A goal-oriented requirements modelling language for enterprise architecture[1]

Dick Quartel[a], Wilco Engelsman[b], Henk Jonkers[b], Marten van Sinderen[c]
[a]Novay {Dick.Quartel }@novay.nl
[b]BiZZdesign {W.Engelsman, H.Jonkers}@bizzdesign.nl
[c]University of Twente {M.J.vanSinderen}@ewi.utwente.nl

Abstract

Methods for enterprise architecture, such as TOGAF, acknowledge the importance of requirements engineering in the development of enterprise architectures. Modelling support is needed to specify, document, communicate and reason about goals and requirements. Current modelling techniques for enterprise architecture focus on the products, services, processes and applications of an enterprise. In addition, techniques may be provided to describe structured requirements lists and use cases. Little support is available however for modelling the underlying motivation of enterprise architectures in terms of stakeholder concerns and the high-level goals that address these concerns. This paper describes a language that supports the modelling of this motivation. The definition of the language is based on existing work on high-level goal and requirements modelling and is aligned with an existing standard for enterprise modelling: the ArchiMate language. Furthermore, the paper illustrates how enterprise architecture can benefit from analysis techniques in the requirements domain.

1. Introduction

Requirements modelling is an important activity in the process of designing and managing enterprise architectures. As mentioned by Brooks [2]: "No other part of the work so cripples the resulting system if done wrong". Nonetheless, most Enterprise Architecture (EA) modelling techniques focus on *what* the enterprise should do by representing 'as-is' and 'to-be' architectures in terms of informational, behavioural and structural models of the different architectural layers (business, application and technical infrastructure). Little or no attention is paid to represent (explicitly) the reasons, i.e., the *why*, behind the to-be architectures in terms of motivations, rationale, goals and requirements.

Also in popular methods for enterprise architecture, such as The Open Group Architecture Framework (TOGAF) [21], goals and requirements are central drivers for the architecture development process. In TOGAF's Architecture Development Method (ADM), requirements management is a central process that applies to all phases of the ADM cycle. The ability to deal with changing requirements is crucial to the ADM, since architecture by its very nature deals with uncertainty and change, bridging the divide between the aspirations of the stakeholders and what can be delivered as a practical solution.

Requirements modelling helps to understand, structure and analyse the way business requirements are related to IT requirements, and vice versa, thereby facilitating the business-IT alignment. For example, the concept of 'goal' in goal-oriented requirements modelling is used to define some desired effect, i.e., *what* should be achieved. In addition, this goal is related to more abstract (business) goals that define *why* the goal is needed, and is also related to more concrete (IT) goals that define *how* the goal can be realized. The explicit definition of these relations facilitates traceability among the motivations and concerns of stakeholders, their goals and the (design) artefacts that ultimately realize the goals. Typically these artefacts are business and IT services, and the processes and applications that support these services.

The explicit modelling of the motivation underlying enterprise architectures using goals, enables new types of analysis from the requirements domain. For example, one can analyse to what extent the enterprise architecture meets the stakeholders' goals, whether these goals may conflict, the impact of revised goals on the enterprise, and vice versa. Furthermore, alternative architectures may be assessed based on the ability to meet stakeholder goals.

In this paper we assume that ArchiMate [17], [20] is used for EA modelling. Basically, ArchiMate allows one to model the products of the enterprise, the value and services that are offered by these products, and the processes, applications and technology that implement the

[1] This work is partly funded by the Dutch Ministry of Economic Affairs, the Dutch Tax Administration and BiZZdesign in the BServed project (http://bserved.telin.nl).

services. An enterprise architecture is structured along two orthogonal dimensions: layers and aspects. The layer dimension decomposes the enterprise into a business, application and technology layer, and the aspect dimension distinguishes between information, behavioural and structural aspects of the enterprise. This work extends the ArchiMate modelling framework with a fourth aspect: the motivation aspect. This aspect is concerned with the goals and intentions of the enterprise. Requirements modelling is positioned within this aspect.

The purpose of this work is to introduce a language, called ARMOR, for modelling the motivation, i.e., goals and requirements, of enterprise architectures. This language should be aligned with the ArchiMate language. Furthermore, we illustrate the potential use of ARMOR for analysing enterprise architectures, while focusing on business-IT alignment issues.

This paper is structured as follows. Section 2 describes the ArchiMate modelling framework, and its extension towards motivation modelling. Section 3 discusses existing languages for requirements modelling, and presents the 'requirements' on ARMOR. Section 4 presents ARMOR, and its implementation in an existing ArchiMate tool. Section 5 discusses extensions of ARMOR for stakeholder and use case modelling. Section 6 illustrates the use of ARMOR by means of an example, including possible analyses. Section 7 discusses related work. And section 8 presents our conclusions.

2. Enterprise architecture

The modelling of an enterprise architecture involves the conceptualization of different aspects of the enterprise and at different levels of abstraction during the life cycle of the architecture. This section structures the involved concepts into domains and viewpoints.

2.1. ArchiMate modelling framework

Figure 1 depicts the modelling framework that underlies the ArchiMate language [17], [20].

