

## Chapter 3

# A Conceptual Framework for Principles

**Abstract** This chapter provides the theoretical core of this book. It is concerned with a conceptual framework for architecture principles and related concepts. It starts by providing some historical background to the concept of principle. We will distinguish between scientific principles that describe laws or facts of nature, and normative principles that start as fundamental beliefs and which are translated to more specific and measurable statements. Based on the distinction between architecture and design, as made in the previous chapter, we will be able to define architecture principles as a subset of design principles. We also include a discussion on the motivation for the use of architecture principles in specific situations. In doing so, we provide a set of typical drivers for their formulation and enforcement. The chapter ends with the discussion of a general strategy to more precisely specify architecture principles and their underlying domain concepts.

### 3.1 Introduction

As argued before, we take the perspective that architecture principles are the cornerstones of enterprise architecture. Several approaches to enterprise architecture indeed position principles as a key ingredient, while some even go as far as to position principles as being the essence of architecture. Architecture principles fill the gap between high-level strategic intentions and concrete designs. The use of architecture principles also invites enterprise architectures to be directed toward the future, while focusing on essential decisions which guide future design decisions.

The goal of this chapter is to provide more background to the concept of architecture principles, while also more clearly defining the concept and its role as the cornerstone of enterprise architecture. To this end, Sect. 3.2 provides a broad discussion of the history of the concept of principle. Section 3.3 then continues by identifying two key flavors of principles, while also relating these to concepts such as *requirements* and *design instructions*. This allows us to clearly define the concept of *architecture principle* in Sect. 3.4, as well as its role in building a bridge from strategy to design. In Sect. 3.5 we turn to the question of how to motivate the (formulation and) enforcement of principles in specific situations. Before concluding, Sect. 3.6 briefly discusses a strategy to more precisely specify architecture principles and the underlying domain concepts it may refer to.

In the course of this chapter, we will incrementally develop a conceptual framework of our understanding of the concept of architecture principles. This conceptual framework is summarized in terms of three complementary fragments depicted in Fig. 3.2 (page 41), Fig. 3.4 (page 47) and Fig. 3.6 (page 55) respectively. The definitions of the concepts included in this framework, are designed to be compatible with existing views on enterprise architecture in general (IEEE 2000; Op 't Land et al. 2008; Dietz 2008; TOGAF 2009), while also taking aboard insights from reported practical case studies on the use and formulation of principles (Davenport et al. 1989; Richardson et al. 1990; Lindström 2006a, 2006b; Op 't Land and Proper 2007; Greefhorst et al. 2007; Greefhorst 2007). As such, the framework presented in this chapter also constitutes a first iteration in a design science (Hevner et al. 2004) driven research effort in which we aim to more clearly define the concept of architecture principles, and develop an associated methodology for defining and describing architecture principles. This first iteration aims to provide a first synthesis of existing views on enterprise architecture and the role of architecture principles.

## 3.2 Background of Architecture Principles

To better understand the nature and use of architecture principles within the field of enterprise architecture, it is important to understand the origins of the term *principle*. We therefore start this chapter with a brief discussion on the history of this term.

The term *principle* is said to originate from the Latin word of *principium* (Meriam-Webster 2003), which means 'origin', 'beginning' or 'first cause'. As summarized in Paauwe (2010), Vitruvius, an architect in ancient Rome, already used the concept of principles to explain what is true and indisputable, and should apply to everyone. Vitruvius considered principles as the elements, the laws of nature that produce specific results. For instance, he observed how certain principles of the human body, such as symmetry and proportion, ensure 'perfection'. The human body was a great source of inspiration to him. He even believed that the principles of the human body should also be applied in the design of gardens and buildings because it would always lead to a perfect result: an ultimate combination of beauty, robustness and usability.

When using principles in the sense of *beginning*, they generally provide insight into the causes of certain effects. These causes can be *laws of nature*, *beliefs* or *rules of conduct*. *Laws of nature* simply *are*, and influence the things we do. Examples of such principles are the *law of gravity* and the *Pauli exclusion principle*. The latter is a quantum mechanical principle formulated by Wolfgang Pauli in 1925. It states that no two identical fermions may occupy the same quantum state simultaneously. Another example, more directly relevant to the design of enterprises, is the principle of *requisite variety* from general systems theory, which states that a regulating system should match the variety of the system that should be regulated (Beer 1985).

*Beliefs* are typically founded in moral values. Examples of such principles are Martin Luther King's *principles of nonviolence*, that were to guide the civil rights movement. In our context, examples of such principles would be: *No wrong doors*

(suggesting that clients should be helped by which ever channel they approach the enterprise) and *The customer is always right*.

*Rules of conduct* are explicitly defined to influence behavior, and are typically based on facts and beliefs. General examples include the *Ten Commandments* from the Bible, e.g. *Thou shalt not murder* and *Thou shalt not commit adultery*. In our context, examples would be: *Clients can access the entire portfolio of services offered by any part of the government by way of all channels through which government services are offered* and *Before delivering goods and services to external parties, we must hold receipt of the associated payment*.

In defining the concept of architecture principle, we aim to remain close to the common interpretation of the term *principle* to prevent confusion. The Webster Dictionary (Meriam-Webster 2003) provides the following interpretations:

- 1a: a comprehensive and fundamental law, doctrine, or assumption b (1): a rule or code of conduct (2): habitual devotion to right principles <a man of principle> c: the laws or facts of nature underlying the working of an artificial device
- 2: a primary source: origin
- 3a: an underlying faculty or endowment <such principles of human nature as greed and curiosity> b: an ingredient (as a chemical) that exhibits or imparts a characteristic quality
- 4: Christian Science: a divine principle: god

The first of these four interpretations will be used as a base for the definitions provided in this chapter.

The use of principles in the context of enterprise architecture can be traced back (at least) to the earlier mentioned PRISM project (PRISM 1986). The PRISM framework is actually a fully principles-based architecture framework. In this context, principles were defined as “*simple, direct statements of an organization’s basic beliefs about how the company wants to use IT in the long term*” (Davenport et al. 1989). Note that in this definition, the operative word is *wants*. It refers to the fact that fundamentally, such principles are used to express a *normative desire*. Even more, it also expresses how these principles will aim to bridge the communication gap between top management and technical experts. PRISM’s concept of principles as well as how they guide the definition and evolution of architectures was its most important and widely accepted contribution.

The PRISM’s notion of principles has strongly influenced other architecture frameworks. The earliest publications referring to the concept of architecture principle (in an enterprise architecture context) can indeed be traced back to the PRISM project (Davenport et al. 1989; Richardson et al. 1990). Furthermore, the HP Global Method for IT Strategy and Architecture (Beijer and De Klerk 2010; Rivera 2007), which was based on works started in 1984 at Digital Equipment Corporation, was almost completely based on the concept of principle brought forward by the PRISM model. Many years later, the PRISM report even influenced the IEEE definition of architecture, as many of the IEEE 1471 committee members (Digital included) were employed by the original sponsors of their earlier work on PRISM. The concept of *architecture principle* as it is defined in TOGAF today is also inspired by the PRISM framework.

### 3.3 Key Classes of Principles

In this section we will define two key classes of principles: *scientific principles* and *normative principles*. In the next section, the class of normative principles will finally be specialized further into *design principles*, while in Sect. 3.4 *design principles* will be specialized further into *architecture principles*.

#### 3.3.1 Scientific Principles

In Sect. 2.2, we already quoted The American Engineers' Council for Professional Development's (ECPD 1941) definition of engineering (which we also used as a base to define *enterprise engineering*). This definition explicitly refers to *scientific principles* as being a core resource in the discipline of engineering: "*the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them ...*". Consider, as an example, the field of civil engineering, an engineering discipline which deals with the design, construction and maintenance of the physical and naturally built environment, including works such as bridges, roads, canals, dams and buildings. In this field scientific principles have always played an important role. A well-known principle in this field is the Archimedes principle, defined by Archimedes in the third century BC. The principle states that "*any object, wholly or partially immersed in a fluid, is buoyed up by a force equivalent to the weight of the fluid displaced by the object*".

Scientific principles are not limited to the field of civil engineering alone. For example, Lidwell et al. (2003) provide a list of 100 *universal principles of design*, consisting of laws, guidelines, human biases, and general design considerations. Examples of principles described that fall into the category of scientific principles are the *exposure effect* and *performance load*. The first principle states that *repeated exposure to stimuli for which people have neutral feelings will increase the likeability of the stimuli*. The latter states *the greater the effort to accomplish a task, the less likely the task will be accomplished successfully*.

