IMPACT OF PRINCIPLES ON ENTERPRISE ENGINEERING

Op 't Land, M. (Martin), Martin.OptLand@capgemini.com
Capgemini, P.O. Box 2575, 3500 GN Utrecht, The Netherlands
Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands

Proper, H.A. (Erik), E.Proper@cs.ru.nl
Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract

Increasingly, organizations make use of enterprise architectures to direct the development of the enterprise as a whole and the development of their IT portfolio in particular. This steering and directing is done by means of principles, which are essentially regarded as constraints on the design space for enterprise engineers, thus guiding them in their design efforts.

In this paper we study the potential constraining effect of principles on the design of enterprises as well as the guidance designers may receive from these principles. We start by providing a brief discussion on the concepts of enterprise architecture and enterprise engineering. We continue by discussing a strategy to make principles specific and measurable enough to indeed allow them to constrain design space. This is followed by a discussion of a number of examples, taken from real-life practice, illustrating the steering effect of principles. Finally, we also briefly pay attention to the process that may be followed in formulating and formalizing principles.

Keywords: Principles, Enterprise Architecture, Enterprise Engineering, Implementation

1 INTRODUCTION

Large organizations increasingly make use of enterprise architectures to direct the development of the enterprise as a whole and IT development in particular (Lankhorst et al 2005). These developments are fuelled even more by government regulations such as the Clinger-Cohen Act in the USA (Clinger-Cohen 1996), which requires government bodies to provide an IT architecture based on a set of architecture principles. A more specific way of expressing the role of enterprise architecture is to state: "architecture is defined as *normative restriction* of *design freedom*" (xAF 2003 p. 25). In most (enterprise) architecture approaches, this constraining is done by means of so-called architecture principles (IEEE 2000, TOGAF 2004). According to The Open Group Architecture Framework (TOGAF 2004), "*Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission*". Such principles typically address the concerns of the key stakeholders within an organization.

While several sources attribute a pivotal role to principles, a precise definition of the concept of principles as well as the mechanisms and procedures needed to turn them into an effective means still lack. Both IEEE (2000) and TOGAF (2004) position principles as a means to guide the design and evolution of systems, while xAF (2003) essentially defines (enterprise) architecture as a set of principles. Nevertheless, no clear definition of principles and associated mechanisms and procedures are given. This is also not the aim of this paper. This paper is, however, part of ongoing research in which we indeed are progressing towards such a definition. The main contribution of this paper is an exploration of the actual use of principles in practice in steering the development of an enterprise. For obvious reasons, this practical use should strongly influence the definition of the concept of principles, as well as the mechanisms and procedures needed to make it into an effective means. Several organizations already apply architecture principles as a means to indeed steer their evolution. At the same time, sources such as IEEE (2000), TOGAF (2004) and xAF (2003) do not provide a discussion on the potential impact of principles on the development of enterprises, i.e. there is no discussion on their actual "steering abilities". This paper aims to provide an initial exploration of this ability.

The remainder of this paper is structured as follows. In the next section, we provide a brief discussion on the concepts of enterprise architecture and enterprise engineering. We then continue by discussing a strategy to make principles specific and measurable enough to indeed allow them to constrain design space. This finally enables us to discuss the main contribution of this paper: an exploration of the impact of principles on the development of enterprises. We do so by discussing a number of examples, taken from real-life practice, illustrating the steering effect of principles. Finally, before concluding, we briefly pay attention to the process that may be followed in formulating and formalizing principles.

2 ARCHITECTURE AND ENTERPRISE ENGINEERING

The aim of this section is to briefly introduce the concepts of architecture and its relation to enterprise engineering. As our starting point, we chose xAF (2003) and Dietz (2005, 2006). First, we focus on the development process of a single system, and the role played by architecture. We then broaden this to the level of an enterprise, involving multiple systems. Finally we pose some challenges which arise when operationalizing architecture as a set of principles.

In a development process several systems are involved. Most prominently, the *object system* (OS), which is the system being designed, engineered and implemented. In addition to the OS, in this paper we will also distinguish a *using system* (US) and the *governing system* (GS). The US is the system that will use the functions or services offered by the OS, once it is operational. The GS governs the design, engineering and implementation of object systems by means of an architecture, where this governing is made concrete in terms of principles.

Dietz (2006) elaborates the relationship between US and OS. According to Dietz, the development of a single homogeneous system of any type can be understood as an instance of the Generic System Development Process (GSDP). This generic process is illustrated in Figure 1.

