An enterprise interoperability framework based on compliance and conformance

José C. Delgado
University of Lisbon, Portugal

ABSTRACT

The existing interoperability frameworks usually take an application-driven, top-down approach, in which the most relevant dimensions of interoperability are optimized for some problem space. For example, The European Interoperability Framework has been conceived primarily to support e-Government services.

With the goal of contributing to the establishment of the scientific foundations of interoperability, this chapter presents a multidimensional interoperability framework, conceived in a generic, bottom-up approach. The basic tenet is to add an interoperability dimension (based on the concepts of compliance and conformance) to an enterprise architecture framework with lifecycle and concreteness as its main dimensions, forming an universal core framework. This core is then provided with an extensibility mechanism, based on a concerns dimension, into which the specific characteristics of applications and their domains can be added to instantiate the framework, now in an application-driven fashion. The most relevant concerns, with sufficient applicability breadth, can be promoted to full dimensions and extend the framework. The use of partial compliance and conformance reduces coupling while still allowing interoperability, which increases adaptability, changeability and reliability, thereby contributing to a sustainable interoperability.

Keywords: Interoperability, Framework, Compliance, Conformance, Enterprise architecture, Lifecycle

INTRODUCTION

Just as any other system, an enterprise is a composition of smaller systems that interact among themselves and with the outside world. In this respect, it must solve the same interoperability problems as any two systems that need to understand each other to achieve meaningful and useful collaboration. In fact, no enterprise can survive alone. It needs to interact with other enterprises and be part of a value network. This means that interoperability is one of the most fundamental issues that any enterprise must deal with, in all its main slants, namely:

- Functionality (guaranteeing that one enterprise understands the requests of another and reacts according to what is expected);

- Non-functional aspects (ensuring adequate service levels, resource management, security, and so on);
- Coupling (reducing it as much as possible, to avoid unnecessary dependencies);
- Reliability (maintaining interoperability, even in the presence of unanticipated failures);
- Adaptability (maintaining interoperability, even when partners change their characteristics).

What distinguishes an enterprise from other systems is not only its high complexity but also the need to redefine itself constantly and to carve its own place in the global enterprise ecosystem. Beating competition through constant renovation and evolution, playing the right cards by balancing factors such as innovation, quality, governance, competitiveness, marketing, customer satisfaction, and so on, is a matter of survival. Basic data interoperability is not enough. Enterprise collaboration demands higher levels of meaningful interaction.

There is no universally accepted definition of interoperability, since its meaning can vary accordingly to the perspective, context and domain under consideration. Although limited to information, the 24765 standard (ISO/IEC/IEEE, 2010) provides the most cited definition of interoperability, as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged". We can generalize this by defining interoperability as "the ability of two or more systems or components to exchange stimuli and to react to them according to some pattern or contract that fulfills all partners' expectations". However, what does this really mean?

There are several frameworks and initiatives conceived to provide insight into what is involved in interoperability. However, most of these efforts attempt to classify interoperability by a single dimension, from high to low level. For example, the LCIM framework (Wang, Tolk & Wang, 2009) uses the following levels: conceptual, dynamic, pragmatic, semantic, syntactic, technical and no interoperability.

The one-dimensional, layered approach to complexity in a framework is classic in software and communication systems. It has the advantage of simplicity but can only express the levels of detail, from more abstract to more concrete, and not the different natures of the various aspects and concepts involved. In addition, it does not provide a justification, or foundation, for the levels used in the framework, nor how they can fit a method or a maturity model, nor which are the conditions to actually achieve interoperability at each level, nor the implications for coupling and adaptability.

Other frameworks, such as the Framework for Enterprise Interoperability (ISO, 2011), resort to more than one dimension, but the extra dimensions pertain to the method that is used to solve the interoperability problem in some context or domain, not just to the framework. This is a consequence of the typical, application-driven approach of these frameworks, which start by defining the problem space (set of applications or systems in which the interoperability problems must be solved) and then derive the corresponding solution space, by establishing a set of guidelines to solve those interoperability problems.

This chapter presents an interoperability framework that takes a different, concept-driven approach. We start by defining a domain independent core, then we provide mechanisms to instantiate it to concrete applications and finally we extend it as required by those applications and their domain. This framework is applicable to any system, (be it simple or very complex, such as enterprises), in its interaction with other systems. It includes several dimensions that reflect orthogonal aspects and which are rooted on fundamental concepts, such as:

- Lifecycle (of the enterprise's architecture or of one its components). No system is created in the operation stage. It must go through several stages of development, such as conception, design, implementation and operation, until someday, at the end of its useful life, it is decommissioned or destroyed;
- Abstraction. Abstracting details is fundamental to deal with complexity. Any system can be described at a high level of abstraction or at full detail. Going from abstract to concrete is the usual path in system design;
- Transaction. In this context, a *transaction* is a set of elementary activities that constitute a basic interaction pattern between two partners, in the roles of *consumer* and *provider*. Typically, it

involves one request from the consumer and a response from the provider. It is during a transaction that interoperability must be accomplished;

- *Compliance* and *conformance*, the two sides of *compatibility*. A system *A* can engage in a transaction (as a consumer) with a system *B* (as a provider) only if *A* is compatible with *B* with respect to that transaction. This implies that *A* must be compliant with *B* (fulfils all the requirements of *B* for placing requests) and *B* must be conformant to *A* (produces responses according to the requirements of *A*);
- Layers of interoperability, which express how much a system must know about the other (how much compatibility is needed) in order to exchange messages meaningfully (in terms of intent, content and reaction).

The main goals of this chapter are:

- To contribute to the scientific foundations of enterprise interoperability, by dissecting it into its main orthogonal components as justified by fundamental concepts in interoperability, while separating the framework from the method;
- To propose and to present a multidimensional framework that caters for this orthogonality;
- To demonstrate the expressive power of this frameworks, by showing that existing frameworks can be mapped onto it;
- To discuss a method in the context of the proposed framework.

This chapter is structured as follows. We start by presenting a multidimensional framework for enterprise architecture. Next, we present the compliance and conformance concepts, in the context of the transactions used to implement interoperability. Then, we incorporate the interoperability dimension in the enterprise architecture framework. We present some guidelines on how a method could exercise the interoperability framework. We also discuss how this framework compares with existing frameworks and how it can contribute to the establishment of a scientific base for enterprise interoperability. Finally, we present guidelines for future work and draw conclusions from this work.

BACKGROUND

Interoperability has been studied in the most varied domains, such as enterprise collaboration (Jardim-Goncalves, Agostinho & Steiger-Garcao, 2012), e-government services (Gottschalk & Solli-Sæther, 2008), military operations (Wyatt, Griendling & Mavris, 2012), cloud computing (Loutas, Kamateri, Bosi & Tarabanis, 2011), healthcare applications (Weber-Jahnke, Peyton & Topaloglou, 2012), digital libraries (El Raheb et al, 2011) and metadata (Haslhofer & Klas, 2010).

Interoperability is as old as networking. When two or more systems need to interact, an interoperability problem arises. A typical approach to deal with it is to consider several layers of abstraction and complexity, along a single dimension. Other frameworks consider several dimensions, to detail issues and concerns.

One of the first systematizations of distributed interoperability was accomplished by the Open Systems Interconnection (OSI) reference model, a standard since 1984 (ISO, 1994), which considers seven layers (Table 1). This pertains mostly to communication issues, with the objective of sending data and reproducing it at the receiver. How that data is interpreted by the receiver and how it reacts to it is left unspecified, encompassed by the topmost layer, Application. Since interoperability must not only deal with data exchange but also meaningful use of information (ISO/IEC/IEEE, 2010), we need to detail the Application layer.

Table 1 depicts the basic structure of several interoperability frameworks that use this layered approach, establishing a rough horizontal correspondence between layers.

Table 1. Comparison between several layered interoperability frameworks

OSI (1994)	C4IF (2006)	Lewis (2008)	Stamper (1996)	LCIM (2009)	EIF (2010)	Monfelt (2011)
Application	Collaboration	Organizational	Social world	Conceptual	Political	SWOT
						Cultural
						Ethical
					Legal	Legal
			Pragmatic	Dynamic	Organizational	Managerial
				Pragmatic		Organizational
	Consolidation	Semantic	Semantic	Semantic	Semantic (includes syntactic)	Adaptation
						Application
Presentation	Communication	Syntactic	Syntactic	Syntactic		Presentation
Session						Session
Transport		Machine	Empirics	Technical	Technical	Transport
Network	Connection					Network
Link						Link
Physical Medium			Physical world			Physical Medium

Peristeras & Tarabanis (2006) proposed a framework (C4IF) based on four layers: Connection (basic use of a channel), Communication (data formats), Consolidation (meaning through semantics) and Collaboration (through compatible processes). It simplifies the lower levels (distinguishing only connectivity and communication) and refines the application layer, distinguishing information semantics from behavior. Lewis, Morris, Simanta & Wrage (2008) proposed a similar framework, with slight differences but basically with the same structure.

Stamper (1996) applied *semiotics* (the study of signs, stemming from linguistics) to the field of information systems and proposed a semiotic ladder, a layered structure in which each layer builds on the previous one (just like a ladder) in increasingly higher levels of abstraction and complexity. Besides the usual syntax and semantics, the pragmatics concept was used to refer to the effect caused by the reception of a message by a receiver. Empirics refer to the lower levels that use the physical world, encompassing details that are well established and less relevant to the understanding of interoperability as a whole. The social world tackle the higher levels, in which people become more involved.

Wang, Tolk & Wang (2009) described the LCIM framework, which follows the semiotic ladder in essence, with the interesting addition of a dynamic layer, which considers evolution along the system lifecycle.

The European Interoperability Framework (EIF, 2010) was conceived as a broad framework for the interoperability of public services and establishes four main interoperability levels (legal, organizational, semantic and technical), with an upper political context that should ensure compatible visions and aligned priorities.

Monfelt, Pilemalm, Hallberg & Yngström (2011) proposed a more detailed framework, by refining the social layer of the semiotic ladder to take care of higher level issues, such as risk (SWOT analysis) and "dependencies on social and organizational aspects concerning cultural, ethical and legal values and existing administrative and managerial issues" (p. 126). This extends the basic OSI reference model with another seven layers. The organizational layer refers to the effect that a message will have (pragmatic meaning of the message) and the adaptation layer refers to the semantics of the message, adapting the new layers to the technical layers provided by the OSI model. Although shown above the Application layer, the new ones actually refine it by considering the issues that it leaves unspecified (ISO, 1994, p. 33).

Several frameworks are defined more on the maturity of systems' interoperability than on layer structure, by establishing a set of levels that give an indication (usually qualitative) of how far interoperability is considered in the abstraction scale. Table 2 illustrates this. Each framework has its own slant and terminology, but overall they possess rather similar structures, ranging from a complete lack of interoperability to full integration.

Table 2. Comparison between several interoperability frameworks in terms of maturity levels

Level	LISI (C4ISR , 1998)	OIM (Fewell & Clark, 2003)	OIAM (Kingston, Fewell, & Richer, 2005)	MMEI (Guédria, Chen, & Naudet, 2009)
4	Enterprise	Unified	Dynamic	Adapted
3	Domain	Combined	Open	Organized
2	Functional	Collaborative	Accommodating	Aligned
1	Connected	Ad hoc	Amenable	Defined
0	Isolated	Independent	Static	Unprepared

Ford, Colombi, Graham & Jacques (2007) developed a six-step mathematical method of measuring interoperability (i-Score), providing a means to identify the gap between an existing interoperability measurement and its optimal value.

Other interoperability frameworks, particularly those conceived for complex systems, such as enterprises, try to complete the scenario by considering more than one dimension of interoperability.

Berre *et al* (2007) described the ATHENA Interoperability Framework (AIF), which builds on previous research, including the IDEAS and INTEROP projects (Chen, Doumeingts & Vernadat, 2008). This is a framework developed mainly for enterprise integration, structured into three parts:

- Conceptual integration, at the levels of enterprise/business, process, service and information/data;
- Application integration, which includes a methodology with methods to support the development of interoperability projects;
- Technical integration, which provides an implementation basis around technologies such as BPEL, Web Services and XML.

Chen (2006) proposed an enterprise interoperability framework, based on three dimensions:

- Concerns (business, processes, services and data), based on the interoperability layers of the Athena framework (Berre *et al*, 2007);
- Barriers, which express the difficulties in making systems interoperable, such as syntactic and semantic differences, and are divided into three categories: conceptual, technological and organizational;
- Approach, the plan to overcome these barriers and to achieve a solution to the interoperability problem. It can be organized in three ways: integrated (a common format is used by all systems), unified (common goals and semantic mapping) and federated (there may be common goals and ontology, but each system must adapt to others dynamically, without any imposed model). This

leads to a barrier-driven method (Chen & Daclin, 2007), oriented towards solving the concrete problem of implementing interoperability between two existing enterprises by removing the perceived barriers to achieve the solution.

The CEN EN/ISO 11354-1 standard (ISO, 2011) defines a Framework for Enterprise Interoperability (FEI) and is based on the interoperability framework described by Chen (2006).

Morris *et al* (2004) proposed an interoperability framework (SOSI) that includes not only the operational dimension but also the programmatic and constructive dimensions, which pertain to prior stages in the systems' lifecycle, namely conception and design, respectively.

Ostadzadeh & Fereidoon (2011) presented an interoperability framework conceived for ultra-large-scale systems, with three dimensions. It tries to include interoperability aspects from the previous frameworks, while adding an enterprise framework dimension:

- Abstract, which draws on the Zachman enterprise framework (Sowa & Zachman, 1992) and answers the classical six questions (what, how, where, who, why, when) in the context of interoperability;
- Perspective, expressing layered interoperability concerns (contextual, conceptual, logical, physical and out-of-context);
- Barriers, extending the SOSI framework with a social slant, to take into account the human factor and cultural issues, but collapsing the three SOSI dimensions into one.

The European project ENSEMBLE (Agostinho, Jardim-Goncalves & Steiger-Garcao, 2011) embodies an effort to formulate a science base for enterprise interoperability. This involves discovering which interoperability problems exist in domains related to interoperability, such as social, applied and formal sciences, and identifying which are the relevant scientific areas underlying interoperability. Twelve areas have been identified in which interoperability is necessary: data, process, rules, objects (in the context of the Internet of Things), software, cultural, knowledge, services, social networks, electronic identity, cloud and ecosystems. In addition, four levels of scientific elements of interoperability (semantics, models, tools and orchestration) have been identified.

Modeling before designing has always been an important tenet. Since the early days of the Zachman framework (Sowa & Zachman, 1992), we understand that enterprise architecture and its underlying principles (Greefhorst & Proper, 2011; Winter & Aier, 2011) are the cornerstones of the enterprises themselves.

Today, several enterprise architecture frameworks exist, with Zachman framework and TOGAF (The Open Group, 2009) as two of the most used, among a wide set (Minoli, 2008) that includes the Department of Defense's Architectural Framework (DoDAF), the Extended Enterprise Architecture Framework (E2AF), the Federal Enterprise Architecture Framework (FEAF), the Treasury Enterprise Architecture Framework (TEAF) and the Integrated Architecture Framework (IAF).