The notion of scientific principle as a generally applicable law that can be used in the design of some artifact, corresponds to the interpretation of principles as *laws or facts of nature underlying the working of an artificial device* from the quoted Meriam-Webster (2003) definition. In line with the definition provided by the American Engineers' Council for Professional, we will indeed refer to these principles as *scientific principles*, leading to the following definition:

□ SCIENTIFIC PRINCIPLE A law or fact of nature underlying the working of an artifact. □

Different engineering disciplines, such as industrial engineering, chemical engineering, civil engineering, electrical engineering, software engineering, and enterprise engineering will have their own corpus of scientific principles. At the same time, these corpora are likely to overlap as well, since a large number of *scientific principles* will be cross-disciplinary in the sense that they will be applicable in

various design disciplines. For instance, the scientific principles from general systems engineering are bound to overlap with other engineering disciplines since these mostly deal with different forms of systems. An example would be the law of  *requisite variety* (Ashby 1956) from general systems theory, which is applicable to the design of enterprises, but equally well to any system in which communication and control plays a role.

Examples of scientific principles for the field of enterprise engineering can be found in sources such as Stafford Beer's viable systems model (Beer 1985), scientific management (Taylor 1911), the  $\phi$ ,  $\tau$ ,  $\psi$  theory (Dietz 2006) underlying the DEMO method, the mechanisms explaining how organizations may be seen as social systems conducting experiments (Achterbergh and Vriens 2009), or the mechanisms that explain how organizations 'emerge' out of human communication (Taylor and Van Every 2010).

As scientific principles essentially represent *design knowledge*, they can also be used as a resource to increase cross-disciplinary knowledge and understanding of design, promote brainstorming and idea generation for design problems, form a checklist of design principles, and to check the quality of design processes and products.

### 3.3.2 Design Principles as Normative Principles

In terms of the earlier quoted Webster's (Meriam-Webster 2003) definition of *principle*, scientific principles correspond to their interpretation as *a law or fact of nature underlying the working of an artificial device*. We take the view that design principles correspond to the interpretation of principles as a *rule of conduct*, where design principles guide/direct the enterprise by normatively restricting design freedom.

Before we properly define *design principles*, we first define the more general class of *normative principles* as:

⌈ NORMATIVE PRINCIPLE A declarative statement that normatively prescribes a property of something. ⌋

This is still quite a general definition. However, below we will see that this will actually allow us to also better relate design principles to concepts such as *business principles* and *IT principles*.

We clearly do not consider scientific principles to be forms of *normative principles*, and *design principles* in particular. As we will show in Sect. 3.4, scientific principles do have a role to play in the creation of enterprise architectures in terms of underpinning design decisions. Even more, they may provide the motivation for the formulation and enforcement of design/architecture principles.

When applying normative principles toward the design of artifacts, we can define the concept of *design principles* as follows:

⌈ DESIGN PRINCIPLE A normative-principle on the design of an artifact. As such, it is a declarative statement that normatively restricts design freedom. ⌋

Note that (Meriam-Webster 2003) defines an artifact to be “*something created by humans usually for a practical purpose*”. In the next section, we will define architecture principle as a specific classes of design principles.

Being normative restrictions of design freedom, design principles act as *rules of conduct* toward the designers of the (to be) constructed artifact since they (normatively) specify how to go about when designing the artifact. When considering the definition of policy used in this book:

□ POLICY A purposive course of action followed by a set of actor(s) to guide and determine present and future decisions, with an aim of realizing goals. □

then design principles, provide the means to define the *purposive course of action* in terms of the declarative statements that normatively prescribe properties of the artifact. This makes it desirable for the description of design principles to also provide guidance to designers that aid them in complying to them. In Sect. 2.8, we already stated that in the context of enterprise transformations, *architecture principles* (being a specific class of design principles) provide the policies needed to steer the transformation process.

Design principles are not the only statements which may limit design freedom. *Requirements*, for example, also limit design freedom. In this book, we define requirements to be:

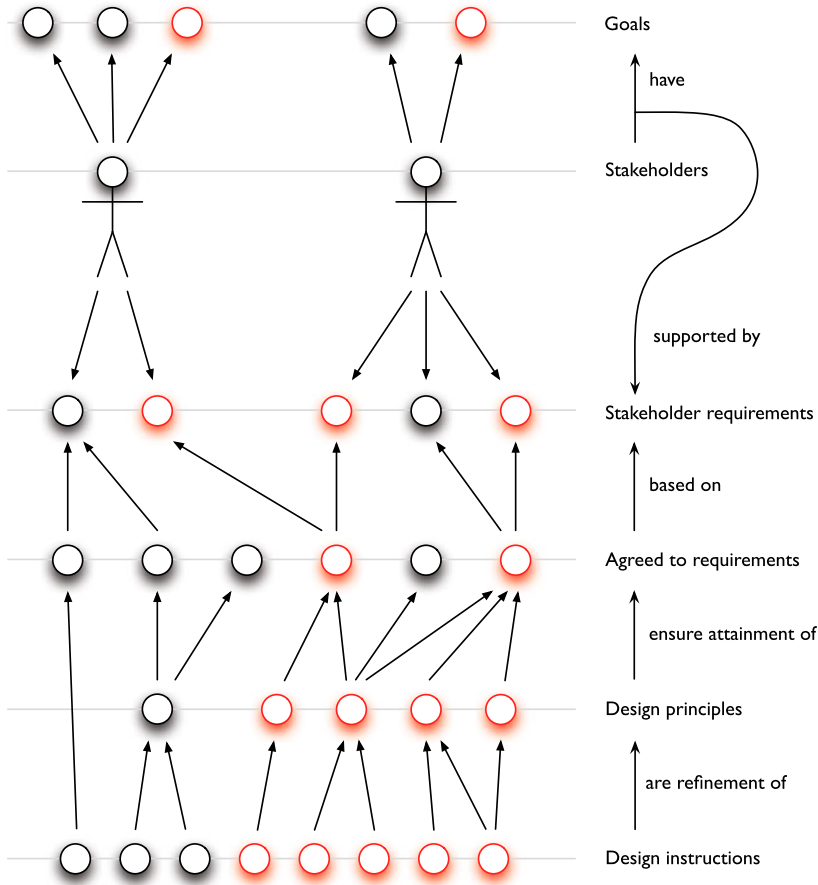
□ REQUIREMENT A required property of an artifact. □

Requirements state *what* (functional or constructional) properties an artifact should have from the perspective of the goals harbored by its stakeholders. The goals of the stakeholders provide the motivation, i.e. the *why*, of the requirements (Yu and Mylopoulos 1994, 1996; Chung et al. 1999). Based on an identification of the goals of the stakeholders, the requirements on the artifact can be derived. Given the requirements, design principles can be used to express the policies that ensure that the design of the artifact indeed meets the requirements. The design principles will focus primarily on addressing essential requirements. Design principles can, however, also address non essential requirements. These relations are exemplified in more detail in Fig. 3.1 (page 37), where the red circles represent essential goals, essential requirements and essential design principles respectively.

In Sect. 3.5 a more elaborate discussion is provided on the drivers underlying the formulation and enforcement of normative principles, also providing a more explicit way to identify what the *essential* goals and requirements are. This discussion will also provide us with a basis to finally properly define *architecture principles* in Sect. 3.4.

The diagram shown in Fig. 3.1 also illustrates the fact that to arrive at a set of requirements, for a given collection of (stakeholder specific) requirements, a negotiation process may be needed to compromise between conflicting needs. It furthermore illustrates the fact that not all requirements might be traceable to explicit goals of stakeholders. Some requirements might simply be too common, addressing a general level of quality required from the artifact being designed.

Since design principles take the form of *declarative statements*, there is a need for statements that provide more tangible guidance to the implementers, while also



**Fig. 3.1** From goals to design instructions

enabling analysis/simulation of a design to assess whether (qualitative and/or quantitative) requirements are met. In other words, instructive statements which more tangibly express *how* the artifact is to be constructed. In the case of enterprises, this would e.g. include: value exchanges, transactions, services, contracts, processes, components, objects, building blocks, et cetera. This typically also involves the formulation of models that act as blueprints of the artifact to be implemented. We will refer to these statements as *design instructions*, since they tell specifically what to do and what not to do in further elaborating the design or actually implementing it:

DESIGN INSTRUCTION An instructive statement that describes the design of an artifact.

The bottom part of the diagram shown in Fig. 3.1 also illustrates the position of design instructions in relation to design principles and requirements.

Design instructions provide a more operational and tangible refinement of the *design principles*. For example, a design principle may state that *stable processes are separated from variable processes*. A design instruction may refine this into stating (either textually or graphically) that *there is a sales process and a separate contract administration process*. In the context of enterprises, design instructions will typically refer to the concepts used in the actual construction of the enterprise, such as: value exchanges, transactions, services, contracts, processes, components, objects, building blocks, et cetera. Enterprises typically use languages such as UML (UML2 2003), BPMN (BPMN 2008), ArchiMate (Iacob et al. 2009), or the language suggested by the DEMO method (Dietz 2006), to express such design instructions. Due to their tangible nature, in terms of actual concepts used in the construction of the enterprise, design instructions allow enterprises to analyse/simulate the effects of different options for the future, as well as analyze problems in the current situation (Lankhorst et al. 2005a).

