

Extending enterprise architecture modelling with business goals and requirements

Wilco Engelsman^{a,b,*}, Dick Quartel^c, Henk Jonkers^a and Marten van Sinderen^b

^a*BiZZdesign, Enschede, The Netherlands;* ^b*Department of Computer Science, University of Twente, PO Box 217, Enschede, AE 7500, The Netherlands;* ^c*Novay, Enschede, The Netherlands*

(Received 2 March 2010; final version received 5 May 2010)

The methods for enterprise architecture (EA), such as The Open Group Architecture Framework, acknowledge the importance of requirements modelling in the development of EAs. Modelling support is needed to specify, document, communicate and reason about goals and requirements. The current modelling techniques for EA 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 EAs in terms of stakeholder concerns and the high-level goals that address these concerns. This article 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 article illustrates how EA can benefit from analysis techniques from the requirements engineering domain.

Keywords: enterprise architecture; requirements engineering; goal modelling; stakeholder concerns; business–IT alignment

1. Introduction

Requirements modelling is an important activity in the process of designing and managing enterprise architectures (EAs). Brooks (1986) mentions, ‘No other part of the work so cripples the resulting system if done wrong’. This quote refers to the design of software architectures, but applies as well and maybe even more so to the elicitation and analysis of the requirements that should be addressed by the architecture design. Nonetheless, most 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 model elements at different architectural layers, e.g. a business, application and technology layer. Little or no attention is paid to represent (explicitly) the motivations or rationale, i.e. the *why*, behind the architectures in terms of goals and requirements.

In contrast to EA modelling techniques, methods for EA, such as The Open Group Architecture Framework (TOGAF) (The Open Group. TOGAFTM Version 9. <http://www.opengroup.org/togaf>), acknowledge that goals and requirements are

*Corresponding author. Email: w.engelsman@bizzdesign.nl

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 information technology (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 intended effect that is desired by some stakeholder, i.e. *what* should be achieved. This goal may be related to more abstract (business) goals that define *why* the goal is needed, and may also be related to more concrete (IT) goals that define *how* the goal can be realised. The explicit definition of these relations facilitates traceability among the motivations and concerns of stakeholders, their goals and the (design) artefacts that ultimately realise the goals. Goals have to be refined into requirements before their realisation can be assigned to some artefact, such as a business service, business process, application service or application component. When talking about requirements modelling in the sequel, we mean the modelling of goals and requirements. For now, goals can be considered as abstract requirements that need to be made more concrete before they can be realised by elements of the EA.

The explicit modelling of the motivation underlying EAs using goals enables new types of analysis from the requirements engineering (RE) domain. For example, one can analyse to what extent the EA 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 their ability to meet stakeholder goals.

In this article, we assume that ArchiMate (Lankhorst *et al.* 2005, The Open Group. ArchiMate[®] Version 1. <http://www.opengroup.org/archimate>), 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 services. An EA 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 motivations or 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 of EAs in terms of goals and requirements. This language is aligned with the ArchiMate language. Furthermore, we illustrate the use of ARMOR for analysing EAs, while focusing on business-IT alignment issues.

The remainder of this article is structured as follows. Section 2 describes the ArchiMate modelling framework and its extension towards motivation modelling. Section 3 discusses the relation between RE and EA. Section 4 provides an analysis for existing languages for requirements modelling, and derives ideas and 'requirements' for ARMOR. Section 5 presents ARMOR in terms of its concepts and their notations. Section 6 illustrates ARMOR by means of a real-life example. Section 7

provides an overview of the available analysis techniques and tool support for ARMOR. Section 8 discusses related work. And Section 9 presents our conclusions.

2. Enterprise architecture modelling

EA is a design or a description that makes clear the relationships between products, processes, organisation, information services and technological infrastructure; it is based on a vision and on certain assumptions, principles and preferences; consists of models and underlying principles; provides frameworks and guidelines for the design and realisation of products, processes, organisation, information services, and technological infrastructure. It comprises a collection of simplified representations of the organisation, from different viewpoints and according to the needs of different stakeholders (Lankhorst *et al.* 2005).

A coherent description of EA provides insight, enables communication among stakeholders and guides complicated change processes (Jonkers *et al.* 2004). ArchiMate (The Open Group. ArchiMate® Version 1. <http://www.opengroup.org/archimate>), an open standard of The Open Group, provides a language to create such descriptions in a precise and formal way. ArchiMate defines concepts for describing architectures at the business, application, and technology layers, as well as the relationships between these layers. Thus, it addresses the ubiquitous problem of business–ICT alignment. ArchiMate originally results from a public/private research project, a cooperation of companies, universities and research institutes.

2.1. ArchiMate modelling framework

Figure 1 depicts the modelling framework that underlies the ArchiMate language (Lankhorst *et al.* 2005, The Open Group. ArchiMate® Version 1. <http://www.opengroup.org/archimate>). 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:

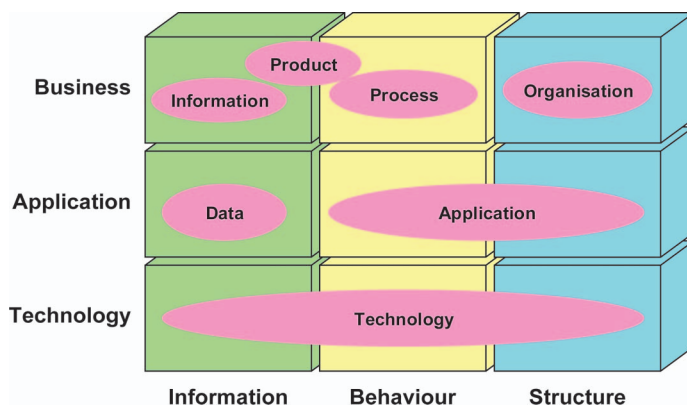


Figure 1. ArchiMate modelling framework.