TOGAF is gaining widespread acceptance as a *de facto* standard in the industry (Dietz & Hoogervorst, 2011). This standard establishes a metamodel and a set of best practices for describing architectures, including ontology and requirements for architecture frameworks and architecture description languages.

Since this chapter contends that interoperability starts on the early stages of architecture development, the BMM, or Business Motivation Model (Malik, 2009), is of particular importance. In fact, it has influenced the architecture framework presented in this chapter. Instead of organizing the architecture into layers, from business down to technology (Winter and Fischer, 2007), the BMM emphasizes the motivations (the reasons why an architecture should be the way it is), the ends (the expectations which should be satisfied) and the means (what should be done to achieve those ends). Each of these types of concerns is derived from the previous one, in a model driven way under a business perspective.

ENTERPRISE ARCHITECTURE AS THE FOUNDATION FOR INTEROPERABILITY

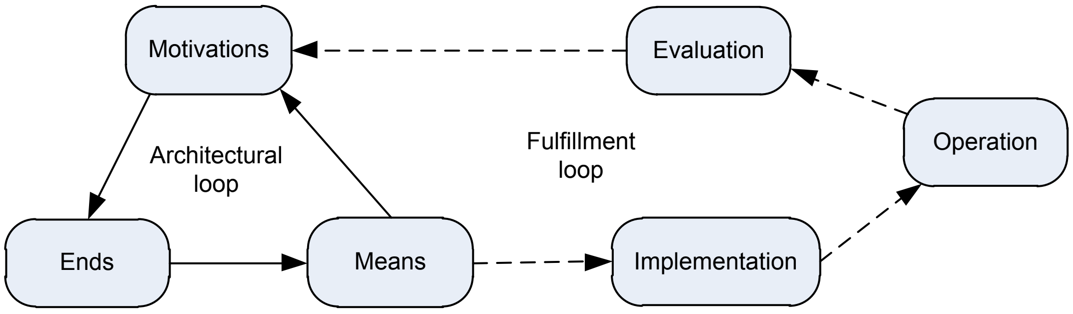
The frameworks mentioned in Table 1 use a single dimension of interoperability. Although this leads to a simple classification scheme, they fail to acknowledge that not all interoperability levels belong to the same dimension and ignore other relevant aspects.

To understand why, we need first to consider an enterprise in isolation and to establish a framework to characterize its enterprise architecture. We will later extend it with the interoperability perspective.

The enterprise architecture lifecycle

Figure 1 depicts the typical lifecycle of the conception and animation of an enterprise architecture, with several stages that reflect different perspectives and occupy specific positions in a graph that is part of an overall method. The loops cater for changes and improvements.

Figure 1. Typical enterprise architecture lifecycle, with stages chained in changeability loops.



The *architectural loop* is inspired by the Business Motivation Model (Malik, 2009) and laid around three main concepts:

- *Motivations*, which emphasize the reasons behind the architectural decisions taken, in accordance with the business needs;
- *Ends*, which express the desires and expectations of stakeholders (e.g., goals and objectives);
- *Means*, which describe the mechanisms used to fulfill those expectations (e.g., actions).

The *fulfillment loop* in Figure 1 models subsequent stages and is based on the organization adopted by classical development methods, such as the Rational Unified Process (Kruchten, 2004):

- The ends and the means of the architectural loop correspond roughly to the analysis and design stages of those methods;
- The *implementation* includes stages such as development, testing and deployment;
- The *operation* corresponds to animating (executing) the system;
- The *evaluation* measures KPIs (Key Performance Indicators) and assesses how well the system meets the expectations caused by the motivations.

A multidimensional architecture framework

The lifecycle of Figure 1 corresponds to a dimension (or axis) of the framework, discretized into six stages. There is another implicit dimension, Evolution, discretized into versions, that stems from repeating the loops, expressing that the architecture can be successively changed and improved. This is not enough to capture all the most important aspects. In particular, we need two additional axes:

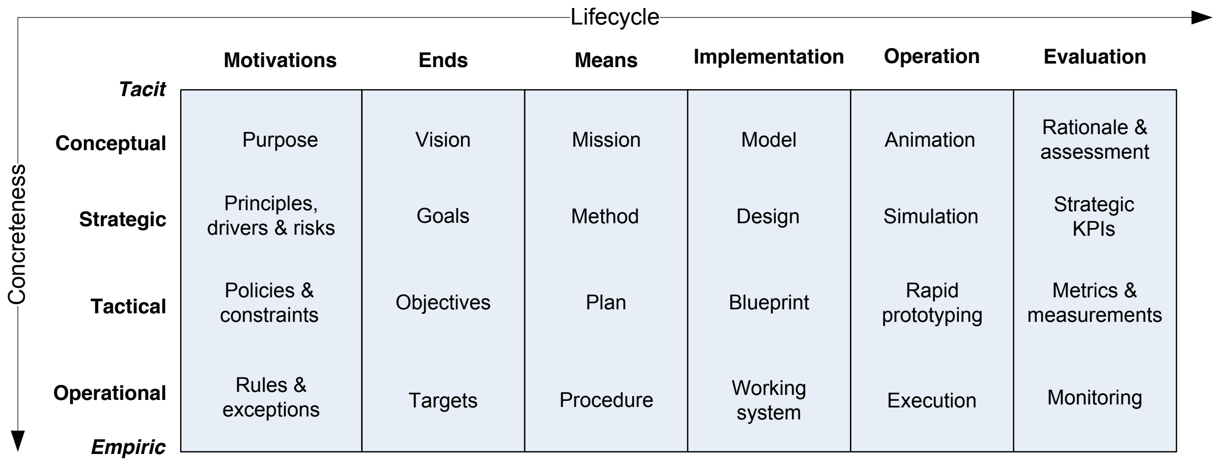
- *Concreteness*. Each stage in Figure 1 can be seen at a very high, abstract level, or at a very detailed, concrete level. We have discretized this axis into four levels: *Conceptual*, *Strategic*, *Tactical* and

Operational. It is common to mix these designations with the Lifecycle axis (embedded into the Motivations, Ends, Means and Implementation stages, respectively). However, note that this common usage corresponds to a specific instantiation of the method, in which tactical decisions are made only after the strategy has been laid out and detailed. In other words, the strategy is used as the motivation for the tactics. However, the motivation for the tactics should be a refinement of the motivation for the strategy and not a consequence of strategy design. That is why we separate concreteness from lifecycle. We can have the Implementation stage at a conceptual level (ideas on how to do it) as well as the Motivations stage at an operational level (justification for the lowest level actions). Stages of the lifecycle and their level of concreteness are orthogonal concepts;

- *Concerns*. The focus words (*what*, *how*, *where*, *who*, *when*, *why*) in the Zachman framework (Sowa & Zachman, 1992) are generic but do not address the entire focus range. Other questions can be made, such as *whence* (where from), *whither* (where to), *how much* (quantitative assessment) and *how well* (qualitative assessment). It is important to be able to express the dynamics of the architecture, its quality (how good it is, quantitatively and qualitatively) and other stakeholders' concerns (financial, regulatory compliance, security, interoperability, domain specific, and so on), both functional and non-functional. The architectural principles of TOGAF (The Open Group, 2009) fit in here. This axis can be organized into categories of concerns, from qualitative to quantitative.

We start by laying down the plane formed by the Lifecycle and Concreteness axes, in Figure 2, and explaining what is the meaning of each cell, which results from crossing the values of both axes and represents one lifecycle stage at a given level of concreteness.

Figure 2. The Lifecycle and Concreteness plane.



	Motivations	Ends	Means	Implementation	Operation	Evaluation
Conceptual	Purpose	Vision	Mission	Model	Animation	Rationale & assessment
Strategic	Principles, drivers & risks	Goals	Method	Design	Simulation	Strategic KPIs
Tactical	Policies & constraints	Objectives	Plan	Blueprint	Rapid prototyping	Metrics & measurements
Operational	Rules & exceptions	Targets	Procedure	Working system	Execution	Monitoring

The Concreteness axis (vertical) considers four levels, from Conceptual, more fuzzy and abstract, down to Operational, more detailed and concrete. Each level in this axis is a refinement of the level above it, by including decisions that turn some abstract aspects into concrete ones. This axis has two opposite thresholds:

- *Tacit*. This is the highest level, above which concepts are too complex or too difficult to describe. It encompasses the tacit knowledge and know-how (Oguz & Sengün, 2011) of the people that conceived the enterprise architecture or manage the enterprise, expressing their insight and implicit expectations and assumptions about the business and enterprise governance;

- *Empiric*. The lowest level, below which details are not relevant anymore and we settle for just using something that already exists and is known to work, such as a standard, an organizational unit or a piece of equipment.

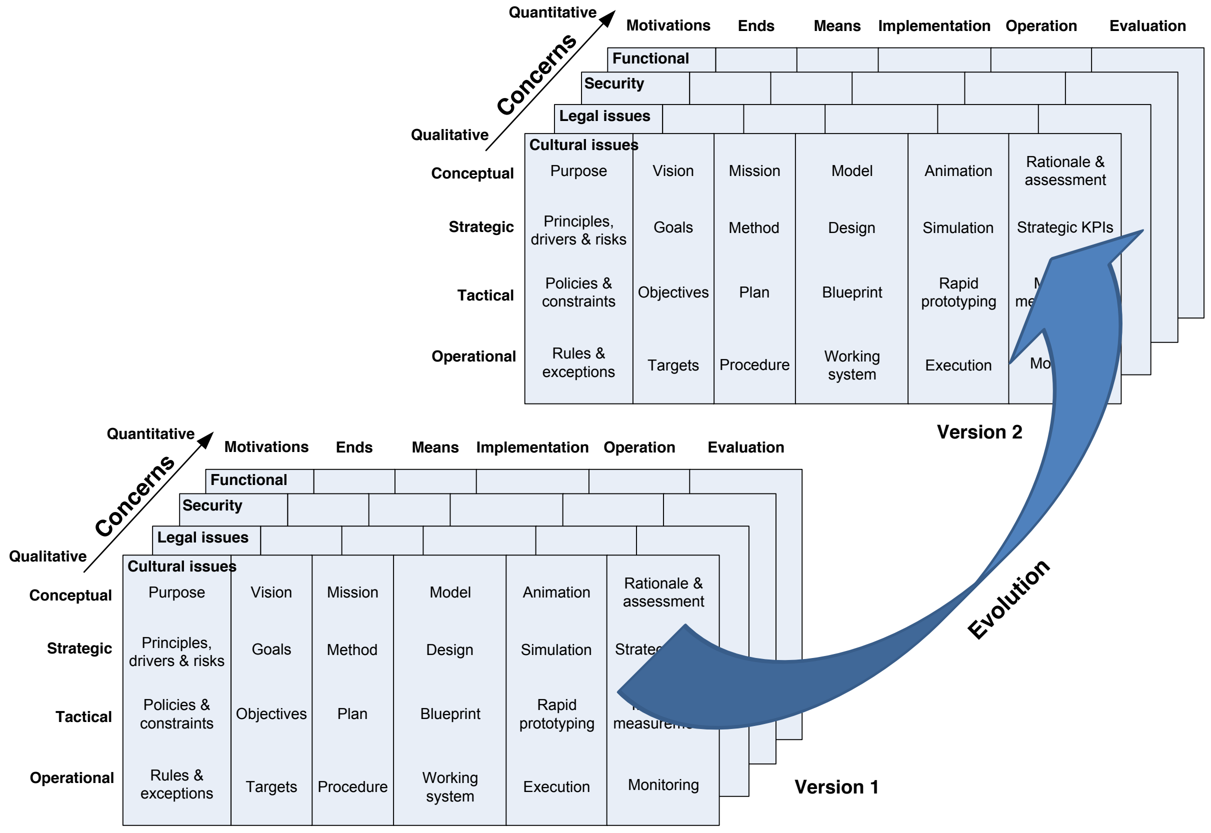
The Lifecycle axis (horizontal) lays down the six lifecycle stages in Figure 1, disregarding the loops, which means that it describes the stages of one iteration (version) of the enterprise architecture. Each column, and the cell at each concreteness level, can be described as:

- *Motivations* – This column is the result of contextual requirements for the design of the system, acting as motivation and justification for the ensuing stages. At the top level, we have the *Purpose*, which reflects the essence and the reason for the existence of this architecture. The *Principles, drivers & risks* refine the *Purpose* by establishing not only the basic tenets that this architecture must obey but also the enablers and inhibitors that foster and limit the characteristics of the architecture (resulting probably from a SWOT analysis). The *Policies & constraints* perform a similar role, but at a lower level and greater detail, which in turn are refined into *Rules & exceptions* (special cases), at the *Operational* level;
- *Ends* – This column corresponds to the expectations established for the architecture and that are justified by the *Motivations*. At the top level, the *Vision* corresponds to the global scenario sought that fulfills the purpose. The vision is refined and decomposed into *Goals*, which express, usually in a qualitative way, a desired state or set of properties to achieve. Goals are in turn refined into *Objectives*, concrete enough to be declared SMART (*Specific*, *Measureable*, *Achievable*, *Realistic* and *Time framed*). *Targets* are lower level and simpler objectives, at the atomic level (which cannot be further decomposed) or close enough;
- *Means* – The cells in this column specify what needs to be done in order to meet the expectations. The *Mission* is the top-level expression of the type of actions to carry out. The *Method* is a refinement of the mission by including the choice of an approach or paradigm. The *Plan* is a graph of steps that make the method concrete by choosing a set of techniques. The *Procedure* is a set of actions that detail the plan by choosing a set of algorithms. The method here refers to the high level plan underlying a specific architecture and should not be mixed up with the architecture development method, which is discussed in section "The method to achieve interoperability";
- *Implementation* – This column specifies the tools, languages and procedures to provide an implementation of the means, by transforming them into workable representations. The *Model* is an overall representation of the architecture chosen to carry out the mission, expressed in some notation, such as UML. The *Design* details the model, structuring it into modules, or subsystems, and establishing their interactions. The *Blueprint* is the detailed description of each module. The *Working system* provides full details (such as data and code);
- *Operation* – This column exercises the system, either to operate it in real conditions or just to get insight about its quality, in both functional (how well does it fulfill the motivations) and non-functional (how much does it comply with a set of fulfillment criteria) terms. At the conceptual level, we can only imagine what will happen by *Animation* of scenarios and use cases (nevertheless, important to check global behavior). *Simulation* can abstract many details by using statistical models and still obtain meaningful insight into the behavior (in statistical terms). *Rapid prototyping* works with partial specifications (not statistical, but not fully implemented modules, either) and enables exercising some specific concerns (user interfaces, for instance). At the lower level, *Execution* corresponds to exercising the completely implemented system, under real conditions, such as running a program;
- *Evaluation* – This is the assessment of the operation, so that changes and improvements can be introduced, if deemed necessary to better fulfill the purpose in the *Motivations* stage. At the top level, *Rationale & assessment* perform reasoning over the scenario and use case animation. *Simulation* provides the inputs to *Strategic KPIs*, which can be checked against the *Principles, drivers & risks* in *Motivations*. *Metrics & measurements* can check how well and how much the *Policies & constraints* in *Motivations* are satisfied. *Monitoring* checks events and serves as the

basis for all higher-level evaluation procedures, assessing whether all *Rules & exceptions* in *Motivations* are dealt with satisfactorily.

