2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops

A Practical Approach to the Formulation and Use of Architecture Principles

D. Greefhorst* and H.A. Proper†‡
*ArchiXL, Amersfoort, The Netherlands
†Public Research Centre – Henri Tudor, Luxembourg
‡Radboud University Nijmegen, Nijmegen, The Netherlands

***Abstract*—This paper describes an approach and accompanying process for the development and use of architecture principles. In doing so, it builds on earlier work in which we defined the concept of architecture principle itself. The approach presented in this paper is based on a combination of experiences gathered from practice, as well as a synthesis of past work and other sources from both academia and industry, including standards such as TOGAF. The approach is practical in the sense that it provides more detail on how to develop architecture principles than offered by other source. Also, it shows the 'magic' involved in how to translate drivers to architecture principles.**

***Keywords*-enterprise architecture; architecture principles;**

I. Introduction

Enterprise architecture, and its associated formulation, implementation and governance processes, are increasingly recognized by organizations as an important capability [1]–[3]. Several approaches to enterprise architecture position principles as an important ingredient [2]–[7], while some even go as far to position principles as being the essence of architecture [8]–[11]. Architecture principles fill the gap between high-level strategic intentions and concrete design decisions.

The concept of architecture principles has not received a lot of research attention [12], while at the same time, there is a need to better understand their essence. In previous work, we therefore focussed on indeed gaining a better understanding of the essence of architecture principle [13], [14]. In this paper we turn our focus to the processes involved in formulating and using principles. In our work on architecture principles, we essentially use a *design science* based research approach [15]. Having a process for formulating and using principles, based on the framework presented in [13], [14], enables us to actually conduct real-life case studies that will validate both the original framework and the associated processes, as well as provide insight for further refinements.

The remainder of this paper is structured as follows. Section II provides a brief summary of the framework presented in [13], [14], providing the reader with a core understanding of the concept of architecture principles as used in the remainder of this paper. In Section III we continue with the identification of some of the core drivers of architecture principles. We will illustrate these in terms of a running example. Section IV then presents the suggested process for formulating and using architecture principles. These processes will also be illustrated by means of an example. Before concluding, we reflect upon the process as described and describe related work.

II. Architecture Principles

In this section we provide a brief summary of the conceptual framework for architecture principles as introduced in our earlier work [13], [14]. This framework also aimed to provide a synthesis of the role, and concept, of principles as used by existing views on enterprise architecture and enterprise engineering [2], [3], [8], [16]. A more elaborate discussion on the concept of principles can be found in our forthcoming book [17].

A. Scientific principles versus normative principles

An important distinction that needs to be made within our field, is the distinction between *scientific principles* and *normative principles*. The American Engineers' Council for Professional Development [18] states that engineering concerns "*the creative application of* **scientific principles** *to design or develop structures, machines, apparatus, or manufacturing processes, or works utilising them . . .* ". These principles are used in a wide range of engineering disciplines such as industrial engineering, chemical engineering, civil engineering, electrical engineering and systems engineering. They can be seen as a form of design knowledge that should be shared, in order to increase the quality of designs. In line with [18], we will refer to these principles as *scientific principles*. We define a scientific principle as "a law or fact of nature underlying the working of an artifact". Principles from general systems theory, such as the law of *requisite variety* [19], are examples of scientific principles that are applicable in an enterprise engineering context.

The other important class of principles are the *normative principles*, which we define as "a declarative statement that normatively prescribes a property of something". In contrast to scientific principles, normative principles are not enforced by nature but require explicit attention to be enforced. We regard architecture principles to be a specific form of normative principle; they guide/direct the enterprise by normatively restricting design freedom. This is in line with the common interpretation of the term. TOGAF [3] states that "principles are general rules and guidelines, intended to be enduring and

978-0-7695-4426-7/11 $26.00 © 2011 IEEE
DOI 10.1109/EDOCW.2011.18

seldom amended, that inform and support the way in which an enterprise sets about fulfilling its mission".

B. Credos and norms

In practice we see architecture principles at several levels of precision. At the start of their life-cycle, normative principles are just statements that express the enterprise's fundamental belief of how things ought to be. At this stage, their exact formulation is less relevant. We call such principles "credos", where we define "credo" as "a *normative principle* expressing a fundamental belief". Credos can be seen as normative principles in their initial stage. They are not yet specific enough to actually use them as a norm.

When enterprises want to use architecture principles as a way to actually *limit* design freedom, the principles need to be more specific. This is when a more precise formulation of the principle becomes important. They need to be formulated in such a way that compliance to them can be assessed. Once normative principles have been (re)formulated specific enough to use them to restrict design freedom, we can start referring to them as a *norm*. We define a norm to be "a *normative principle* in the form of a specific and measurable statement".

