Trakia Journal of Sciences, Vol. 15, Suppl. 1, pp 108-114, 2017

Available online at:
http://www.uni-sz.bg

ISSN 1313-7069 (print)
ISSN 1313-3551 (online) **doi:10.15547/tjs.2017.s.01.020**

INTEGRATION OF "BUSINESS DECISION MODELING" IN COMPANIES FROM BULGARIAN REALITY

P. Slavova*

Department of Industrial Business, University of National and World Economy, Sofia, Bulgaria

ABSTRACT

A problem that many companies in Bulgaria are facing is that the rules, originating from legislation or business policies eventually end up in many different places in the company, with many opportunities for misinterpretation along the way. Traditionally, in most information-intensive companies, their business logics is often hidden in the business processes, manuals or work instructions, or hard-coded in their (succeeded) information systems. This has led to complex and inflexible processes and systems, and as a result, the inability to quickly respond to changing circumstances, regulations and requirements as a part of the Bulgarian reality. In this paper, we explain the role of business decisions and rules in an integrated company design approach, using "The Decision Model" for implementation-independent modeling of the business logic. When modeling business logic, business processes and information as separate, coequal domains, loosely coupled through a limited set of linking elements, the resulting designs become much more flexible and manageable. By first specifying them in an implementation-independent way, it becomes easier to verify whether a design actually meets the requirements of the business. And once there is agreement on the correctness of the design, different implementations can be derived from it. Business agility, i.e., the capability of companies in Bulgaria to rapidly adapt the organization to a changing environment and changing requirements, becomes increasingly important. Although business rules approaches receive a lot of attention as a way to achieve this, their potential is not yet fully exploited. The Decision Model provides an implementation- and technology-independent way to model business logic, which can be loosely coupled with enterprise architecture, business process and information modeling.

Key words: business process management, decision making, innovations in organization, business rules, technical language, information models and tools

INTRODUCTION

Separating the business logics from processes and systems has led to more agility of organizations, and at the same time plays an important role in managing organizational complexity. In this context, it is important to note that business rules are not just specifications that can be executed by a rule engine. Business decisions and rules play a role in all phases of the design and the development chain of business processes and supporting IT applications. At the enterprise architecture level, rule sets and decisions are identified as essential elements of an architecture, which can be related to other elements such as business processes, business objects and application components. At the organization design level, business rules and decisions are specified in more detailed, but still in an implementation-independent way. These implementation-independent models can be translated to specific notations used in the various implementation platforms.

A problem that many companies in Bulgaria are facing is that the rules, originating from legislation or business policies eventually end up in many different places in the company, with many opportunities for misinterpretation along the way. Traditionally, in most information-intensive companies, their business logics is often hidden in the business processes, manuals or work instructions, or hard-coded in their (succeeded) information systems. This has led to complex and inflexible processes and systems, and as a result, the inability to quickly respond to changing circumstances, regulations and requirements as a part of the Bulgarian reality.

***Correspondence to**: *Petia Slavova, Department of Industrial business, University of National and World Economy, Sofia 1700, Bulgaria, "8th December" str., e-mail: p_fin@abv.bg*

In recent years, business rules approaches have received a lot of attention. These approaches promise to offer a solution to the above-mentioned problems, by "separating the know from the flow". Business rules are considered as assets in their own right, and the same are developed and managed separatly from processes and information. However, in practice, this promise is often not fulfilled, due to a number of reasons:

1. There is still a lot of confusion as to what a business rule actually is, and what different types of business rules exist.
2. Business rules are often specified in a very detailed way, in one of the proprietary languages of a rule engine. As a result, they usually have a 'technical' flavor, which makes it difficult for business stakeholders to verify them, and they are tied to a specific implementation platform.
3. Business rule specifications, and the tools that support them, are often poorly integrated within the existing process and information models and tools.

In this paper, the role of business decisions and rules in an integrated company design approach are explained by using The Decision Model (1) for implementation-independent modeling of the business logics. While modeling business logics, business processes and information used as separate, coequal domains, loosely coupled through a limited set of linking elements, leads to the fact that the resulting designs become much more flexible and manageable. By first specifying them in an implementation-independent way, it becomes easier to verify whether a design actually meets the requirements of the business. And once there is an agreement on the correctness of the design, different implementations can be derived from it.