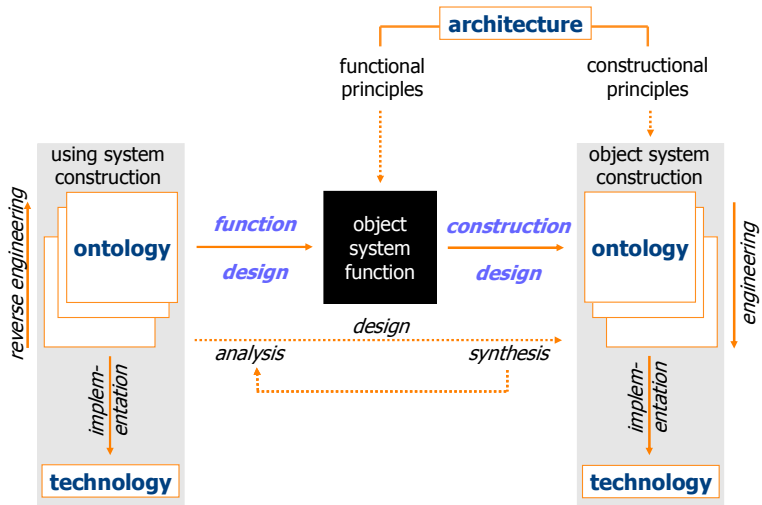

Figure 1 Generic System Development Process (after Dietz 2005 p. 23)

The development of an OS is considered to consist of a *design*, an *engineering* and an *implementation* phase. The design phase comprises a function design and construction design. *Function design*, the first step in the design of the OS, starts from the construction of the US and ends with the function of the OS. Function design delivers the requirements of the OS, so a black-box model of the OS. This black-box model clarifies the behavior of the OS in terms of (functional) relationships between input and output of the OS. The function model of the OS does not contain any information about the construction of the OS. *Construction design*, the second step in the design of the OS, starts with the specified function of the OS and it ends with the construction model of the OS. Construction design bridges the mental gap between function and construction, which means establishing a correspondence between systems of different categories: the category of the US (where the function of the OS is defined) and the category of the OS. Construction design delivers an ontology, the highest level white-box model, of the OS. This white-box model clarifies the internal construction and operation of the OS in terms of collaboration between its elements to deliver products to its environment. By an *ontology* or ontological model of a system we understand a model of its construction, which is completely independent of the way in which it will be implemented using some underlying technological infrastructure. The *engineering* of a system is the process in which a number of white-box models are produced, such that every model is fully derivable from the previous one and the available specifications. Engineering starts from the ontological model, produces a set of subsequently more detailed white-box models and ends with the implementation model. By *implementation* is understood the assignment of technological means to the elements in the implementation model, so that the system can be put into operation. By *technology* we understand the technological means by which a system is implemented. A wide range of technological means is available, varying from human beings and organizational entities via ICT (e.g. phone, email, computer programs) to industrial robots, vacuum cleaners, cars, drilling machines and screw drivers.

An important assumption for enterprise architecture is that the design freedom of designers is undesirable large xAF (2003 p. 25), and that as a consequence, the systems thus developed would typically be suboptimal in use, maintenance and costs. Suppose for instance that the OS is an automated information system *X* and the US is some department *Y* of the enterprise. Suppose furthermore that the project which has to develop the OS involves the relevant parties from both OS and US. Then it is reasonably to expect that the OS in itself will meet the needs of the US. However, even if this is the case, it is

quite conceivable (and in practice quite common) that the resulting OS is suboptimal from the perspective of the enterprise. For instance, although information system *X* may proper support to department *Y*, it may not connect well to other information systems of the enterprise or one of the third parties in an extended enterprise. Or, it may work and connect well, but does not do so in a future proof manner. Or, the development of system *X* has led to the introduction of new, costly or difficultly maintainable components. Of course, the typical cause of such phenomena is that the interests of far more stakeholders than just the direct representatives of US and OS have to be guarded. This is where the governing system (GS) needs to step in.

It is the responsibility of the GS to provide guidance to the development process in a way which balances and safeguards the interests, concerns and objectives of all stakeholders, by restricting the design freedom of a class of object systems. This is the starting point for xAF (2003 p. 25), who defines *architecture* (1) conceptually as a *normative restriction* of *design freedom* and (2) operationally as a consistent and coherent set of design *principles* that embody general requirements, where these general requirements hold for a class of systems. As benefits of such a governing based on architecture, xAF (2003 p. 10) a/o mentions improved integration, adaptability and agility. CIAO! (2006) and TOGAF (2004) add to these benefits ease of collaboration, interoperability and a free flow of information.