C. Architecture principles versus design principles

Another distinction that is relevant for normative principles is that between *architecture principles* and *design principles*. As suggested by the IEEE [16] and TOGAF [3] definitions of architecture, the architecture level should focus on fundamental aspects. An enterprise architecture should provide an elaboration of an enterprise's strategy, while focussing on the core concerns of the stakeholders. As such, an architecture is typically positioned at a level concerned with a class of systems. A design focuses on the remaining requirements and design decisions pertaining to a specific system being developed, which will typically have a limited impact on the key concerns of the stakeholders.

Fehskens [20] states that architecture should explicitly address alignment, relating the role of architecture to the mission. He redefines architecture as "*those properties of a thing and its environment that are necessary and sufficient for it to be fit for purpose for its mission*". In his view, architecture should focus on what is essential, on "the stuff that matters". This equates to those properties that are necessary and essential. This is also what distinguishes architecture from design; architecture is really essential design. A different architecture implies a different mission, whilst different designs may address the same mission.

The distinction between design and architecture, also allows us to distinguish between *architecture principles* and *design principles*. We define a design principle as "a *normative principle* on the design of an artifact". As such, it is a declarative statement that normatively restricts design freedom. In contrast, we define an architecture principle as "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.

III. Motivating architecture principles

Architecture principles do not just drop out of the sky. Depending on the specific situation, different drivers will lead to the formulation (and enforcement) of architecture principles. These drivers will ultimately 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.

A. Sources for finding motivation

There is no universal agreement yet 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 [21], [22].

Another source of inspiration is the business motivation model [23]. As its name suggests, this model provides important concepts to express business motivations. The model was initially created to provide the motivations behind business rules, but can also be used to find the motivation for architecture principles.

TOGAF [3] 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 [7] 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 [24] also provides an interesting source of inspiration for motivating architecture principles. This approach suggests nine building blocks to describe the rationale of how an organization creates, delivers and captures value. Given that these choices determine the business model of the organization, they should also lead to essential properties to be met by the enterprise.

B. Drivers as motivation for architecture principles

Based on the sources mentioned above, as well as our own experiences in practice, we propose six types of drivers for the formulation of architecture principles: *goals & objectives*, *Values*, *issues*, *risks*, *potential rewards*, and *constraints*.

1) Goals & objectives: These are targets that stakeholders within and outside an enterprise seek to meet. They can be very high-level, such as *decrease costs*, but 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. Goals and objectives should be the main drivers for architecture principles. Without ends, any means will do.

2) Values: 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 [25] and IEEE 1061 [26] are a good source of inspiration for these quality attributes. The formulation of a desired property can be used to describe how values should be expressed in practice.

3) Issues: These 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*".

4) Risks: These are very much comparable to issues; they are essentially problems/issues 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 desired properties. 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 *there are single points of failure in the infrastructure that may lead to unavailability of IT systems*.

5) Potential rewards: These drivers are essentially what is referred to as business opportunities in [27]. 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.

6) Constraints: 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. 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*.

IV. Generic process for principles

We propose a generic process that handles the entire life-cycle of architecture principles. We have distilled this process from existing methods, research and case studies on architecture principles. An initial process was presented in 2007 [28], which was enriched with our experience with developing architecture principles at an insurance company [29]. We elaborated and validated the process in a workshop held in 2007 [30]. Later, we started a working group on architecture principles in which additional research and experiments were conducted [31]–[33]. The process as described is based on the results of this working group, as well as on additional experience with clients in the public and financial sectors. Inspiration has also been drawn from literature on policy making [34] and requirements management [22], [35]. As such, the presented approach is a synthesis of existing views and experiences.

We do recognize that the process still needs refinement. Most of our experience has been focused on the determination and specification of architecture principles, and the drivers on which they are based. That means that more experience and evaluation is needed with the other subprocesses. Also, in practice it remains difficult to apply the process with enough depth and formality, due to limited time that is typically available. Most of the time only a subset of the drivers are determined and architecture principle specifications are restricted to a basic structure. We believe that more formality could deliver more value.

The process itself consists of eight subprocesses (see Figure 1).

Determine drivers – where the relevant inputs for determining architecture principles are collected, such as the goals and objectives, issues and risks.

Determine principles – where the drivers are translated to a list of (candidate) architecture principles. At this stage the architecture principles can be considered credos.

Specify principles – where the candidate principles are specified in detail, including their rationale and implications. This subprocess translates architecture principles from credos to norms.

Classify principles – where architecture principles are classified in a number of dimensions to increase their accessibility.

Validate and accept principles – where architecture principles, their specifications and classifications are validated with relevant stakeholders and formally accepted.

Apply principles – where architecture principles are applied to construct models and derive design decisions in downstream architectures, requirements and designs.

Manage compliance – where architects ensure that the architecture principles are applied properly, and dispensations for deviations may be given.

Handle changes – where the impact of all sorts of changes

on the architecture principles is determined and new method iterations may be initiated.