For the implementation-independent design of the business logics, The Decision Model has adopted. It shows that this formalism is applicable at various levels of abstraction, and how it can be combined with enterprise architecture modeling standards such as ArchiMate (2), business process modeling standards and languages such as BPMN (3) and AMBER (4), and information modeling standards such as UML class diagrams. Once an implementation-independent design has been completed, tested and validated with the business stakeholders, it can be mapped on various implementation platforms. Depending on the characteristics of the design and the target platforms, it is often possible to (partly) automate this mapping.

In the remainder of this paper, we first position business rules in the context of related concepts such as principles, requirements, and business policies in Bulgaria. It is also described a classification of the most common types of business rules that exist, and we present an overview of the existing business rule notations and standards. Next, it is described the role of business logic in the enterprise architecture. Finally, the view on the integrated, model-based design of business logics, processes and information, is shared, illustrated with an example.

Positioning and classification of business rules

The trouble with the term 'business rule' is that it can be refered to various kinds of rules that are very different in their nature. Rules may be defined at "design time" or at "runtime" (i.e., operational business rules). Furthermore, at both levels, several types of rules may be distinguished.

Design-time rules can be considered as a specific type of requirements or design constraints. These requirements may be realized by operational business rules, but this is not the only option: they may also be realized by, e.g., "hard-coding" them into business processes or IT applications.

An operational business rule may realize a business policy for a specific situation: a business policy has a general validity within a certain context, while a specific solution is applied to a business rule. This is very similar to the structure at design time, where a requirement or constraint (specific) may realize a design principle (generic). Both design principles and requirements (and, indirectly, also business policies and operational business rules) realize business goals and drivers for change.

Types of business rules

Operational business rules can be classified into a number of categories:

- *Process* rules prescribe the order in which activities are performed, in a declarative way. Variants of process rules include *production* rules (*if ... then ...*) and *event-condition-action* (ECA) rules (*when ..., if ..., then*). It is obvious that process rules as an alternative way to specify process constraints that we typically specify in process models, albeit in a procedural way. Therefore, these types of rules will not be further consider in this article.

An example of a process rule:

When a customer calls for a reservation, makes the reservation, *provided that* the customer is not on the blacklist.

- Different types of dynamic *constraints*, including information or data constraints, timing constraints, constraints on actor or role assignments, etc.

Examples of constraints include:

- The stock of product X should always be at least 100.
- A re-assessment must always be performed by a different person than the original assessment.
- A customer should always receive a response to a question within two working days.

- Business rules to derive a conclusion based on a set of input data ("facts"). These types of rules, which are refered to the term *decision* rules, will be the focus of this particular article. Decision rules represent business logics, and can be considered orthogonal to the process logics. The conclusion may be a result of any type: e.g., a yes/no decision, a selection from a list of options, the result of a calculation, an arbitrary text string, or a list of values.

Examples of decision rules include:

- A customer aged < 20 has a risk profile 'low'.
- The premium for a cancellation insurance is 6% of the total price.

Business rule notations and standards

There are many notations and standards to express business rules, decisions and vocabularies, occupying different places in the development chain. At the computation-independent level, structured (semi-) natural language formalisms such as RuleSpeak are used. At the platform-independent design level, there are a number of approaches for decision modeling, such as decision tables, decision trees, and The Decision Model, and the Production Rule Representation (PRR) (5) to express a class of process rules. The OMG standard Semantics for Business Vocabulary and Business Rules (SBVR) (6) can be positioned at both the computation-independent and platform-independent level. At the platform-specific level, most of the business rule platforms use their own proprietary notations **(Figure 1).**