The Concerns axis expresses the various aspects under consideration in the system, from a more qualitative to a more quantitative emphasis. Each concern corresponds to a Lifecycle and Concreteness plane. The Evolution axis corresponds to a new batch of planes, expressing a new version of the system. Figure 3 illustrates these axes.

Figure 3. The Concerns and Evolution axes.



The framework can be extended by promoting concerns to axes, if they are relevant enough. This can happen if the concern:

- Can be discretized into several situations, stages or values;
- Has a variability that is relevant across the entire span of existing axes;
- Refers to issues orthogonal to those dealt with by other axes.

In general, the architecture framework is multidimensional but, for the sake of simplicity, the following sections consider only a subset of axes at a time.

Discussion and rationale

Strategy and tactics are usually bound to the notion of course of action (Malik, 2009) and therefore it may seem strange that the Motivations stage, for example, can be described at the strategic, tactic and even operational levels. However, as motivations are progressively detailed and made more concrete, they have to include some notion of approach. The Strategic and Tactical levels of the Motivations stage, for instance, describe the motivations for the strategy and tactics to use in subsequent stages.

In the same way, policies and rules appear in the Business Motivation Model (Malik, 2009) as directives in the course of action stemming from tactics and motivated by a SWOT-type assessment. In our opinion, however, they should appear first in the Motivations stage and not just afterwards, in the Means stage.

In fact, we can look at each horizontal plane (parallel to the plane formed by the Concerns and Lifecycle axes) in Figure 3 as global descriptions of the system at increasingly depths of detail and concreteness. The top plane (Conceptual) is a view that describes essentially the main concepts, ideas and features, both in the various stages (Motivations, Ends, Means, and so on) and in stakeholder's concerns. The next plane (Strategic) is a refinement of the previous one, with decisions that reflect a strategic approach by making it partially more concrete. The same happens in the two lower planes (Tactical and Operational), with increasingly level of refinement and detail. Each plane can lead to different lower planes, according to the decisions taken and approach followed.

These planes correspond roughly to the main rows of the Zachman framework (*planner*, *owner*, *designer* and *builder*). TOGAF also includes a dimension of level of detail (and strategy as the top level), but with refinement essentially linked to structure, with the strategic level architecture decomposed into architectural units at the segment (business areas) and capability (business units) levels. Aspect refinement in TOGAF is accomplished by considering different types of architectures, or domains (*business*, *data*, *application* and *technology*), which may have its justification in terms of current technology (namely, by separating data and applications, which assumes the process paradigm) but that in practice tend to act as architectural silos. The Concreteness axis in Figure 2 supports both structural decomposition and aspect refinement.

Note that, in our framework, both the Lifecycle and Concreteness axes reflect features of the system itself (nature of considerations and level of refinement) and not stakeholder's perspectives of the system (Zachman) nor architectural units, types or domains (TOGAF).

Vertical planes that are parallel to the Concreteness-Lifecycle plane correspond to concern oriented views (for example, looking at everything from the security or from the legal point of views). Vertical planes that are parallel to the Concreteness-Concerns plane correspond to considering every issue in each stage (for example, the ends, namely goals and objectives, to achieve under the various concerns and at various levels of detail).

Like most architectural frameworks (Minoli, 2008), ours is recursive (Greefhorst & Proper, 2011), in the sense that progression downwards the concreteness axis can be made by aspect refinement or structural decomposition. In the latter case, each architectural subsystem can be modeled by the same framework. The architecture concept can thus refer to a set of interacting enterprises, an enterprise, a department, a business unit, and so on, down to the level of a small module in an application. It all depends on the granularity envisaged.

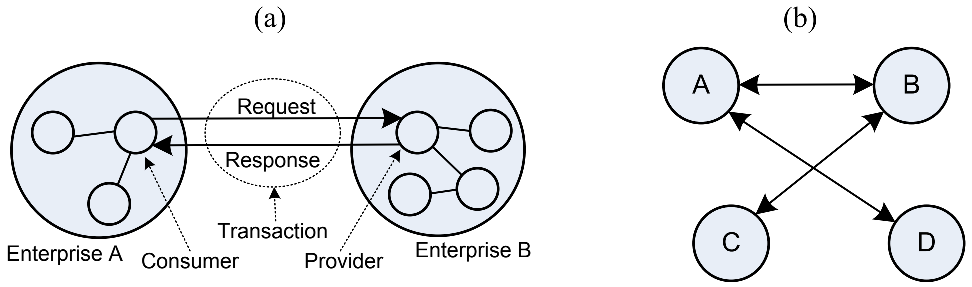
TRANSACTIONS AND CHOREOGRAPHIES

Interactions between enterprises occur in the Operation stage of the lifecycle (Figure 1), in the form of transactions, but the fact is that interoperability starts at the beginning of the lifecycle, with the motivations to interact with other enterprises.

A transaction is a primitive pattern (predefined sequence) of messages exchanged between two interacting partners, one in the role of consumer, which initiates the pattern by a request, and the other in the role of provider, which satisfies that request and typically provides a response. The same enterprise can act as a consumer and as a provider, in bidirectional interactions. A *choreography* (Bravetti & Zavattaro, 2009) is a contract between two or more partners that expresses a composition of transactions.

Figure 4 illustrates a request/response transaction and a multi-partner choreography. Figure 4a shows that the interaction between enterprises usually involves only specific (sub)systems and not the enterprise as a whole. Nevertheless, since systems are recursively composed of other systems, the rules are the same. Figure 4b shows a simplified representation of a choreography.

Figure 4. Illustration of a transaction (a) and of a choreography (b)



Dietz (2006) established a standard basic transaction pattern, in which the provider, upon receiving a request, can either send a *promise* (to honor it) or decline it, whereas the consumer can either accept or reject the response. If the interaction is asynchronous, a *future* (which the provider's response will later replace) can be returned to the consumer immediately, upon sending the request. Table 3 illustrates some of the basic message types that a transaction can entail.

Table 3. Examples of message types.

Message category/type	Description
Request	*Initial request in each transaction*
React	React to message, no answer expected
React & respond	React to message and answer/notify
Amendment	*Further information on an already sent request*
Cancel	Cancel the execution of the request
Response	*Response to the request*
Answer	A value returned as a response (replaces the future, if any)
Resource fault	A value returned as the result of an exception
Protocol fault	A value resulting from a protocol error (or failure in partner)
Notification	*Information of completion status*
Promise	Confirm will to honor the request
Denial	Confirm rejection of request
Done	Request completed but has no value to reply
Cancelled	Confirm cancellation of request

To achieve interoperability, messages need to be accepted and understood at all required levels, within a transaction or a choreography. Some enterprises resort to mediators to convert data formats, protocols or some other aspect. In this case, Figure 4a still applies, with the mediator as just another partner involved, producing two relationships (Enterprise *A* to mediator and mediator to Enterprise *B*).

Meaningfully exchanging a message (the basic component of a transaction) between consumer and provider, in one direction or the other, entails the following main aspects:

- *Intent*. Sending the message must have a given intent, inherent to the transaction it belongs to and related to the collaboration sought by two interacting enterprises;
- *Content*. This concerns the generation and interpretation of the content of a message by the sender, expressed by some syntax and semantics (including ontology) in such a way that the receiver is also able to interpret it;
- *Transfer*. The message content needs to be successfully transferred from the context of the sender to the context of the receiver;

- *Reaction*. This concerns the reaction of the receiver upon reception of a message, which should produce effects according to the expectations of the sender (either functionally and non-functionally, including issues such as security and exceptions caused by failures).

This is valid both at higher levels of detail, which typically correspond to business choreographies, and at lower levels, which typically implement data communication protocols.

COMPATIBILITY: COMPLIANCE AND CONFORMANCE

The ability of two enterprises to engage successfully in a transaction, in the roles of consumer and provider, means that they are *compatible* with respect to that transaction and in those roles. In the same way, an enterprise is said to be compatible with a choreography if it fulfills all the requirements of all the roles that enterprise needs to perform in that choreography. The same enterprise can act both as consumer and as provider at multiple times in a given choreography.

In many cases, systems are made compatible by design, i.e., conceived and implemented to work together. The problem with enterprises is that they are complex and evolve in an independent way, which means that ensuring compatibility at the enterprise level is not an easy task. A typical and pragmatic solution is to resort to Web Services and XML data, sharing schemas and namespaces.

More flexible solutions involve discovering Web Services similar to what is sought, by performing schema matching with similarity algorithms (Jeong, Lee, Cho & Lee, 2008), and ontology matching and mapping (Euzenat & Shvaiko, 2007). Manual adaptations are usually unavoidable.

The *compatibility* notion introduces a different perspective, stronger than similarity but weaker than commonality (resulting from using the same schemas and ontologies). The trick is to consider partial (instead of full) compatibility, by considering only the intersection between what the consumer needs and what the provider can offer. If the latter subsumes the former, the degree of compatibility required by the consumer is feasible, regardless of whether the provider supports additional features or not.

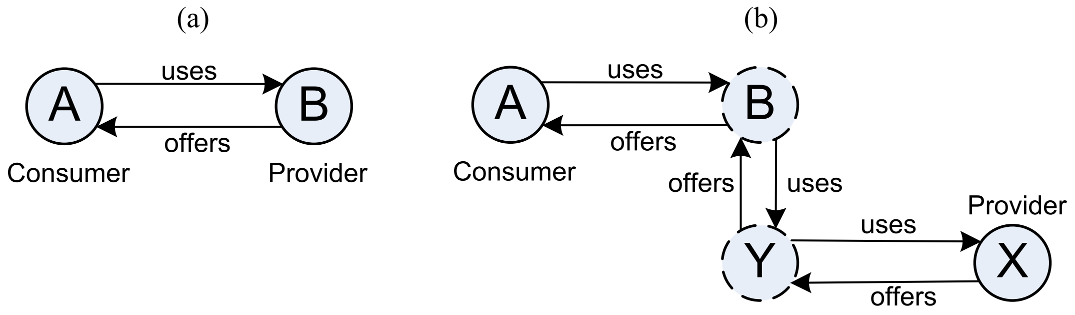
Compatibility (of a consumer with a provider) entails the following concepts:

- *Compliance* (Kokash & Arbab, 2009). The consumer must satisfy (*comply with*) the requirements of the provider for sending requests to it, without which these cannot be honored;
- *Conformance* (Adriansyah, van Dongen & van der Aalst, 2010). The provider must fulfill all the expectations of the consumer regarding the effects of a request (including eventual responses), therefore being able to take the form of (*to conform to*) whatever the consumer expects it to be.

In full compatibility, the consumer can use all the provider's features. In partial compatibility, the consumer uses only a subset of the provider's features, which means that compliance and conformance need only be ensured for that subset.

As an example, consider Figure 5a, in which an enterprise *A* (in the role of consumer) has full compatibility with enterprise *B*, in the role of provider. *A* uses only the features that *B* offers and *B* offers only what *A* uses. However, in Figure 5b, the provider of *A* is now enterprise *X*, which has been designed for full compatibility with enterprise *Y*, in the role of consumer. The problem is that *A* was designed to interact with a provider *B* and *X* was designed to expect a consumer *Y*. In other words, *B* is how *A* views provider *X* and *Y* is how *X* views consumer *A*. In this case, *B* and *Y* are specifications, not actual systems. How can *A* be made compatible with *X* so that it becomes able to interact with it?

Figure 5. Full (a) and partial (b) compatibility.



There are two necessary conditions:

- *Compliance*. *B* must comply with *Y*. Since *A* complies with *B* and *Y* with *X*, this means that *A* complies with *X* and, therefore, *A* can use *X* as if it were *B*, as it was designed for;
- *Conformance*. *Y* must conform to *B*. Since *X* conforms to *Y* and *B* to *A*, this means that *X* conforms to *A* and, therefore, *X* can replace (take the form of) *B* without *A* noticing it.

In this example, partial compatibility has been achieved by *subsumption*, with the set of features that *A* uses as a subset of the set of features offered by *X*. This inclusion relationship, without changing characteristics, allows compatibility to be transitive across the interacting partners and specifications in Figure 5b.

Partial compatibility can also be achieved by *translation*, in which a mediator adapts features by changing names, formats, or any other characteristic (including mashups of features) between actual interacting partners. In this case, compatibility is not transitive.

Compatibility is also not commutative, since the roles of consumer and provider are different and asymmetric by nature. However, nothing prevents two interacting partners from switching roles in a symmetric way, by using and offering identical features in a reciprocal fashion, which is typical of certain interaction protocols. These are just special cases.

We should also note that:

- Compliance and conformance are general concepts, in terms of satisfying requirements and expectations, respectively, and can be applied to behavior, data or any other specifications that somehow become related (such as enterprise strategies, for example), either at a high level of abstraction or at a low and detailed level. Wherever there is a relationship, compliance and conformance are present;
- Compatibility, which encompasses both compliance and conformance, is thus a concept that is more general than interoperability, which applies to the Operation stage of the lifecycle, in which the operational interactions occur. Compatibility applies to any relationship in any stage of the lifecycle, even if the Operation stage is never reached (such as in models and documentation).

The greatest advantage of partial compatibility is the ability to tune up the right level of interoperability, achieving a balance between two opposing goals:

- Enterprises need to cooperate and therefore must share some set of specifications, so that information can flow between them, be properly understood and produce the intended effects;
- However, shared specifications mean coupling and constraints to diversity (universally adopted standards are not always the typical case) and lifecycle evolution;

The more specifications are shared, the greater the coupling. Therefore, compatibility should be reduced to the minimum necessary compliance and conformance and no more than that. On the other hand, an enterprise that needs less compliance and/or less conformance has better chances of being able to participate in a choreography (Delgado, 2012). This is an important aspect when discovering potential partners for a collaboration contract.

A MULTIDIMENSIONAL INTEROPERABILITY FRAMEWORK

Armored with the multidimensional enterprise architecture framework of Figure 3 and the concepts of compliance and conformance introduced in the previous section, we now set out to improve on the linear structure of the interoperability frameworks of Table 1 to reach a multidimensional interoperability framework.

A linear perspective of interoperability

We start by addressing ourselves the following question: if an enterprise *A* (in the role of consumer) wants to interact with another *B* (in the role of provider), what must *A* know about *B*, and vice-versa?

If we consider again Table 3, which describes the various types of messages that can be used to build a transaction, and the aspects that it entails (intent, content, transfer and reaction), it is easy to assert that a lot needs to be involved to ensure that *A* complies with what *B* requires and *B* conforms to what *A* expects.

As any complex issue, interoperability may be considered at various levels of abstraction, which means discretizing it into layers. In our framework, we consider the layers described in Table 4, which details and extends the layer schemes used in Table 1.