We now turn to the level of an enterprise. We define an *enterprise* as a heterogeneous system, constituted as the layered integration of three aspect systems, namely the Business system, the Informational system and the Datalogical system (after Dietz 2006 p. 115). Dietz (2006 p. 71) defines *enterprise engineering* as "the whole body of knowledge and know-how regarding the development, implementation and operational use of enterprises, as well as their practical application in engineering projects". The term "enterprise engineering" is used here in a broader sense, like in civic, electrical and mechanical engineering. Note the difference with the notion of engineering in the narrow sense as introduced in GSDP: (1) it now plays a role in all life-cycle phases of a system and (2) the type of system involved is now the (heterogeneous) system called the enterprise.

As *enterprise architecture* we will consider the architecture, so operationally the set of principles, which is applicable to the enterprise. As a consequence, its principles can influence any aspect system of the enterprise (Business, Informational, Datalogical), the integration of those aspect systems and any life-cycle phase (development, implementation, operational use, maintenance etc.) of the systems.

As defined before, principles embody general requirements holding for a class of systems. xAF (2003 p. 10) illustrates this difference between general and special requirements as follows. Suppose we require for all cars of a car factory that the maximum speed should be at least 80 km/hour. For a specific car type, we could additionally require that its maximum speed be at least 180 km/hour. This example of a functional requirement has also a constructional counterpart in terms of material use (see Table 1).

perspective	General	special
functional	• accounting should be in conformity with European law • the maximum speed of cars should be at least 80 km/hr	• this accounting system should handle $ and € • the maximum speed of this car should be at least 180 km/hr
constructional	• ICT-applications should be component based • in cars, minimal 25% of the material should be synthetic	• this system must be programmed in C^{++} • the body of this car should be fully synthetic

Table 1 General and special requirements (xAF 2003 p. 10)

Having explored the concepts of enterprise engineering and enterprise architecture / principles, several questions emerge, which we will discuss in the following sections:

- how do principles, usable as a steering instrument for the GS, look like?
- how do principles restrict and impact enterprise engineering?
- what process should we follow to formulate (a set of) principles?

3 OPERATIONALIZING DESIGN RESTRICTIONS: PRINCIPLES

There is no clear definition yet of the concept of principles. As mentioned before, providing such a definition is also not the goal of this paper. In this paper we aim to explore the impact of principles on the development of enterprises. To operationalize this impact, however, principles should be specific enough. This section therefore briefly surveys some considerations in making principles specific enough to enable them to steer development.

Some generally accepted quality requirements for a (set of) principle(s) are (Davenport et al. 1989, Tapscott and Caston 1993, IEEE 2000, TOGAF 2004, xAF 2003 p. 25):

- the set of principles should be consistent and coherent;
- the amount of principles in a set should be few;
- a principle should have a rationale, be stable, be specific, and be measurable/falsifiable.

Consider, as an illustration, Table 2. This set of example principles, taken from a real-life case, has been documented using the IAFv3 (2005) style of documenting principles.

doc-part	definition	example (building on USACO 2000, principle 7)
name	essence of the rule, easy to remember	Re-use before buy before build
description	brief, clear & precise/unambiguous statement of the principle	We will consider re-use of existing applications, systems, and infrastructure before investing in new solutions. We will build only those applications or systems that will provide clear business advantages and demonstrable cost savings.
motivation	rationale behind the principle (benefits, intentions, relationship with other principles)	• Use and availability of effective packaged solutions is increasing. • Using tested solutions reduces risks. • Reduces the total cost of ownership.
implication	impact of the principle, e.g. on other principles, design, maintenance	• Software license agreements and system development contracts should be written to allow for re-use across State government. • "The definition of "reusable" will include solutions available from other government entities (e.g., other states, federal government, etc.). • Areas that provide clear advantages and businesses cost savings are likely to require quick adaptation. • Must identify the areas in which the State is seeking to distinguish itself.
assurance	what & how will be measured to verify that this principle is achieved	Degree of re-use and extern buying will be measured and benchmarked against other states

Table 2 Documentation-standard for principles (IAFv3, 2005)

The core of a principle is formed by the description, which syntactically takes the form of a rule. In current practice, this usually takes the form of informal statements such as (taken TOGAF 2004):

> *Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations.*

This is a necessary basis to meaningfully complement the documentation of the principle with ingredients such as a rationale, ways of measuring compliance, etc.

When using architecture principles as the core element in enterprise architecture, informal statements as exemplified above arguably do not provide enough precision to concretely limit design space. As a result, they only have a limited power as a steering instrument. The call can already be heard for

SMART (Specific, Measurable, Achievable, Relevant, Time-bound) treatment of architecture principles.