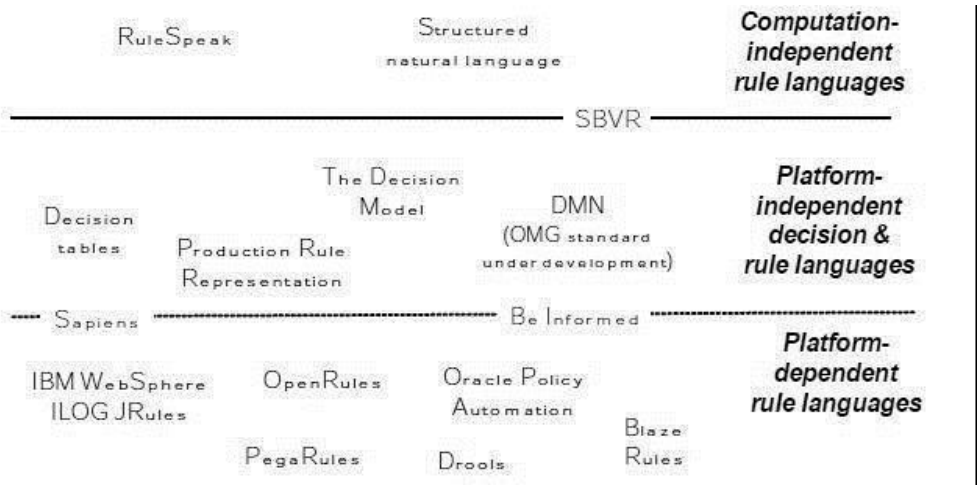

Figure 1. Platform-specific level

Business logics in the enterprise architecture

Enterprise architecture refers to an organization-wide architecture that integrates business, information, application and technology aspects. It specifies the principles and high-level structure that guide the organization design. Enterprise architecture models may coexist and be related to various other types of models and specifications, at different levels of abstraction, which together characterize the organization. These other models can be roughly positioned at the strategy, design or execution level. On the one hand, enterprise architecture is a strategic instrument that supports executive decision-making and guides the design of new solutions in an organization. On the other hand, the enterprise architecture shows the high-level structure of the organization in terms of its components and their interrelationships, and as such ones can be seen as a first step towards organizational design.

Since business decisions and rules are important assets concerning Bulgarian companies that should be managed within an organization and which are related to other assets such as business processes and IT systems, it is important to incorporate them in the enterprise architecture. In the Zachman

framework (7), the requirements of business rules are most naturally positioned within the Motivation ("Why") column.

For an enterprise architecture modeling, the ArchiMate language (8)(9), a standard of The Open Group has been adopted. Design-time business rules, as explained in the previous section, can be modeled as specific types of requirements or constraints in the ArchiMate Motivation extension. However, operational decisions and business rules haven't been yet supported in ArchiMate. These concepts would most naturally fit in the ArchiMate Core language, and have a structure that more or less mirrors the structure in the Motivation domain. A *decision* is a specific type of behavior element, i.e., a process or function that is governed by a set of decision rules. *Directives* are specific types of passive structure elements, and may be subdivided into *business policies* and *business rules* (grouped in *rule sets*). A business policy is a generic operational guideline that is valid within a certain context, while a business rule is a guideline for a specific situation. A business policy can be seen as the operational counterpart of a design principle, while a business rule is the operational counterpart of a requirement. Note, however, that at the enterprise architecture level of granularity, it is probably more appropriate to model business rule sets rather than individual business rules.

The **Figure 2** below illustrates how elements in an enterprise architecture model, expressed in ArchiMate (extended with a business decision and rule set concept), can be linked to more detailed design models in different domains.