PRISM (1986) recognizes the existence of more specific statements; standards are specific rules or guidelines for implementing models. They are the most detailed aspect of architecture, and the primary activity in which names and named—of vendors, databases, applications and people. Infrastructure standards specify component selection and connectivity for particular environments. Data standards describe structures, data definitions, redundancies, and security considerations for databases. Application standards prescribe tools and environments, and can also mandate programming practices and structures for developed software. Organization standards describe support and management structures and staffing requirements for the delivery of information services.

Hoogervorst (2009) also distinguishes design principles from standards, where a standard is a predefined design norm, which includes design patterns. The statement to use such standards should also be considered as being a form of design instruction.

### 3.3.3 From Credos to Norms

Normative principles (such as design principles) can be classified in several dimensions based on their topical focus, i.e. the domain where the principle states a norm about. In our field, this can typically be done in terms of the cells of an architecture framework. In addition to the topical focus of a principle, we also distinguish two flavors of normative principles based on the level of precision (a form of detail) at which they have been formulated. This distinction will be especially useful in practical settings as they correspond to two important levels of ambitions at which these principles can be formulated and enforced.

When considering the design/architecture principles included in case studies (Davenport et al. 1989; Richardson et al. 1990; Lindström 2006a, 2006b; Lee 2006; Greefhorst et al. 2007; Greefhorst 2007; Bouwens 2009) one can indeed observe a variation in the level of precision at which these normative principles have been



formulated. As an illustration, consider the following examples exhibiting an increasing level of precision:

- “We are committed to a single vendor environment” (Davenport et al. 1989).
- “System structure and IS/IT availability shall enable mergers, acquisition, and establishment on new sites” (Lindström 2006b).
- “Customers: We only service customers who pay their bill” (Lee 2006).
- “When determining information systems solutions, the preferred order of selection should be an existing system, a purchased application package, in-house development, then outside services” (Richardson et al. 1990).

At the start of their life-cycle, normative principles are just statements that express the fundamental belief of how things ought to be. At this stage, their exact formulation is less relevant. This is in line with intentions behind TOGAF and the Zachman framework, where the architecture process starts with the creation of an architecture vision. In this phase, architecture is very future-oriented and mostly a creative process. Architecture principles are used as a means to express a vision, which is mostly based on personal beliefs of the stakeholders involved in the envisioning. They can be seen as normative principles in their initial stage. They are not yet specific enough to actually use them as a norm. In other words; assessing compliance of architectures and designs to these principles is not feasible. They are primarily used as a source of inspiration. Examples of normative/design principles in this phase, taken from practical cases, are:

- We should follow citizen logic.
- Work anywhere; anytime.
- Reuse as much as possible.
- Applications should be decoupled.

Normative principles in this phase can best be referred to as being a *credo*:

CREDO A normative-principle expressing a fundamental belief.

The Webster dictionary (Meriam-Webster 2003) defines *credo* as: “*a set of fundamental beliefs; also: a guiding principle*”. This is very close to the definition of principle provided by Beijer and De Klerk (2010): “*A fundamental approach, belief, or means for achieving a goal ...*”. In our context, *credos* are things an enterprise consciously chooses to adopt. They represent the fundamental beliefs or assumptions underpinning further design decisions. This allows enterprises to provide a first elaboration of an enterprise’s strategy toward the desired design of the enterprise.

When an enterprise aims to use normative principles as a way to actually *limit* design freedom, the formulation of these principles need to be more specific. In other words, they need to be formulated in such a way that compliance to them can be assessed. This starts with a reformulation of the principle statement, but extends to other properties. The specification will at least need to contain the rationale and implications of the statement, and preferably also definitions of terminology used, as well as guidance on how to assess the compliance of a design to the principle.

The examples given previously could be reformulated as follows to make them more specific:

- The status of customer requests is readily available inside and outside the organization.
- All workers are able to work in a time, location and enterprise independent way.
- Before buying new application services, it must be clear that such services cannot be rented, and before building such application services ourselves, it must be clear that they can not be purchased.
- Communication between application services will take place via an enterprise-wide application service bus.

Once credos have been (re)formulated such that they are specific enough, we can start to refer to them as a *norm*:

NORM A normative principle in the form of a specific and measurable statement.

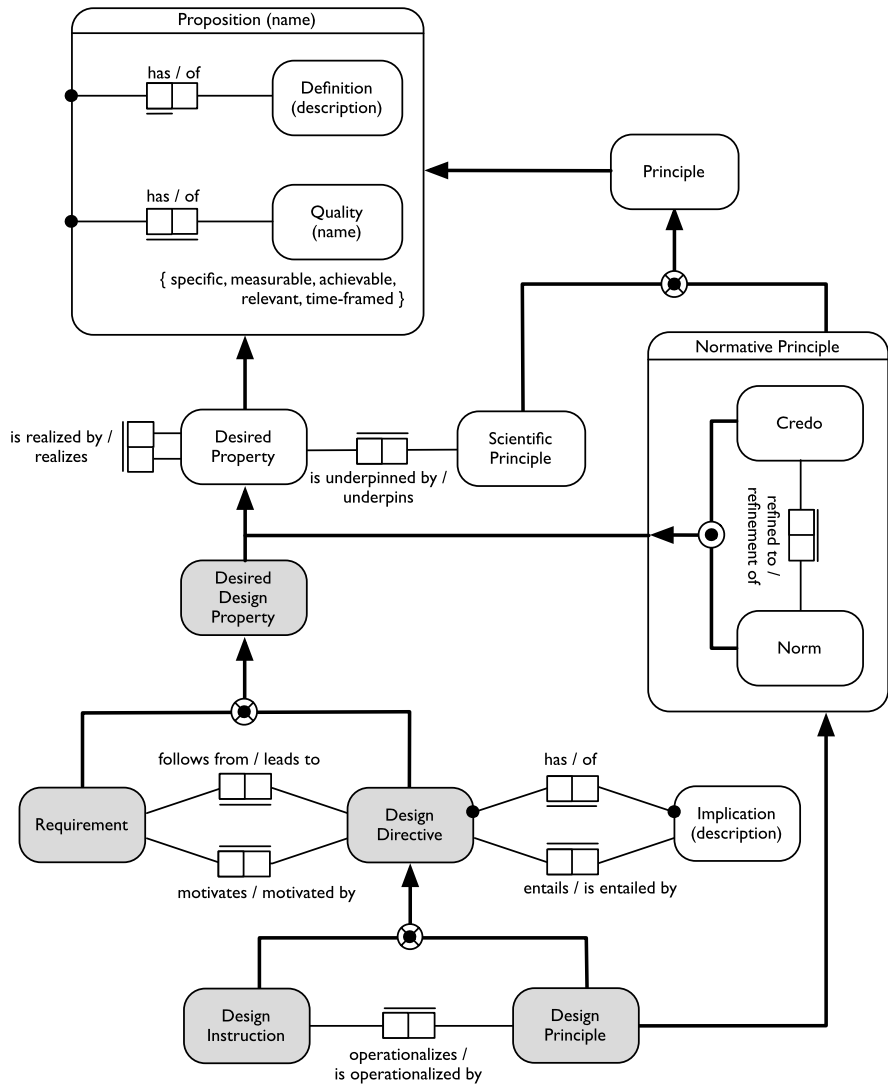
The Webster dictionary (Meriam-Webster 2003) defines a norm as: *a principle of right action binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behavior*. Norms can also be regarded as a tactic by which (the intention of) a *credo* can be enforced.

TOGAF defines an architecture principle as “*a qualitative statement of intent that should be met by the architecture*”. We take the stance that TOGAF requires architecture principles to be in the form of *norms*.

### 3.3.4 Conceptual Framework

As a summary, Fig. 3.2 (page 41) provides an ontological framework positioning *scientific principles*, *normative principles*, *design principles*, *design instructions*, *credos* and *norms*. In this diagram, we have used the Object Role Modeling (Halpin and Morgan 2008) notation as this notation provides a rich semantic modeling technique that is well suited to the modeling of ontologies (Trog et al. 2006), such as the conceptual framework for principles. In Fig. 3.2 we have also applied an abstraction/attribution mechanism to more compactly represent complex objects (Campbell et al. 1996; Creasy and Proper 1996), such as *proposition* and *normative principle*. In this notation, objects (entity types) are shown as boxes with rounded corners, relationships (fact types) are represented as two rectangles that show the roles that the objects play in both directions, and specialization relationships as lines that end with an arrow. Entity types that are attributed to other entity types are represented inside of a larger box with rounded corners, such shown in the case of *proposition* and *normative principle*.