In Bommel et al. (2006) a first step was taken in making principles more measurable by formalizing their formulation in a form of restricted natural language. The authors investigated the possibility of using a *business rule*-like approach for this purpose. This investigation was motivated by the strong parallel between *business rules* and principles. Consider for example the following statements taken from the business rules manifesto (Ross 2003):

- *Rules must be explicit. No rule is ever assumed about any concept or fact.*
- *Rules should be expressed declaratively in natural-language sentences for the business audience.*
- *Business rules should be expressed in such a way that they can be validated for correctness by business people.*
- *Business rules should be expressed in such a way that they can be verified against each other for consistency.*
- *Formal logics, such as predicate logic, are fundamental to well-formed expression of rules in business terms, as well as to the technologies that implement business rules.*
- *Rules define the boundary between acceptable and unacceptable business activity.*
- *More rules is not better. Usually fewer good rules is better.*

Business rules are traditionally aimed at constraining first order behavior. Principles can, however, be regarded as second order business rules constraining second order behavior of systems, i.e. constraining their design/evolution space. A possible formalization of the TOGAF example principle would be:

- Each Enterprise-function has access to Data which some User [who supports that Enterprise-function] needs for some Duties
- Each Organization has access to Data which some User [that belongs to that Organization] needs for some Duties

The language used in this example is the Object Role Calculus (ORC) (Hoppenbrouwers et al. 2005), which a further evolved variant of RIDL (Meersman 1982) and Lisa-D (Hofstede et al. 1993). As another example, consider the following TOGAF principle:

> *Development of applications used across the enterprise is preferred over the development of duplicate applications which are only provided to a particular organization.*

A possible formalization of this principle would be:

- If an Application [that is used in some Organization] results from some Development, and this Application is not a duplicate of another Application [that is used in another Organization than the former], then that Development is preferred by the Enterprise that includes both Organizations and both Applications.

An important question in this example is the way one would have to measure when one application is *a duplicate of another* application. In making such principles SMART, proper mechanisms should be defined to determine whether one application is a duplicate of another one, or more appropriately, whether one set of applications is a duplicate of another set. And more generally, in the aspect system Business one would like to measure when one process is *a duplicate of another* process, in order to detect process and organizational redundancy.

4 PRACTICAL APPLICATIONS OF PRINCIPLES

Principles have been introduced as a means for the coherent and consistent steering of change in an enterprise. In this section we will elaborate how those principles influence the enterprise engineering in practice. We saw already that documentation of the implications of a principle are required (IAFv3, 2005). To approach this finding of implications systematically, we first clarify the field of activity of an Enterprise Engineer by introducing the Enterprise Engineering Framework (EEF). In this framework, every cell represents a certain area of attention. Then we will show for a few real-life principles the impact of such a principle, using the cells of the EEF.

In the EEF, as we said in §2, an enterprise is treated as a heterogeneous system, comprising an integrated whole of three aspect systems, namely the Business system, the Informational system and the Datalogical system (after Dietz 2006 p. 115). Each of those three homogeneous systems, including its integration, is part of the field of activity of an Enterprise Engineer, and therefore spans the dimension *System Type* in the EEF (see Figure 2).

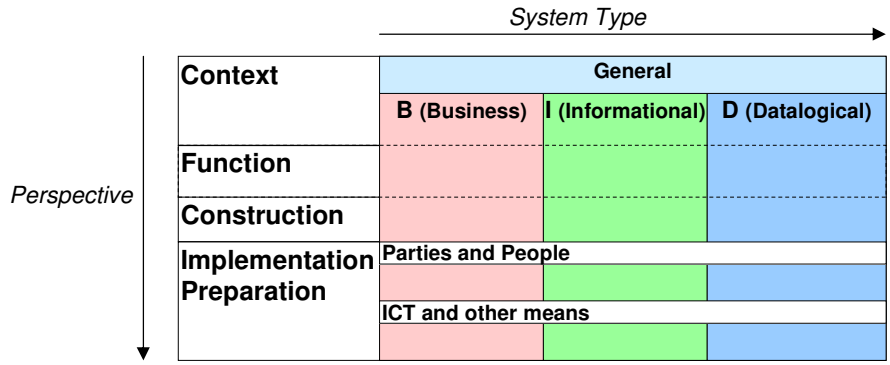

Figure 2 Enterprise Engineering Framework (EEF)

The second dimension of the EEF, the *perspective*, is mainly based on GSDP, which we discussed in §2. The functional models and requirements of an object system, i.e. its black-box perspective, reside in *Function*. The white-box models and specifications of an object system reside in two parts:

- *Construction* contains the implementation-independent model – ontology – of the object system;
- *Implementation Preparation* contains the implementation model of the object system.