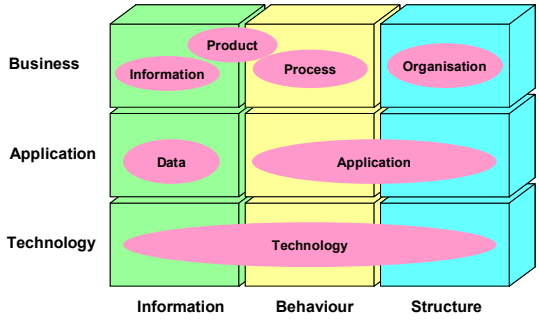

Figure 1. ArchiMate modelling framework

This framework decomposes an enterprise along two dimensions: *layers*, which represent successive abstraction levels at which an enterprise is modelled, and *aspects*, which represent different concerns of the enterprise that need to be modelled. The layer dimension distinguishes three main layers:

- *business* layer, which offers products and services to external customers that are realised in the organisation by business processes;
- *application* layer, which supports the business layer with application services that are realised by (software) application components;
- *technology* layer, which offers infrastructural services (e.g., processing, storage and communication services) that are needed to run applications, and are realized by computer and communication devices and system software.

The aspect dimension distinguishes the following modelling aspects:

- *structure* aspect, which represents the actors (systems, components, people, departments, etc.) involved and how they are related;
- *behaviour* aspect, which represents the behaviour (e.g., processes and services) that is performed by the actors, and the way the actors interact;
- *information* aspect, which represents the problem domain knowledge that is used by and communicated between the actors through their behaviours.

The structuring into dimensions allows one to model an enterprise from different viewpoints, where a *viewpoint* ([11],[10]) is characterized by one's position along each dimension. A viewpoint represents a certain perspective on the enterprise that is of interest to one or more stakeholders. A stakeholder typically focuses on a (small) range along each of the dimensions. The intersection of these ranges spans a viewpoint. For example, each cube in Figure 1 represents the intersection of a single layer and single aspect. A viewpoint may span multiple or only part of a layer or aspect. Furthermore, depending on the choice of viewpoints, they may (and often will) overlap.

Each viewpoint comprises a number of concepts that are used to model an enterprise architecture covering the levels of abstraction and aspects represented by that viewpoint. Accordingly, overlapping viewpoints may comprise overlapping concepts. In order to define, maintain and apply concepts for EA modelling in a structured and consistent way, these concepts are organized in orthogonal, i.e., non-overlapping 'viewpoints', called *domains*. Each domain represents a conceptual model (set of concepts) that covers a particular viewpoint, however, the idea is to choose the domains such that overlap with other domains is minimized. For

example, the ellipses in Figure 1 represent common modelling domains that have been defined for ArchiMate.

The consistency among viewpoints is not addressed in this paper. An approach to address consistency is described in [6].

2.2. Extended framework

ArchiMate focuses on the modelling of *extensional* and *intensional* properties of an enterprise, in terms of informational, behavioural and structural architecture elements. Extensional properties model, e.g., the products and services that are offered, and intensional properties model how they are offered by processes and applications.

To support the modelling of *intentional* properties an extension of the ArchiMate framework is proposed, as depicted in Figure 2. This extension comprises the motivation and meaning aspects, and the value layer.

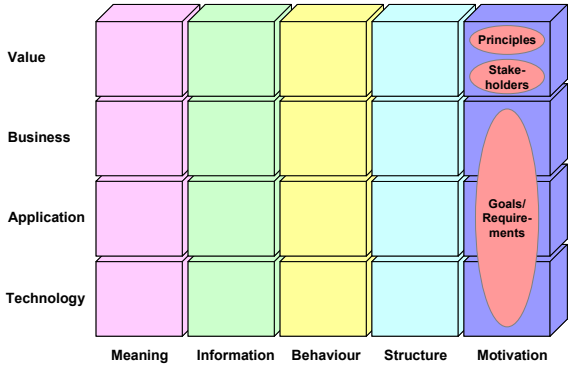

Figure 2. EA modelling framework

The *value* layer represents the value of the services and products that are offered to customers. For example, existing work on value modelling, such as [8], [28] can be positioned in this layer. The 'Value' concept of ArchiMate fits in this layer, and could be extended to model specific types of value, such as cost, and networks of value exchange.

The *meaning* aspect represents concerns that are related to the semantics of enterprise (architecture) artefacts. For example, different ontologies could be used by the enterprise and its customers, or even within the enterprise. The 'Meaning' concept of ArchiMate fits in this aspect, and could be extended to model how an enterprise handles these ontological differences.

The *motivation* aspect is concerned with the goals and intentions of the enterprise. ArchiMate does not provide any concepts for this aspect. This paper is concerned with the motivation aspect. The value layer and meaning aspect are topics of a forthcoming paper. The remainder of this section discusses the domains within the motivation aspect. This aspect resembles the motivation (or why) column of the Zachman framework [26].

Stakeholder domain. This domain models the stakeholders of the enterprise, including their concerns and the assessment of these concerns. A concern is interpreted as some area of attention or interest. For example, a CEO may be concerned with executing the mission of the enterprise, a CIO with the clarity of the enterprise architecture and its ability to adapt to change, and a system's manager with the capacity and reliability of the computing and networking platforms used within the enterprise. These concerns may be assessed using a SWOT analysis. For example, this analysis may reveal that the enterprise's architecture lacks traceability, which makes it difficult to handle change.

In addition, the users or customers of the enterprise may be considered as stakeholders. Customers may be concerned with, e.g., the diversity of the products and services that are offered or the privacy of their information. Also these concerns may be assessed (not necessarily in terms of SWOT – Strengths, Weaknesses, Opportunities and Threats) to reveal customer needs.

Stakeholders and their concerns may be identified from the enterprise's business plan (cf. section 3.1).

Principles domain. This domain models amongst others the vision, mission, strategies, policies, principles and guidelines of the enterprise, constituting the high-level constraints for the design of the enterprise architecture. The current definition of the domain [9] merely identifies the need for defining a vision, mission, strategies, policies, principles and guidelines, but lacks guidance on how this can be done. The Business Motivation Model discussed in section 3.1 could be helpful in this.

