Giving Meaning to Enterprise Architectures;
Architecture Principles with ORM and ORC

P. van Bommel, S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper and Th.P. van der Weide

Institute for Computing and Information Sciences, Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
{P.vanBommel, S.Hoppenbrouwers, E.Proper, Th.P.vanderWeide}@cs.ru.nl

Abstract. Rule-driven systems development emphasizes the use of formalized, declarative rules for the mainstay of its models. The basic underlying techniques are decades old, but now their application, that used to concern only operational system and process definition (business logic, data structure) is being extended to much higher-level items such as policies and architecture principles (we focus on the latter here). When using ORM and Object Role Calculus (ORC) for formal modelling of architecture principles, the underlying logical principles of the techniques may lead to better insight into the rational structure of the principles. Thus, apart from successful formalization, the quality of the principles as such can be improved. We provide some examples and discussion based on the analysis of principles taken from the The Open Group's Architecture Framework (TOGAF).

1 Introduction

Model-driven system development is a major direction in information systems development today. Roughly speaking, it advocates the modelling of various aspects of enterprises as a basis for the design of both the detailed, operational organization of the enterprise (mostly process engineering) and the IT to support it. Model-driven IT development can be traditional (engaging human developers), but in an increasing number of cases fully automated creation (generation) of software from models is strived for [2].

Increasingly, organizations make use of enterprise architectures to direct the development of the enterprise as a whole and IT development in particular [8]. These developments are fuelled by requirements such as the Clinger-Cohan Act in the USA[1], which force government bodies to provide an IT architecture based on a set of architecture principles.

One of the key roles of enterprise architecture is to steer the over-all enterprise/system development within a large organization (enterprise). A more specific way of expressing this is to state that "Architecture serves the purpose of

[1] http://www.cio.gov/Documents/it_management_reform_act_Feb_1996.html

constraining design space"[2]. In most (enterprise) architecture approaches, this constraining is done by means of so-called architecture principles [7, 10]. These principles usually take the form of informal statements such as (taken from [10]):

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

According to the TOGAF architecture framework [10], "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 concerns of the key stakeholders within an organization. In this case, a stakeholder may be highly concerned about the organization's ability to flexibly deploy their workforce over different work locations.

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. Therefore, they have limited power as a steering instrument. The call can already be heard for for SMART[3] treatment of architecture principles. Both in view of their formulation and their enforcement, formalizing principles in a rule-like fashion can be expected to bring the SMART objectives closer. What is more, if architecture and development are complex, and demands on quality, performance, and agility are high, formalization of such rules will enable their embedding in a fully rule-based modelling setup. This may include capabilities for simulation of alternative architectures and their impact, quantitative analysis, and formal verification of and reasoning about and with rules (for example, weeding out contradictions and inconsistencies, or deriving new facts). These and other advantages claimed by rule-based approaches (most prominently, the Business Rules Approach or BRA [12]) may thus also become available to system development under architecture.

It has been argued by some architects that architecture principles should never be formalized, since this would lead to them being too restrictive. They should "leave room for interpretation". We would argue, however, that sharp definition and careful, rational composition of rules should not be mistaken for overly detailed regulation. Even the sharpest formalization of a high-level principle merely sets constraints; if the principle is general enough, ample room is left for more details, at lower levels of design, within those constraints.

In this case paper we do not discuss any further the question whether formalization of architecture principles in a rule-driven development setup is a good idea or not. Instead, we assume that it is at the least an idea worthwhile exploring. What we focus on is the idea that formalization, when properly and systematically performed, may also lead to better analysis of certain patterns of meaning underlying the principles, and thereby to improvement of the (formulation of) the principles as such –even of their informal formulations.

[2] See: `http://www.xaf.nl`

[3] Specific, Measurable, Achievable, Relevant, Time-bound; a common mnemonic used in project management.

We base our account on the ORM and ORC (Object Role Calculus) approach because of its formal foundations, its close relation to the BRA, and its long running affiliation with cooperative domain modelling involving varied, often non-technical domain experts. The *Object-Role Calculus* [6] (ORC) is an evolved variant of RIDL [9]. Two earlier variants where Lisa-D [5], which provided a multi-sets based formalization of RIDL, and ConQuer [11, 1], which provides a more practical approach (that is, from an implementation point of view). The ORC aims to re-integrate the Lisa-D and ConQuer branches of RIDL.