The Perspective-dimension of EEF stops at the level of engineering in the narrow sense. The implementation, in which technological means are actually assigned to the elements of the implementation model, is out of scope for EEF.

Would the Perspective-dimension only consist of function, construction and implementation preparation, then something would be missing. Indeed, those three perspectives contain inherent properties about the object system itself. Following the distinction between function and purpose (Dietz 2006 p. 60), we need additional space for the relationship of the object system with its stakeholders, to express e.g. purpose and value of that object system for its stakeholders. Therefore the perspective *Context* has been added.

We illustrate the line of reasoning of the Perspective-dimension by an example of the B-system of Rijkswaterstaat. Rijkswaterstaat is an agency of the Ministry of Transport, Public Works and Water Management, which constructs, manages, develops and maintains the Netherlands' main infrastructure networks. For Dutch citizens, an important value is to keep "dry feet", in other words, make sure the land is not flooded by water. Rijkswaterstaat could support this by several products/services, e.g. by facilitating large-scale evacuations or by sustaining the coastline; the last strategy will in turn need other products/services like sand suppletion, artificial reefs, etc. Given the choice of product/service, a delivery-chain and actor-roles may be discerned for e.g. sand suppletion. Finally a decision has to be made who supplies the sand and checks the result. We summarize the conclusions, using the Perspective-dimension:

- Context: value for Dutch citizen = "To keep dry feet";
- Function: sustaining the coastline, using as products sand suppletion or artificial reefs;
- Construction: model of actor-roles and delivery-chain for sand suppletion;
- Implementation-preparation: shows which market party supplies sand on request; shows which part of Rijkswaterstaat decides where and when to supply sand and to check the result.

On every level in the Perspective-dimension, principles can make a difference. On the implementation level, this is easy to see: two bike-suppliers, which are the same in the perspective of Construction, will have a different Implementation in organization and parties, depending if they chose for or against the principle "outsource all production" (Op 't Land 2006 p. 6). The choice for the principle "we should be agile" will touch all Perspective-levels: it will give the customer the feeling his interests are early and easily heard and make the employee proud of working for such a state-of-the-art organization (Context), the organization should be proactive in its value-propositions (Function), therefore the organization should discern roles for requirements-elicitation and agility-measurement (Construction) and the organization should execute the role requirements-elicitation in co-operation with its customer (Implementation – parties & people), supporting by innovative Group Decision Support systems (Implementation – ICT and other means).

We will now focus on a way in which the impact of principles on Enterprise Engineering can be detected more completely and systematically, using the introduced EEF. As an example-principle we use the Dutch government expression "de overheid vraagt niet naar de bekende weg", roughly translatable to "the government doesn't ask the same question twice". This means a citizen or company should inform the Dutch government only once about the same fact (e.g. his income, cars, personal data etc.); after that such a citizen/company may assume any Dutch governmental organization (so on national, provincial, municipal or district water board level) is informed. This principle is specific and falsifiable; as soon as any Dutch public institution asks you information on the same subject twice, you know they are trespassing. In Table 3 we have collected part of the impact-analysis of this principle on Enterprise Engineering.

	Business	Informational	Datalogical
Context	• reduction in tax burden for citizen • cost reduction for government • more efficiency in government • simplifying use of government services by citizen • government should monitor privacy • government should prevent abuse of linking data	• government will be responsible for secure sharing and saving of data • also European authorities will request information	• collected data should be retrievable in several formats • data-exchange is based on standard-formats • data should be available as backup • technology infrastructure should be available 24*7
Function	• a government which ... o is customer-oriented o knows what's she is talking about o has its house on order o is serious with her customers and the customer data • separate governments act as a unity • improved lead time of government-services (Quality of Business)	• it must be possible to share data with European authorities • improved information (Quality of Infomation: response time, actuality, reliability, completeness etc.)	• data must be shared in several (standard-) formats • data of the citizen should be shared only when encrypted • combining data from different sources
Construction		New I-roles: • discerning authentic registers (e.g. car license plates, trade register etc.) • coordinating information	New D-roles: • data manager • data collector

	Business	Informational	Datalogical
		re-use • certifying quality of digital service-delivery and security • authorization-manager • auditor: who has seen which information?	
Implementation / parties & people	• more time for officals to improve decision-making (saving time on data collection and interpretation)	• less time spent for getting information	• no longer record the same data again & again in several formats/systems ⇒ less people required
Implementation / ICT and other means	• less physical guichets required • *DIGID* (= Dutch unique electronic citizen id)	• Connecting government-websites to authentic registrations • Pre-fill tax forms with available information • Implementation of IDA (Interchange of Data between Administrations)	• less questionnaires • national infrastructure for (public) data-exchange ○ e.g. an ESB • redundant infrastructure • standardization of data storage