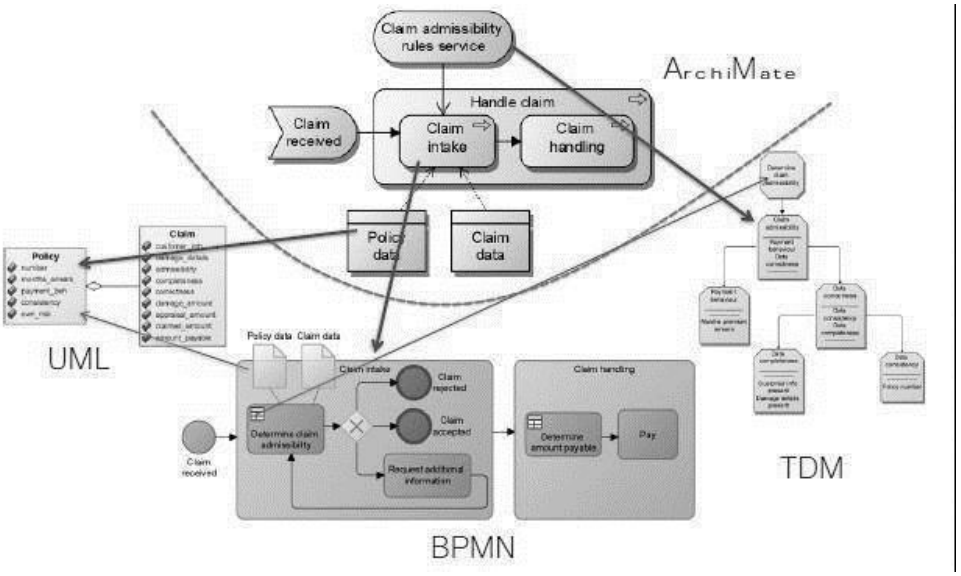

Figure 2. Elements of enterprise architecture model

Integrated design of business logics, processes and information

In this section, the integrated approach, including a conceptual framework, methods and tool support, to the design, validation, analysis and management of business logics (decisions and rules) in relation to processes and information is outlined. These domains are described in an implementation-independent way, but the resulting designs can be translated to various implementation platforms. It is also shown how existing techniques and standards for describing decisions and rules can be reused.

In order to describe the operations of a business, three interrelated domains are distinguished: the *process* domain, the *information* domain and the *logic* domain. (The *organization* or the *actor* domain might be included as a fourth domain, but for simplicity, it is left out of this article.) While the elements in these domains are interrelated, there is no inherent order in which these domains have to be developed. In addition to the model concepts, the elements within each of these domains, there is also a number of concepts that connects the domains: processes may refer to *information items* that represent instances of information entities modelled in the information domain; processes may also refer to *decisions* that are specified in terms of business rules in the logic domain. Finally, business rules make use of *business vocabulary* that links the terms and facts used to specify the rules to the information entities and attributes in the information domain.

The Decision Model

The Decision Model (TDM) , (1) is a platform- and technology-independent intellectual template for perceiving, organizing, and managing the business logic behind a business decision. It is basically a decision table-based approach to modeling decision logic, but with a number of specific characteristics and principles with respect to the structure of the tables and the integrity of the logic. These tables are accompanied by a graphical notation to model the dependencies between the rule families.

Example

Consider the generic claim intake process of ArchiSurance, a favorable insurance company. In order to determine whether a claim will be admitted, a number of criteria has been formulated:

- A damage claim will only be accepted if the customer's premium payment behavior is good: i.e., if the customer has no premium arrears of more than two months.
- If the claim data is incomplete (i.e., claim data is missing or customer data is missing), a claim is accepted provisionally. The claim data must be complemented with the missing elements, and it is checked again on completeness.
- Also, if the claim data is inconsistent, a claim is accepted provisionally. The data must be corrected, and the claim data is checked again on correctness. A particular considerable consistency check is that a policy number should consist of exactly six digits.

Below **(Figure 3),** a "traditional" process model is shown that incorporates this logics. Even for this simple example, the several choices make the model unnecessarily complex. Also, because the logics is "hidden" in the process, it is difficult to change and maintain the rules. For example, if we decide to change the definition of what good payment behavior is (no premium arrears of more than one month, instead of more than two months), it has to be found where this condition occurs in the process model (which may be several times), and change the conditions everywhere.