In the resulting framework depicted in Fig. 3.2, we have added several generalizations leading to a generalization hierarchy. *Design principles* and *design instructions* have been generalized to *design directives* in general, since they both direct the (further) design of an artifact by expressing directives on *how* the artifact is to be designed/implemented. The OMG’s business motivation model (BMM 2006)



**each** Design Directive **which** has **an** Implication **which** is entailed by **some** Design Directive, **must be** realized by **the latter** Design Directive

**each** Design Directive **which** leads to **some** Requirement, **must be** realized by **this** Requirement  
**each** Requirement **which** motivates **some** Design Directive, **must be** realized by **this** Design Directive

**each** Design Principle **which** is operationalized by **some** Design Instruction, **must be** realized by **this** Design Instruction

**each** Credo **which** is refined to **some** Norm, **must be** realized by **this** Norm

Fig. 3.2 Core terminology

also uses the notion of *directive* as the most general form of guidance/regulation. Analogously, we have introduced *desired design property* as a generalization of *requirement* and *design directive*, since both of them express desired properties of a to be designed artifact in terms of *what* the constructed artifact should be like and *how* its design will ensure these requirements respectively. Since not all *normative principles* are *design principles*, and *normative principles* are desired properties in general, a further generalization of *desired design properties* and *normative principles* to *desired properties* in has been introduced as well. Furthermore, the concept of *principle* generalizes *normative principles* and *scientific principles*. Finally, the concept of *proposition* provides a further generalization of *principles* and *desired property*, since both essentially are propositions.

The encircled crosses in Fig. 3.2 are used to signify a mutual exclusive specialization. For example, *requirement* and *design directive* are mutually exclusive specializations of *desired property*, which means that a given desired property can not be both a design directive and a requirement. The black dot in the middle of the cross, as is the case with the specialization of *design directive* to *design instruction* and *design principle*, is used to indicate that it is a complete specialization in addition to being an exclusive specialization. In this case it means that **each** of the *design directives* is **either** a *design instruction* **or** a *design principle*. Since the border between *credos* and *norms* cannot be drawn explicitly, there is no mutual exclusiveness between these forms of *normative principles*. Nevertheless, as indicated by the encircled black dot, each of the *normative principles* must be a *credo* or a *norm*. Finally, as *principle* and *desired property* by definition overlap, since *design principles* are both forms of principles and desired properties, there is no mutual exclusiveness there as well.

Each of the *propositions* must have a *quality* and a *definition* (signified in the diagram by a black dot at the base of the relationship), while they have at most one *definition* (signified by the short bar on the relationship). The *qualities* that can be associated to a proposition are limited to the criteria from the (overloaded) SMART acronym<sup>1</sup>: *specific*, *measurable*, *achievable*, *relevant* and *time framed*. Table 3.1 (page 43) summarizes which qualities are to be held by the different flavors of propositions. In this table, we have used the following definitions of the SMART criteria:

**Specific** The proposition should be formulated clearly, while also defining the concepts used in its formulation.

**Measurable** The validity of the proposition with regards to the domain it states a property about should be measurable. Defining these measures is an integral part of the proposition.

**Achievable** Obtaining/maintaining validity of the proposition should be achievable given a reasonable amount of effort involving skills, means and time.

**Relevant** Achieving/having the validity of the proposition should be relevant. Note: later we will see how the relevance of an *architecture principle* can be argued in terms of the concepts shown in Fig. 3.6 (page 55).

<sup>1</sup>See [http://en.wikipedia.org/wiki/SMART\\_criteria](http://en.wikipedia.org/wiki/SMART_criteria).

**Table 3.1** Relevant qualities of propositions

	specific	measurable	achievable	relevant	time framed
norm	✓	✓	✓	✓	✓
credo	✓	✓	✓	✓	✓
design instruction	✓	✓	✓	✓	✓
requirement	✓	✓	✓	✓	✓
scientific principle	✓	✓			

**Time framed** There should be a time frame associated to the desired validity of a proposition, making explicit when compliance is required. Needless to say that such a time frame could run from months to ‘forever’. For example, normative principles typically have a longer time frame of application than requirements for a specific system.

As can be seen from the table, credos are not required to be specific or measurable. As discussed before, this is precisely what distinguishes them from norms. Requiring scientific principles to be achievable, relevant or time framed is not meaningful; they are statements that will always hold.

*Desired properties* in general may be formulated at different levels of granularity, where *desired properties* formulated at a higher level of granularity may be realized by a number of *desired properties* at a lower level of granularity. This allows for a stratified introduction of multiple levels of desired properties at different levels of granularity with regards to realization and implementation detail. For example, in the context of architecture frameworks one sees how architecture principles in one view may be based on the principles in a preceding view. Consider, as an illustration, ITSA (Beijer and De Klerk 2010). In the ITSA framework, architecture principles are defined in four views (business, functional, technical and implementation). The architecture principles in one view may be based on architecture principles in the view preceding it. *Scientific principles* can be used to underpin *desired properties*.

The realization relation between *desired properties* ‘reappears’ in different shapes in the lower parts of the generalization hierarchy depicted in Fig. 3.2. Introducing/enforcing a *design directive* will have consequences in terms of limiting further design decisions. Therefore, each *directive* must have some *implication*, while some of these *implications* may actually entail additional *desired properties*. *Requirements* can be used to motivate *design directives*, while at the same time *design directives* may also lead to the introduction of more refined *requirements*. Finally, *Normative principles* are operationalized by *instructions*, while *credos* may be refined to *norms*. The textual constraints at the bottom of Fig. 3.2 govern the connection between these specializations of the general realization relationship between *desired properties*, to the general one.

Finally, the *desired design property*, *design directive*, *design instruction*, *design principle* and *requirement* object types in this diagram have a grey background to signify the fact that they will re-appear in an additional schema fragments providing more context to principles (Figs. 3.4 and 3.6).

### 3.4 Architecture Principles as Pillars from Strategy to Design

In this section we finally define the concept of *architecture principle*, while also positioning architecture principles as pillars under the bridge from strategy to design as provided by enterprise architecture. Based on the discussions in this section, we will further extend the conceptual framework from Fig. 3.2.

#### 3.4.1 Architecture Principles

In Sect. 2.4.2, a distinction was made between design and architecture. Where an architecture focuses on essential requirements that typically also transcend the scope of specific projects, the design fills in the remaining aspects to meet the specific requirements that apply to the scope of a single project. This allows us to define architecture principles as a further specialization of design principles in general, based on their inclusion in an architecture:

ARCHITECTURE PRINCIPLE A design principle included in an architecture. As such, it is a declarative statement that normatively prescribes a property of the design of an artifact, which is necessary to ensure that the artifact meets its essential requirements.

It should be noted that the distinction between architecture and design is orthogonal to the distinction between *requirement* (what), *normative principle* (declarative how) and *instruction* (operational how). So, in principle, the concepts of *requirement* and *design instruction* could be specialized to *architecture requirement* and *architecture instruction*, respectively, based on their inclusion in an architecture. However, as our focus is on architecture principles, we will not do so.

#### 3.4.2 Business and IT Principles

TOGAF (2009) considers architecture principles as a subclass of IT principles, and the latter as a subclass of enterprise principles. We *strongly* disagree with this stance since enterprise architecture should holistically describe an enterprise including its business and IT aspects. Only those architecture principles that are related to the IT aspect can be a subclass of IT principles. Conversely, more architecture principles exist than just IT related architecture principles. IT principles are normative principles which provide policies to govern IT in general, but not all of these principles might be relevant from an architecture or design perspective. The same holds for business principles. Some of these might be ‘business architecture principles’, while not all of them need to be architecture principles. Schekkerman (2008) also concurs that architecture principles are a subset of business and IT principles, and not the other way around.

An important thing to note here as well is the meaning of ‘business’ in the word ‘business principle’. One could define ‘business’ as being the company, firm or enterprise as a whole. Accidentally, when translating business in to Dutch or German,

one most often uses the word ‘Bedrijf’ or ‘Betrieb’, respectively, which generally immediately refers to the company, firm or enterprise as an entity. When using this interpretation, IT principles and architecture principles in general could all be called business principles, because they refer to some aspect of ‘the business’ in terms of ‘the firm as a whole’.