Table 3 Part of the impact-analysis of the principle "the government doesn't ask the same question twice", visualized in EEF-structure

We have now shown the impact of one principle on Enterprise Engineering in the EEF. Elaborating the impact for more than one principle might lead to contradictions in implications. Consider, for example, the following two principles: (A) "we prefer standard-packages over tailored solutions" and (B) "we prevent vendor lock-in". In an efficient market these two principles need not be contradictory. However, in the specific market for operating systems of desktop-computers, currently not many alternatives exist. Comparable contradictions in implications could be found when both the principles "have flexibility in service-levels" (rationale: customer friendliness) and "standardize the work" (rationale: cost savings) are effective. By making visible the possible contradictions in implications, the governing system (GS) is supported in prioritizing, either at the level of principles or at the level of the specific case.

Using a framework such as the EEF provides several benefits. It helps to systematically find the impact of principles in a short time-frame; e.g. most of the results in table 3 were produced by a student-team in 7 hours, applying EEF for this real-life case for the first time. As a consequence it helps to detect potential contradictions in principles, thus enabling prioritizing in principles or clarifying that prioritizing for this contradiction has to be done on a case-by-case basis. Finally the EEF supports traceability; in a cause-effect-reasoning, it makes clear what consequences are drawn from what principles, thus enabling impact-of-change analysis the moment principles are no longer valid.

5 FORMULATING PRINCIPLES

An important remaining question is *how to* arrive at a set of (SMART) principles? In this section we explore two alternative approaches to the formulation of principles (eventually we actually aim to integrate these two alternatives). The first alternative takes the perspective that the formulation of a set of principles is a collaborative process involving several participants. In other words, it is a collaborative process. Nabukenya et al. (2006) discusses a general process for the collaborative formulation or policies (such as business rules and architecture principles). This general process can be specialized to architecture principles as follows:

1. Formulation of the objectives for having the principles.
2. Identify & prioritize key objectives for principles

3. Formulate candidate principles that address the stated objectives.
4. Identify & prioritize key principles
5. Elaborate key principles
6. Evaluate completeness of set of principles (go back to 4 if needed).

In the first step, the aim is to make explicit the goals for creating the principles in terms of stakeholders' concerns and objectives that should be addressed. This activity may be conducted by each of the participants individually. Once these have been identified, the participants are then required to collectively identify and prioritize the key objectives that should be addressed by the principles. This set then scopes the next activity in which candidate principles are listed and gathered, which again should be done by the participants individually. Each of the principles should be motivated in terms of the objectives it addresses. The next step is to identify and prioritize the key principles, which should again be done collaboratively. The resulting set of principles should be elaborated in the sense that each principle should be made specific and measurable as discussed in §3. This should be done collectively. Finally, the participants should evaluate the set of principles produced to see if they indeed meet the set objectives.

The second alternative is described in Op 't Land (2005). This alternative is explicitly based on *cause-effect analysis* and has been applied in both practical and educational contexts:
1. agree on the scope and the relevant stakeholders;
2. identify concerns of the stakeholders, based on their role and personal interests;
3. identify candidate-principles, derived from vision, mission and strategy of the scope;
4. identify implications of the candidate-principles, using an enterprise engineering framework;
5. detect contradictions in the principles, including the underlying assumptions;
6. define sufficiently different scenario's and find no-regret principles;
7. prioritize according to the different stakeholders and their typical concerns;
8. give an integrated advice on the over-all prioritization of principles.