2 Architecture Principles, Rules, and Formalization

In their Architecture Framework (TOGAF), the Open Group [10] lists five criteria that distinguish a good set of principles:

1. **Understandable:** The underlying tenets can be quickly grasped and understood by individuals throughout the organization. The intention of the principle is clear and unambiguous, so that violations, whether intentional or not, are minimized.
2. **Robust:** Enable good quality decisions about architectures and plans to be made, and enforceable policies and standards to be created. Each principle should be sufficiently definitive and precise to support consistent decision making in complex, potentially controversial, situations.
3. **Complete:** Every potentially important principle governing the management of information and technology for the organization is defined. The principles cover every situation perceived.
4. **Consistent:** Strict adherence to one principle may require a loose interpretation of another principle. The set of principles must be expressed in a way that allows a balance of interpretations. Principles should not be contradictory to the point where adhering to one principle would violate the spirit of another. Every word in a principle statement should be carefully chosen to allow consistent yet flexible interpretation.
5. **Stable:** Principles should be enduring, yet able to accommodate changes. An amendment process should be established for adding, removing, or altering principles after they are ratified initially.

We will use the following two example principles, also taken (rather arbitrarily) from TOGAF, throughout the remainder of this paper:

Data is Shared (TOGAF1): "Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations."

Common Use Applications (TOGAF2): "Development of applications used across the enterprise is preferred over the development of duplicate applications which are only provided to a particular organization."

Now we suggest the reader briefly compare the criteria for good architecture principles with the following articles selected from the Business Rules Manifesto[12], describing the nature of business rules:

3.2 Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.

3.3 Rules must be explicit. No rule is ever assumed about any concept or fact.

4.1 Rules should be expressed declaratively in natural-language sentences for the business audience.

5.1 Business rules should be expressed in such a way that they can be validated for correctness by business people.

5.2 Business rules should be expressed in such a way that they can be verified against each other for consistency.

5.3 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.

7.1 Rules define the boundary between acceptable and unacceptable business activity.

8.4 More rules is not better. Usually fewer good rules is better.

8.5 An effective system can be based on a small number of rules. Additional, more discriminating rules can be subsequently added, so that over time the system becomes smarter.

We have no space here to discuss in detail the apparent match of the TOGAF "good principles" characterizations and the BRA in view of our exploration of the formalization of architecture principles, but trust the reader can observe for herself at least a clear similarity of the rationales behind principles and rules.

There is one striking difference between TOGAF principles and BRA rules. The Business Rule Approach aims to help create agile information systems. Rules should be easily changeable, and preferably automatically lead to system adaptations; this should greatly improve agility in business-IT alignment. This sharply contrasts the explicitly phrased Stability characteristic of Principles. However, obviously the possible agility that results of the BRA does not imply that rules have to change often; in fact, many business rules are extremely static.

Note that modality of rules plays an important part in dealing with rule formalization [4]. This is expected to hold also for architecture principles. However, we do not go into modality issues here.

There is one article in the Business Rule Manifesto that deserves some further discussion in view of our formalization goal:

3.1 Rules build on facts, and facts build on concepts as expressed by terms.

This statement, sometimes referred to as the "Business Rule Analysis Mantra", explicitly points towards the approach to formalization shown in the remainder of the paper. Starting from the middle level of analysis, ORM is explicitly designed to deal with the formalization of facts. It is no coincidence that in the BR Manifesto, facts are mentioned so explicitly: ORM is an important means of analysis and representation used in the business rules community [3]. The third level of analysis, term level, is not discussed in detail here but can be seen as the "ontological level" at which intensional and mostly lexical meaning is added to the predicate structures represented in ORM. A typical language/framework used for modelling this level in the BR community is SBVR [13]. The first level of analysis boils down to adding constraints to the basic ORM role/predicate structures. If constraints are not too complex, they can be expressed as ORM graphical constraints (internal or external); a way of expressing (more) complex constraints verbally is by using Object Role Calculus or ORC (see next section).

3 Stating Principles: ORM and ORC

In this section we scrutinize two sample architecture principles taken from [10]. Each time, the goal is to interpret the sample architecture principle as an ORM/ORC expression.