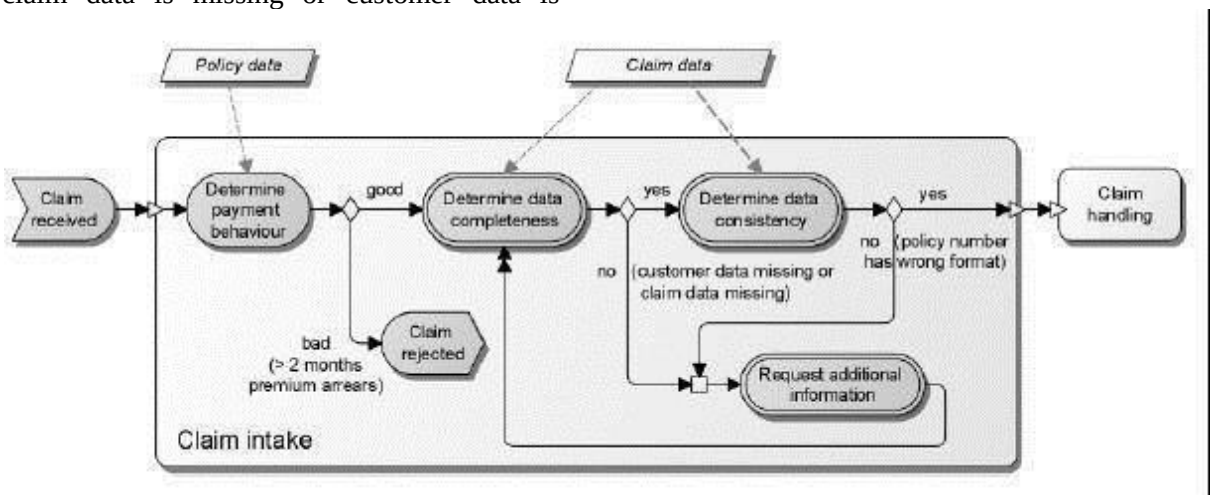

Figure 3. Traditional process model

By separating the business logics from the process flow, the process model can be simplified considerably, concentrating all the business logics into a single decision action ("determine claim admissibility"). Data completeness and data consistency are combined into data correctness: data is correct if and only if it is both complete and consistent.

Messages can be added to specific rules to provide additional information as to why a decision has a certain outcome. This is, among others, useful if the same outcome can be the result of different combinations of conditions; for example, data is incorrect if it is either incomplete or inconsistent (or both).

Implementation

Up to this moment, the business process and business logics specifications have still been purely implementation-independent. The choice whether they will be performed manually, hard-coded in software systems or mapped onto process and rule execution platforms is still open.

As a first step towards implementation, the information is structured, both input data and derived results, in terms of entities and attributes, e.g., using a UML class diagram.

As a proof-of-concept, a mapping from The Decision Model to the open source rule execution environment OpenRules® has been implemented, using the Excel-based input format of OpenRules. The business logics can

then be executed in a stand-alone version of the OpenRules rule engine, within an Eclipse integrated development environment, or embedded in Java software.

The Decision Modeler

The Decision Modeler provides all the functionality needed to model, analyze and test business logics, independent of specific implementation choices or execution environments **(Figure 4).** Because it is a part of an integrated business design modeling tool suite, it is straightforward to relate the business logics to other types of models, such as business processes, information, enterprise architecture, and goal and requirement models.