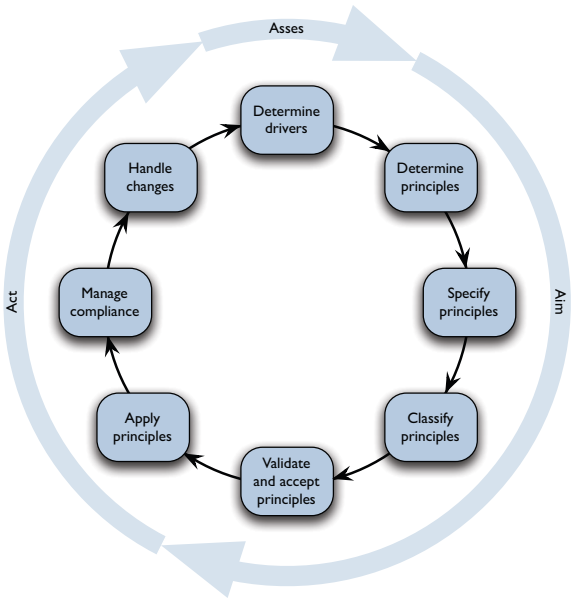

Figure 1. Generic process for handling architecture principles, adopted from [17], © Springer-Verlag Berlin Heidelberg 2011.

In the remainder of this section, we provide a summary of each of these subprocesses. A more elaborate discussion is included in our forthcoming book [17].

In the remainder of this section, we will discuss the subprocesses in more detail. Note, however, that the presented generic process is required to be translated to an organization-specific process catering for the situation at hand. We will illustrate a number of subprocesses with a case, which revolves around a municipality. The case is based on a real-world example, in which architecture principles were defined for the information systems and technology area of the municipality.

A. Determine drivers

In Section III we have described the drivers for architecture principles: goals, objectives, values, issues, risks, potential rewards and constraints. In this subprocess, the drivers that are relevant in a specific context are identified and described. Drivers are ideally defined outside the scope of the architecture function. In practice, however, they do need to be gathered explicitly before architecture principles can be identified. Drivers that are not explicitly documented may have to be elicited from stakeholders. It is the role of the architect to ensure that the definitions of these drivers are current, and to clarify any areas of ambiguity.

The exact nature of the goals depends on the exact scope and context of the architecture engagement. A selection of the most important drivers at hand is made, leaving other drivers implicit. In order to identify issues, carefully look for topics that have caused a lot of discussion in the past. It is important to identify any differences in opinions. Drivers may be found/uncovered by studying existing internal and external documentation, as well as by asking stakeholders.

We recognize the importance of all drivers identified earlier, but do feel that using them all may lead to an overly complex process. The goals and issues are the basic drivers that should be addressed. Others may be added in later iterations. Most drivers can also be (re)formulated as goals. For example, if efficiency is an important value in an organisation, then a goal may be to increase the efficiency. Also, the specific drivers that should be used are very dependent on the specific organizational context and motive for the architecture engagement.

Having identified the types of drivers, the next step is to determine which information on these drivers is needed in order to determine the architecture principles. Often, drivers are poorly documented or the documentation is hard to find. In such a case one needs to find the stakeholders and/or the subject matter experts that do have the understanding. A more structured approach may even be necessary. Depending on the type and amount of information it may be necessary to perform a market analysis, study documentation, conduct interviews, organize workshops or even to organize questionnaires. The identified drivers should be validated with the stakeholders. What may seem a driver for one stakeholder, may seem irrelevant to someone else. Also, stakeholders may value drivers differently, and prioritization of drivers is advised. In the engagement of stakeholders drivers will be moulded, combined, split, removed or added until all stakeholders are satisfied with the result.

The final step in the determination of drivers is their explicit specification in the form of an architectural requirement. This results in a list of statements with a unique identification, that is the basis for the determination of architecture principles. It thereby enables traceability from drivers to architecture principles, as well as requirements management of these drivers.

We will now turn to our case, a municipality, to illustrate the activity of determining drivers. As mentioned before, the focus of the engagement was the definition of a set of architecture principles for the information systems and technololgy area. Various documents were studied, and interviews with information advisors were held to uncover the drivers. The goals that were found have been filtered for their relevance to information systems and technology, and clustered into five themes: reducing expenses, improving electronic and telephone services, implementing a case-oriented style of working, moving to a new location, collaborating and sharing services in the region. In addition, issues have been identified in the current information systems and technology landscape. A report had already been written

by a consulting firm that identified issues in the process of providing permits, which is an core process for the municipality. In particular, the process lacked standardization, did not provide enough management information and led to too many complaints by citizens. Other issues identified included the lack of alignment between business and IT, a high pressure on the IT department, and an incomplete and inconsistent documentation of configuration items.

B. Determine principles

After having determined the drivers it is possible to determine the architecture principles. This is where the 'magic' comes in; how to translate drivers to architecture principles? What makes this process complex is that there are different types of drivers, and that they may be formulated in many different ways. We see the following three basic activities when determining architecture principles:

Generate candidate principles – generates a list of candidate architecture principles that realize the drivers.