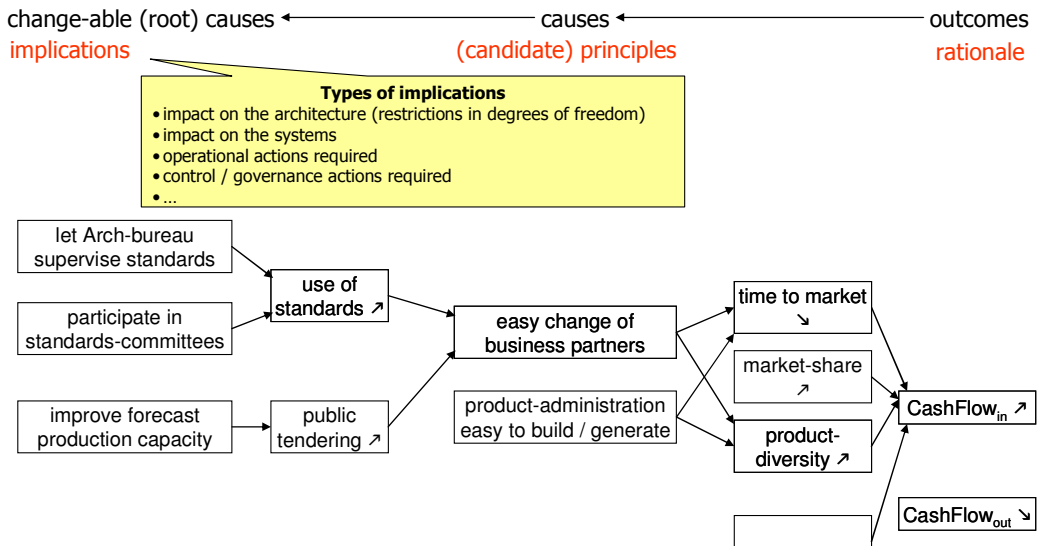

Figure 3 Part of a cause-effect diagram for principles (Op 't Land, 2005)

The first step limits the area of concern and the people to be involved and reckoned with. In step 2 we realize that stakeholders are people: they have a role (e.g. CFO) and at the same time they have personal interests (e.g. "wants to get promotion within 2 years in that area"). Step 3 is executed to safeguard that principles are in line with and steered by the vision, mission and strategy of the area scoped. Step 4 is the exercise we demonstrated in §4, using the EEF. In step 5 it is important to not only detect contradictions in the principles, but especially to articulate the assumptions under which those contra-

dictions occur; those assumptions might be subject to change indeed! For example, as we pointed out in §4, the principles "standard-packages preferred" and "no vendor lock-in" need not be contradictory in the situation of an efficient market; by making this assumption explicit, stakeholders become aware of their dependency on market-efficiency or of their opportunities to stimulate the market. In step 6 scenario planning is applied to principle formulation. A few extreme alternative futures (= scenarios) are drafted and the principles are tested against them, using the assumptions from step 5. If a principle works in every scenario, it is a no-regret principle and consequently accepted; otherwise, either extra investments are done to still let the principle work or "early warning indicators" are installed to see which scenario becomes more probable. In step 7, for all stakeholders the principles are prioritized according to their concerns; by comparing the difference in priorities also there no-regret principles are found and underlying assumptions clarified. In step 8, the no-regret principles are accepted and on the other principles negotiations take place, supported by the insights of the different interests.

The pivot in this approach is a cause-effect analysis, starting from concerns of stakeholders and ending in implications. Figure 3 shows a part of such an analysis, starting from the concern to *increase incoming cash flows*. This concern can be answered by *decreasing time-to-market*, *increasing market-share* and *increasing product-diversity*. Both *easy change of business partners* and *product administration should be easy to build or generate* positively influence *time-to-market* as well as *product-diversity*. In its turn, the *easy change of business partners* is positively influenced by *increasing use of standards* and *increase public tendering* etc. Somewhere "in the middle of the diagram" the candidate-principles relatively easily emerge, at the same time clarifying (1) the rationale of each principle, (2) the implications of each principle and (3) the mutual coherence or contradiction of principles.

As a next step, we aim to better relate and integrate both ways of working into a more explicit and elaborate way of working catering for the underlying cause-effect analysis and the collaborative nature of principle formulation processes.

6 CONCLUSIONS AND FURTHER RESEARCH

In this paper we have studied some cases from industry, focusing on the impact of principles on the design of enterprises as well as the guidance, which designers may receive from these principles. We have explored how, using an enterprise engineering framework such as the EEF, the impact of principles can be systematically detected in a short time, thus clarifying mutual coherence, potential contradictions and providing traceability. The process of formulating principles can base itself on more generic collaborative patterns in the area of strategy formulation. A promising technique here is the use of cause-effect diagrams, because it readily connects concerns and rationales of stakeholders with implications.

We have also identified some key areas that need further work. To make this more explicit, we are currently working along the following lines to broaden and deepen the principles mechanism:

1. Gather documented examples, originating from a diversity of sectors, concerning the use of principles in real-life practice. Note that with "use" we do not only refer to the act of putting them on paper, but to documented impact on enterprise engineering. In other words, documenting their shown ability to limit design space during enterprise engineering.
2. Further elaborate on strategies to formalize principles and their underlying domain concepts.
3. Get explicit insight in the impact of levels of principles. Some principles in an organization originate from higher organizational levels, others from industry standards, again others from external authorities. How do the principles from this several levels and steering lines interact?
4. Work towards a more explicit definition of principle, while meeting the requirements put on them from practical use. This should include a (formal) language in which to express the underlying constraints on design space allowing/forcing for specific and measurable formulations.
5. Mechanisms to indeed enforce principles and guide designers in their design activities. How to actually steer by using principles, e.g. by organizationally embedding the formulating of principles and the issuing of building permits.