As mentioned before, the Object-Role Calculus (ORC) aims to re-integrate the Lisa-D and ConQuer branches of RIDL. It therefore also has a configurable definition of its semantics in the sense that a distinction is made between four abstraction layers, and that at each layer specific choices can be made with regards to the semantic/syntactic richness of the language. The bottom level is a *counting layer* concerned with an algebra defining how the the occurrence frequency of results of ORC expressions should be combined. The next layer up, the *calculus layer*, defines logical predicates, connectives and an associated inference mechanism. The next layer, the *paths layer*, deals with paths through an ORM schema including connectives enabling the construction of non-linear paths. Finally, the fourth layer is the *presentation layer*. At this level, the path expressions from the paths layer are presented either graphically, or verbalized using a textual language. In this section we only show expressions at the presentation level. We will do this either graphically (corresponding to the traditional graphical constraints), or textually in a naturalized but fully formalized format that is a slightly enriched version of traditional Lisa-D.

We will now stroll through the examples and provide some comments on issues raised during analysis of the principles. We provide an interpretation based on our own knowledge of architecture issues and, admittedly, on guesses. In a real modelling context, such guesses would of course have to be systematically validated by the relevant stakeholders. All graphical (ORM) information we provide is concentrated in Figure 1. Please note that none of the internal uniqueness and total role constraints in the diagram could be directly derived from the principles; they too are interpretations and therefore educated guesses that would have to be validated. The only constraints in the diagram that were more directly derived from the analysis of the principles are the *external* constraints in the 'TOGAF Principle 1' ORM model (the upper half of Figure 1).

Data is Shared (TOGAF1): "Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations."

Concerning TOGAF1: what is an enterprise function? TOGAF does not provide a definition. According to a (presumably related) ArchiMate [8] definition,