Requirements domain. This domain models the goals, requirements and expectations that further constrain the design of the enterprise architecture. These goals, requirements and expectations may originate from the constraints set in the principles domain or from the assessment of concerns in the stakeholders domain. This assessment may reveal strengths, weaknesses, opportunities or threats that need to be addressed by means of changing existing goals or setting new ones.

3. Requirements engineering

Requirements Engineering (RE) is, simply said, concerned with the process of finding a solution for some problem. This concern can be approached from a problem-oriented view, which focuses on understanding the actual problem, and a solution-oriented view, which focuses on the design and selection of solution alternatives [29].

3.1. Problem-oriented RE

Problem-oriented RE originates from systems engineering, emphasizing the modelling and analysis of the problem domain. A requirements model describes the experienced problematic phenomena, the relations between these phenomena, why they are seen as problematic and by whom. A popular problem-oriented RE approach is Goal Oriented RE (GORE) [1]. Goals are considered as high-level objectives of some organization or system. They capture the reasons why a system is needed and guide decisions at various levels within the enterprise. Well-known GORE techniques are i* [31] and KAOS [5].

GORE enables a number of analyses. Firstly, it facilitates reasoning about the purpose of a proposed solution. Goal models can be analyzed to demonstrate which goals realize other goals and which goals conflict or negatively contribute to other goals. Secondly, GORE demonstrates the contribution of the proposed solution to the actual need. This can be combined with traditional techniques like Viewpoint-Oriented RE (VORE) [25][7]. Viewpoints can be used to analyze if all the required views for a solution are satisfied. Furthermore, SWOT analyses can be used to demonstrate the value of a proposed solution to business stakeholders (cf. section 7).

3.2. Solution-oriented RE

Solution-oriented RE represents the more traditional software engineering view on requirements engineering. A requirements model typically describes the context of the system to-be, the desired system functions, their quality attributes, and alternative configurations or refinements of these functions and attributes. These alternatives are analysed and compared to decide which one is the best solution to the problem.

Traditional approaches are structured analysis (SA) [24] and object-oriented analysis (OOA) [12]. Structured analysis focuses on the flow of data and control of the system to-be. Object-oriented analysis applies object-modelling techniques to analyze the functional requirements of the system to-be. An important OOA technique is use-case elicitation and specification. Use cases capture the solution behaviour in terms of interaction scenarios between the system and its user.

Problem- and solution-oriented RE can be considered as two consecutive or complementary phases; also denoted as the early and late requirements phase in [31]. A language for requirements modelling should preferably support both phases and facilitate models to be related for purposes of refinement and analysis.

4. Requirements modelling

Besides its alignment to ArchiMate, the ARMOR language should be based on or aligned with existing languages for requirements modelling. Our intention is not to introduce a new language persé, but one that meets our modelling requirements. These requirements are described first, followed by an overview of the following techniques for goal modelling: the Business Motivation Model [3], the i* framework [31], and the KAOS notation from [22].

4.1. Language requirements

The following list gives an overview of our 'requirements' for a requirements modelling language:

- Re-use of concepts and ideas from existing languages and methods for goal modelling.
- Alignment with ArchiMate.
- Enable *documentation*, *communication* and *reasoning* about requirements.
- *Ease of use*. ARMOR should be easy to learn, understand and apply, especially while its main application is for documentation and communication purposes. Therefore, we aim at a lean and general purpose language that supports a limited set of generic goal modelling concepts.
- *Extensible*. It should be possible to extend ARMOR with specialized concepts and associated analysis techniques. This would allow users to choose between a basic and advanced versions of ARMOR.
- *Traceability*. Adaptation to change is an important requirement for enterprise architectures. In order to support impact of change analysis, abstract goals should be traceable to the more concrete goals and design artefacts such as services and processes that implement these abstract goals; and vice versa.

4.2. Business motivation model

The Business Motivation Model (BMM) provides a structure of concepts for developing, communicating and managing business plans. The concepts can be used to model (i) the factors that motivate a business plan, (ii) the elements that constitute the business plan, and (iii) the relationships between these factors and elements. The BMM has been developed by the Business Rules group and has been adopted as an OMG standard in 2005.

The central notion of the BMM is *motivation*. An enterprise should not only define in its business plan what approach it follows for its business activities, but also *why* it follows this approach and what results it wants to achieve. Figure 3 depicts an overview of the Business

Motivation Model. The following three major parts are distinguished:

- *Ends*, which describe the aspirations of the enterprise, i.e., what the enterprise wants to accomplish;
- *Means*, which describe the action plans of the enterprise to achieve the ends, and the capabilities that can be exploited for this purpose.
- *Influencers*, which describe the assessment of the elements that may influence the operation of the enterprise, and thus influence its ends and means.

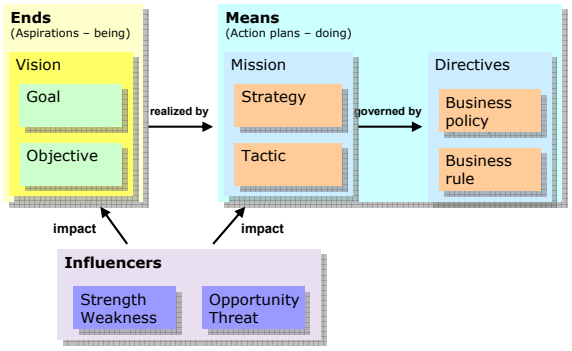

Figure 3. BMM overview

4.3. i*

The i* framework focuses on concepts for modelling and analysis during the early requirements phase. It emphasises the "whys" that underlie system requirements, rather than specifying "what" the system should do.