Select relevant principles – selects those architecture principles that are relevant to the specific architecture.

Formulate principle statements – specializes or generalizes the candidate architecture principle statements into the proper abstraction level.

These activities are typically used in combination, where there is also a logical flow from generation, through selection to formulation. The following subsections describe the activities is more detail, and show specific approaches and techniques that can be applied.

1) Generate candidate principles: The generation activity is where architecture principle determination starts. Basically three different approaches to generation exist: derivation of architecture principles from the drivers that were identified earlier, elicitation of domain knowledge and harvesting of existing architecture principles. We describe these three approaches in turn.

Deriving architecture principles from drivers ensures that these are properly motivated, which is very relevant to get commitment from stakeholders. The idea behind the activity is that architecture principles are a realization of some driver. They are a means to an end. A certain amount of creativity is needed in this activity in order to generate candidate architecture principles. By comparison: what we refer to as 'derivation' is comparable to what is called 'refinement' in the Goal-Oriented Requirements Engineering approach [22]. In that approach, 'how' questions are positioned as a means to refine goals into requirements or subgoals. Since not all our drivers are goals, and we look for architecture principles we propose the following questions per type of driver:

For goals and objectives – What is needed to attain the goal or objective?

For issues – What is needed to solve the issue?

For values – What is needed to realize this value?

For risks – What is needed to minimize the probability or the impact of the risk?

For potential rewards – What is needed to attain the potential reward?

For constraints – What is needed to enforce the constraint?

Elicitation of domain knowledge is an approach that does not replace, but can be used in combination with derivation. Domain knowledge is essential to truly understand the drivers and to come up with the proper solutions, and requires input from subject matter experts. Domain knowledge is often tacit, but may also be documented in the form of scientific principles. Such scientific principles describe potential solutions, and as such can be considered as architecture principles in their first inception. In that sense they are an important part of the knowledge that the architect brings into the process. Nevertheless, in depth domain knowledge should be gathered from subject matter experts.

Harvesting existing architecture principles also provides a starting set for new architecture principles. In contrast to the existing architecture principles that were mentioned in the previous subprocess and previously in this section, the architecture principles we refer to here are those that have not formally been agreed upon. As a result, they cannot be used as drivers. An interesting source is the set of solution architectures, which may contain architecture principles that can be reused. Ideally, an architecture repository exists in the organization that provides a collection of architecture materials from previous architecture experiences or external sources. Note that it is not recommended to solely depend upon harvesting existing architecture principles since the most important architecture principles may be missed.

2) Select relevant principles: Given that the previous activity was executed, selection starts with a list of candidate architecture principles. This list needs to be filtered, so that only the architecture principles that are relevant are included. Also, limiting the number of architecture principles is important to limit the time required from stakeholders, and to ensure accessibility of the resulting architectural description. Do not be afraid to throw away architecture principles that do not really express an essential choice and/or are not specific enough for the organizational context. Not all candidates may be at the right level of genericity. This is not yet relevant at this stage; formulation of the real architecture principle statement will be performed in the next activity. Selection can be seen as a form of prioritization, that is executed at an early moment in time.

An important thing to do at this stage is to filter out things that are not really architecture principles. A brainstorm typically results in all sorts of statements, which includes architecture principles but also actions, requirements, strategic decision, business principles, IT principles and more detailed design principles. The following questions

are relevant in order to determine whether the statement is really an architecture principle:

- Does it describe a functionality that is needed? In that case it is probably a (functional) requirement.
- Does it describe something that needs to be done? If it does, then it is probably an action.
- Are there objective arguments that support it? If it does not, then it is probably a (potential) strategic decision or business principle?
- Does it have impact on the design of the organization and/or the IT environment? If if does not, then it is probably a business principle (if it influences the daily business operations) or an IT principle (if it influences the daily IT operations).
- Does it have impact on the design of multiple systems? If it does not, then it is probably a more detailed design principle or design decision.

3) Formulate principle statements: The previous activities have led to a list of candidate architecture principle statements, which may not exactly be at the level of architecture principles and/or match the organizational context. This activity transforms the statements to the right level of abstraction, and finds a balance between genericity and specificity of the architecture principles. Although the end-result of this activity needs to be validated, most of the work in this activity can be performed by an architect in solitude.

An important insight is that architecture principles can be regarded as generic requirements that can apply to a number of solutions [9]. Previous activities may have come up with statements that apply to a different scope than that of the architecture, and generalization or specialization of these statements may be needed. Architecture principles should apply to all solutions that match the scope of the architecture. It is important to carefully determine the extent of generalization that is needed. You should not generalize too much, since that can have a counter-productive effect. In particular, such architecture principles may have implications that were not foreseen and undermine the credibility of the architecture.