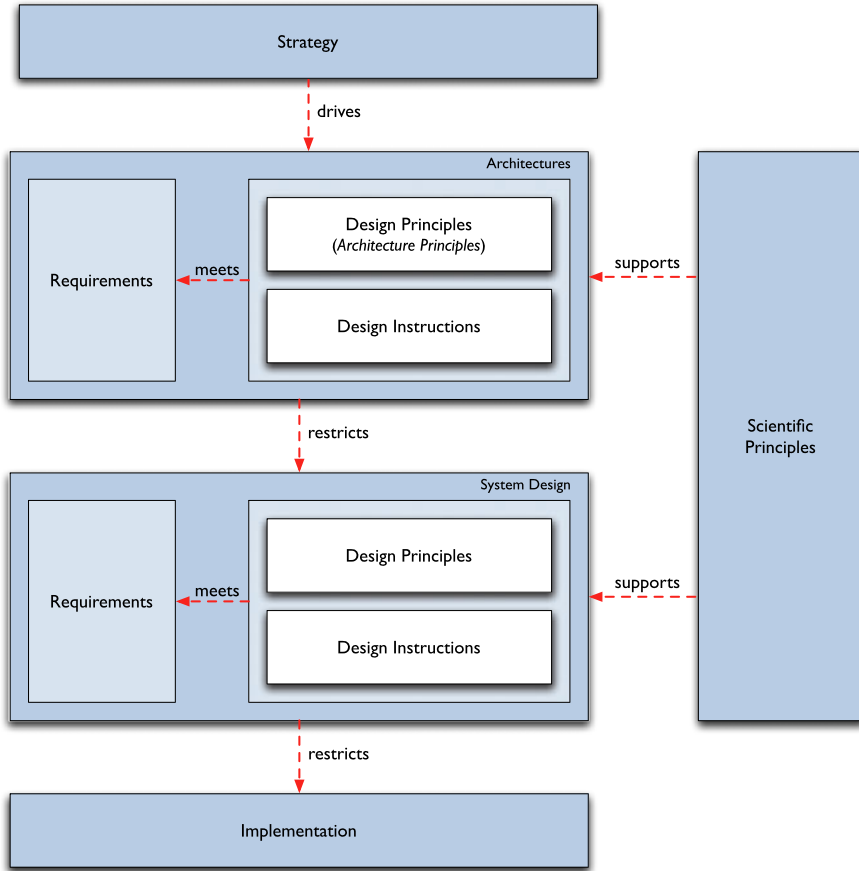
One could also refer to the ‘business’ as being those aspects of a company, firm, or enterprise, that pertain to the essential activities it engages in as a means of economical<sup>2</sup> livelihood. In general, the business level/column in architecture frameworks refers to ‘business’ from this perspective. In this case, IT principles are quite different from business principles.

One may also wonder how *business rules* fit in all of this. This depends very much on one’s definition of business rules. The SBVR (2006), a well-known standard in the business rules community, defines a business rule as: “*a rule that is under business jurisdiction*”. Those *desired properties* (see Fig. 3.2 (page 41)) that are defined under business jurisdiction would then be forms of business rules. This would certainly apply to *business principles* in general, and potentially to *architecture principles*. This of course also depends strongly on what is meant by the words ‘jurisdiction’ and ‘business’ in particular. IT architecture principles should be ‘owned’ by business stakeholders, even though they deal with the IT aspect and not the business aspect of an enterprise. Does this mean they are still under ‘business jurisdiction’? Furthermore, the above discussion on the interpretation of ‘business’ also applies here. The *Terms and Definitions* section of SBVR (2006) provides no clear definition of their understanding of ‘business’. The related standard of the Business Motivation Model (BMM 2006) also does not clarify the issue. Interestingly enough, it defines enterprise as *a business or company*, which seems to suggest an interpretation of ‘business’ as firm or company as a whole, including their IT aspects. However, when considering the line of reasoning suggested in BMM (2006) it seems that with ‘business’ in ‘business rule’ one refers to those rules that can be motivated in terms of risks/influences on the essential activities it engages in as a means of economical livelihood. This is strengthened by statements such as “*For the Sake of the Business, Not Technology*”.

In sum, from “*a rule that is under business jurisdiction*” (SBVR 2006) we take the position that *business rules* are intended as *desired properties* of an enterprise, where these *desired properties* are motivated directly in terms of risks/influences on the enterprise’s business in terms of the essential activities it engages in as a means of economical livelihood. This means that in terms of Fig. 3.2 (page 41), some business rules may turn out to be *requirements*, some may be *design principles*, and some may even be specific enough to be *design instructions*. One may expect most business rules to refer to the business aspect only, but it is not impossible for business rules to pertain to the IT aspects, as long as their motivation originates from the business: “*For the Sake of the Business, Not Technology*” (BMM 2006).

---

<sup>2</sup>Which does not only have to refer to money. We refer here to the exchange of scarce goods and/or services, which may include money, but also societal esteem, happiness, physical wellbeing, et cetera.



**Fig. 3.3** Architecture as a bridge from strategy to design

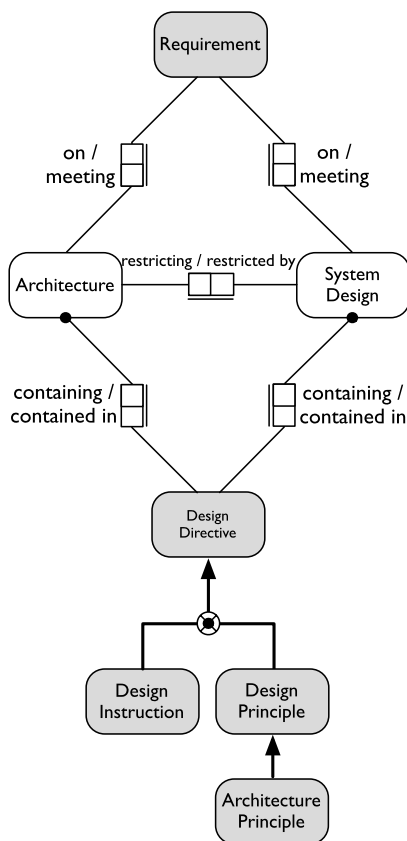
### 3.4.3 Bridging from Strategy to Design

With the above definitions in place, we can now refine the discussion provided in Sect. 2.4.2 on the role of enterprise architecture as a means to bridge from strategy to design. In doing so, we combine Fig. 2.5 (page 18) and Fig. 3.1 (page 37) to the situation as depicted in Fig. 3.3 (page 46). This diagram illustrates the flow from enterprise strategy via architectures, to the design of some specific system within the *system of systems* that constitutes the enterprise, to that system's implementation. The diagram also makes the role of requirements, design principles and design instructions at both the architecture and design levels more explicit. It furthermore shows how scientific principles support the creation of architectures and designs.

Figure 3.3 also shows the fact that architecture principles do not exist in isolation. They are based on all sorts of other artifacts, such as the business strategy and business drivers, the existing environment and (anticipated) external developments.



**Fig. 3.4** Extended conceptual framework



**Subtype defining rule:**

- An Architecture Principle is a Design Principle **which** is contained in **some** Architecture

They also influence all sorts of other artifacts, such as guidelines, architecture instructions, design requirements, design instructions, and implementations. Architecture principles really bridge between strategy and operations; they are primarily an alignment instrument. They are formulated based on knowledge, experience and opinions of all sorts of people in the organizations; senior management, as well as the people that do the actual work. This mixture of people is also the target audience of normative principles. In that sense, the definitions of normative principles also provide a common vocabulary for the organization.

As a further illustration of the flow from strategy to design, we use a fictitious insurance company. Their strategy is based on operational excellence. To this end they have formulated the objective to cut costs with 20% within two years, which can be considered an architectural requirement. Based on this architecture requirement they have defined an architecture principle which states that “*business processes are standardized and automated*”. Although they could not find any scientific principles to support this, they had good experiences with process standardization in other organizations. The architecture principle is translated to specific design in-

structions on their claims handling process in terms of a series of ArchiMate (Iacob et al. 2009) models. These instructions define the specific activities which must be present in all claims handling processes. A new claims handling system is designed to support the standardized claims handling process. A requirement for this system is that it integrates with the recently developed customer portal. The lead designer strongly believes that business rules should be defined and implemented separately from other application functionality in this claims handling system and therefore defines the design principle that *business rules are defined in a business rules engine*. He also provides more specific design instructions on how to actually define these business rules, by prescribing the specific constructs in the business rules engine that should be used. These design instructions are used by the developers that use the rules engine to implement the system.

Finally, as discussed in Sect. 2.4.3, the situation depicted in Fig. 3.3 should not be mistaken to be a top-down steering approach only. Architecture principles can be used as a *control* mechanism. However, by observing how emergent processes within a (networked) enterprise may lead to violation of existing principles, or even the emergence of (the need for) new architecture principles. As such, the mechanism of architecture principles can be used as an *indicator* mechanism as well. Admittedly, the remainder of this book focuses mainly on operationalizing the top-down steering aspect of architecture principles. By focusing on the essential requirements, the use of architecture principles allows/invites enterprises to think carefully about what to regulate in a design-first style and what to leave up to emergence, or to even take measures that enable desirable emergence.