The i* framework has been developed to model and reason about organizational environments and their information systems. The central notion is the *intentional actor*. Actors within an organization are viewed as having intentional properties such as goals, beliefs, abilities and commitments. Actors depend on each other to achieve goals, to perform tasks and to use resources. Furthermore, actors are strategic and will try to rearrange these dependencies to deal with opportunities and threats.

Two types of models are distinguished: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. A Strategic Dependency model describes the dependencies among actors in an organizational context. A dependency models an agreement between two actors, where one actor (the depender) depends on another (the dependee) to fulfil a *goal*, perform a *task* or deliver a *resource* (the dependum). A dependency may involve a *soft goal*, which represents a vaguely defined goal with no clear criteria for its fulfilment.

The Strategic Rationale model describes stakeholder interests and concerns, and how they can be addressed by various configurations of systems and environments. An SR model adds more detail to the SD model by looking "inside" actors to model internal intentional relationships. Intentional elements, i.e., *goals*, *tasks*, *resources* and *soft goals*, appear both as external dependencies and as internal elements. Intentional elements can be linked by *means-end relations* and *task decompositions*. A third type of link is the *contribution* relation, which represents how well a goal or task contributes to a soft goal.

The i* framework allows various types and levels of analysis [30], for example, to assess the ability, workability, viability and believability of goals and tasks.

4.4. KAOS

KAOS is a methodology for requirements engineering. In comparison to i*, KAOS seems to focus more on the late requirements phase. Having said this, the goal concept in KAOS does allow one to model the motivations, i.e., the why, behind system requirements. But, in contrast to i*, KAOS seems less concerned with modelling the 'intentions' of actors.

The key concept underlying KAOS is goal. [14] defines a *goal* as "a prescriptive statement of intent that the system should satisfy through cooperation of its agents". Here, an agent can be any actor involved in the satisfaction of the goal, e.g., an existing information system, an application to be developed, or a human user.

Goals can be defined at different abstraction levels. Higher level goals and lower level goals are related through refinement relations, which define what lower-level goals are needed to satisfy a higher level goal. At the same time, these refinement relations define the justification for (why) a lower level goal is introduced.

Typically, a (high-level) goal requires the cooperation of multiple systems. One important outcome of requirements engineering is the decision which goal can be automated (partly) and which not. A goal that is assigned to a system-to-be, such that the system is made responsible for the satisfaction of a goal, is called a *requirement*. Instead, a goal that is assigned to the environment of the system-to-be is called an *expectation*. Unlike requirements, expectations can not be enforced by the system-to-be.

In KAOS, a conflict relation can be used to model that the satisfaction of one goal prevents the satisfaction of another goal (and vice versa). An obstacle can be used to represent a situation that hinders or *obstructs* the satisfaction of some goal or requirement. An obstacle may be *resolved* by other goals.

Further, KAOS allows the modelling of properties of the problem domain: domain hypotheses, which describe properties that are expected to hold, and domain invariants, which describe properties that always hold.

KAOS supports various kinds of analysis, such as traceability, completeness, formal validation, refinement checking, and risk, threat and conflict analysis [14].

4.5. Observations

The following observations aim at guiding the decisions about the concepts that should be supported by ARMOR, which are discussed in section 5.

The BMM can not be considered a true requirements modelling language. The model focuses on business plans, which may involve high-level goals and objectives. A business plan that is developed using the BMM can be used as a starting point for (early-phase) requirements engineering. Elements of the BMM, such as goals and strategies, but also strengths, weaknesses, opportunities and threats that result from the analysis of business influencers, may serve as sources or motivations for high-level goals.

The i* framework focuses on the early requirements phase and is an expressive language, allowing various types of analysis. However, the expressiveness of the language and corresponding rich notation, may be experienced as (too) complex and prevent people from using it [30]. Other observations are:

- i* focuses on modelling the intentions of agents (actors) and allows the analysis of these intentions, concerning intentional concepts such as ability, workability, viability and believability;
- the distinction between a means-end relationship and a decomposition relationship in i* in terms of semantics and consequence for further design steps is not always clear and may lead to confusion;
- a similar remark can be made about the distinction between goals and tasks;
- i* distinguishes between the internal intentions of an actor, and its external intentions in terms of dependencies on other actors. This is consistent with the distinction between the internal and external perspective on system design in ArchiMate.

The KAOS graphical notation [22] seems to be less complex and easier to use than i*. This comes at the price of less expressivity, such as the inability to model the extent to which a goal contributes to another goal (although this ability can be introduced). In the tradeoff between expressivity and ease-of-use, this work prefers the latter for the basic version of ARMOR. Other observations are:

- KAOS does not use a separate actor model, but introduces the actors in the goal model via responsibility assignment relations;
- KAOS distinguishes between goals that typically must be satisfied by multiple cooperating agents, and requirements that are assigned to individual agents. This distinction corresponds to the distinction between activities and inter-activities (interactions, collaborations) in ArchiMate.

5. Language definition

In order to align the conceptual model of ARMOR with existing requirements modelling languages, the following approach is followed:

1. Determine the common concepts underlying the languages studied in section 3 and use these concepts as basis for ARMOR. This may involve the abstraction of concepts of one language to relate them to concepts of another language.
2. Extend the basic concepts of ARMOR in case its expressiveness is insufficient.

In these steps, criteria like ease-of-use and suitability of the proposed concepts for the EA domain are considered as well. Furthermore, a 'minimal' set of generic concepts is strived for in order to keep ARMOR broadly applicable and to facilitate modifications and extensions later on when more experience has been gained with the use of ARMOR.