In the municipality we use as an example, the determination of architecture principles was a two-step effort. The initial identification was performed by the architects that were responsible for the engagement. They used the drivers and issues they found earlier, and translated these to candidate principles by asking themselves what would be needed to reach the goals or to solve the issues. A reference architecture for municipalities [36] was used as a source of inspiration for these candidate principles. Subsequently, a workshop was organized with a number of information advisors and project leaders. The participants in the workshop were asked what was needed to reach the goals and solve the issues, in a brainstorm fashion. This resulted in a list of terms and statements of various types. After the brainstorm, all these terms and statements were classified as being: 1) a candidate architecture principle, 2) an action or 3) a functional requirement. The candidate architecture principles were clustered into groups, and a statement was formulated for all these clusters. Subsequently, the list of candidate principles defined beforehand were discussed in the group, and combined with the list that the group came up with. The statement was moulded until all workshop participants were satisfied. Diagrams were constructed after the workshop to show the relationship between the architecture principles and the goals and issues they were derived from.

The following architecture principles were determined:

- The front-office is the channel-indepenent entry point for all services
- Applications and infrastructure are standardized and reused
- Applications are bought and not custom developed
- Singular acquisition and multiple use of information
- Information items have a clear owner
- The municipality works process- and case-wise
- Employees have access to all relevant information, independent of their time and location
- Applications and infrastructure use open standards
- Information is stored centrally and digitally
- Information development is executed under architecture
- The infrastructure is consolidated and virtualized

C. Specify principles

After the architecture principles have been determined they need to be specified in more detail. Further detailing of the architecture principle is a prerequisite for actually using it to restrict design freedom, and converts the architecture principle from a credo to a norm. This means that all relevant attributes that have been chosen need to be described. Actually using architecture principles to restrict design freedom may not be needed in all situations. The determination of architecture principles also builds common understanding and commitment for certain issues, which may be sufficient in certain situations. This also depends on the architecture maturity and culture of the organization.

Typically architecture principles are specified in an iterative manner. In the previous subprocess the statement has been defined. This is the most essential part of the architecture principle, and in some situations it may even be sufficient. When the statement is constructed in a collaborative process, chances are that it needs to be refined later on to ensure that it expresses the exact intentions. It is at this moment in time that other attributes will also be drafted by an architect, starting with the basic structure that contains the rationale and implications. A number of sessions with different stakeholders may be necessary to refine the specification. Other attributes can be added later.

As part of the specification process, architecture principles may be prioritized. This is especially relevant in determining

the guiding architecture principles. These are the most fundamental ones. Those that truly make a difference, are the hardest to change and are closest to the drivers. Determining the guiding architecture principles is important since top-level architectures should only contain a limited number of architecture principles. A rule of thumb is to have no more than 10 guiding architecture principles. More than that decreases the accessibility of the architecture, and obfuscates the importance of the most important architecture principles.

In the municipality described earlier, the drafting of architecture principle specifications was performed in the same workshop in which they were identified. Workshop participants looked at the statement, and interactively determined its most important consequences. Although the formulation of these consequences was not finalized in the session, some time was spend for every consequence ensuring that the right wording was used. After the workshop, the architects refined the architecture principle specifications. They reformulated some of the sentences, and added the relationship to the drivers as well as other motivation as rationale for the principles.

An example of an architecture principle resulting from this activity for this municipality:

The infrastructure is consolidated and virtualized

Rationale:

- Supports the goal to reduce expenses
- Consolidation reduces costs and maintenance effort

Implications:

- Applications are installed on the consolidated and virtualized environment
- Support for virtualization is a selection criterium for applications

D. Classify principles

After the architecture principles have been specified it is useful to classify them in terms of some framework, to ease their accessibility and maintainability. For example, along (a subset of) the dimensions of the architecture framework used in the organisation. The importance of classification depends on the number of architecture principles, their breadth of application and the ambition level for handling architecture principles. At a low ambition level there are probably only a limited number of architecture principles, and adherence to them is not formalized. At a high ambition level there could be hundreds of architecture principles, scattered around a large number documents, owned by different stakeholders and governed by a formalized process. This is when classification of architecture principles becomes important. It increases their accessibility, by providing an inherent navigation structure in them. You can find them based on their classification. Ideally, they are stored inside an architecture repository where they can be traced to other artifacts and included in queries and impact analyses. A diagram that contains the classification (typically a diagrammatic representation of the architecture framework) can be shown as an entry point for people that want to query the repository.

Since the number of architecture principles identified for the municipality was small, there was no need to classify them in terms of a framework.

E. Validate and accept principles

Architecture principles are important since they provide the means to govern changes in the organization. A large part of the organization should be able to understand them, commit to them and act accordingly to them. Given their importance, it is clear that they need to have a high quality. This means that validation as well as formal acceptance of architecture principles is essential. Although described as a separate subprocess, all previous subprocesses should also include some form of validation. This builds commitment for them with the stakeholders, and prevents rejection in the validation subprocess. However, describing it as a separate subprocess stresses that it is also a formal subprocess that provides a quality gateway.