### 3.4.4 Extended Conceptual Framework

As a summary, Fig. 3.4 (page 47) shows how the discussions provided in this section further extend the model fragment from Fig. 3.2 (page 41). An *architecture* restricts a *system design*. Both an *architecture* and a *system design* have to meet *requirements*, while both contain *design directives* in the form of *design principles* and *design instructions*. Each *architecture* and each *system design* must contain at least some *design directive*. Depending on their inclusion in an *architecture*, *design principles* are specialized into their respective design or architecture counterparts. This is formalized by the sub-type defining rule, shown at the bottom of Fig. 3.4.

One of the postulates of the enterprise engineering manifesto (CIAO 2010) states that an architecture should be concerned with a coherent, consistent, and hierarchically ordered set of normative principles for a particular class of systems. This book suggests a slight modification in that it takes the perspective that an architecture is concerned with coherent, consistent, and hierarchically ordered set of design directives.

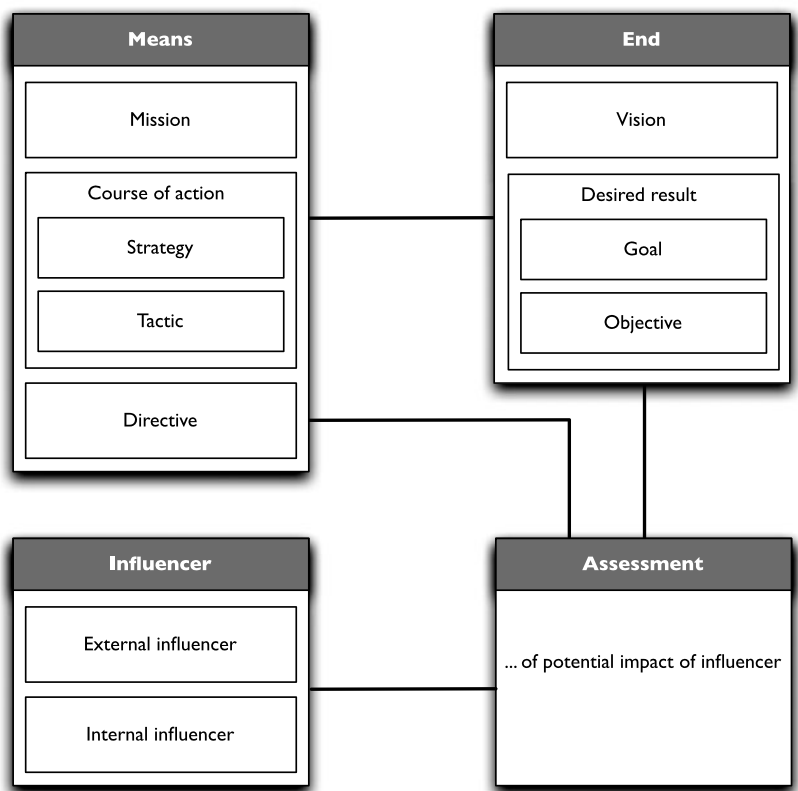


Fig. 3.5 Business motivation model

### 3.5 Motivating Architecture Principles

Architecture principles do not just fall out of the sky. Depending on the specific situation, different drivers will lead to the formulation (and enforcement) of design principles, and architecture principles in particular. Especially in the case of architecture principles, these motivations will originate from the goals and objectives embedded in the strategy.

In this section we provide a closer examination of the motivation for formulating and enforcing architecture principles, including the underlying drivers for their motivation. Although we have not investigated it explicitly, these drivers likely also apply to other *desired properties*, including *requirements*, *design principles* and *design instructions*.

### 3.5.1 Sources for Finding Motivation

There seems to be no universal agreement on the types of drivers that exist to motivate architecture principles. Nevertheless, much inspiration can be found in various existing models and approaches.

The field of requirements engineering has produced a number of methods and techniques that can also be applied to the motivation of architecture principles. Most notably, goal oriented requirements engineering (Yu and Mylopoulos 1994; Van Lamsweerde 2001; Regev and Wegmann 2005; Rifaut and Dubois 2008).

The business motivation model (BMM 2006) provides important concepts to express motivation. The model was initially created to provide the motivations behind business rules, but can also be used to find the motivation for architecture principles. Figure 3.5 highlights the core motivational concepts from the business motivation model (BMM 2006). As suggested by the business motivation model, an important step for the motivation of directives is the assessment of risks. This idea is brought to enterprise architecture by Engelsman et al. (2010) who state that architecture principles are based on an assessment of stakeholder concerns. An assessment represents the outcome of the analysis of some concern, revealing the strengths, weaknesses, opportunities, or threats (SWOT) that may trigger a change to the enterprise architecture. We believe that this provides only a subset of relevant drivers for architecture principles.

PRISM (1986) perceives the organization's technology values as the main determinant of architecture principles, but also recognizes that they are not the sole determinant. The organization's business situation and condition, its market position, competitive environment, and other elements of business strategy all impact technology values and principles directly, and in turn therefore underlie the entire architectural endeavor. It also states that architecture principles must be informed by, but not determined by, an assessment of the current state and future direction of technology. Such an assessment provides the options for technology-based principles.

TOGAF (2009) provides the following list of drivers: *enterprise principles*, *IT principles*, *enterprise mission and plans*, *enterprise strategic initiatives*, *external constraints*, *current systems and technology* and *computer industry trends*.

ITSA (Beijer and De Klerk 2010) distinguishes three types of drivers: *pains* (identifies what is wrong in the current situation), *directives* (what is stated as a constraint by other authors) and *opportunities* (a business opportunity). These three drivers are translated into SMART goals, that provide the motivation for architecture principles.

The Business Model approach described by Osterwalder and Pigneur (2009) also provides an interesting source of inspiration for motivating architecture principles. The basis for this approach is the Business Model Canvas, a tool for describing, analyzing and designing business models. The canvas provides nine building blocks to describe the rationale of how an organization creates, delivers and captures value. These building blocks are: customer segments, value propositions, channels, customer relationships, revenue streams, key resources, key activities, key partnerships

and cost structure. Given that these choices determine the business model of the organization, they should also lead to essential properties to be met by the enterprise.

In our further elaboration of motivations for the formulation (and enforcement) of desired properties, we will base ourselves mainly on the concepts provided by the business motivation model (BMM 2006). The business motivation model uses the concepts shown in Fig. 3.5 (page 49) to motivate business rules and business policies, which are generalized to the concept of *directives*. In particular we see that directives are formulated based on strategies and tactics, support goals and objectives, and are motivated by the potential impact of internal and external influencers. External influencers are the environment, technology, regulation, suppliers, customers, competitors and partners. Internal influencers are infrastructure, issues, assumptions, habits, corporate values, management prerogatives and resources.

The motivational concepts shown in Fig. 3.5 can be applied in our context as follows. An enterprise transformation effort is likely to impact on the stakes of many different stakeholders. For example, stakes of owners, sponsors, people working in the enterprise, clients, et cetera. Consider a stakeholder with a stake in the outcome of an enterprise transformation. Then it is fair to assume that this stakeholder has some goals/objectives which are potentially impacted by the outcome of the transformation. From the perspective of these goals, the transformation process has an ideal behavior. This behavior can refer to all aspects of an enterprise transformation, be it the changes to the enterprise, products produced, the actual process itself, et cetera. Whether or not the transformation exhibits this ideal behavior, is likely to be influenced by both internal and external factors. These potential ‘impacts’ may spark the stakeholder into (trying to) regulate the transformation and/or the potential influence. Needless to say that there will not be just one stakeholder. This means that the desire to regulate the transformation may lead to conflicts between stakeholders who have different goals with regards to the transformation.

For each influence, a risk assessment may show that this influence has a potential undesired impact on the goals of some stakeholder(s) (BMM 2006; Van Bommel et al. 2007). In other words there is some set of risks posed by the influence on the goals of the stakeholder. If the expected impact of the identified risks is high enough, a concern will be raised with the stakeholder. Multiple risks may even strengthen the specific concern of a stakeholder. When the expected impact of the risks is indeed high enough, the stakeholder(s) will be motivated to introduce regulations, i.e. the formulation and enforcement of desired properties that prevent the risks from occurring. The benefits of these desired properties can be expressed as the reduction of the expected impact of the risks (Van Bommel et al. 2007). Based on the risk assessment, as well as potential benefits, a MoSCoW (Stapleton 1997)<sup>3</sup>

---

<sup>3</sup>The capital letters in MoSCoW stand for:

- M** *Must* have this.
- S** *Should* have this if at all possible.
- C** *Could* have this if it does not affect anything else.
- W** *Won't* have this time but *would* like in the future.

style assessment can be made, yielding the essential requirements that should e.g. be addressed by an enterprise architecture.