Table 4. Layers of interoperability in a transaction. ▬ *- Request.* ▭ *- Response.*

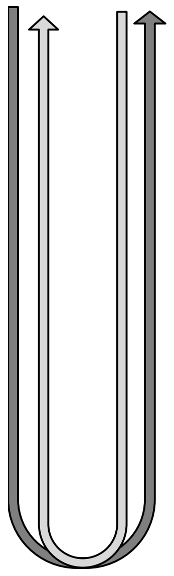
Category	Layer	Artifact (Consumer)	Interaction channel	Artifact (Provider)
Symbiotic (nature)	Coordination	Governance		Governance
	Alignment	Joint-venture		Joint-venture
	Collaboration	Partnership		Partnership
	Cooperation	Outsourcing		Outsourcing
Pragmatic (context)	Contract	Choreography		Choreography
	Workflow	Process		Process
	Interface	Service		Service
Semantic (meaning)	Inference	Rule		Rule
	Knowledge	Semantic network		Semantic network
	Ontology	Concept		Concept
Syntactic (notation)	Structure	Schema		Schema
	Primitive type	Primitive object		Primitive object
	Serialization	Message format		Message format
Connective (protocol)	Messaging	Message protocol		Message protocol
	Routing	Gateway		Gateway
	Communication	Network protocol		Network protocol
	Physics	Equipment		Equipment

Table 4 depicts a transaction, in which a consumer sends a request to a provider, which executes the request and sends a response to the consumer. This involves an interoperability ladder, with a set of layers of interoperability, from a very high level of abstraction (each enterprise has assumptions and expectations regarding the nature of the intent of the other in engaging in some interaction) down to a very low level (any message must physically reach the other enterprise). The goal of the U-shaped message paths (in both requests and responses) depicted at the center of the table is twofold:

- To show that two enterprises do not interact directly, except at the lowest level. Any message needs to go down the interoperability ladder all the way to the bottom, at the sender, so that the message flows through the communication channel in some physical format, and then needs to be reconstructed at the receiver, climbing the interoperability ladder all the way up until its intent is understood;
- Any message goes through all levels, both in sender and receiver. Whether it pertains to some low level protocol or to some high level business choreography, it must be sent through the lowest level (physical channel communication) but, at the same time, it has been sent with some intent and serves a given purpose. The transaction is just an instantiation of a general interaction pattern, which means that it goes through all layers. Choreographies will be compositions thereof.

The Category column in Table 4 organizes the most interrelated interoperability layers (Layer column) into sets, with the following meaning:

- *Symbiotic*. This category expresses the interaction nature of two interacting enterprises in a mutually beneficial agreement. This can be a tight coordination under a common governance, if the enterprises are controlled by the same entity, a joint-venture agreement, if the two enterprises are substantially aligned, a collaboration involving a partnership agreement, if some goals are shared, or a mere value chain cooperation, instantiated as an outsourcing contract;
- *Pragmatic*. The interaction between a consumer and a provider is done in the context of a contract (the nature of which in not known in this category), which is implemented by a choreography that coordinates processes, which in turn implement workflow behavior by orchestrating service invocations;
- *Semantic*. Both interacting enterprises must be able to understand the meaning of the content of the messages exchanged, both requests and responses. This implies compatibility in rules, knowledge and ontologies, so that meaning is not lost when transferring a message from the context of the sender to that of the receiver;
- *Syntactic*. This category deals mainly with form, rather than content. Each message has a structure, composed by data (primitive objects) according to some structural definition (its schema). The data in messages need to be serialized to be sent over the channel, using formats such as XML or JSON;
- *Connective*. The main objective in this category is to transfer a message from one enterprise to the other, regardless of its content. This usually involves enclosing that content in another message with control information and implementing a message protocol (such as SOAP or HTTP) over a communications network, according to its own protocol (such as TCP/IP) and possibly involving routing gateways.

Formally, each transaction must satisfy compatibility (compliance and conformance) at all layers, both in the request and in the response, which can be viewed in two orthogonal directions:

- Horizontally, between the same layer in the consumer and in the producer. This is merely an abstraction mechanism, hiding the lower layers, as if the interaction occurred in that layer. Programs 1 and 2, in section “A simple wrap-up example”, illustrate the Service and Schema layers;
- Vertically, between adjacent layers in each interacting partner. An artifact in an interoperability layer is obtained by (abstracting the details of) the composition of artifacts in the layers below. For example, a choreography coordinates a set of processes and a process is a set of service invocations. The vertical relationship between the Choreography and Process layers, for example, means that a process must comply with a choreography in the role of consumer and conform to it in the role of provider.

It is interesting to note that these horizontal and vertical relationships have essentially the same principles that have already been stated by the OSI reference model (OSI, 1994). The most relevant changes are the distinction between compliance and conformance and the discrimination of the Application layer into a range of interoperability layers.

In practical terms, it is very difficult to consider all the interoperability layers in a systematic way for all transactions, which means that the tacit and empiric limits, already used in the concreteness axis of the multidimensional framework if Figure 2, are again helpful here. For example, a business-oriented perspective may consider the empiric level (below which details are not that relevant) at the semantic category, whereas a more implementation-oriented perspective may see the pragmatic category as the tacit level (assuming someone is taking care of behavior and intentions of the interacting partners). In practice, only a range of layers is considered explicitly in each context, but Table 4 needs to consider all layers for completeness. The fact that a Web Service, for instance, does not deal explicitly with semantics does not mean that the corresponding interoperability layers are not there. Simply, it is a tacit aspect, unverified by the system.

From bottom to top, each interoperability layer adds new knowledge of one interacting partner about the other partner's interoperability characteristics. For example, at the lowest layer only the physical level protocol is known. Layers up to Choreography have a reasonably universal meaning. Therefore, we detail only the layers above, in the Symbiotic category, which is dealt with explicitly only by interacting partners that understand why they are engaging in an interaction (all others do it tacitly):

- Cooperation. This just a marriage of convenience, limited to a contract to supply something at the exchange of something else (such as a payment). Mutual interoperability knowledge is limited to the set of choreographies necessary to implement that contract. This is typical of business interactions in value chains;
- Collaboration. This involves an interaction agreement towards some compatible (or even common) goals and not just a supply contract, entailing explicit compliance and conformance between the goals of the interacting enterprises. This is what happens in partnerships;
- Alignment. This is even a more profound interaction, in which there must be explicit compatibility not only in goals but also in the motivations to pursue them. Alignment leads to interactions such as those found in joint-ventures;
- Governance. This corresponds the highest level of mutual interoperability knowledge, because not only goals and motivations, but also the actual governance, and by extension evolution into new versions of the enterprise architecture, must be compatible. This layer is only found in partners that experience a tight coordination, such as found in merging operations, subsidiary enterprises or departments of the same enterprise.

We also note that:

- The composition ladder across interoperability layers, in the Artifact columns of Table 4, is not necessarily as linear as expressed. For example, the Schema artifact in the Structure layer composes Primitive objects (necessary in any system), but can also compose higher-level objects, namely the artifacts in the layers of the Semantic category. This is a natural consequence of the recursive nature of system (de)composition. In addition, composition can be both in space (module composition) and time (interaction protocols);
- The relationship between interacting partners needs not be symmetric and the level of mutual interoperability knowledge (highest layer dealt with explicitly) needs not be the same when consumer and provider roles are reversed in subsequent transactions.

These considerations lead to the concept of *interoperability profile*, which is a set of interoperability layers dealt with explicitly in a given interaction (between a tacit and an empiric treatment). Some profiles can occur frequently enough to create a pattern. Examples of interoperability profiles:

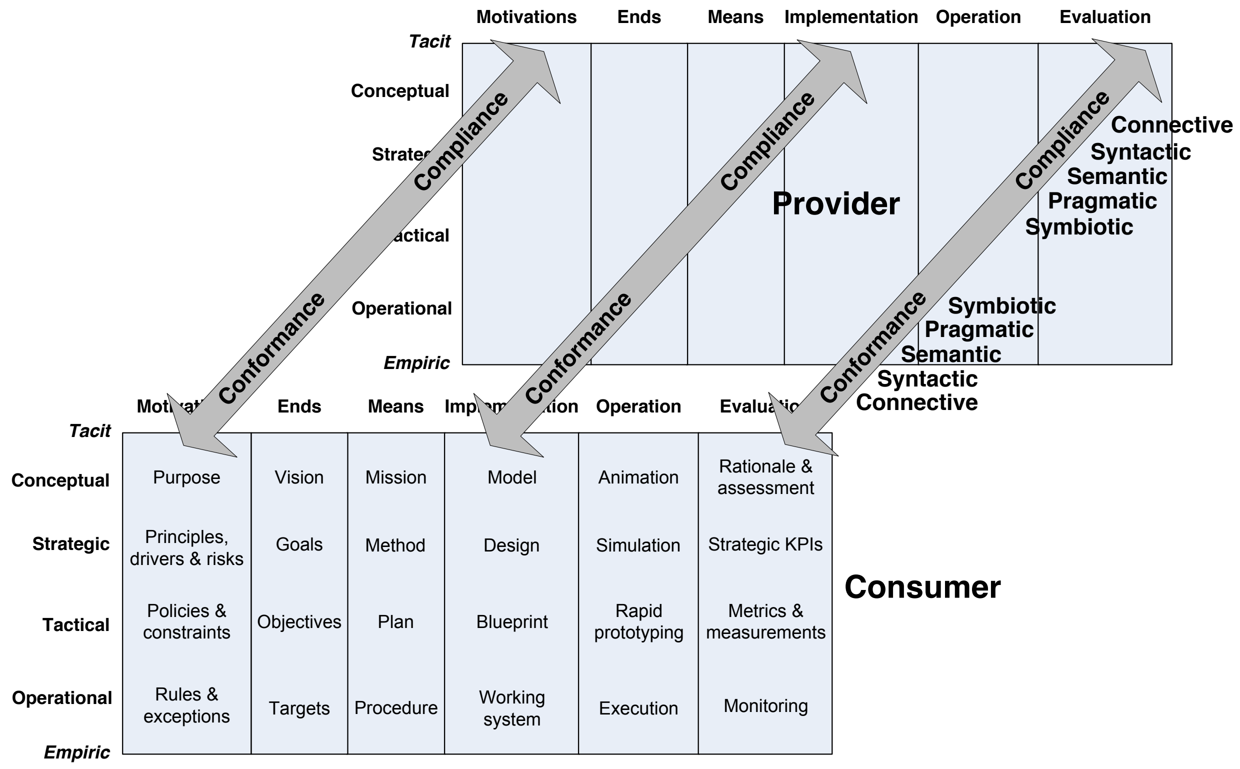
- Value-chain, partnership, joint-venture and subsidiary, corresponding to the four symbiotic layers in Table 4;
- Service, corresponding to the Service layer but without the Semantic layers (which are tacit);
- Semantic service, which now includes at least part of the Semantic layers;
- Normative, which involves semantics but not pragmatics (used for documentation).

Combining interoperability and enterprise architecture in a framework

How does the linear interoperability framework of Table 4 fit the multidimensional enterprise architecture framework of Figure 3? In other words, how can we combine both into just one multidimensional interoperability framework?

Figure 6 illustrates the answer to this question. The basic tenet is to recognize that interoperability, one of the possible concerns in the enterprise architecture framework of Figure 3, fulfills all the criteria, stated in section "A multidimensional architecture framework", to be promoted to an axis of its own.

Figure 6. A multidimensional interoperability framework. Only three axes are shown and the provider has been simplified.



The interoperability axis is orthogonal to the others and relates two enterprise architectures (one as consumer, the other as provider) in corresponding features. For simplicity, Figure 6 omits the Concerns and Evolution axes and considers only one plane (Lifecycle and Concreteness axes) for each enterprise architecture. These planes are generic, but they should correspond to the same concern in both enterprise architectures. Again, for simplicity, the details of the provider's plane are omitted but are identical to the consumer's.

The interoperability axis, which relates the two planes, has two directions: compliance (consumer to provider) and conformance (provider to consumer). This is an innovation with respect to other axes and stems directly from the dichotomy between compliance and conformance. This axis is discretized into the interoperability layers of Table 4, in both directions. For simplicity, only the interoperability categories are shown. The highest level is represented at the origin (the middle) of the axis, much like the origin of the Lifecycle and Concreteness axes in Figure 2 are represented at top left of the plane. What makes sense for any method is to start at the origin of the framework at the highest abstraction level and progress outwards by refining and detailing all axes.

Figure 6 raises some interesting questions, such as:

- Where should interoperability fit (in what perspective and at what concreteness level of the planes)?
- How do we assert the orthogonality of the axes, or why the interoperability layers are not simply the concreteness axis applied to the interoperability concern?
- How do we cater for the fact that interoperability is usually not dealt with in an uniform and monotonic way across the range of layers? For example, the pragmatic, syntactic and connective layers are the easiest to tackle, with semantic and symbiotic typically left to documentation only;
- Where do we fit in non-functional concerns, such as security and legal issues?

To answer these questions, the interpretation of Figure 6 should be done in the following way:

- Interoperability makes sense in all cells of the consumer and provider planes (not just in the Operation stage) and should be done on a homologous base. For example, the Purpose (motivation, at a conceptual level) of the consumer to send a request to the provider must be compliant with the Purpose (homologous cell) of the provider in accepting and honoring that request. Naturally, the same needs to happen in the opposite direction, but now in relation to conformance. Identical reasoning can be made for other cells, such as Goals, Plan and Working System. If interactions actually occur in the Operation stage, what occurs is a consequence of what has been designed, which means that the interoperability must be considered from the initial stages of the lifecycle. Even the Evaluation stage makes sense for interoperability, by using the adequate metrics (such as those relative to SLAs) to measure how well the interoperability works;
- Although both interoperability and concreteness axes progressively deal with an increasing amount of details (as other axes), they refer to different issues. The concreteness axis measures the refinement in the development of the system, from fuzzy specifications (many decisions yet to take) to full details (everything is known). The interoperability axis is just a way of dealing with complexity, a zoom in or out of the interoperability aspects, according to the various layers. The concreteness level at which each interoperability layer is defined will certainly vary during the executing of the method used to develop the system, but the hierarchical organization of the interoperability layers can be the same right from the start;
- The orthogonality of the interoperability and concreteness axes is also the answer to the third question. Some interoperability layers are easier than others to deal with. Communication protocols, syntax formats and code entail the most well-known aspects, with semantics and organizational issues typically dealt with in a tacit (implicitly assumed) or empiric (based on existing specifications, such as standards) way. The framework must be able to express this reality, because the designers are unable to specify everything at once and intermediate versions must be workable until improvements can be made. Each interoperability layer can be specified at any concreteness level, from tacit to empiric, as illustrated by Figure 6;
- The enterprise architecture framework of Figure 3 includes one plane for each concern, be it functional or non-functional. This means that the interoperability framework interconnects each pair of corresponding planes in each enterprise with the interoperability axis. Figure 6 represents only one of these pair of planes, but which is generic and applicable to any concern. Security, for example, is like any other concern: it must follow the lifecycle stages, be described at various levels of details and respect the compliance and conformance rules as described above, otherwise interoperability will not be possible in a secure way. Equivalent reasoning can be made about legal issues, including regulatory compliance, or any other concern.