This subprocess should include an architecture review process. Depending on the context, this can be a highly formalized process performed by specific personnel, or it can be a review process that is organized by the original author of the architecture principles. The result of the review process should be discussed, agreed upon and signed off in an architecture board with management representatives of all major departments. This ensures that the architecture principles have a formal status, and that management will support them in any discussions and potential escalations.

An important part of the review process are the quality criteria that can be used to determine the quality of the architecture principles. We propose to use: specific, measurable, achievable, relevant and time-framed as the key quality criteria. For sets of architecture principles the proposed quality criteria are: representative, accessible and consistent. The review process as well as the criteria should however be customized and refined to the organizational context. The same holds for the architecture compliance review process that is discussed later in this section.

In the municipality we described, the validation of architecture principles was performed in two steps. The first step was to send them to the participants of the workshop that was held earlier, in order to determine whether they were comfortable with the final formulation and/or were missing anything important. The second step was to send them to the management team of the IT department, and present them at a management meeting. In this meeting management was asked to validate and accept the architecture principles, after which they were finalized and communicated formally to the organization.

F. Apply principles

Since the proof of the pudding is in the eating, it is strange that very little has been documented about the actual application of architecture principles. How do architects and designers actually use the architecture principles to base their own artifacts upon? Depending on the architecture review process is probably too late since a lot of decisions will already have been made at that moment. Also, turning back decisions that have already been made requires a lot of energy that can better be spent in a constructive way. This section therefore provides some ideas on how to actually use architecture principles in a constructive way.

Using architecture principles requires a good understanding on the artifacts that are impacted by them. The architecture itself, as well as any downstream architectures need to translate the architecture principles to more detailed architecture principles and instructions, as well as to models of various aspect areas. Solution requirements need to be validated for compliance with the architecture principles. Also, new solution requirements may be derived from the architecture principles. Solution designs contain all sorts of models, detailed design principles, design instructions and design decisions that need to be validated against the architecture principles and that may partly be generated from the architecture principles. In practice, the usage of architecture principles suggested above is entirely a manual process, depending on the knowledge and experience of professionals. An important part of the application of architecture principles is what we call "transformation".

We distinguish two types of transformation. The first transformation is *derivation*, where more specific directives are identified that realize an architecture principle. This requires the transformation of architecture principle to statements that are relevant in a more specific context. Actually, it can be seen as an approach to identify more specific implications of the architecture principle and instructions that follow from it. The implications that are part of the generic architecture principle provide a good starting point. In a solution architecture, this insight leads to an approach in which architecture principles from upstream architectures are transformed to requirements for the specific solution. Explicitly documenting this transformation in the solution architecture is advised, since this enables traceability from the architecture principles to the solution. Such information can also be used in a compliance review; it can easily be determined whether and how architecture principles are adhered to. Enterprise architects may have performed this transformation before a project starts, and provide it as an input to the project.

Another form of transformation is from architecture principles to models and their diagrammatic representations. This transformation applies to models at various levels, such as enterprise architecture models, solution architecture models and design models. The value of architecture principles in this transformation is that they can become the rationale behind a number of elements in the model, thereby increasing its quality and value. In the most basic form this may be a model that is a direct consequence of the application of the architecture principle itself, which may only be possible for a limited number of architecture principles. In particular, the architecture principle needs to have a direct impact on the identification of specific design elements or relationships between elements. This especially holds for architecture principles that focus on the need to distinguish between several elements or types of elements. An example in the business domain is the distinction between front-office and back-office process areas. These architecture principles can easily be modeled and visualized in diagrams with the elements identified. Downstream models such as design models must then respect the distinction between the elements.

G. Manage compliance

Although the intention is that people adhere to the architecture principles, a process needs to be in place to manage compliance to them. What makes this even more important is the fact that, there can always be good reasons to deviate from architecture principles since not all situations and consequences can be taken into account during their specification. It can even be valid to adjust the architecture principles based on insight originating from specific situations.

Architecture compliance assessments provide management with insight on the actual implementation of the architecture. It provides them with an instrument to highlight potential problems, and the opportunity to act upon it before it is too late. It also provides the architects with the much needed insights on the actual impact of their architecture. An effective architecture compliance process is executed at various moments in the life-cycle of projects, starting from the moment when an initial project definition becomes available, and ending upon project completion to ensure project insights are harvested. Also, an architecture compliance process needs an overall architectural governance framework in order to be effective. Amongst others, this implies a clear architectural organization such as an architecture board, and clear roles and responsibilities.

Architecture principles are the primary enablers for an effective architecture governance. They express what is really important, what the consequences for the organization are and how we can assure their implementation. You can see them as self-contained pieces of architecture governance that come with their own description on how they should be governed. In that sense they are much easier to govern than architecture models. It is up to the reader of an architecture model to interpret how to ensure compliance with it. By simply looking at an architecture model, one

cannot see which part of the model is important, why it is so important, what happens when not adhering to the model and how that model propagates to design models. These characteristics make architecture principles an ideal architectural governance instrument.