### 3.5.2 Drivers as Motivation for Architecture Principles

Based on the sources mentioned above, as well as our own experiences in practice, we propose the following types of drivers for the formulation of architecture principles:

**Goals & objectives** targets that stakeholders seek to meet, many of these will be embedded in the strategy of the enterprise.

**Values** fundamental beliefs shared between people in an enterprise.

**Issues** problems that the organization faces in reaching the goals.

**Risks** problems that may occur in the future and that hinder the enterprise in reaching its goals.

**Potential rewards** chances and their potential reward for enterprises.

**Constraints** restrictions that are posed by others inside or outside the enterprise, including existing normative principles.

We will describe these drivers, their origins and characteristics in more detail below.

*Goals & objectives* are targets that stakeholders within and outside an enterprise seek to meet. They can be very high-level, such as *decrease costs*. They can also be very specific, such as *decrease IT development costs with 10% within one year*. In line with the business motivation model, we will refer to these latter goals as objectives. Objectives are required to be SMART, where as goals are not. Goals and objectives can be very strategic, resulting from a SWOT analysis of the organization (comparable to goals derived from opportunities in ITSA (Beijer and De Klerk 2010)). They can also be more tactical and operational, focused on specific areas within an enterprise. They can be hierarchical as well; a goal can be a means for a higher level goal. In that sense, strategies and tactics as defined by the business motivation model are also considered goals from the perspective of architecture principle development. Goals and objectives should be the main drivers for architecture principles. Without ends, any means will do.

*Values* are also important drivers for desired properties. Fundamentally, values are expressed in terms of quality attributes such as: reliability, trustworthiness, transparency, sustainability, efficiency, flexibility, privacy, et cetera. Quality frameworks such as ISO 9126 (ISO 2001) and IEEE 1061 (Software & Systems Engineering Standards Committee 1998) are a good source of inspiration for these quality attributes. The importance of values (and the associated quality attributes) as drivers for desired properties, is stressed by a number of sources (Graves 2009; Vermeulen 2009; Bouwens 2009; PRISM 1986). The formulation of a desired property can be used to describe how values should be expressed in practice. PRISM (1986) describes values as a set of underlying attitudes and perspectives that shape the organization's fundamental approach to information systems. They cut across

technology domains, and are even more long-lived than principles. Vermeulen (2009) has collected architecture principles from a number of sources and developed a 'generator' that, based upon a scoring of core values, generates a list of most appropriate architecture principles. Bouwens (2009) also shows how some values are really a combination of other values. Graves (2009) states that it is important to distinguish required, espoused and actual values. Mismatches in these values can be used to determine the priority of the value and the architecture principles that are based on it.

*Issues* are particularly relevant drivers for architecture principles, and comparable to the pains as identified by ITSA. The business motivation model defines issue as *an internal influencer that is a point in question or a matter that is in dispute as between contending partners*. In a more general sense, issues are anything that hinders an enterprise in reaching its goals. They exist at all levels, from strategic to tactical and operational. An example of an operational issue is *IT systems do not reach the availability requirements as set forth in the Service Level Agreement*. Including them as drivers enables operational employees to provide relevant input to the architecture, and thereby involve them in the process. It provides an opportunity for the architect to contribute to problems that people are confronted with in their daily work, and is an important step in the acceptance of architecture principles.

*Risks* are very much comparable to issues; they are problems that *may* occur. The reduction of risks is an important motivation for directives. These risks are thus also an important driver for the formulation of architecture principles. There are various classes of risks. As an example, based on the Basel II accord (BIS 2004), financial institutions should manage credit, market and operational risks. These operational risks are particularly relevant from an architectural perspective; credit and market risks are mostly handled by financial departments. Basel II defines operational risk as the risk of loss resulting from inadequate or failed internal processes, people and systems, or from external events. It is up to the architect to identify the most important risks. The focus should be on those risks that hinder the enterprise in reaching its goals. An example of a risk is that *there are single points of failure in the infrastructure that may lead to unavailability of IT systems*. For all risks identified, a risk analysis needs to be performed in which the impact and probability of occurrence is determined. Only those risk that have a high impact and/or probability need to be included as driver. At a more aggregated level, certain categories of risks may lead to the formulation of desired properties. Especially security risks are relevant. In an enterprise that seems unaware of security risks, an architecture principle that states that *information is secured according to laws and regulations* is very relevant. The role of risks as a driver to architecture principles has been explored in more detail in Van Bommel et al. (2007).

*Potential rewards* are essentially what is referred to as business opportunities in Rivera (2007). In other words, some event or initiative that has a potential benefit/reward to the enterprise. In this sense, a potential reward is the inverse of a risk. In the business motivation model, a SWOT assessment leads to the estimate of a potential impact, which is either a risk or a potential reward. A potential impact significant to an assessment can provide impetus for the formulation of architecture principles.

*Constraints* are also commonly recognized as drivers for architecture principles. They identify those things that were defined by others and that cannot be changed by the architect. They may come from outside the enterprise, such as laws, policies and regulations provided by government. An example of such a constraint is the policy that is defined by the Dutch government that states that open source products are preferred over commercial products when they are equally suitable. Constraints may also come from (senior) management. Such constraints are called ‘management prerogatives’ in the business motivation model, which defines them as *an Internal Influencer that is a right or privilege exercised by virtue of ownership or position in an enterprise*. An example constraint that could be defined by management is that *all non-core activities will be outsourced*.

Most constraints actually correspond to some externally enforced *desired property*. An important class of such desired properties are normative principles that have been formulated in the context. These normative principles may be refined into more specific normative principles that realize them. In that sense, pre-existing normative principles, that are to be enforced as constraints, may be drivers for the formulation of other normative principles.

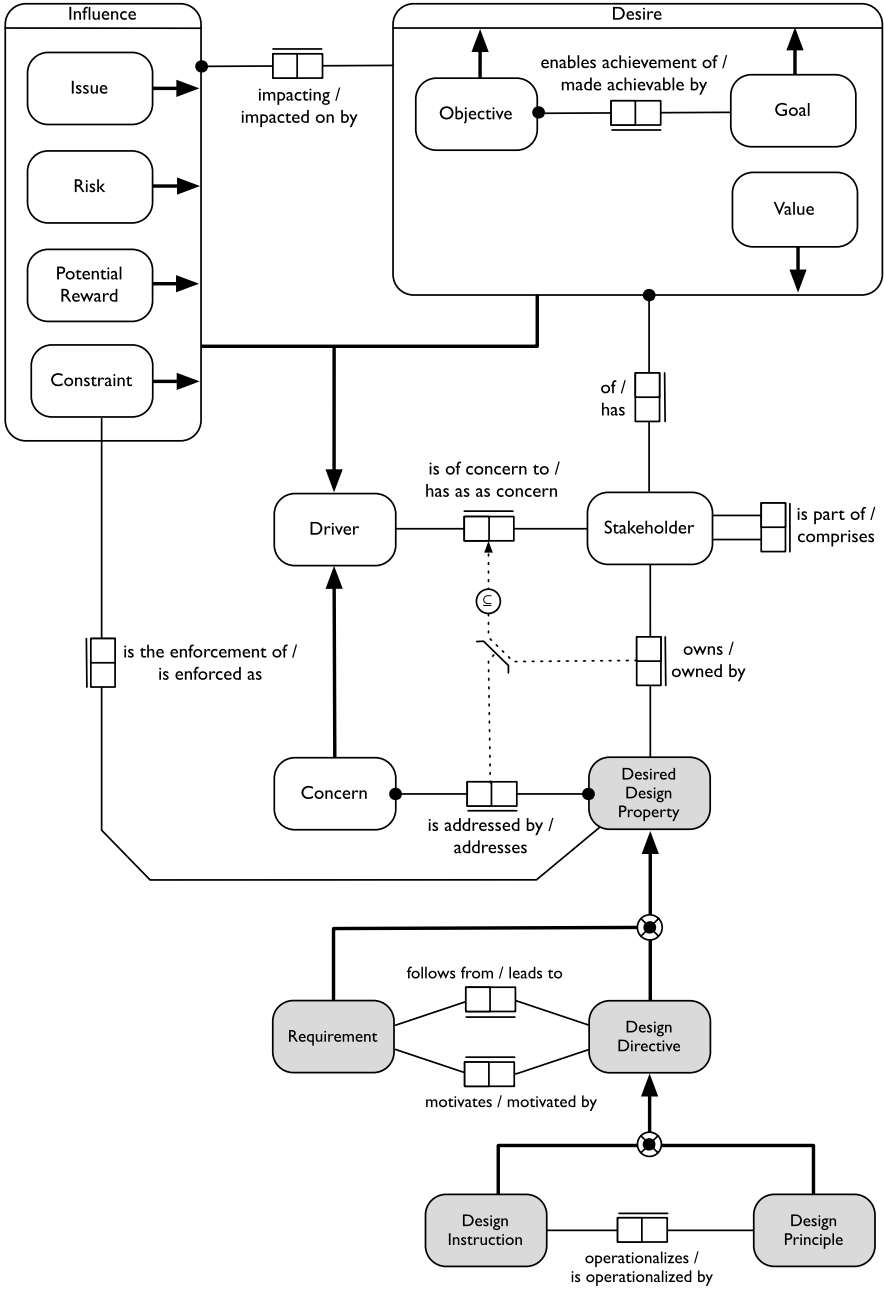
### 3.5.3 Extended Conceptual Framework

As a summary, Fig. 3.6 provides a conceptual framework of the key concepts introduced in this section. *Goals, objectives* (which must enable the achievement of some goal), and *values* are generalized to *desires*, where each *desire* must be the desire of some *stakeholder*. The *desires* may be influenced (positively or negatively) by an *influence*, where we have four specific kinds of *influences*: an *issue*, a *risk*, a *potential reward* and a *constraint*.