5.1. Supported concepts

The following describes the concepts supported by ARMOR, including their motivation.

Goal concept. The key concept is the concept of goal, which is supported by BMM, i* and KAOS. A goal is defined as some desired effect in the problem domain, or some desired properties of a solution.

Furthermore, the goal concept can be used as an abstraction or generalization of other concepts:

- The concepts of vision and objective in BMM can be modelled as an abstract (high-level) and concrete (low-level) goal, respectively. Also the concepts of mission, strategy and tactic can, from a goal-oriented perspective, be seen as (sub-)goals that are obtained by 'operationalizing' the concepts of vision, goal and objective, respectively.
- The concept of task in i* can be modelled as a concrete goal that defines how (part of) a more abstract goal can be satisfied.
- The goal concept in KAOS is an abstraction of the requirement and expectation concepts, since it abstracts from the agent (actor) to which the goal can be assigned.

An abstract notion of goal reduces the number of required concepts. However, this may be at the expense of precision and intuition. For example, a designer of a business plan does not only think in terms of 'goals', but specializes in terms of strategies, tactics, objectives, etc.

For a similar reason, we want to distinguish between goals that can and can not (yet) be assigned to actors. Therefore, ARMOR supports both the concepts of goal and requirement, where a requirement is defined as a goal that can be assigned to a single system The concept of expectation is not supported explicitly, but can be modelled as a special type of requirement, i.e., one that can be assigned to an environment actor.

The distinction between hard and soft goals is made both in i* and KAOS (and implicitly in BMM via the distinction between goals and objectives). This distinction is considered significant and is therefore also supported in ARMOR. In particular, soft goals are useful in the evaluation of alternative designs.

Goal refinement. BMM, i* and KAOS all support the refinement of goals into sub-goals. Moreover, BMM and i* distinguish two types of refinement relations: means-end relationships and decomposition relations. The need to be able to make this distinction is however not always clear. Furthermore, the distinction is sometimes considered confusing: when should a refinement be considered as a decomposition and when as a means-end relationship?

Therefore, currently only the more abstract refinement relation is supported in ARMOR, with the possibility to specialize this relationship later on if this is felt necessary.

Conflicts, obstacles and qualitative contributions. Both i* and KAOS allow one to model that some goal or situation may have a negative influence on the satisfaction of another goal.

- KAOS supports the conflict relation and the obstruct relation in combination with the obstacle concept. Furthermore, the resolution relation can be used in KAOS to resolve, i.e., 'dissatisfy', an obstacle;
- i* supports the contribution relation to model positive and negative influences on the satisfaction of soft goals. These influences are defined in qualitative terms, e.g., using the range: ++, +, +/-, -, --.

The obstruction of goals by obstacles are not modelled as part of a goal model in ARMOR. An obstacle is considered the result of the assessment of some stakeholder concern, like the assessment of an influencer as a threat or weakness in the BMM. The modelling of assessments should however be supported by ARMOR – not as part of the goal domain – but as part of the stakeholders domain; see also section 6.1.

The following properties can be modelled as part of goal models in ARMOR:

- Positive and negative contributions (influences) on the satisfaction of hard and soft goals, in order to facilitate the evaluation of alternative goal refinements. The need to be able to qualify the strength of the contribution, and in what detail may depend on the situation at hand. Therefore, it should be easy to introduce different qualification ranges, such as the range 0..10 or the range ++, +, +/-, -, -- mentioned above.
- A conflict between two goals G1 and G2, such that the satisfaction of G1 inhibits the satisfaction of G2, and vice versa. A conflict is only possible between hard goals (and requirements), since the criteria for the satisfaction of soft goals is unclear; i.e., it is unclear when the satisfaction of a soft goal inhibits the satisfaction of another goal.

Assumptions. The refinement of some goal may be based on certain assumptions about (elements in) the problem domain. i* and KAOS introduce the notions of assumption, belief and domain property for this purpose. Since it is considered useful to make such assumptions explicit, ARMOR supports the general notion of 'assumption'.

5.2. Meta-model

Figure 4 depicts the abstract syntax, or meta-model, of ARMOR. Most ARMOR concepts from section 5.1 are represented one-to-one by an abstract language element (i.e., UML class). Instead, assumptions are represented by an attribute of the goal concept.

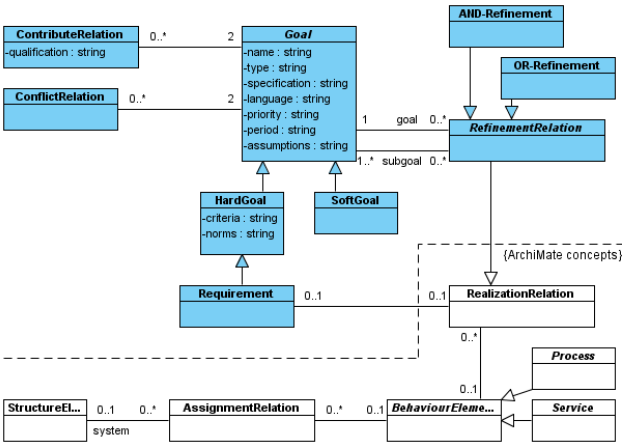

Figure 4. ARMOR meta-model

The idea is to use ARMOR in combination with ArchiMate. Therefore, the actor and assignment relation concepts are 'borrowed' from ArchiMate. The realization relation of ArchiMate is used to represent refinement and to link a requirement to design artefacts, such as the services and processes that implement the requirement. These artefacts are also modelled using the ArchiMate language.