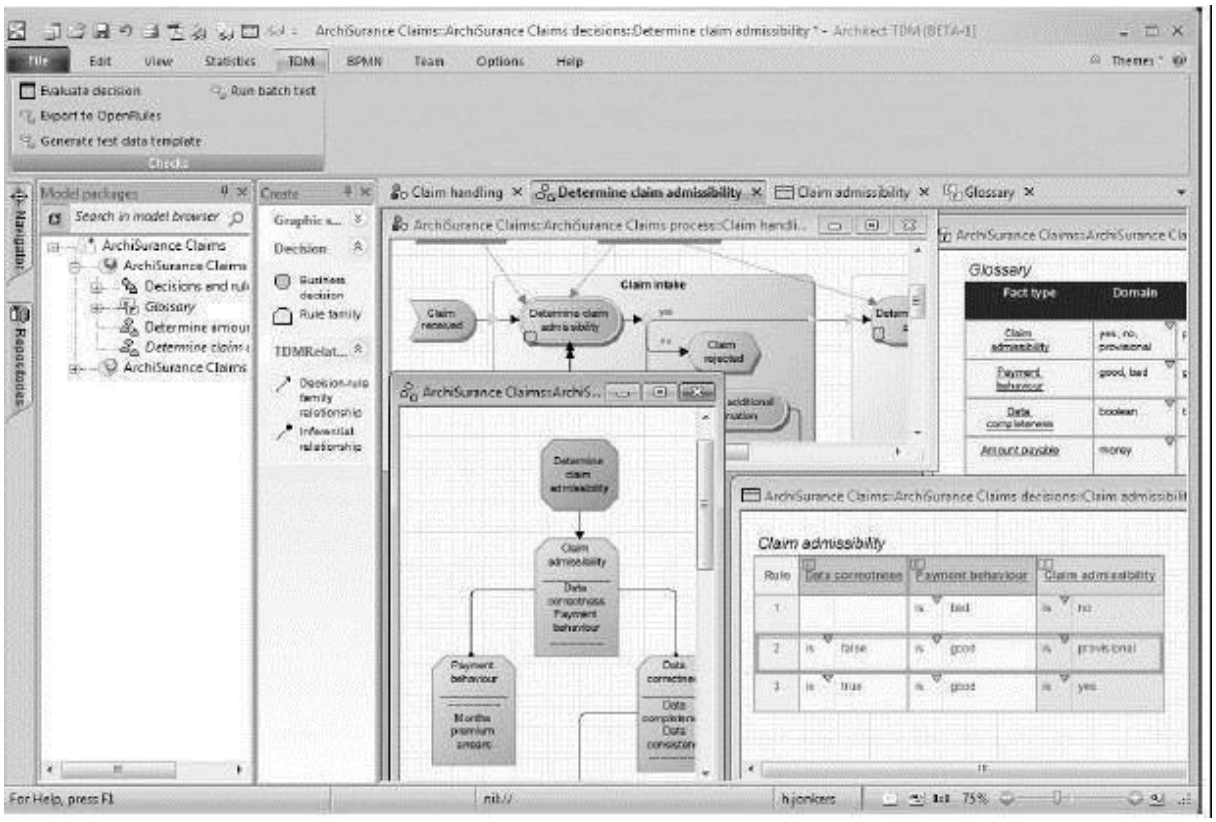

Figure 4. The Decision Modeler view

The main functionality of The Decision Modeler includes:

- A graphical editor to model the structure of a business decision from a Decision Model view
- The rule families re-usage in multiple Decision Model views
- Specification of the decision logics in rule family tables, supporting expressions in conditions and conclusions
- Massage supporting
- Fact types definitions in a glossary, providing a fact - type list
- Several correctness and consistency checks
- Built-in rule engine for (batch) business - logic testing
- Export to the OpenRules execution environment

Furthermore, the generic functionality of the tool suite is available, such as documentation of model elements, models reporting to HTML and RTF, model queries, multi-user support and version control, and the inter-model relationships creation.

CONCLUSIONS

Business agility, i.e., the companies' capability in Bulgaria to rapidly adapting the organization to a changing environment and changing requirements, becomes increasingly important. Although business rules approaches receive a lot of attention as a way to achieve this goal, their potential is not yet fully exploited. This is due to, among others, a lack of common understanding of what business rules are, how they are related to the business processes, and a lack of tools to specify business logics in a comprehensible, implementation-independent way, in relation to other domains.

The Decision Model provides an implementation- and technology-independent way to model business logics, which can be loosely coupled with enterprise architecture, business processes and information modeling. It matches a graphical representation of the

main decision structure with a well-defined and intuitive tabular format in order to define the actual logics. The resulting models can be tested at the implementation-independent level (possibly together with the implementation-independent process specifications). Therefore, it is believed that The Decision Model, used in a combination with the other modeling domains, to have the potential to overcome the above-mentioned problems.

It has been illustrated with an example how this triumvirate of business logics, processes and information may work in practice, in a model-based and a tool-supported way. Once the implementation-independent design has been completed and validated, it can be translated (semi-) automatically to a variety of execution platforms, but also to, e.g., work instructions (for non-automated decisions) or hard-coded application software.

Using an approach as presented in this particular article, the trajectory from legislation and business strategy, through operational processes and systems becomes much more transparant and traceable than in a traditional development approach. This results in higher-quality solutions, because it becomes much easier to verify whether the underlying business requirements have been fulfilled, and in considerable improvement in the business flexibility.

REFERENCES

1. B. von Halle & L. Goldberg, The Decision Model: A Business Logic Framework Linking Business and Technology, CRC Press, 2009.
2. The Open Group, ArchiMate® 2.0 Specification, Van Haren Publishing, 2012. Also available on http://www.archimate.org
3. Object Management Group, Business Process Model and Notation (BPMN), Version 2.0 (formal), 2011.
4. H. Eertink, W. Janssen, P. Oude Luttighuis, W. Teeuw and C.A. Vissers, "A business process design language", World Congress on Formal Methods, 1999, pp. 76-95.
5. Object Management Group. Production Rule Representation (PRR), Version 1.0 (formal), 2009.
6. Object Management Group, Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0 (formal), 2008.
7. Zachman International, The Zachman Framework for Enterprise Architecture 3.0, http://www.zachman.com/about-the-zachman-framework
8. H. Jonkers, D. Quartel, B. van Gils and H. Franken, Enterprise Architecture with TOGAF® 9.1 and ArchiMate® 2.0, White Paper, BiZZdesign, 2012.
9. The Open Group, TOGAF® Version 9.1, Van Haren Publishing, 2011. Also available on http://www.togaf.org