H. Handle changes

Although architecture principles should be stable, new insights and developments may surface that have impact on them. These insights typically come from architecture compliance reviews, but can also come from various other processes and sources. In general, architects are responsible for continuously monitoring all potential drivers as mentioned in Section III. From that responsibility they are an important source for potential changes, but not the only source. A change management process is needed to guide the organization in handling all these drivers for change. The most important part of such a process is a classification scheme of types of changes, that provides guidance on the appropriate steps to take. In particular, smaller changes can be handled by simply 'patching' the architecture principles, whilst bigger changes may require a new architecture development iteration. Also, there should be a standard periodic architecture refreshment cycle in which changes can be incorporated. TOGAF proposes to classify required architectural changes into simplification changes, incremental changes and re-architecting changes.

Architecture principles are largely self-contained and that this provides an opportunity for small-scale release management. In particular, architecture principles could be published and updated independently of each other. This does require a well thought-through release strategy with specific attention to the publication mechanism and the way people are informed of recent changes. Ideally, there is a central architecture repository that is available to all employees through an Intranet, where they can see the architecture principles and changes to them. This is also an opportunity to implement a feedback mechanism where people can comment on architecture principles, request changes or discuss with peers on specific experiences.

V. Related work and conclusions

A. Related work

Nabukenya et al. [37] discuss a general process for the collaborative formulation of policies. This general process is specialized to architecture principles by Op 't Land [28]. The focus of this process however, is on the determination and specification of architecture principles, and not so much on what happens to them afterwards. The same holds for the approach as described in [7], although they do provide deeper insights into architecture principles in general.

TOGAF [3] provides an Architecture Development Method, which also includes steps for the development of architecture principles. TOGAF does not make the handling of architecture principles explicit in all phases and steps. Our generic process is also more detailed than the ADM. The latter does not distinguish between determining, specifying, classifying and validating principles. Also, the actual usage of architecture principles and their governance is not explicit in the ADM.

B. Conclusions

In this paper we have discussed an approach for the formulation and use of architecture principles. The presented approach extend earlier work [13], [14], in which we defined the concept of architecture principles itself. The presented process consists of eight subprocesses starting with the determination of drivers, and ending with the handling of changes.

The process that is presented should help organizations in better understanding the activities and steps involved in architecture principle development and application. Having this approach in place, enables us to conduct real world experiments that will validate both the original framework and the associated processes. Work is now underway to indeed evaluate the process in the context of architecture engagements of ArchiXL.

References

[1] M. Lankhorst *et al.*, *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, 2005.

[2] M. Op 't Land, H. Proper, M. Waage, J. Cloo, and C. Steghuis, *Enterprise Architecture – Creating Value by Informed Governance*, ser. Enterprise Engineering Series. Springer, Berlin, Germany, 2008.

[3] *The Open Group – TOGAF Version 9*. Van Haren Publishing, Zaltbommel, The Netherlands, 2009.

[4] T. Davenport, M. Hammer, and T. Metsisto, "How executives can shape their company's information systems," *Harvard Business Review*, vol. 67, no. 2, pp. 130–134, March 1989.

[5] G. Richardson, B. Jackson, and G. Dickson, "A Principles-Based Enterprise Architecture: Lessons from Texaco and Star Enterprise," *MIS Quarterly*, vol. 14, no. 4, pp. 385–403, 1990. [Online]. Available: http://www.jstor.org/stable/249787

[6] R. Wagter, M. Van den Berg, J. Luijpers, and M. Van Steenbergen, *Dynamic Enterprise Architecture: How to Make It Work*. New York, New York: Wiley, 2005.

[7] P. Beijer and T. De Klerk, *IT Architecture: Essential Practice for IT Business Solutions*. Lulu, 2010.

[8] J. Dietz, *Architecture – Building strategy into design*. The Hague, The Netherlands: Netherlands Architecture Forum, Academic Service – SDU, 2008. [Online]. Available: http://www.naf.nl

[9] J. Hoogervorst, *Enterprise Governance and Enterprise Engineering*. Heidelberg, Germany: Springer, 2009.

[10] "PRISM: Dispersion and Interconnection: Approaches to Distributed Systems Architecture, Final Report," CSC Index, Inc. and Hammer & Company, Inc., Cambridge MA., Tech. Rep., 1986.

[11] L. Fehskens, "What the "Architecture" in "Enterprise Architecture" Ought to Mean," in *Open Group Conference Boston*. The Open Group, July 2010.

[12] C. Fischer, R. Winter, and S. Aier, "What is an Enterprise Architecture Design Principle? Towards a consolidated definition," in *Proceedings of the 2nd International Workshop on Enterprise Architecture Challenges and Responses*, Yonezawa, Japan, 2010.