5.3. Concrete syntax

The ARMOR language has been implemented in the BiZZdesign Architect tool as an extension of ArchiMate. Table 1 depicts the concrete syntax used for ARMOR, including part of the concrete syntax of ArchiMate.

Table 1. ARMOR concrete syntax

Abstract	Concrete	Abstract	Concrete
Hard goal	Hard goal	Soft goal	Soft goal
Requirement	Requirement	Refinement/ Realization	
And-refinem.		Or-refinem.	
Contribute	+/-	Conflict	
Business service	Business service	Business process	Business process
Used by		Association	

6. Stakeholders and use cases

Modular extensions of ARMOR have been developed to support the modelling of the Stakeholders domain, as well as the modelling of use cases and business rules. In this paper, we discuss the modelling of stakeholders and use cases briefly.

6.1. Stakeholders domain

Figure 5 depicts the meta-model of the stakeholders domain (see section 2.2) and its relation to the goal concept. The relationships between stakeholders, their concerns, and the assessments of these concerns are mapped onto the association relation of ArchiMate.

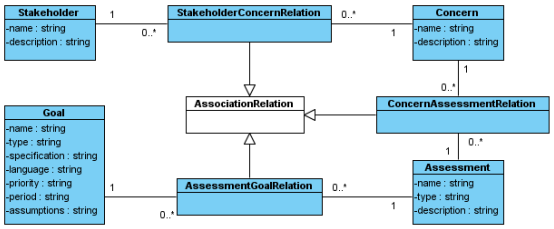

Figure 5. Stakeholders – meta-model

6.2. Use case domain

The modelling of use cases is strongly related to the modelling of goals and requirements. Use cases are generally used as a technique to elicit and specify system requirements. A use case describes the interactions between a system and some external actor, i.e., user [12]. This user typically initiates the use case having some goal in mind. This goal is satisfied when the use case completes successfully.

Multiple, alternative sequences of interactions (called scenarios) may satisfy the goal. In addition, a use case may describe alternative sequences of interactions that handle failure, e.g., exception or error handling. By specifying only interactions, the system is considered as a 'black box', abstracting from internal detail.

Figure 6 depicts the meta-model of the use-case domain. Since a use case is defined as a type of requirement, the use-case domain is a sub-domain of the requirements domain.

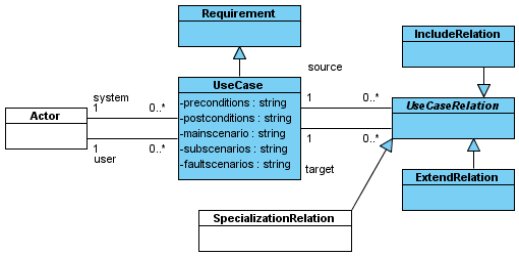

Figure 6. Use cases – meta-model

6.3. Concrete syntax

Table 2 depicts the concrete syntax for the stakeholders and use-case domains.

Table 2. Extension ARMOR concrete syntax

Abstract	Concrete	Abstract	Concrete
Stakeholder	Stakeholder	Concern	Concern
Assessment	Assessment	Use case	Use case
Include	<<include>>	Extend	<<extend>>

7. Application of ARMOR

The introduction of ARMOR enhances enterprise architecture modelling with reasoning and analysis techniques from the domain of requirements engineering. This section illustrates the application of ARMOR and discusses possible analyses. Due to space limitations, we consider the following example issues in enterprise architecture: (i) traceability of stakeholder concerns, (ii) evaluation of alternative architectures, and (iii) detection of conflicting interests and solutions.

Traceability of stakeholder concerns. Figure 7 depicts an ARMOR model that represents two stakeholders of some insurance company, called PRO-FIT, including their concerns, and some assessments of these concerns.

For example, both senior management and the service & IT department are concerned with customer satisfaction. A periodical assessment of these concerns has revealed several threats and weaknesses: leaving customers, lack of insight in insurance portfolios, and insufficient support for portfolio management. The latter is addressed by defining the goal to improve portfolio management, which is refined into the sub-goal to enable on-line portfolio management. This sub-goal supports another higher-level goal, i.e., the use of on-line services to expose PRO-FIT's insurance products. This goal addresses the innovation concern. One of the assessments of this concern presented the SOA paradigm as an opportunity to achieve innovation and to improve the automated handling of insurance processes.

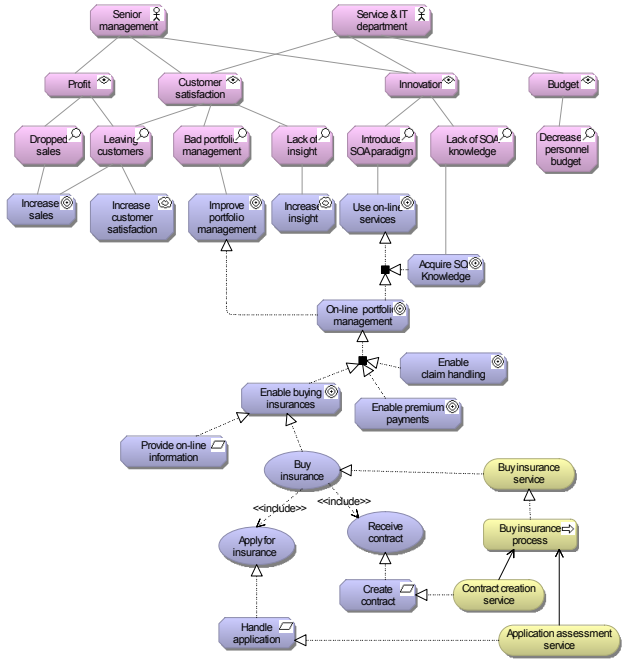

Figure 7. Traceability of stakeholder concerns

In addition, the ARMOR model illustrates the further refinement of goals into sub-goals, use-cases and system requirements, which are realized by means of business services and processes. This enables the forward tracing from stakeholder concerns to the services/processes (and possibly the supporting applications and technology) that 'solve' these concerns, and the backward tracing from services/processes to the goals and concerns that motivated this solution.

