Moreover, elements of our framework have been advocated by others too, as referenced in this article.

Arguably, society is largely a society of enterprises. Put differently, societal well-being is to a considerable extent determined by the performance and conduct of enterprises. Adequate performance and conduct (or the opposite) thus have far reaching consequences. For such adequacy, proper enterprise design is crucial. TOGAF aims to address this issue, but fails to do so. As said before, the major cause of this is the lack of a coherent underlying theory. We cannot continue to deal with the immense problems we are facing in an unprofessional way. It is time to practice a profession that is built on sound theoretical foundations, along the lines we have sketched in this paper. Hopefully, our reflection contributes to raise awareness of TOGAF's shortcomings and to rebuild it on solid grounds. This is vitally important in view of the crucial role of enterprises for customers, employees, stakeholders, and the society at large.

References

Ackoff, R.L., *Ackoff's Best: His Classic Writings on Management*, New York, Wiley 1999

Albani, A., Dietz, J.L.G.: Enterprise Ontology based Development of Information Systems, in: *International Journal for Internet and Enterprise Management*, vol. 7, no. 1, 2010

Albani, A., Terlouw, L., An Enterprise Ontology-Based Approach to Service Specification, in: *IEEE Transactions on Services Computing*, Oct-Dec 2010

Aveiro, D., Rito Silva, A., Tribolet, J., Extending the Design and Engineering Methodology for Organizations with the Generation Operationalization and Discontinuation Organization, in: Winter, R. et. al., *Global Perspectives on Design Science Research*, LNCS 6105, Springer 2010

Aveiro, D., Rito Silva, A., Tribolet, J., Towards a G.O.D. Organization for Organizational Self-Awareness, in: Albani, A. et. al. (eds.*), Advances in Enterprise Engineering IV*, LNBIP 49, Springer 2010

Barjis, J., Kolfschoten, G.L>, Verbraeck, A., Collaborative Enterprise Modeling, in: Proper et. al. (eds.), *Advances in Enterprise Engineering II*, LNBIP 28, Springer 2009

Bertalanffy, L. von, *General Systems Theory*, New York, George Braziller 1969

Bryan, L.L., Joyce, C.I., Better strategy through organizational design, *McKinsey Quarterly*, No. 2, 2007

Bunge, M., *Treatise on Basic Philosophy Volume 4: A World of Systems*, Dordrecht, D. Reidel Publishing Company 1979

Daft, R.L., *Organization Theory and Design*, Mason, South-Western Publishing, 2001.

Dietz, J.L.G., Albani, A., Basic notions regarding business processes and supporting information systems, in: *Requirements Engineering*, vol 10 no 3, November 2005, pp 175-183

Dietz, J.L.G., *Enterprise Ontology*, Berlin, Springer, 2006.

Dietz, J.L.G., *Architecture: Building Strategy into Design*, The Hague, SDU Publishing 2009

Doucet, G., Gøtze, J., Saha, P., Bernard, S., *Coherence Management: Architecting the Enterprise for Alignment, Agility and Assurance*, Bloomongton, AuthorHouse 2009

Doz, Y., Thanheiser, H., Regaining Competitiveness: A Process of Organizational Renewal, In: Hendry, J., Johnson, G., Newton, J., *Strategic Thinking: Leadership and the Management of Change*, Chichester, Wiley 1993

Drucker, P., *Management*, New York, Harper 1985

GERAM, *Generalized Enterprise Reference Architecture and Methodology*, Version 1.6.3, IFIP-IFAC Taskforce 1999

Gharajedaghi, J., *Systems Thinking*, Boston, Butterworth Heinemann 1999

Hoogervorst, J.A.P., Quality and Customer Oriented Behavior: Towards a Coherent Approach for Improvement, Delft, Eburon 1998

Hoogervorst, J.A.P., Enterprise Governance and Enterprise Engineering, Berlin, Springer 2009

Jackson, M.C., *Systems Thinking*, Chichester, Wiley 2003

Jong, J. de, Dietz, J.L.G., Understanding the Realization of Organizations, in: Albani, A. et. al. (eds.), Advances in Enterprise Engineering IV, LNBIP 49, Springer 2010

Kaplan, R.S. en D.P. Norton, *Strategy Maps*, Boston, Harvard Business School Press, 2004

Kotter, J.P., Leading Change: Why Transformation Efforts Fail, *Harvard Business Review*, Vol. 71, No. 2, 1995, pp. 59-67

Lawrence, P.,Lorsch, J., *Organization and Environment*, Boston, Harvard Business School Press 1967

Leinwand, P., Mainardi, C., *The Coherence Premium*, Harvard Business Review, June 2010

Luthans, F., *Organizational Behavior*, New York, McGraw-Hill 1992

Maier, M.W., Rechtin, E., *The Art of Systems Architecting*, Boca Renton, CRC Press 2002

Martin, J., *The Great Transition. Using the Seven Principles of Enterprise Engineering to Align People, Technology and, Strategy*, New York, American Management Association 1995
McGregor, D.M., *The Human Side of Enterprise*, New York, McGraw-Hill 1960
Miles, R.E., Coleman, H.J., Douglas Creed, W.E., Keys to Success in Corporate Redesign, *California Management Review*, Vol. 37, No. 3, 1995, pp. 128-145
Mintzberg, H., *The Rise and Fall of Strategic Planning*, New York, The Free Press, 1994
Nadler, D.A., Tushman, M.L., *Competing by Design: The Power of Organizational Architecture*, New York, Oxford University Press 1997
Nuffel, D. van, Mulder, H., Kervel, S. van, Enhancing the Formal Foundations of BPMN by Enterprise Ontology, in: Albani, A. et. al. (eds.), *Advances in Enterprise Engineering III*, LNBIP 34, Springer 2009
Peters, T.J., Waterman, R.H., *In Search of Excellence*, New York, Warner Books 1982
Pettigrew, A., *Success and Failure in Corporate Transformation Initiatives*, In: Galliers, R.D., Baets, W.R.J., Information Technology and Organizational Transformation, Chichester, Wiley 1998
Rechtin, E. *Systems Architecting of Organizations*, Boca Raton, CRC Press, 2000
Scott Morton, M.S., *The Corporation of the 1990s. Information Technology and Organizational transformation*, New York, Oxford University Press 1991
Schein, E.H., Organizational Culture and Leadership, New York, Wiley 2004
The Open Group, *TOGAF Version 9*, Zaltbommel, Van Haren Publishing 2009a
The Open Group, The Open Group Business Architect Certification Program, San Francisco, 2009b
Terlouw, L. Dietz, J.L.G., A Framework for Clarifying Service-Oriented Notions – How to Position Different Approaches, in: *Enterprise Modeling and Information Systems Architectures*, Vol. 5, No. 1, July 2010
Ven, K., Verelst, J., The Adoption of DEMO: A Research Agenda, in: Albani, A. et. al. (eds.), *Advances in Enterprise Engineering III*, LNBIP 34, Springer 2009
Weinberg, G.M., *An Introduction to General Systems Thinking*, New York, Dorset House Publishing 2001
Winograd, T., A Language/Action Perspective on the Design of Cooperative Work, *Human-Computer Interaction*, Vol. 3, No. 1, 1988, pp. 3-20
Winograd, T., Flores, F., *Understanding Computers and Cognition: A New Foundation for Design*, Boston, Addison-Wesley 1987