The *desires* and *influences* together form the *drivers* for the formulation of *desired properties* in general. *Drivers* may be a concern to some *stakeholder*. When this is indeed the case, the *driver* becomes a *concern*, which must be addressed by some *desired property*. *Constraints* may actually be the enforcement of some externally formulated *desired property*. *Drivers* are initially translated to *requirements*, that provide a manageable representation for these *drivers*. These *requirements* may lead to *design principles*. The essential requirements lead to architecture principles (not shown in the figure).

Even though it makes sense to ensure that each *desired property*, is owned by some *stakeholder*, one cannot simply state this as a general rule. However, especially when included in an architecture, it does make sense to assure ownership. Enforcement of the desired property will certainly benefit from this. When a *desired property* is owned by a *stakeholder*, then this must be based on an underlying *concern* of this *stakeholder*. More precisely, if a *stakeholder* has ownership of a *desired property*, then there must be a *concern* of that *stakeholder* which is addressed by that specific *desired property*. This is expressed formally in Fig. 3.6 by means of the subset constraint marked with the  $\subseteq$  symbol.

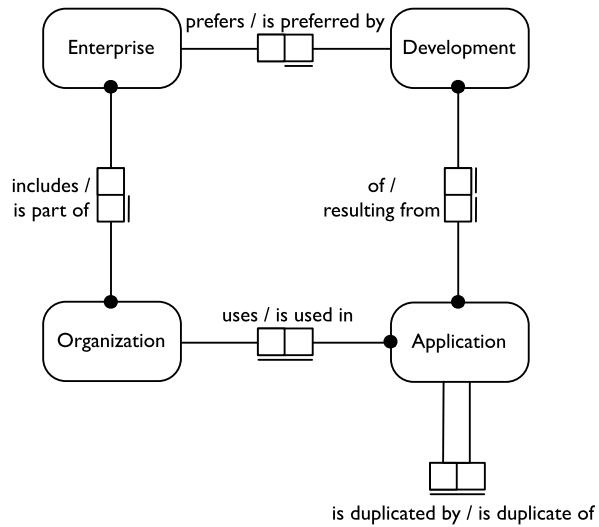




**Subtype defining rule:**  
- A Concern is a Driver which is a concern to some Stakeholder

Fig. 3.6 Motivating architecture principles

**Fig. 3.7** ORM representation of concepts underlying an architecture principle



### 3.6 Formal Specification of Normative Principles

In this final section, we briefly discuss ways of more formally specifying normative principles. Although architecture principle specifications are discussed in detail in the next chapter, we feel that the formality of the current chapter is a better place for a discussion on the formal specification of architecture principles. In Van Bommel et al. (2006), the authors describe how Object Role Modeling (ORM) (Halpin and Morgan 2008) and Object Role Calculus (ORC) (Hoppenbrouwers et al. 2005), essentially a formalized version of SBVR (2006), can be used to formalize normative principles. Even more, they also argue that the mere fact of formalizing normative principles already leads to interesting feedback on the original informal formulation. The authors illustrate this by means of two examples taken from TOGAF. Consider for example, the principle suggested by TOGAF:

**Common use applications:** *“Development of applications used across the enterprise is preferred over the development of duplicate applications which are only provided to a particular organization.”*

Figure 3.7 shows the ORM representation of the domain concepts underlying this architecture principle. The actual architecture principle should unambiguously express a norm in terms of these objects and facts. In creating Fig. 3.7, one is also invited to more carefully define the terminology used. What is an *organization*, what is a *enterprise*, what is their relationship, what does it mean for an application to be *duplicate*, what does it mean for an application to be *used across the enterprise*, et cetera. Questions that also need answering if one seriously aims to enforce such an architecture principle, and even when one only uses this principle as a means of guidance. Without proper definitions of the basic terms, guidance can be difficult.

For the sake of the example, it is assumed that organizations are the composing parts of an enterprises, while “*applications being used across the enterprise*”

is interpreted as applications being used in two or more organizations within that enterprise. In addition, we model the notion of ‘duplication’ as a distinct fact. Lexically, it corresponds to some measure or judgment concerning great similarity in functionality of two applications. Another issue is the interpretation of the term ‘preferred’. For simplicity’s sake it is assumed, maybe naively, that a development is either preferred or not. However, in practice it seems more realistic to provide a rated interpretation, for example by counting the number of duplicates occurring (decreasing preference), or the number of times a single application is used in different organizations being one or larger (increasing preference as the count goes up). This would more actively encourage actual development of applications that are used in more than one organization.

In terms of the terminology from Fig. 3.7, we now have:

**if an** Application [ **that** is used in **an** Organization ] results from **some** Development  
**and that** Application is **not a** duplicate of **another** Application  
**which** is used in **another** Organization  
**then that** Development is preferred by **the** Enterprise  
**which** includes **both** Organizations

In the analysis leading up to this formalization, it became clear that “*duplications*” and “*use across organizations*” related to essentially different concepts (the first to similarity in functionality between different applications, the second to distributed use of the same application). Consequently, it was deduced that “*duplications*” alone could do the job in capturing the intention of this principle:

**if an** Application results from **some** Development  
**and that** Application is **not a** duplicate of **another** Application  
**then that** Development is preferred by **the** Enterprise

This boils down to the simple informal rule “*no duplicate applications*”.

As argued in Van Bommel et al. (2006), such an analysis generally leads to a better understanding, and even improvement of normative principles. It helps in providing them clear and unambiguous meaning. Experiments with students (Chorus et al. 2007) lead to similar conclusions. However, it also raises the question whether stakeholders can/should be confronted with the formalized notation. In SBVR (2006), it is argued that business rules which are formalized in such a style can indeed be validated by domain experts, not requiring formal skills. In practice, formalized specifications are not yet common ground for specifying architecture principles. We believe that SBVR-like formalization of normative principles in terms of languages such as RIDL (Meersman 1982), Lisa-D (Ter Hofstede et al. 1993), ConQuer (Bloesch and Halpin 1996) or Object-Role Calculus (Hoppenbrouwers et al. 2005) is primarily a tool for architects, enabling them to improve the quality of architecture principles, while potentially enabling validation by stakeholders.

In general, normative principles are best described in terms of structured text, at a minimum involving a clear *normative* statement. It is imperative that principles can be understood by a broad audience, and more specifically a mixed group of stakeholders. Using an SBVR-like style, might provide a balance between formality and understandability by a broad audience. This does, however, require further study and evaluation. Additionally, a concise motivation, as well as an indication

of the consequences, are also highly recommended. There is a wide range of other attributes (meta-data), such as the application area the principle pertains to, that can be associated to normative principles, and aids in their formulation and governance. These are discussed in more detail in Chap. 4.

### 3.7 Key Messages

- The concept of principle has a long history.
- An important distinction has to be made between *scientific principles* and *normative principles*.
- Architecture principles are *design principles* that focus on how the design of an enterprise will meet the essential requirements.
- Architecture principles are declarative statements, that can be made more specific using *design instructions*. The latter can take the form of architecture models in a language such as ArchiMate.
- Architecture principles allow enterprises to build a bridge from the strategy to the more specific designs.
- Architecture principles, and desired properties in general, can be motivated based on several drivers.
- Drivers are desires of stakeholders and influences that may impact these desires.

Architecture Principles

The Cornerstones of Enterprise Architecture

Greefhorst, D.; Proper, H.A.

2011, XV, 197 p., Hardcover

ISBN: 978-3-642-20278-0