Viewpoint Oriented RE (VORE) [25] may help to analyze which goals originate from (the view of) a specific stakeholder and how the proposed solution addresses the concerns of the stakeholder. A complete analysis requires the identification of all major concerns. [13] provides a set of pre-determined viewpoint categories which can be used as a starting point for viewpoint identification and analysis. Other techniques that can be used to assess how well stakeholder goals can be satisfied are workability, viability and ability analyses [31].

Evaluation of alternative architectures. The model in Figure 8 depicts two alternative ways to realize the 'Improve portfolio management' goal, i.e., by providing on-line services or by offering customers a personal assistant. In addition, the (expected) contribution of these alternatives to the soft goals 'Increase customer satisfaction' and 'Increase insight' has been modelled. Based on the contribution to these soft goals, the best alternative seems to be the first one. However, when also taking into account that the personnel budget must decrease, and thus efficiency should improve, a different decision may be made since the first alternative contributes negatively to this. Using soft goals for architecture evaluation is originally done by the NFR framework [4]. Goal analysis techniques [4], [14] can be used to evaluate architectural alternatives.

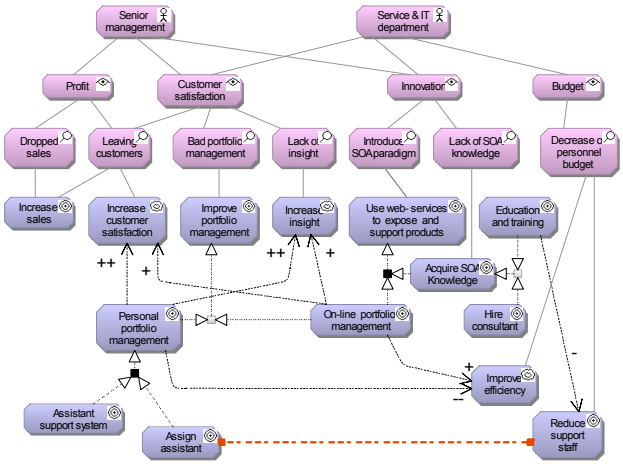

Figure 8. Evaluation of alternative architectures

Detection of conflicting interests and solutions. The model in Figure 8 shows a conflict between the goal to assign personal assistants and the goal to reduce support staff. A conflict is considered stronger than a negative contribution, i.e., assigning assistants is not possible without reducing staff (and vice versa), whereas the improvement of efficiency might still be achieved in other ways. The early detection of conflicts avoids implementation costs of goals that ultimately can't be realized. This also holds for the detection of goals that contribute negatively to other goals, with the distinction that the relative impact of conflicting goals is expected to be higher. [16] reviews generic types of inconsistency that can arise during requirements elaboration and integrated them into a generic framework. Viewpoints are

a recommended mechanism for conflict identification and resolution [16][19].

8. Related work

As mentioned in the introduction, requirements management plays a central role in TOGAF's Architecture Development Method [21]. TOGAF provides a limited set of guidelines for the elicitation, documentation and management of requirements, primarily by referring to external sources. TOGAF's content metamodel, part of the content framework, defines a number of concepts related to requirements and business motivation; however, this part has been worked out in little detail compared to other parts of the content metamodel, and the relation with other domains is weak. Also, the content framework does not propose a notation for the concepts.

The Integrated Architecture Framework (IAF) is Cap Gemini's architectural framework [18]. Like TOGAF this framework also recognizes the importance of requirements for enterprise architectures. IAF recognizes requirements at both the contextual and conceptual level. At the contextual level they identify "business requirements" that answer the why question and at the conceptual level they provide more detailed requirements. But IAF lacks a detailed description of how to represent either business requirements or the more detailed requirements. It mainly lacks concept definition and a requirements language to represent the requirements.

[23], [30] propose to use i* as a problem investigation technique for architecture design and business modelling. This way the motivation for architectural elements is linked to their implementation. [30] illustrates the potential benefit of using BMM and i* in combination to support intentional modelling and analysis of enterprise architectures. This work does not consider the integration or alignment of these languages with existing enterprise modelling languages. [8] extends intentional modelling with value modelling, by combining the i* framework and the e^3 value methodology.

Concerning tool support for enterprise architecture, many tools claim to support requirements modelling (e.g., System Architect and Powerdesigner). However, this support is often limited to the documentation of requirements as structured lists, or the modelling of use cases. Furthermore, they do not offer graphical modelling techniques, nor the integration with other modelling domains.

A relevant tool in the field of requirements modeling is Enterprise Architect from Sparx [27]. Enterprise Architect is primarily an UML modeling tool focused on software engineering. But it also supports a Zachman Framework extension. For modeling the motivation of the Zachman framework Enterprise Architect relies on goal modeling techniques as well, but at the level of the BMM framework. Therefore it lacks GORE based concepts as used by ARMOR. Secondly the link with the actual architectural models is weaker than ARMOR's. For example, with ARMOR it is possible to explicitly model the realization relation between a business service and its use-case. This use-case is associated with a requirement or refined goal. This way ARMOR realizes traceability from business goals, through requirements to architectural elements.