Interoperability maturity model

The main purpose of an interoperability maturity model is to express how well a given enterprise explores the potential of interoperability when interacting with another enterprise. This is typically done in a simple scale, from not being able to interoperate at all to full knowledge of the partner's characteristics relevant to interoperation.

Table 2, in the Background section, summarizes several existing maturity models, which typically use five levels. The number of levels used to discretize the maturity of an enterprise interaction is really not that important. The relevant aspect is that each level brings something substantially new and that the granularity of the levels is not so fine or so coarse that discrimination between levels becomes doubtful or difficult.

In our framework, the maturity level of an enterprise interaction expresses how high it climbed the ladder of interoperability layers of Table 4 before tackling layers at the tacit level. Recall that all layers are always present, although some are dealt with empirically and others tacitly, at the lower and upper extremes of the range, respectively. This means that the interoperability categories in Table 4 provide a natural classification of maturity level, in a scale of six levels, without the need to resort to a different set of level names. Table 5 expresses this scale. A given interaction is classified in the highest level that it deals with explicitly. If needed, the layers of interoperability in Table 4 provide a refinement of this scale.

Table 5. Interoperability maturity levels

Level	Name	Description
5	Symbiotic	Partners understand the nature of the interaction and the intent, purpose and goals of the interlocutor
4	Pragmatic	Partners are able to specify their reaction to messages and transactions
3	Semantic	Partners understand the meaning of the a message and reason on it, but not the reaction of the interlocutor to it
2	Syntactic	Partners can only understand the structure of the messages, but not the meaning of its components
1	Connective	Partners can exchange messages, but do not understand the format or meaning of their content
0	Incompatible	There is no compliance and conformance at any level, not even basic communication protocols

The maturity of interoperability is also an expression of the degree attained in the relationship between several fundamental concepts, namely:

- The enterprise architecture is the main architectural component, which defines the enterprises that need to interoperate;
- The interoperability framework provides a means by which the enterprise architectures can be analyzed and systematized from the point of view of their interaction;
- The method (described in the next section) to use the framework to achieve interoperability between two enterprise architectures.

The fact that Table 5 mimics the Category column of Figure 4 is no coincidence. True enterprise integration, up to the highest abstraction levels, can only be achieved through proper design or adaptation of the enterprise architectures, by using a framework and methd such as those proposed in this chapter, which consider all the main aspects and dimensions involved right from the motivations (even before the conception and design) and not as an afterthought solution.

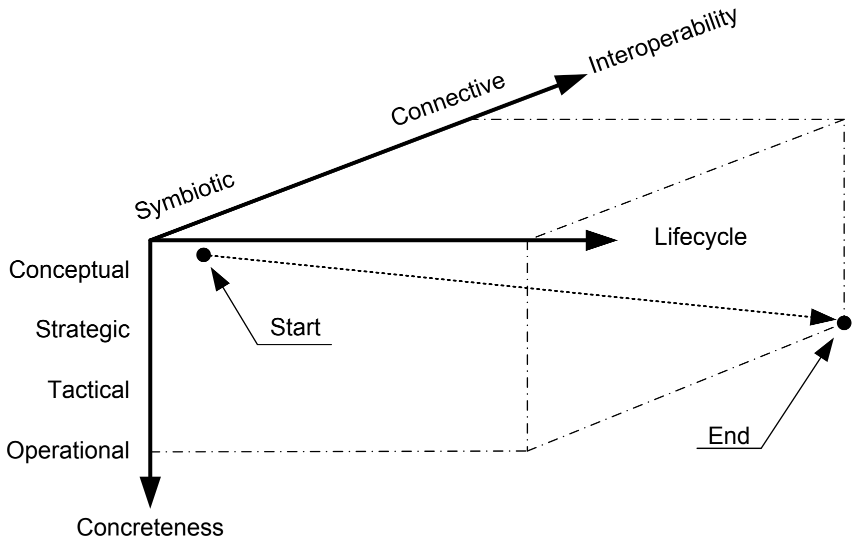
The linear structure of Table 5, with just one value, is a simplification of what the framework can provide. Looking at the maturity concept as an assessment indicator, we could define a more elaborate maturity model, with one indicator for each of the axes in the framework. For simplicity, this is not done in this chapter.

THE METHOD TO ACHIEVE INTEROPERABILITY

A method is needed to exercise the multidimensional interoperability framework, with the goal of designing, implementing or improving the interaction between two enterprises. The basic goal is to move every axis from its origin (essentially, ideas) to the other extreme, where everything is refined and established, eventually recycling the design through successive versions.

Figure 7 illustrates this, by representing the main axes of the framework for each concern. For simplicity, the Concerns and Evolution axes are not represented and compliance and conformance have been represented in the same direction in the Interoperability axis. The method is pictured as the path taken from the initial steps, near the origin and with very little detail (referred to as *start*), down to the last stages (Operation, Evaluation), full level of concreteness and lowest layer of interoperability (referred to as *end*). Unlike what Figure 7 seems to imply, this path is most likely not linear.

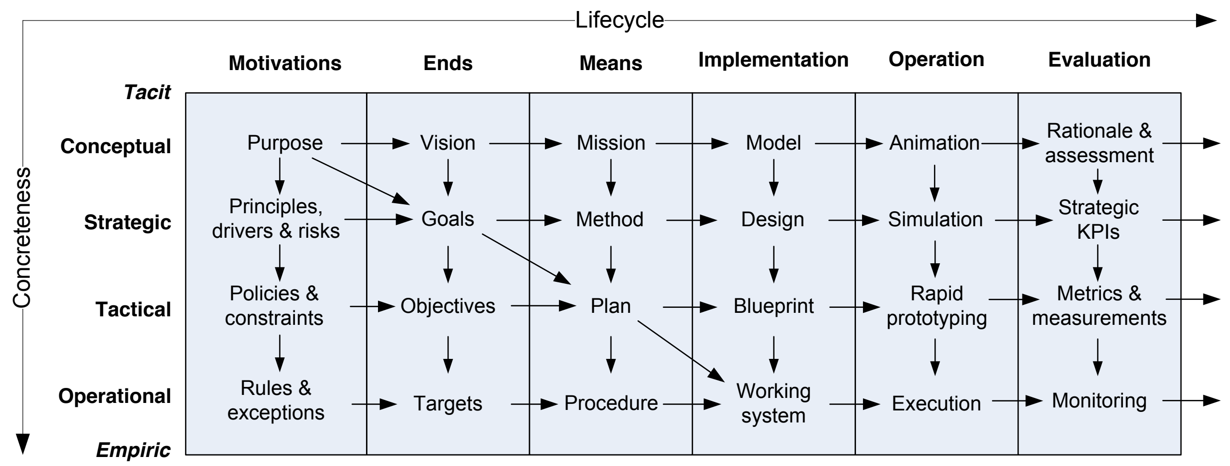
Figure 7. The method as the path taken along the axes of the framework.



The method should be model driven, with the lifecycle as the main guide, based on the architectural and fulfillment loops of Figure 1. The *consistency*, *alignment* and *traceability* throughout these loops are fundamental. A stage S_i must be derived from the previous one S_{i-1} by adding further detail and turning abstract aspects into concrete ones, but maintaining conformance to S_i. This means supporting at least all the requirements imposed on S_i, forward propagating any constrains and adding new ones, while maintaining the ability to trace back constraints to the stage that took the decisions that originated them.

Figure 8 represents the front plane of Figure 7 and illustrates the possible transitions along the lifecycle, in this plane. Each cell represents one stage at a given level of concreteness and the arrows represent possible transitions between cells. Only rightwards and downwards transitions are represented, but the evaluation stage can loop leftwards to the motivations stage, as indicated in Figure 1.

Figure 8. Transitions along the front plane of the framework.



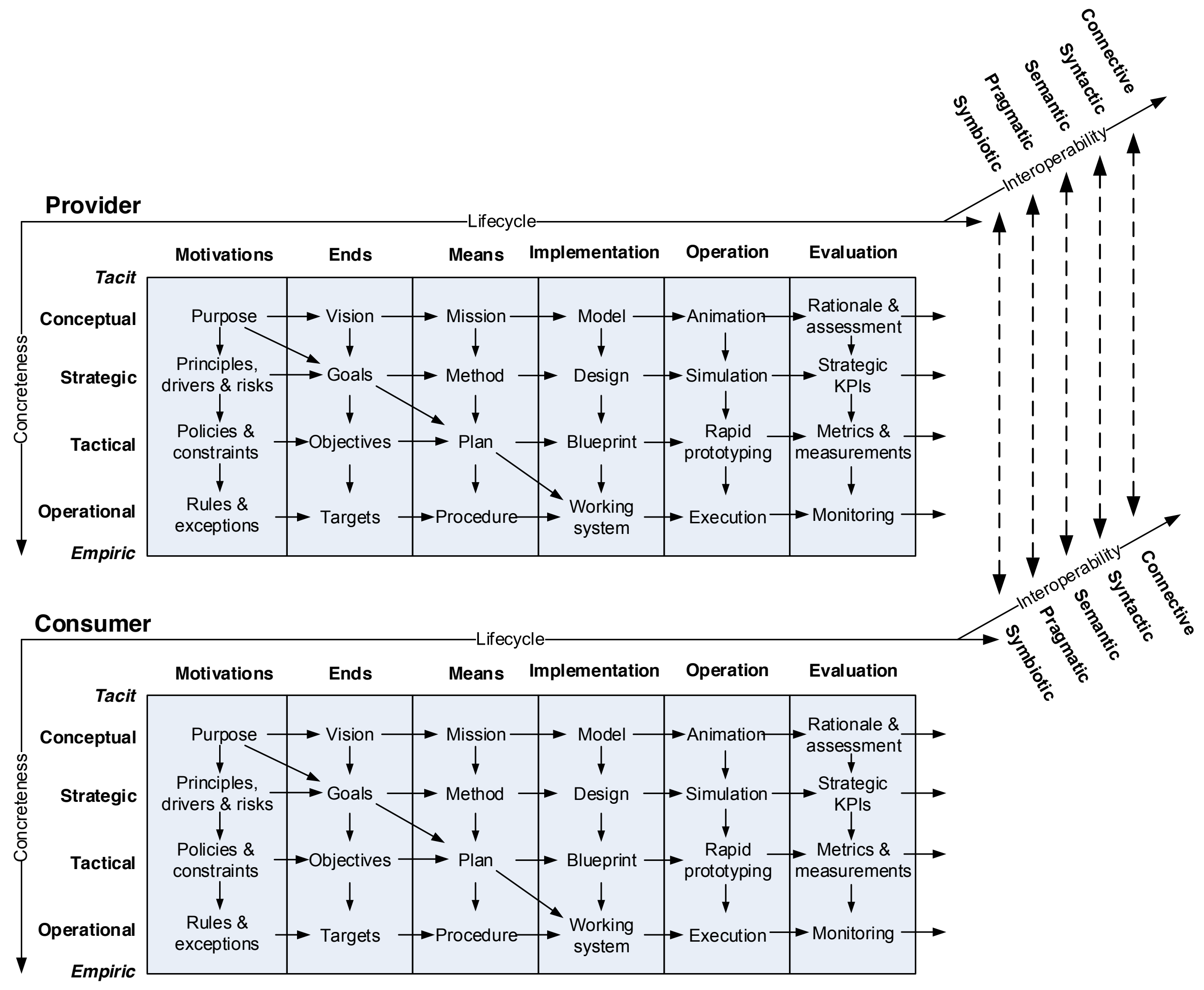
Transitions are not limited to vertical or horizontal moves, not even to adjacent cells, but that means leaping over intermediate steps and therefore greater effort and less support (from the development method) in performing those transitions. Figure 3 exemplifies this by showing direct transitions from *Purpose* to *Goals*, *Goals* to *Plan* and *Plan* to the *Working system*. Usually, this corresponds to laying down goals and defining and programming processes directly from them.

We describe three of the possible paths in Figure 8:

- *Breadth-first*. Go along the conceptual level until the *Implementation* column is reached and then plunge until the *Working system*. This is the worst path. Initially all goes well, since we are evolving only with concepts, but when we want to detail we have no support from intermediate steps, so we need to decide everything on the spot and on the fly, without a guiding map;
- *Depth-first*. Work as much as possible at the *Motivations* column, successively detailing from Purpose down to *Rules & exceptions*, and then follow right at this detailed level until the *Working system*. This seems the best path, since everything is thoroughly justified, but crashes into complexity, loses sight of overall picture and takes the risk of deciding everything at high level, unaware of whether the Means support those decisions;
- *Straight line*. Traverse the front plane in a straight line (*Purpose*, *Goals*, *Plan* and *Working system*). By evolving along the Lifecycle axis at the same time that concreteness is increased, this is a tradeoff between the previous two paths.

Equivalent reasoning can be made when introducing the Interoperability axis in two interacting partners (provider and consumer), as illustrated by Figure 9 (which represents this axis on the right side just to avoid cluttering). Interoperability between the two partners must be achieved in every cell of the front plane and in all layers of interoperability. The dashed lines express compliance in the consumer to provider direction and conformance in the provider to consumer direction.

Figure 9. Confronting the interoperability frameworks of two enterprises as part of the method to achieve interoperability.



Again, a breadth-first approach can be followed, by staying in the symbiotic layer until the working system is reached and only then dealing with the details of the lower interoperability layers. A depth-first approach, more radical, would involve agreeing on interoperability details right from the conception of the interaction between the partners. A more balanced approach would be following the axes in diagonal, refining them as design moves along.

In practice, the depth-first approach is not so radical. Since interoperability is a difficult issue that in many cases is built as an afterthought over existing enterprise architectures, in many cases existing integration technologies (such as Web Services and REST) take a central role in the method and impose on the remaining aspects, which usually works as a pragmatic approach but diminishes agility and the sustainability of interoperability.

Figure 9 is not the full picture. It should be repeated for every concern, functional and non-functional (see Figure 3) and everything repeated again along the Evolution axis, by iterating the lifecycle into a new version. Since two or more interacting partners are involved, the method would still have to be applied to each of them, but considering their interaction and the influence of each of them on the others (the choreography).

This is a complex scenario and the most adequate path to follow will depend a lot on the enterprise architectures and on the level of integration sought, but a reasonable approach would be to step along the following guidelines:

- The interoperability method should complement the existing enterprise architecture method, not replace it. The cells in the front plane of Figures 8 and 9 should be reasonably identifiable in that method, providing a mapping between the front plane and the framework underlying that method;
- Most likely, the enterprise architecture method will transition from the Purpose to the Working System cells. In each transition, consider the Interoperability axis. This means that compatibility between the consumer and the provider must be checked in cells such as Purpose, Goals, Plan and Working System (if the straight-line path is followed);
- Consider the interoperability layers as low as needed. For example, it is natural to check only at the symbiotic level that the purposes of both enterprises are compatible, but the goals will probably rely on different ontologies and the plan may depend on the architectural model (such as SOA or REST) used to achieve interoperability;
- Consider the most important concerns right from the start, but the rest should be introduced as needed and as possible. Complexity should be dealt with gradually;
- Use interoperability standards (such as Web Services) whenever possible and as adequate.