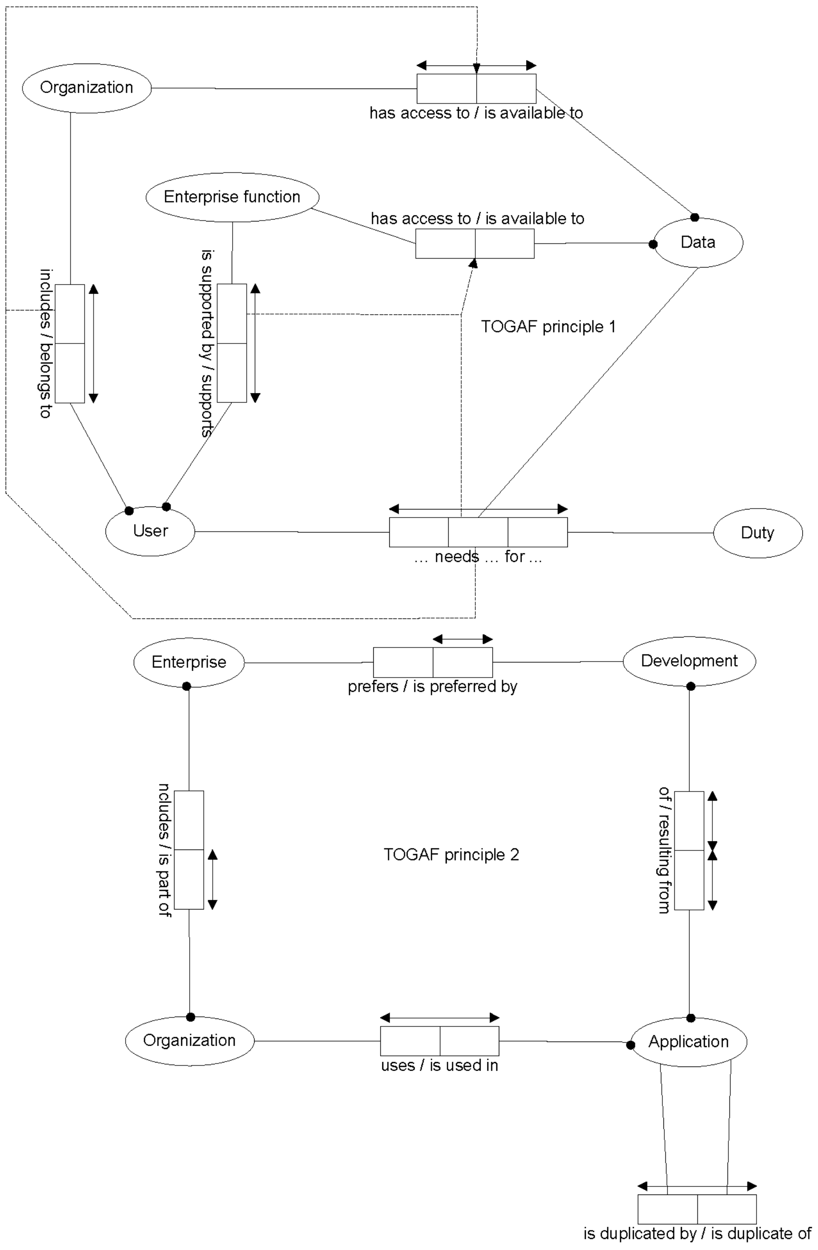

Fig. 1. ORM models underlying the TOGAF principles

a business-function "offers functionality that may be useful for one or more business processes". We presume that in the same vein, enterprise functions are production activities that are part of one or more of the enterprise's operations.

Another issue concerns the meaning of "therefore". It does not seem to be equivalent to a regular logical *imply*, but rather something like "p is enabled by q". We assume that because users need to perform their duties they have access to data, and that users are both part of organizations and support enterprise functions (see Figure 1; the paths in the diagram can be best traced by focusing on the capitalized words in the formulations below; these words correspond to object types in the diagram). We thus have two related rules, both implied by TOGAF1:

1.a Each Enterprise-function has access to Data which some User [that supports that Enterprise-function]
needs for some Duties

1.b Each Organization has access to Data which some User [that belongs to that Organization]
needs for some Duties

In this interpretation, we assume that there are two ways to decompose an enterprise: into functions and into organizations. Either decomposition type now requires its own data access rules. Obviously, this apparent redundancy in the model could be avoided by making explicit that both organizations and enterprise functions are "enterprise parts" and then make one rule for enterprises parts. However, this might mean a considerable infringement of the domain model/language for sake of elegant modelling. The two issues may represent two related but separate concerns that require explicitly separate formulation in the eye of the stakeholders. Therefore, the newly suggested component-rules would have to be validated. For now, we would suggest to maintain the rule as a conjunction of 1.a and 1.b:

1.c Each (Enterprise-function or Organization) has access to Data needed by some User
[that (supports or belongs to) that (Enterprise-function or Organization)] for some Duty

The TOGAF1 example has allowed us to show how a moderately complex architecture principle can be analyzed using ORM/ORC, through a reasonably straightforward interplay between analysis, questions, and propositions. TOGAF2 will show that more complex situations may occur, in which the ability to perform basic formal reasoning can be helpful (we will return to this briefly at the end of this section).

Common Use Applications (TOGAF2): "Development of applications used across the enterprise is preferred over the development of duplicate applications which are only provided to a particular organization."

The TOGAF2 conceptual analysis is related to that of TOGAF1. We again assume that organizations are part of enterprises. We interpret "applications being used across the enterprise" as applications being used in two or more organizations. In addition, we model the notion of "duplication" as a distinct predicate. Lexically, it corresponds to some measure or judgement concerning great similarity in functionality of two applications. Another issue is the interpretation of the term “preferred”. We assume, maybe naively, that a development is either preferred or not. However, in practice it seems possible to provide a rated interpretation, for example by counting the number of duplicates occurring (decreasing preference), or the number of times a single application is used in different organizations being 1 or larger (increasing preference as the count goes up).

In correspondence with Figure 1, we now have:

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

So this is our our formalized interpretation of the original TOGAF2a rule. However, as we were performing the ORM analysis of TOGAF2, it became clear that “duplications” and “use across organizations” relate to essentially different concepts (the first to similarity in functionality between different applications, the second to distributed use of the same application). Consequently, we saw that logically, “Duplication” alone could do the job:

2.a If an Application results from some Development, and that Application is not a Duplicate of
   another Application, then that Development is preferred by the Enterprise.

This boils down to the simple informal rule “no duplicate applications”. Rule 2.a is stronger than rule 2, which it subsumes (i.e. makes it redundant). Its extensional interpretation includes duplicates that occur *within one organization*. To make absolutely sure this is correct (obvious thought it may seem), we should therefore validate rule 2.b:

2.b If an Application A [that is used in some Organization] results from some Development,
   and that Application is not a duplicate of any other Application that is used in the same Organization,
   then that Development is preferred by the Enterprise that includes Application A.