6. A real-life tested way of working for finding/formulating principles, ensuring shared understanding and shared commitment for the impact of these principles, taking both collaborative and cause-effect reasoning into account,
7. Mechanisms by which principles can be rationalized in terms of an enterprise's strategy. One would typically expect principles to follow from an enterprise's strategy. However, how this is to take place exactly needs further investigation.

References

Bommel, P. van, Hoppenbrouwers, S.J.B.A., Proper, H.A., and Weide, Th.P. van der (2006). Giving Meaning to Enterprise Architectures - Architecture Principles with ORM and ORC. In Proceedings of the OTM Workshops 2006, (R. Meersman, Z. Tari, P. Herrero et al., Eds.), LNCS, Springer Berlin Heidelberg.

CIAO! (2006) Research Program on Cooperation, Interoperability, Architecture and Ontology. http://www.ciao.tudelft.nl/

Clinger-Cohen (1996), IT Management Reform Act, 1996, USA. http://www.cio.gov/Documents/it_management_reform_act_Feb_1996.html

Davenport, T.H., Hammer, M. and Metsisto J. 1989, How executives can shape their company`s information systems. *Harvard Business Review*, 67(2):130-134, March 1989. doi:10.1225/89206

Dietz J.L.G. (2005). The third wave. Plenary presentation for the Dutch National Architecture Congress 2005 (LAC2005). See http://www.lac2005.nl/Uploads/Files/Dietz.pdf.

Dietz J.L.G. (2006). Enterprise Ontology – theory and methodology. Springer Berlin Heidelberg.

Hofstede, A.H.M. ter, Proper, H.A., and Weide Th.P. van der (1993). Formal definition of a conceptual language for the description and manipulation of information models. Information Systems, 18(7):489-523, October 1993.

Hoppenbrouwers, S.J.B.A., Proper, H.A., and Weide, Th.P. van der (2005). Fact Calculus: Using ORM and Lisa-D to Reason About Domains. In Proceedings of the OTM Workshops 2005, (R. Meersman, Z. Tari, P. Herrero, Eds.), LNCS, Springer Berlin Heidelberg.

IAFv3 (2005) Integrated Architecture Framework version 3.9 Archifacts Reference. Material of the course "IAF Essentials". Capgemini

IEEE (2000), *Recommended Practice for Architectural Description of Software Intensive Systems*. Technical Report IEEE P1471--2000, IEEE, Piscataway, New Jersey, USA, September 2000.

Lankhorst, M.M. et al (2005). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, 2005. ISBN 3540243712

Meersman, R. (1982). The RIDL Conceptual Language. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, 1982.

Nabukenya, J., Bommel, P. van and Proper, H.A. (2006), Collaborative Policy-Making Processes, Technical Report: ICIS-R6036, December, Radboud University Nijmegen, The Netherlands.

Op 't Land, M. (2005) Principles and Architecture Frameworks. Educational material of University-based Master *Architecture in the Digital World*. Radboud University Nijmegen, The Netherlands

Op 't Land, M. (2006). Applying Architecture and Ontology to the Splitting and Allying of Enterprises: Problem Definition and Research Approach. In Proceedings of the OTM Workshops 2006, (R. Meersman, Z. Tari, P. Herrero et al., Eds.), LNCS, Springer Berlin Heidelberg.

Ross, R.G., editor (2003). *Business Rules Manifesto*. Business Rules Group, November 2003. Version 2.0. http://www.businessrulesgroup.org/brmanifesto.html.

Tapscott, D. and Caston, A. (1993). *Paradigm Shift - The New Promise of Information Technology*. McGraw-Hill, New York, New York, USA, 1993. ASIN 0070628572.

TOGAF (2004). *TOGAF - The Open Group Architectural Framework*, 2004. http://www.togaf.org

USACO (2000). Enterprise Architecture Strategies, Conceptual Architecture Principles, USA/State of Connecticut, version 8/22/2000, http://www.ct.gov/doit/LIB/doit/downloads/conarch.pdf

xAF (2003). Extensible Architecture Framework version 1.1 (formal edition); report of the NAF-working group xAF. See http://www.naf.nl/content/bestanden/xaf-1.1_fe.pdf