[13] H. Proper and D. Greefhorst, "The Role of Principles in Enterprise Architecture," in *Proceedings of the 5th Workshop on Trends in Enterprise Architecture Research, TEAR 2010, Delft, The Netherlands*, ser. Lecture Notes in Business Information Processing, H. Proper, M. Lankhorst, M. Schönherr, J. Barjis, and S. Overbeek, Eds., vol. 70. Springer, Berlin, Germany, November 2010, pp. 57–70. [Online]. Available: http://www.springer.com/business+%26+management/business+information+systems/book/978-3-642-16818-5

[14] ——, "Principles in an Enterprise Architecture Context," *Journal of Enterprise Architecture*, no. February, pp. 8–16, February 1011.

[15] A. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, pp. 75–106, 2004.

[16] "Recommended Practice for Architectural Description of Software Intensive Systems," The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, Tech. Rep. IEEE P1471:2000, ISO/IEC 42010:2007, September 2000.

[17] D. Greefhorst and H. Proper, *Architecture Principles – The Cornerstones of Enterprise Architecture*, ser. Enterprise Engineering Series. Springer, Berlin, Germany, 2011. [Online]. Available: http://www.springer.com/business+%26+management/business+information+systems/book/978-3-642-20278-0

[18] "The Engineers' Council for Professional Development," *Science*, vol. 94, no. 2446, p. 456, November 1941.

[19] S. Beer, *Diagnosing the System for Organizations*. New York, New York: Wiley, 1985.

[20] L. Fehskens, "Re-Thinking architecture," in *20th Enterprise Architecture Practitioners Conference*. The Open Group, October 2008.

[21] E. Yu and J. Mylopoulos, "Understanding 'why' in software process modelling, analysis, and design," in *Proceedings of the 16th international conference on Software engineering, Sorrento, Italy, Los Alamitos, California*. Los Alamitos, California: IEEE, 1994, pp. 159–168.

[22] A. Van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *Proc. RE'01: 5th Intl. Symp. Req. Eng.*, 2001.

[23] "Business Motivation Model (BMM) Specification," Object Management Group, Needham, Massachusetts, Tech. Rep. dtc/06–08–03, August 2006.

[24] A. Osterwalder and Y. Pigneur, *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Amsterdam, The Netherlands: Self Published, 2009.

[25] *Software engineering – Product quality – Part 1: Quality model*, 2001, iSO/IEC 9126–1:2001.

[26] Software & Systems Engineering Standards Committee, "IEEE Std 1061-1998 - IEEE Standard for a Software Quality Metrics Methodology," IEEE Computer Society, Tech. Rep., 1998.

[27] R. Rivera, "Am I Doing Architecture or Design Work?" *It Professional*, vol. 9, no. 6, pp. 46–48, 2007.

[28] M. Op 't Land and H. Proper, "Impact of Principles on Enterprise Engineering," in *Proceedings of the 15th European Conference on Information Systems*, H. Österle, J. Schelp, and R. Winter, Eds. University of St. Gallen, St. Gallen, Switzerland, June 2007, pp. 1965–1976.

[29] D. Greefhorst, *Ervaringen met het opstellen van architectuurprincipes bij een verzekeraar (Experiences with the formulation of architecture principles at an insurance company)*, ser. ICT bibliotheek. The Hague, The Netherlands: Academic Service – SDU, 2007, no. 35, ch. 2, pp. 53–62, in Dutch.

[30] D. Greefhorst, H. Proper, and F. Van den Ham, "Principes: de hoeksteen voor architectuur – Verslag van een workshop op het Landelijk Architectuur Congres 2007 (Principles: The Cornerstone of Architecture – A report of a workshop held at the Dutch National Architecture Congres 2007," *Via Nova Architectura*, 2007, in Dutch. [Online]. Available: http://www.via-nova-architectura.org

[31] M. Van den Tillaart, "Propositions into a Framework," Master's thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2009.

[32] J. Kersten, "Propositions," Master's thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2009, in Dutch.

[33] K. Van Boekel, "Architectuurprincipes: Functie en Formulering (Architecture Principles: Function and Formulation," Master's thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2009, in Dutch.

[34] P. Sabatier, Ed., *Theories of the Policy Process*. Boulder, Co.: West view Press, 1999.

[35] S. Robertson and J. Robertson, *Mastering the Requirements Process*. Reading, Massachusetts: Addison Wesley, 1999.

[36] A. E. i TEAMS, "GEMMA Thema's en Kernprincipes - voor gemeentelijke proces– en informatiearchitectuur," EGEM, Tech. Rep., April 2009, In Dutch.

[37] J. Nabukenya, P. Van Bommel, and H. Proper, "A theory–driven design approach to collaborative policy making processes," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS-42), Los Alamitos, Hawaii*. IEEE Computer Society Press, 2009.