In practice, Figure 9 does not entail designing the enterprise architectures from scratch, but rather adaptations or the design of adapters and mediators. The method is applicable to a full system or to a subsystem, as well as to a design from scratch or to an adaptation that involves another iteration in the enterprise architecture.

The method also needs to consider the issue of what should come first when modeling and designing: conformance or compliance. Should a consumer comply with an already existing provider, adapting to the provider's interface requirements, or should a provider conform to a role previously specified by a choreography?

Given the discoverable nature usually attributed to the service paradigm, many authors tend to opt for the second option (Diaz & Rodriguez, 2009), in a polymorphic setting. Given a role in a choreography, any discovered service that is found to be conformant to that role can be used to fulfill that role. However, given the variability of software services, in particular if non-functional aspects are considered, how many services can we expect to conform to an independently specified role?

A more realistic scenario, even in a distributed context, considers that designing a system is a combination of top-down and bottom-up strategies, in which the expectations of the system have to be matched against and mapped onto already existing subsystems and the services they offer, with new services needed when there is no match or mapping. This means that conformance (bottom-up) and compliance (top-down) have to be dealt with together, in an overall strategy that constitutes the art and science of design.

COMPARISON WITH OTHER INTEROPERABILITY FRAMEWORKS

The usefulness of any framework lies in its expressive power. It should be able to describe any instantiation of the problem domain for which it has been conceived. The objective of this section is to show how several interoperability frameworks can be mapped onto ours and be described by it. Although not an exhaustive exercise, it gives a hint on how these mappings can be undertaken.

We start with the frameworks that express a linear perspective of interoperability, such as those described by Table 1. This table presents one dimension of level of detail, from high-level and abstract to low-level and concrete. This suggests a vertical composition mechanism, in which each layer is composed of artifacts in the layers below it. However, the following difficulties arise:

- Composition alone is not sufficient to fully describe a layer in terms of the artifacts in lower layers, particularly in the higher layers (social, cultural, ethical, legal, etc.);
- Not all interoperability layers are dealt with at the same time. Some are more dynamic and pertain to operation time, whereas others are more static and must be dealt with at conception or design time;
- An enterprise (or its subsystem relevant to interoperability) is not a monolithic artifact with a single facet. It has several slants and aspects, under which interoperability may need differentiated treatment. This is usually the case of non-functional aspects, such as reliability, security and context-oriented behavior.

Multidimensional interoperability frameworks, such as those mentioned in the Background section, have appeared to cater for this diversity and variability. Some give more relevance to enterprise architecture than others do, but most are heavily influenced by the interoperability problem-solving issue, which corresponds to the method of making two enterprises or services interoperable. In a way, these frameworks start from the real world, in a set of contexts and domains, identify which types of artifacts need to be made interoperable and conceive an architectural space of dimensions oriented towards the problems to be solved in those contexts and domains.

Our approach is different, guided by the following principles:

- To consider the interoperability as a generic system problem, context and domain independent, but contemplating interacting systems of arbitrarily high complexity, such as enterprises;
- To introduce real world contexts and domains as instantiations of the generic interoperability problem, by not only using the provisions of the framework itself (dimensions, namely Concerns) but also the method that exercises it, to solve a given problem;
- To clearly separate the framework (the organization and classification scheme of the various aspects of the enterprises that are relevant to interoperability) from the method (the plan to make two or more enterprises interoperable). In this chapter we focus on the framework, although the method is generically described;
- To elect the enterprise architecture as the foundation ground for interoperability, by starting with a set of orthogonal dimensions, as illustrated by Figure 3, with lifecycle stages and levels of concreteness as the main structuring guidelines;
- To organize interoperability as a mapping from one enterprise architecture to another, as expressed by Figure 6. In other words, each interoperability issue arises from the need to relate the corresponding issues of two interacting enterprises;
- To separate clearly what is relevant to the interoperability mechanism proper, in a domain independent way (which corresponds to understanding the intent, content and transfer mechanism of a message, as well as the reaction of the interlocutor, in the context of a generic transaction), from what is domain specific (such as social, cultural, ethical and legal issues);
- To separate clearly the roles of consumer and provider in an interaction, which discriminates compliance from conformance, instead of considering generic, symmetric relationships.

Under these principles, the mapping of the interoperability frameworks of Table 1 in our framework is done essentially in the following way:

- OSI (1994), C4IF (Peristeras & Tarabanis, 2006), Lewis (2008) and LCIM (Wang, Tolk & Wang, 2009) have a straightforward mapping, with essentially similar layers at the lower levels and with our framework detailed the higher layers in these frameworks. In the case of LCIM, the Dynamic layer corresponds roughly to the Symbiotic category of layers in Table 4 and the Conceptual layer, dealing with documentation, is mapped onto the left side of the Lifecycle-Concreteness plane of our framework, in which the system is designed and models and documentation are produced;
- In the framework of Stamper (1996), there is a good mapping to our framework up to the Pragmatics layer. The Social world layer is mapped onto a plane in the Concerns axis of our framework;

- The political and legal layers in the European Interoperability Framework (EIF, 2010) are also mapped onto planes in the Concerns axis;
- Something similar happens in the case of the framework of Monfelt (2011). The Legal, Ethical and Cultural layers are mapped onto planes in the Concerns axis. The SWOT analysis belongs to the method, which means that it is best mapped onto our method, by performing the analysis in the context of several Concerns planes.

The mapping of the multidimensional frameworks can be done in the following way:

- In the ATHENA framework (Berre *et al*, 2007), the conceptual integration, with the levels of enterprise/business, process, service and information/data, has a straightforward mapping onto our interoperability layers, in Table 4. The application and technical integrations, however, involve activities that are best mapped onto the method, not the framework;
- In the framework proposed by Chen (2006), the concerns dimension (business, processes, services and data) correspond to our interoperability dimension and are mapped onto layers in Table 4. The other dimensions, the barriers to overcome and the approach taken fit better the method than the framework. Nevertheless, there is an interesting mapping from the types of approach to the higher layers of our interoperability dimension, in Table 4. These types are integrated (a common format is used by all systems), unified (common goals and semantic mapping) and federated (there may be common goals and ontology, but each system must adapt to others dynamically, without any imposed model). These can be mapped onto interoperability profiles and therefore be described by the framework prior to exercising the method;
- The SOSI framework (Morris *et al*, 2004) include the lifecycle dimension, which can be easily mapped onto the Lifecycle dimension in our framework;
- The framework of Ostadzadeh & Fereidoon (2011) is based on an enterprise architecture framework, like our own. The Zachman questions map partly onto the Lifecycle stages (Motivations is why, Ends is what, Means is how) and partly onto the Concerns axis (who, where and when). The layered interoperability concerns (contextual, conceptual, logical, physical and out-of-context) map directly onto our interoperability layers in Table 4 and the remaining dimension is identical to SOSI (expressing the lifecycle), with the addition of a cultural aspect, which we map onto a Concerns plane.

The project ENSEMBLE (Agostinho, Jardim-Goncalves & Steiger-Garcao, 2011) follows a methodology to establish a scientific base for enterprise interoperability, rather than an interoperability framework. In this respect, the ENSEMBLE's approach is complementary to ours, coming from the opposite direction. Essentially, it adopts a top-down approach by gathering information on the real world domains that pose interoperability problems and then deriving the scientific base to solve them. Our approach, bottom-up, entails establishing a rationale for what interoperability is, as an universal concept relating entities of any level of complexity and domain, and deriving a generic framework that allows us to structure and to organize the aspects in real world interoperability problems, as a tool that can be a precious aid in solving them.

The main challenges in the top-down approach are:

- To deal with the massive scale of interoperability problems;
- To be able to identify which are the basic interoperability elements (the equivalent of a periodic table) that underlie them all in an orthogonal way;
- To derive (de)composition patterns that are able to map typical problems onto those elements, so that a set of commonly used techniques can be applied to solve them.

ENSEMBLE has defined twelve scientific areas of interoperability (Popplewell, 2011), which can be organized into the following categories:

- Data, process, rules and knowledge have a domain and system independent perspective;

- Objects, software, services, cloud and ecosystems are domain independent but refer to systems, with interoperability problems similar to any other system, albeit with varying levels of complexity;
- The cultural, social networks and electronic identity levels are domain dependent, with social networks including a system perspective.

Other areas in which interoperability is relevant and encompassing, such as security, need to fit in and increase the complexity of the scenario.

This variability emphasizes the need to separate concerns, striving for the orthogonality between the meaning of each layer of interoperability, area of concern, domain and class of systems. Therefore, we see the top-down approach as an application-driven effort that ensures that the interoperability problem scenario is complete and the bottom-up approach as an effort to structure that scenario by identifying the orthogonal aspects involved.

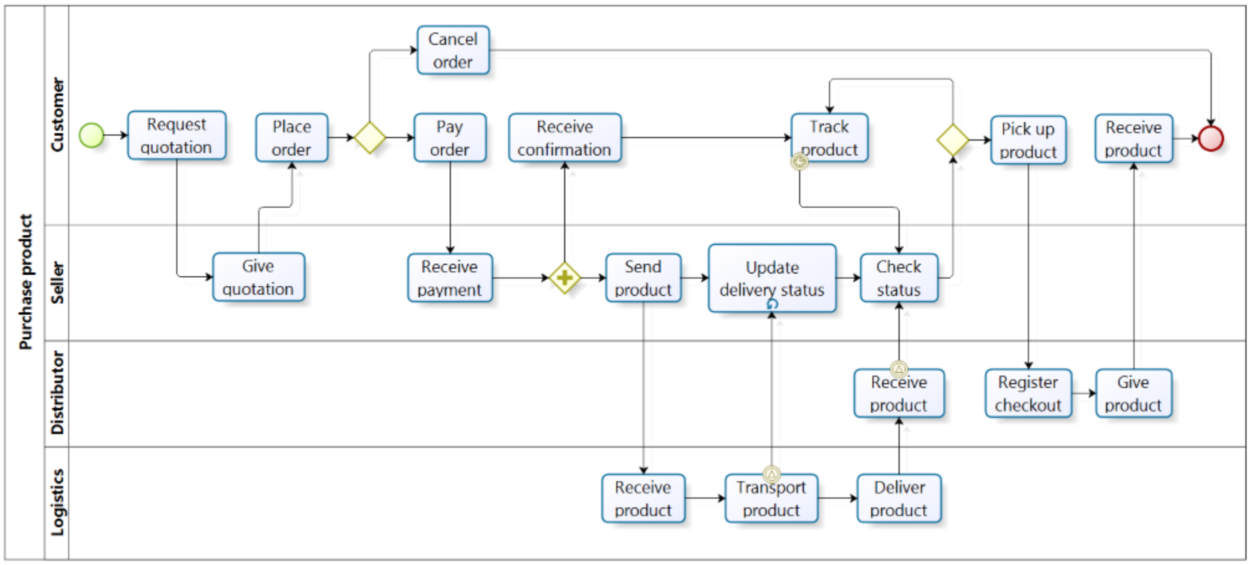
The taxonomy developed by the ENSEMBLE project can be mapped onto our framework in the following way:

- Domain independent interoperability areas (data, process, rule and knowledge) map directly onto interoperability layers in Table 4;
- System/module oriented areas (objects, software, services, cloud and ecosystems) map onto system/module composition (Concreteness axis) and interoperability layers;
- Domain oriented areas (cultural, social networks and electronic identity), as well as applications and issues in the three main neighboring domains (social, applied and formal sciences) map onto the Concerns axis;
- Of the scientific elements of interoperability:
  - Semantics maps onto the corresponding category in Table 4;
  - Orchestration maps onto the pragmatic and particularly the symbiotic layers;
  - Models and tools map onto interoperability profiles, which may or may not include actual message exchange, depending on whether the goal is just to have format compatibility (compliance and conformance) or actual interoperation, in tools.

A SIMPLE WRAP-UP EXAMPLE

Figure 10 considers a simple example of a collaboration between several partners to achieve some common goals. A customer requests a quote for a product from a seller, places an order, pays for it and picks it up at a local distributor when it arrives. The collaboration involves the customer, the seller, the distributor and a transporter. We assume that the customer is a company and electronic interaction is done by services that have their API available.

Figure 10. A simple example to illustrate the interoperability framework and method.



The typical way of solving the interoperability problem in such an example entails the following:

- The customer searches for a seller providing the product(s) needed;
- The customer downloads the seller's API (in either SOA or REST) and develops a client matched for it, as well as the necessary adapter(s) to enable its enterprise architecture to deal with this client;
- If semantic information is available, the customer may use it to improve the programmatic treatment of interoperability;
- Same thing for the interface with the distributor;
- The higher levels of interoperability are dealt with manually, through documentation;
- If the seller or the distributor change something in the functionality of their interfaces, the client program at the customer needs to be changed accordingly (to reflect the new schemas).

This is a classical view of interoperability as integration, in which the schema of the data exchanged is the centerpiece and partners are made interoperable for the full variability range of that schema, even if the interaction actually exercises only a fraction of that variability. This leads to unnecessary coupling.

The framework and method described in this chapter propose a minimalist vision, reducing interoperability to the minimum coupling needed and, at the same time, widening the range of partners with which interoperability can be established.

To illustrate this claim, consider the interoperability needed between the customer and the seller, in Figure 10. For the customer, this involves not only finding a suitable seller but also avoiding lock-in with that seller, by allowing the seller to be replaced by another with a compatible interface.

Current technologies, such as Web Services and the tools that deal with them, have not been conceived with compliance and conformance in mind, but through an example we can give an idea how that can be done. Consider the seller as a Web Service offering several operations, including one (which we will call `getQuote`) that allows the customer to implement its RequestQuote activity in Figure 10.

Program 1 contains a fragment of the seller's WSDL file, showing only the relevant parts for the interface. Only the operation `getQuote` is depicted, to keep it simple. It receives the specification of a product (the model and the seller's department it belongs to) and the information returned includes a quote for each product of that model found, with an optional boolean that indicates whether that product is in stock. A client is needed at the customer to deal with this Web Service.

```
<types>
     <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
               targetNamespace="http://example.com/schema/seller1"
               xmlns="http://example.com/schema/seller1"
               elementFormDefault="qualified">
          <xs:element name="product" type="Product"/>
          <xs:element name="prodInfo" type="ProdInfo"/>

          <xs:complexType name="Product">
               <xs:sequence>
                    <xs:element name="model" type="xs:string"/>
                    <xs:element name="department" type="xs:string"/>
               </xs:sequence>
          </xs:complexType>
          <xs:complexType name="ProdInfo">
               <xs:sequence minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="ID" type="xs:string"/>
                    <xs:element name="cost" type="xs:decimal"/>
                    <xs:element name="inStock" type="xs:boolean"
                                   minOccurs="0"/>
               </xs:sequence>
          </xs:complexType>
     </xs:schema>
</types>
<interface name="Seller_1">
     <operation name="getQuote"
                    pattern="http://www.w3.org/ns/wsdl/in-out">
          <input messageLabel="In" element="seller1:product"/>
          <output messageLabel="Out" element="seller1:prodInfo"/>
     </operation>
</interface>
```

Program 1. A fragment of the WSDL of the seller's Web Service.