2.a would make 2.b redundant (just as it subsumes 2). However, perhaps this logic-based assumption should not be embraced too rashly. It seems equally reasonable to assume that 2 was put forward (and not 2b) to emphasize the preferred status of “acrossness” in application development. However, even if this were the case, and therefore principle 2 (or rather, the natural language equivalent thereof) was maintained, then still we would like rule 2.a and rule 2.b to be explicitly validated to safeguard the correct interpretation of principle 2.

Note that some assumptions made in the discussion above could be, and to some extent would need to be, formally verified or proven. For example:

– Does (1.a AND 1.b) indeed amount to 1.c?
– Does 2.a indeed logically subsume 2? Also, does 2.a indeed logically subsume 2.b?
– Does 2.b indeed fail to be logically covered by 2?

Formal reasoning in order to answer such questions is possible with ORC, as we have demonstrated in [6] (we cannot include the actual formal exercise here, for reasons of both space and relevance). A sufficiently accessible way of performing such formal reasoning would, in our conviction, be a welcome contribution to a set of tools that aims to support the formalization of principles.

4 Conclusion

In this case paper, we demonstrated and discussed the formal analysis, using ORM and ORC, of architecture principles. We provided two example analyses based on principles taken from the TOGAF architecture framework. First, we discussed why it seems a good idea to, at the least, explore the possibilities for formalizing architecture principles as declarative rules, inspired on the Business Rules Approach. Next, we reported in some detail the experiences and results of analyzing the two principles.

We found not only that such analysis is quite possible, but more importantly that it can lead to better understanding of and even improvement of the principles as such, so apart from their formalization. Using ORM and ORC for principle analysis helps give clear and unambiguous meaning to those principles. In our example case, it led to reconsideration of formulations (both informal and formal) that did not occur when we first read the principles in their original natural language form. However, one has to take care not to discard formulations that make explicit some specific stakeholder concern, even if they lead to redundancy in the model resulting from formal analysis.

We have based the interpretations that inevitably underlie each analysis on assumptions that were educated guesses; in a real analysis process, every assumption should have been validated by relevant stakeholders. However, we have also seen that the approach used makes it very clear which precise assumptions (and formulations thereof) are to be validated. Also, ORC provides us with means, beyond mere intuition, for verifying whether presumed logical relations between propositions in fact hold true. Instead of demonstrating such proofs in detail in this paper (which we have already done elsewhere, for similar cases), we identified them and indicated how they would help back up and consolidate the analysis of principles.

References

1. A.C. Bloesch and T.A. Halpin. ConQuer: A Conceptual Query Language. In B. Thalheim, editor, *Proceedings of the 15th International Conference on Conceptual Modeling (ER'96)*, volume 1157 of *Lecture Notes in Computer Science*, pages 121–133, Cottbus, Germany, EU, October 1996. Springer, Berlin, Germany, EU.
2. D.S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing.* Wiley, New York, New York, USA, 2003.
3. T.A. Halpin. Business Rules and Object Role Modeling. *Database Programming and Design*, 9(10):66–72, October 1996.

4. T.A. Halpin. Business Rule Modality. In T. Latour and M. Petit, editors, *Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'06), held in conjunctiun with the 18th Conference on Advanced Information Systems 2006 (CAiSE 2006)*, pages 383–394, Luxemburg, Luxemburg, EU, June 2006. Namur University Press, Namur, Belgium, EU.
5. A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.
6. S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Fact Calculus: Using ORM and Lisa–D to Reason About Domains. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005*, volume 3762 of *Lecture Notes in Computer Science*, pages 720–729, Agia Napa, Cyprus, EU, October/November 2005. Springer, Berlin, Germany, EU.
7. Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471–2000, The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, USA, September 2000.
8. M.M. Lankhorst and others. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU, 2005.
9. R. Meersman. The RIDL Conceptual Language. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, EU, 1982.
10. The Open Group. *TOGAF – The Open Group Architectural Framework*, 2004.
11. H.A. (Erik) Proper. ConQuer–92 – The revised report on the conceptual query language LISA–D. Technical report, Asymetrix Research Laboratory, University of Queensland, Brisbane, Queensland, Australia, 1994.
12. R.G. Ross, editor. *Business Rules Manifesto*. Business Rules Group, November 2003. Version 2.0.
13. SBVR Team. Semantics of Business Vocabulary and Rules (SBVR). Technical Report dtc/06–03–02, March 2006.