9. Conclusions and future work

We have presented a language, called ARMOR, for modelling goals and requirements in enterprise architectures. The origin of high-level goals is modelled in terms of stakeholders, their concerns and the (SWOT) assessments that are addressed by the goals. Goals are refined into (alternative sets of) sub-goals, via goal trees. Low-level goals (requirements) are related to the services, processes and applications that implement the requirements. This enables forward and backward traceability of goals and requirements.

The ARMOR language is based on existing requirements modelling languagues and is aligned with the standard enterprise modelling language ArchiMate. This brings existing theory and analysis techniques to the domain of enterprise architecture modelling.

Currently we apply ARMOR combined with an architecture-driven requirements engineering approach in consultancy projects. These projects help to validate and improve ARMOR and the associated approach. Furthermore, our future work aims at the formalization of ARMOR and the elaboration of various analysis techniques, using existing work such as the work referred to in this paper.

References

[1] A.I. Antón. Goal-based requirements analysis. Proceedings of the Second International Conference on Requirements Engineering, pp. 136-144, 1996.

[2] F. Brooks. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20, 4, pp. 10-19, 1986.

[3] Business Rules Group. *The Business Motivation Model – Business Governance in a Volatile World*. Release 1.3, http://www.businessrulesgroup.org.

[4] L. Chung. *Non-Functional Requirements in Software Engineering*. Springer, 1999, ISBN 0-7923-8666-3.

[5] A. Dardenne, A. van Lamsweerde, S. Fickas. Goal-Directed Requirements Acquisition. *Science of Computer Programming*. Vol. 20, pp. 3-50, 1993.

[6] R.M. Dijkman, D.A.C. Quartel, M.J. van Sinderen. Consistency in Multi-Viewpoint Design of Enterprise

Information Systems. *Information and Software Technology* (IST) 50(7-8), pp. 737-752, 2008.

[7] A. Finkelstein, M. Goedicke, J. Kramer, and C. Niskier. Viewpoint oriented software development: Methods and viewpoints in requirements engineering. *Lecture Notes in Computer Science*, Vol. 490, pp. 29-54, 1991.

[8] J. Gordijn, E. Yu, B. van der Raadt. e-Service Design using i^* and e^3 value Modeling. *IEEE Software*, Vol. 23, No. 3, pp. 26-33, 2006.

[9] M.E. Iacob, H. Franken, H. van de Berg. *Enterprise Architecture Handbook – method, language and tools*, BiZZdesign Academy Publishers, 2007.

[10] ISO. ISO/IEC 10746. The ISO Reference Model for Open Distributed Computing.

[11] ISO. ISO/IEC 42010. Systems and software engineering – Recommended practice for architectural description of software-intensive systems.

[12] I. Jacobson. The use-case construct in object-oriented software engineering. *Scenario-based Design: Envisioning Work and Technology in System Development*, pp. 309-338, 1995.

[13] G. Kotonya and I. Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996

[14] A. van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 5–19, New York, NY, USA, 2000. ACM Press.

[15] A. van Lamsweerde. Requirements Engineering: From Craft to Discipline. *Proceedings FSE'2008: 16th ACM Sigsoft International Symposium on the Foundations of Software Engineering*, Atlanta, November 2008

[16] A. van Lamsweerde, R. Darimont and E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering", *IEEE Transactions on Sofware Engineering, Special Issue on Inconsistency Management in Software Development*, Nov. 1998.

[17] M. Lankhorst, et al. *Enterprise Architecture at work: modelling, communication and analysis*. Springer, 2005.

[18] A. Mulholland, and A.L. Macaulay. Architecture and the integrated architecture framework. *Whitepaper,* Capgemini, 2006.

[19] B. Nuseibeh, J. Kramer and A. Finkelstein, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specifications", *IEEE Transactions on Software Engineering*, Vol. 20 No. 10, Oct. 1994, 760-773.

[20] The Open Group. ArchiMate® Version 1. http://www.opengroup.org/archimate.

[21] The Open Group. TOGAF™ Version 9. http://www.opengroup.org/togaf.

[22] Respect-IT. A KAOS Tutorial, V1.0, http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf.

[23] R. Samavi, E. Yu, and T. Topaloglou. Strategic reasoning about business models: a conceptual modeling approach. *Information Systems and E-Business Management*, pp. 1-28, 2008.

[24] K. Schoman, and D.T. Ross. Structured Analysis for requirements definition. *IEEE Transactions on Software Engineering*, 3(1), 1977.

[25] I. Sommerville, P. Sawyer, and S. Viller. Viewpoints for requirements elicitation: a practical approach. *Proc. Third IEEE International Conference on Requirements Engineering (ICRE 98)*, 1998.

[26] J.F. Sowa, J.A. Zachman. Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 31, No. 3, p. 590, 1992.

[27] Sparx Systems. MDG Technology For Zachman Framework User Guide. http://www.sparxsystems.com.au/downloads/pdf/ZFUserGuide.pdf.

[28] H. Weigand, P. Johannesson, et al. Strategic analysis using value modeling – the c3-value approach. Proc. of the 40th Annual Hawaii Int. Conf. on System Sciences (HICSS'07), p. 175c, 2007.

[29] R. Wieringa. Requirements engineering: Problem analysis and solution specification (extended abstract). *Lecture Notes in Computer Science*, 3140:13-16, 2004.

[30] E. Yu, M. Strohmaier, and X. Deng. Exploring Intentional Modeling and Analysis for Enterprise Architecture. *Proceedings of the EDOC 2006 Workshop on Trends in Enterprise Architecture Research* (TEAR 2006). 2006.

[31] E. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*. Jan. 6-8, 1997, Washington D.C., USA, pp. 226-235.