Now, in Program 2, consider the equivalent WSDL fragment in another seller's Web Service, compatible with this one but not identical. Its `getQuote` operation provides the same overall functionality and has the same name, so that a syntactical interface match is possible (for simplicity, this example does not tackle semantics). It also obtains a quote on the product, specified by a description and the product category it belongs to, and returns up to three pairs of product ID and respective price. The description can be made with one or two strings, so that brand and model, for example, can be separated if required, and the category is optional but supports two strings so that the product's division and unit, for example, can be specified.

```
<types>
     <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
               targetNamespace="http://example.com/schema/seller1"
               xmlns="http://example.com/schema/seller2"
               elementFormDefault="qualified">
          <xs:element name="productSpec" type="ProductSpec"/>
          <xs:element name="productInfo" type="ProductInfo"/>

          <xs:complexType name="ProductSpec">
               <xs:sequence>
                    <xs:element name="description" type="xs:string"
                                   minOccurs="1" maxOccurs="2"/>
                    <xs:element name="category" type="xs:string"
                                  minOccurs="0" maxOccurs="2"/>
               </xs:sequence>
          </xs:complexType>
          <xs:complexType name="ProductInfo">
               <xs:sequence minOccurs="0" maxOccurs="3">
                    <xs:element name="productID" type="xs:string"/>
                    <xs:element name="price" type="xs:decimal"/>
               </xs:sequence>
          </xs:complexType>
     </xs:schema>
</types>
<interface name="Seller_2">
     <operation name="getQuote"
                    pattern="http://www.w3.org/ns/wsdl/in-out">
          <input messageLabel="In" element="seller2:productSpec"/>
          <output messageLabel="Out" element="seller2:productInfo"/>
     </operation>
</interface>
```

Program 2. A fragment of the WSDL of another seller's Web Service.

These two WSDL fragments correspond to two different schemas and the usual way to invoke the respective services would be to generate two different clients, one for each service. However, by looking at both the WSDL fragments, we can notice that:

- The `In` element of the `getQuote` operation in Program 1, of type `Product`, complies with the `In` element of the operation in Program 2, of type `ProductSpec`. In other words, supplying a `model` and `department` with one string each is within the requirements of the operation in Program 2;
- The `Out` element of the operation in Program 2, of type `ProductInfo`, conforms to the `Out` element of the operation in Program 1, of type `ProdInfo`. This means that all the values returned by the operation in Program 2 will be expected as if they were returned by the operation in Program 1.

The conclusion is that, although the services are different, a client generated to invoke the service that Program 1 belongs to can also be used, without changes, to invoke the service in Program 2, as long as both `getQuote` operations are consistent in the interoperability layers above Syntactic.

Compliance and conformance are checked structurally (Delgado, 2012), in which complex types are checked recursively, component by component, until primitive types are reached. Corresponding components need to either have the same name in the same ontology or be mapped one to the other when reconciling ontologies. In the general case, compliance and conformance must hold at all interoperability layers, as shown in Figure 9.

The example of Figure 10 can also be used to argue that the multidimensional framework proposed in this chapter is better than one-dimensional frameworks such as those in Table 1, essentially due to the following reasons:

- The enterprise architecture and its lifecycle are always the starting point, guiding the development of interoperability in a model driven way. This is actually nothing new as a good practice, but the organization presented for the front plane in Figure 8, coupled with the rationale for the method, with the set of desirable transitions, provide a modeling map that will emphasize where the relevant subsystems fit and where interoperability is needed. Interoperability is clearly not a mere operational issue;

- Making compliance and conformance explicit at all layers, in Table 4, but in an integrated way with the enterprise architecture, as expressed by Figure 9, reminds the designers of that enterprise architecture that no enterprise is isolated and interoperability is an integral part of that enterprise architecture as well;
- An enterprise architecture is not limited to electronic services, but includes physical resources and even human roles, which means that interoperability must be ensured in these areas as well. As an example, the products transported in the Transport process in Figure 10 must comply with the containers used, and these must conform to what the products to transport need;
- Another aspect is the overall choreography expressed in Figure 10. The role performed by each enterprise (as determined by its enterprise architecture) must comply with (as a consumer) and conform to (as a provider) that choreography.

The main conclusion is that interoperability and enterprise architecture are not dissociable.

CONTRIBUTIONS TO A SCIENTIFIC BASE FOR ENTERPRISE INTEROPERABILITY

The development of a scientific base for any domain involves necessarily the systematization of thought in that domain. This chapter seeks to complement existing efforts towards this purpose (Jardim-Goncalves, Grilo, Agostinho, Lampathaki & Charalabidis, 2013) by trying to discover which are the fundamental concepts underlying interoperability, much in the same way as Physics goes deep down to the smallest scale to understand the Universe in the largest scale.

In the general sense, interoperability stems from two conflicting goals:

- Systems need to interact to accomplish collaboration, either designed or emergent. Mutually understanding partners creates dependencies on others that hamper system evolution;
- Systems need to be independent to evolve freely and dynamically. Independent systems do not understand each other to be able to interact.

Enterprises are (complex) systems and have exactly the same issue, with the additional problem that agility (which translates to fast evolution) is a critical requirement to the survival of even the largest enterprises. These are actually quite vulnerable, because their complex architecture has woven a large web of dependencies.

This chapter contends that, for systems in general and enterprises in particular:

- Compliance and conformance, expressed in Figure 5, are the two basic concepts that underlie all interactions;
- The basic reusable pattern is the transaction, as illustrated by Figure 4a;
- There is a set of primitive message types, as depicted in Table 3, which by composition can yield a set of primitive transaction patterns, which by composition in turn can yield arbitrarily complex choreographies;
- Compliance and conformance extend into non-operational stages of the lifecycle, in which case what is involved is the use of a specification, rather than message interaction, as shown in Figure 5a. Interoperability is then extended into the concept of compatibility. Regulatory compliance is an example;
- Partial compliance and conformance constitute a solution to the conflicting goals mentioned above, allowing to tune up the desired level of interoperability according to the constraints and requirements of the problem at hand. Only the features of the interacting systems relevant to interaction, as depicted in Figure 5b, contribute to dependencies. The principle of substitution, applicable here, shows that inheritance and polymorphism, typical of programming languages, is just another expression of compliance and conformance.

The importance of compliance and conformance can be seen in various examples:

- The fact that interoperability is inherently asymmetric becomes clearer and more explicit. In the literature, interoperability is commonly taken as symmetric and reversible, but this is just a consequence of the fact that an enterprise takes part in choreographies sometimes as a consumer, other times as a provider;
- Partial compliance and conformance reduce coupling between systems because less restrictions are imposed on the systems. This compares very favorably with an interoperability based on schema sharing, such as when using XML, in which interoperability needs to be maintained for the full spectrum of variability of the schema, even if only a very small subset of document types are exchanged;
- Sustainable interoperability is easier with partial compliance and conformance, because they support polymorphism and the substitution principle in a distributed fashion, as Programs 1 and 2 illustrate. The basic tenet is that, by reducing coupling to a bare minimum, all the remaining features are free to change. This increases the ease of design (the range of suitable interacting partners increase), adaptability (adaptations can also be partial and as needed), changeability (the features with dependencies on others are in a smaller number and more explicit) and reliability (if a partner fails, it is easier to get an alternative which ensures compliance and conformance). Delgado (2012) established coupling and adaptability metrics which show the benefits of partial compliance and conformance;
- Strategic alignment is usually seen as a requirement for enterprise interoperability and collaboration. Again, this is just too much coupling. Their strategies and goals just need to be partially compatible (compliant and conformant), not necessarily aligned;
- Structural compliance and conformance (Delgado, 2012) constitute a solution to checking system interoperability in a distributed environment, in which the lifecycles of interacting systems are not synchronized. Using names, even if belonging to a previously agreed ontology, is not adequate due to dynamic changeability. The trick is to agree on some upper ontology (an universal set of concepts), and then to specify everything (data, behavior, goals, and so on) in terms of composition of those concepts. Compliance and conformance can then be verified by checking, recursively, the corresponding structures of the components that are relevant to the interoperability in interacting partners;

This richness of properties justifies that compliance and conformance take a central role in the framework and in the method presented in this chapter.

FUTURE RESEARCH DIRECTIONS

Compliance and conformance are basic concepts in interoperability and can be applied to all domains and levels of abstraction and complexity. Although work exists on its formal treatment in specific areas, such as choreographies (Adriansyah, van Dongen & van der Aalst, 2010), an encompassing study needs to be conducted on what is the exact meaning of compatibility (compliance and conformance) at each of the interoperability layers of Table 4. Their formal definition, across all layers, needs to be made in a systematic way, building on previous work.

The interoperability framework presented in this chapter needs to be improved and made more complete, namely in the Concerns axis, to include relevant concerns such as security and common domain-specific aspects and problems, such as those uncovered by other frameworks and those being systematized by ENSEMBLE (Agostinho, Jardim-Goncalves & Steiger-Garcao, 2011).

The method to exercise the framework needs to be detailed and structured in a systematic way, with a comparative case study regarding other interoperability methods being used today, namely those that are part of specifications promoted by relevant bodies, such as the European Interoperability Framework (EIF, 2010) or standardized, such as the Framework for Enterprise Interoperability (ISO, 2011).

The recognition of interoperability as a science, expressed by efforts such as the ENSEMBLE project, constitutes an opportunity to conduct all these efforts in a systematic and organized way, as a contribution to the establishment of a base to that science.

CONCLUSIONS

This chapter has presented an interoperability framework, conceived with the purpose of providing a systematic way to organize and structure the various aspects that interoperability entails, with orthogonality of concepts, domain independence, large-scale complexity and extensibility as the main goals. Their achievement has been sought in the following way:

- Orthogonality, by considering multiple dimensions, each for an aspect or concern orthogonal to others, with all combinations valid and, as a whole, able to support the full spectrum of applications;
- Domain independence, by basing the framework on dimensions with universal applicability (such as lifecycle, level of concreteness and layers of interoperability), with domain dependent aspects as part of a concerns dimension, orthogonal to the others;
- Large-scale complexity, by contemplating not only recursive system composition (any system is a composition of other systems) and but also very complex aspects, up to the tacit, human level of understanding and reasoning;
- Extensibility, by allowing any concern (in the dimension of concerns) to be promoted to a full dimension, if its relevance and applicability breadth justifies it (criteria for this have been described).

The approach taken is generic and bottom-up, from universal architectural concepts to domain and context dependent concerns, as a complement to the more common approach, application-driven and top-down. For example, the European Interoperability Framework (EIF, 2010) has been conceived specifically for e-Government level services. The advantage of the bottom-up approach is that it leads to an open-ended framework, with an universal core that can then be instantiated and extended at will and as required in each context and domain, in which it can be complemented by a top-down approach that systematizes the problems in each context and domain.

We hope that this framework and the approach we took serve as a contribution to the establishment of the scientific foundations of interoperability, in particular in the enterprise context.

REFERENCES

Adriansyah, A., van Dongen, B., & van der Aalst, W. (2010). Towards robust conformance checking. In Muehlen, M. & Su, J. (Eds.) *Business Process Management Workshops* (pp. 122-133). Springer Berlin Heidelberg.

Agostinho, C., Jardim-Goncalves, R., & Steiger-Garcao, A. (2011). Using neighboring domains towards setting the foundations for Enterprise Interoperability science. In *Proceedings of the International Symposium on Collaborative Enterprises* (CENT 2011).

Berre, A. *et al* (2007) The ATHENA Interoperability Framework. In Gonçalves, R., Müller, J., Mertins, K. & Zelm, M. (Eds.) *Enterprise Interoperability II* (pp. 569-580). London, UK: Springer

Bravetti, M. and Zavattaro, G. (2009) A theory of contracts for strong service compliance, *Journal of Mathematical Structures in Computer Science*, 19(3), 601-638.

C4ISR (1998) Levels of Information Systems Interoperability (LISI). C4ISR Architecture Working Group (AWG), Department of Defense. Retrieved February 25, 2013, from http://www.eng.auburn.edu/~hamilton/security/DODAF/LISI.pdf

Chen, D. (2006). Enterprise interoperability framework. In Missikoff, M., De Nicola, A. and D'Antonio F. (Eds.) *Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability*.

Chen, D. & Daclin, N. (2007) Barriers driven methodology for enterprise interoperability. In Camarinha-Matos, L., Afsarmanesh, H., Novais, P. & Analide, C. (Eds.) *Establishing The Foundation Of Collaborative Networks* (pp 453-460). Springer US

Chen, D., Doumeingts, G. & Vernadat, F. (2008) Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, 59(7) 647–659

Delgado, J. (2012) Structural interoperability as a basis for service adaptability. In: Ortiz, G. & Cubo, J. (Eds.) *Adaptive Web Services for Modular and Reusable Software Development: Tactics and Solutions* (pp. 33-59) IGI Global, Hershey PA

Diaz, G. and Rodriguez, I. (2009) Automatically deriving choreography-conforming systems of services, in *IEEE International Conference on Services Computing* (pp. 9-16), IEEE Computer Society Press.

Dietz, J. (2006) *Enterprise Ontology: Theory and Methodology*, Springer-Verlag Berlin Heidelberg

Dietz, J. and Hoogervorst, J. (2011) A critical investigation of TOGAF - based on the enterprise engineering theory and practice, in Albani, A., Dietz, J. and Verelst, J. (Eds.) *Advances in Enterprise Engineering V*, (pp. 76-90), Springer-Verlag, Berlin Heidelberg.

EIF (2010) European Interoperability Framework (EIF) for European Public Services, Annex 2 to the Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of Regions 'Towards interoperability for European public services', Retrieved June 22, 2013, from http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf

El Raheb, K. *et al* (2011) Paving the Way for Interoperability in Digital Libraries: The DL.org Project. In Katsirikou, A. & Skiadas, C. (Eds.) *New Trends in Qualitive and Quantitative Methods in Libraries* (pp. 345-352). Singapore: World Scientific Publishing Company.

Euzenat, J. & Shvaiko, P. (2007). *Ontology matching*. Berlin: Springer.

Fewell, S. & Clark, T. (2003). Organisational interoperability: evaluation and further development of the OIM model. Defence Science And Technology Organisation. Edinburgh (Australia). Retrieved February 25, 2013, from http://www.dtic.mil/dtic/tr/fulltext/u2/a466378.pdf

Ford, T., Colombi, J., Graham, S., & Jacques, D. (2007) The interoperability score. In *Proceedings of the Fifth Annual Conference on Systems Engineering Research*, Hoboken, NJ.

Gottschalk, P. & Solli-Sæther H. (2008) Stages of e-government interoperability. *Electronic Government, An International Journal*, 5(3), 310–320

Greefhorst, D. and Proper, E. (2011) *Architecture principles: the cornerstones of enterprise architecture*, Springer-Verlag, Berlin Heidelberg.

Guédria, W., Chen, D. & Naudet, Y. (2009) A maturity model for enterprise interoperability. In Meersman, R., Herrero, P. and Dillon T. (Eds.) *On the Move to Meaningful Internet Systems Workshops* (pp. 216-225). Springer Berlin/Heidelberg.

Haslhofer, B. & Klas, W. (2010). A survey of techniques for achieving metadata interoperability. *ACM Computing Surveys*, 42(2), 7:1-37

ISO/IEC (1994) *ISO/IEC 7498-1, Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, 2nd edition, International Standards Office, Geneva, Switzerland. Retrieved February 25, 2013, from http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

ISO/IEC/IEEE (2010) Systems and software engineering – Vocabulary. International Standard ISO/IEC/IEEE 24765:2010(E). First Edition (p. 186) Geneva, Switzerland

ISO (2011) CEN EN/ISO 11354-1, Advanced Automation Technologies and their Applications, Part 1: Framework for Enterprise Interoperability. International Standards Office, Geneva, Switzerland

Jardim-Goncalves, R., Agostinho C. & Steiger-Garcao, A. (2012). A reference model for sustainable interoperability in networked enterprises: towards the foundation of EI science base. *International Journal of Computer Integrated Manufacturing*, 25(10), 855-873.

Jardim-Goncalves, R., Grilo, A., Agostinho, C., Lampathaki, F. & Charalabidis, Y. (2013) Systematisation of Interoperability Body of Knowledge: the foundation for Enterprise Interoperability as a science. *Enterprise Information Systems*, 7(1), 7-32

Jeong, B., Lee, D., Cho, H. and Lee, J. (2008) A novel method for measuring semantic similarity for XML schema matching, *Expert Systems with Applications*, 34, 1651–1658

Kim, D., and Shen, W. (2007) An Approach to Evaluating Structural Pattern Conformance of UML Models. In *ACM Symposium on Applied Computing* (pp. 1404-1408), ACM Press.

Kingston, G., Fewell, S., & Richer, W. (2005). An organisational interoperability agility model. Defence Science And Technology Organisation. Edinburgh (Australia). Retrieved February 25, 2013, from http://www.dtic.mil/dtic/tr/fulltext/u2/a463924.pdf

Kokash, N. and Arbab, F. (2009) Formal Behavioral Modeling and Compliance Analysis for Service-Oriented Systems, in *Formal Methods for Components and Objects*, Boer, F., Bonsangue, M. and Madelaine, E. (Eds.). *Lecture Notes In Computer Science*, 5751 (pp. 21-41), Springer-Verlag, Berlin, Heidelberg

Kruchten, P. (2004) *The rational unified process: an introduction*, Pearson Education Inc., Boston, MA.

Lewis, G., Morris, E., Simanta, S., Wrage, L. (2008) Why Standards Are Not Enough To Guarantee End-to-End Interoperability. In Ncube, C. and Carvallo, J. (Eds.) *Seventh International Conference on Composition-Based Software Systems* (pp. 164-173). IEEE Computer Society Press

Loutas, N., Kamateri, E., Bosi, F. & Tarabanis, K. (2011) Cloud computing interoperability: the state of play, in Lambrinoudakis, C., Rizomiliotis, P. and Wlodarczyk T. (Eds.) *International Conference on Cloud Computing Technology and Science* (pp. 752-757), IEEE Computer Society Press.

Malik, N. (2009) Toward an Enterprise Business Motivation Model, *The Architecture Journal,* 19, 10-16.

Minoli, D. (2008) *Enterprise Architecture A to Z*, Auerbach Publications, Boca Raton, FL.

Monfelt, Y., Pilemalm, S., Hallberg, J. & Yngström, L. (2011) The 14-layered framework for including social and organizational aspects in security management. *Information Management & Computer Security*, 19(2), 124-133

Morris, E. et al (2004). System of Systems Interoperability (SOSI): final report. Report No. CMU/SEI-2004-TR-004. Carnegie Mellon Software Engineering Institute. Retrieved February 25, 2013, from http://www.sei.cmu.edu/reports/04tr004.pdf

Oguz, F. and Sengün, A. (2011) Mystery of the unknown: revisiting tacit knowledge in the organizational literature. *Journal of Knowledge Management*. 15(3). 445 – 461

Ostadzadeh, S. & Fereidoon, S. (2011) An Architectural Framework for the Improvement of the Ultra-Large-Scale Systems Interoperability, In *International Conference on Software Engineering Research and Practice*, Las vegas, NV.

Peristeras, V. & Tarabanis, K. (2006) The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C4IF) For Information Systems Interoperability. *IBIS - International Journal of Interoperability in Business Information Systems*, 1(1) 61-72

Popplewell, K. (2011). Towards the definition of a science base for enterprise interoperability: a European perspective. *Journal of Systemics, Cybernetics, and Informatics*, 9(5), 6-11.

Sowa, J. and Zachman, J. (1992) Extending and formalizing the framework for information systems, *IBM Systems Journal*, 31(3), 590-616.

Stamper, R. (1996) Signs, Information, Norms and Systems. In Holmqvist, B., Andersen, P., Klein, H. and Posner, R. (Eds.): *Signs of Work* (pp. 349–397). Berlin, Germany: de Gruyter.

The Open Group (2009) TOGAF Version 9 – The Open Group Architecture Framework (TOGAF), The Open Group.

Wang, W., Tolk, A. & Wang, W. (2009) The levels of conceptual interoperability model: Applying systems engineering principles to M&S. In Wainer, G., Shaffer, C., McGraw, R. & Chinni, M. (Eds.), Spring Simulation Multiconference. Article no.: 168. Society for Computer Simulation International. San Diego, CA

Weber-Jahnke, J., Peyton, L. & Topaloglou, T. (2012) eHealth system interoperability. *Information Systems Frontiers*, 14(1), 1-3

Winter, R. and Aier, S. (2011) How are enterprise architecture design principles used?, in *International Enterprise Distributed Object Computing Conference Workshops* (pp. 314-321), IEEE Computer Society Press

Winter, R. and Fischer, R. (2007) Essential layers, artifacts, and dependencies of enterprise architecture, *Journal of Enterprise Architecture*, 3(2), 7–18.

Wyatt, E., Griendling, K. & Mavris, D. (2012) Addressing interoperability in military systems-of-systems architectures. In Beaulieu, A. (Ed.) *International Systems Conference* (pp. 1-8) IEEE Computer Society Press

ADDITIONAL READING SECTION

Athanasopoulos, G., Tsalgatidou, A. and Pantazoglou, M. (2006) Interoperability among Heterogeneous Services, in *International Conference on Services Computing* (pp. 174-181), IEEE Computer Society Press

Bell, M. (2008) Service-Oriented Modeling: Service Analysis, Design, and Architecture, John Wiley & Sons, New York.

Berkem, B (2008) From the Business Motivation Model (BMM) to Service Oriented Architecture

(SOA), Journal of Object Technology, 7(8), 57-70.

Brocke, J., Schenk, B. & Sonnenberg, C. (2009) Organizational implications of implementing service oriented ERP systems: an analysis based on new institutional economics. In Abramowicz, W. (Ed.) *12th International Conference Business Information Systems*, Poznan, Poland (252–263). Berlin, Germany: Springer-Verlag

Earl, T. (2005) Service-oriented architecture: concepts, technology and design. Upper Saddle River, NJ: Pearson Education

Earl, T. (2007) *SOA: Principles of Service Design*, Prentice Hall PTR, Upper Saddle River, NJ.

Earl, T. (2008) *Principles of service design*. Boston, MA: Pearson Education

Fricke, E. & Schulz, A. (2005) Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. *Systems Engineering*. 8(4), 342-359.

Ganguly, A., Nilchiani, R. & Farr, J. (2009) Evaluating agility in corporate enterprises. *International Journal of Production Economics*, 118(2), 410-423.

Gehlert, A., Bramsiepe, N. and Pohl, K. (2008) Goal-Driven Alignment of Services and Business Requirements, in *International Workshop on Service-Oriented Computing Consequences for Engineering Requirements* (pp. 1-7), IEEE Computer Society Press.

Havey, M. (2005) *Essential business process modeling*. Sebastopol, CA: O’Reilly

Hoogervorst, J. (2004) Enterprise Architecture: Enabling Integration, Agility and Change, *International Journal of Cooperative Information Systems*, 13(3), 213–233.

Hoogervorst, J. (2009) *Enterprise Governance and Enterprise Engineering*, Springer-Verlag, Berlin.

Juric, M. and Pant, K. (2008) Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture, Packt Publishing, Birmingham, UK.

Khadka, R. *et al* (2011) Model-Driven Development of Service Compositions for Enterprise Interoperability, In van Sinderen, M. and Johnson, P. (Eds.), *Lecture Notes in Business Information Processing*, 76 (pp. 177-190), Springer Berlin Heidelberg.

Kurpjuweit, S. and Winter, R. (2009) Concern-oriented Business Architecture Engineering, in *ACM Symposium on Applied Computing* (pp. 265-272), ACM Press.

Lankhorst, M., Proper, H. and Jonkers, H. (2009) The Architecture of the ArchiMate Language, in Halpin, T. *et al*. (Eds.) *Enterprise, Business-Process and Information Systems Modeling* (pp. 367-380), Springer-Verlag, Berlin.

Läufer, K., Baumgartner, G. and Russo, V. (2000) Safe Structural Conformance for Java. *Computer Journal*, 43(6), 469-481. Oxford University Press.

Loutas, N., Peristeras, V. and Tarabanis, K. (2011) Towards a reference service model for the Web of Services, *Data & Knowledge Engineering*, 70, 753–774.

Lovelock, C. & Wirtz, J. (2007) *Services marketing: people, technology, strategy*. Upper Saddle River, NJ: Pearson Prentice Hall

Markov, I. and Kowalkiewicz, M. (2008) Linking Business Goals to Process Models in Semantic Business Process Modeling, in *International Enterprise Distributed Object Computing Conference* (pp. 332-338), IEEE Computer Society Press.

Papazoglou, P., Traverso, P., Dustdar, S. and Leymann, F. (2008) Service-oriented computing: a research roadmap, *International Journal of Cooperative Information Systems*, 17(2), 223–255.

Patten, K., Whitworth, B., Fjermestad, J. & Mahinda, E. (2005) Leading IT flexibility: anticipation, agility and adaptability. In Romano, N. (Ed.) *Proceedings of the 11th Americas Conference on Information Systems*, Omaha, NE, 11–14, Red Hook, NY: Curran Associates, Inc.

Perepletchikov, M., Ryan, C., Frampton, K. and Tari, Z. (2007) Coupling Metrics for Predicting Maintainability in Service-Oriented Designs, in *Australian Software Engineering Conference* (pp. 329-340) , IEEE Computer Society Press.

Quartel, D., Engelsman, W., Jonkers, H. and van Sinderen, M. (2009) A goal-oriented requirements modelling language for enterprise architecture, in *International conference on Enterprise Distributed Object Computing* (pp. 1-11) IEEE Press.

Regev, G. and Wegmann, A. (2005) Where do goals come from: the underlying principles of goal-oriented requirements engineering, in *International Conference on Requirements Engineering* (pp. 353-362) IEEE Press.

Ross, A., Rhodes, D. and Hastings, D. (2008) Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value, *Systems Engineering*, 11 (3), 246-262.

Schekkerman, J. (2006) How to survive in the jungle of enterprise architecture frameworks. Bloomington, IN: Trafford Publishing.

Shroff, G. (2010) Enterprise Cloud Computing: Technology, Architecture, Applications, Cambridge University Press, Cambridge, UK

Spohrer, J., Vargo, S., Caswell, N. & Maglio, P. (2008) The Service System is the Basic Abstraction of Service Science. In Sprague Jr., R. (Ed.) *41st Hawaii International Conference on System Sciences*. Big Island, Hawaii, 104, Washington, DC: IEEE Computer Society

Turnitsa, C. (2005). Extending the levels of conceptual interoperability model. In *IEEE Summer Computer Simulation Conference*. IEEE Computer Society Press

van der Aalst, W. (1999) Process-oriented architectures for electronic commerce and interorganizational workflow, *Information Systems*, 24(8), 639-671.

van Lamsweerde, A.(2001) Goal-Oriented Requirements Engineering- A Guided Tour, in *International Symposium on Requirements Engineering* (pp. 249-262), IEEE Computer Society Press.

Xu, X., Zhu, L., Kannengiesser, U. and Liu, Y. (2010) An Architectural Style for Process-Intensive Web Information Systems, in *Web Information Systems Engineering*, *Lecture Notes in Computer Science*, 6488 (pp. 534-547), Springer-Verlag Berlin Heidelberg.

Xu, X., Zhu, L., Liu, Y. and Staples, M. (2008) Resource-Oriented Architecture for Business Processes, in *Software Engineering Conference* (pp. 395-402), IEEE Computer Society Press.

KEY TERMS & DEFINITIONS

Consumer: A role performed by a system *A* in an interaction with another *B*, which involves making a request to *B* and typically waiting for a response.

Provider: A role performed by a system *B* in an interaction with another *A*, which involves waiting for a request from *A*, honoring it and typically sending a response to *A*.

Transaction: Primitive pattern (predefined sequence) of messages exchanged between two interacting systems, one in the role of consumer, which initiates the pattern by a request, and the other in the role of provider, which honors that request and typically provides a response.

Choreography: Contract between two or more systems, which establishes how they cooperate to achieve some common goal through a composition of transactions.

Compliance: Asymmetric property between a consumer *C* and a provider *P* (*C* is compliant with *P*) that indicates that *C* satisfies all the requirements of *P* in terms of accepting requests.

Conformance: Asymmetric property between a provider *P* and a consumer *C* (*P* conforms to *C*) that indicates that *P* fulfills all the expectations of *C* in terms of the effects caused by its requests.

Compatibility: Asymmetric property between a consumer *C* and a provider *P* (*C* is compatible with *P*) that holds if *C* is compliant with *P* and *P* is conformant to *C*.

Interoperability: The ability of a consumer *C* to be partially or fully compatible with a provider *P*. By composition, it can also refer to multilateral compatibility between several systems, interacting in the context of some choreography.

Interoperability framework: Set of principles, assumptions, rules and guidelines to analyze, to structure and to classify the concepts and concerns of interoperability.

Layers of interoperability: Organization of interoperability concepts and concerns along a single dimension, in layers of monotonically varying degree of complexity and abstraction.

Enterprise architecture framework: Set of guidelines, best practices and tools to analyze, to classify, to structure and to describe the architecture of an enterprise.

Interoperability method: Set of steps to be taken to derive an interoperable enterprise architecture from an initial problem statement or from an already existing enterprise. This is typically used in conjunction with an interoperability framework.

Lifecycle: Set of stages that a system goes through, starting with a motivation to build it and ending with its destruction. Different versions of a system result from iterations of these stages, in which the system loops back to an earlier stage so that changes can be made.