- *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 realised 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* (ISO 1998, 2007) is characterised 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 EA 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 organised in orthogonal, i.e. non-overlapping 'viewpoints', called *domains*. Each domain represents a set of concepts that is used to model systems from a particular viewpoint. For example, the ellipses in Figure 1 represent common modelling domains that have been defined for ArchiMate.

The consistency among viewpoints (or domains) is not addressed in this article. An approach to address consistency is described by Dijkman *et al.* (2008).

2.2. Extended framework

ArchiMate focuses on the modelling of *extensional* and *intentional* properties of an enterprise, in terms of informational, behavioural and structural architecture elements. Extensional properties model a system from an external perspective, e.g. the products and services that are offered. Intentional properties model the system from an internal perspective, e.g. how the products and services are supported 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 consists of the *motivation aspect*, which resembles the motivation (or why) column of the Zachman framework (Sowa and Zachman 1992).

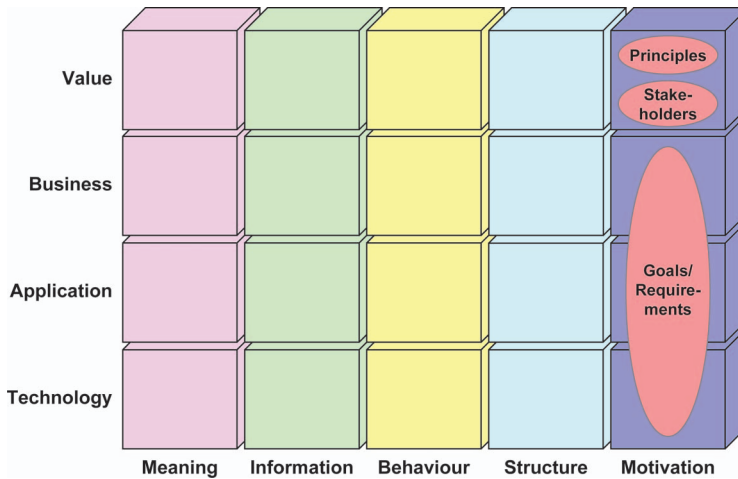


Figure 2. Extended EA modelling framework.

ArchiMate does not provide any concepts for modelling the motivation aspect. The remainder of this section introduces three modelling domains within the motivation aspect: the stakeholder domain, the principles domain and the requirements domain.

2.2.1. 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 EA 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 strengths, weaknesses, opportunities and threats (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) to reveal customer needs.

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

2.2.2. Requirements domain

This domain models the goals, requirements and expectations that constrain the design of the EA. These goals, requirements and expectations typically originate from the assessment of concerns in the stakeholders domain. This assessment may reveal strengths, weaknesses, opportunities or threats that need to be addressed by changing existing goals or setting new ones.

2.2.3. Principles domain

This domain models the principles of the enterprise. Principles are normative restrictions on the EA. The principles use business drivers of the organisation, found in the stakeholder domain and requirements domain. The current version of the domain merely identifies the need for such a domain. Section 8.1 will provide more information about the positioning of principles related to ARMOR.

3. Business requirements and enterprise architecture

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 from a *solution-oriented view*, which focuses on the design and selection of solution alternatives (Wieringa 2004).

3.1. Problem-oriented RE

RE as problem analysis stems from systems engineering and emphasises the investigation and documentation of the problem domain. A requirements model describes the experienced problematic phenomena, the relations between these phenomena, why they are seen as problematic and who experience these problems. A popular problem-oriented RE approach is goal-oriented RE (GORE) (Antón 1996). Goals are considered as high-level objectives of some organisation 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* (Yu 1997) and KAOS (Dardenne *et al.* 1993).

GORE enables a number of analyses. Firstly, it facilitates reasoning about the purpose of a proposed solution. Goal models can be analysed to demonstrate which goals realise 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) (Finkelstein *et al.* 1991, Sommerville *et al.* 1998). Viewpoints are useful to break up the requirements model in smaller, more manageable parts. A requirements model can be considered complete if all the required views combined satisfy the needs of the stakeholders. 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 RE. 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) (Schoman and Ross 1977) and object-oriented analysis (OOA) (Jacobson 1995). SA focuses on the flow of data and control of the system to-be. OOA applies object-modelling techniques to analyse 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.

3.3. Problem chains

When we look at RE from an engineering perspective we can identify the so-called *problem chains*, where each chain links a problem to a solution such that the solution is considered again as a problem by the next chain. For example, a business analyst may investigate a business problem and specify a business solution for this problem. This new solution most likely needs IT support, and therefore becomes a problem for the IT analyst. Figure 3 illustrates this engineering perspective.

Problem chains link RE to EA. This is illustrated in Figure 4. The *why* column represents the problem-oriented view and defines the business needs, goals, requirements and use-cases that should be addressed. The *what* column represents the solution-oriented view in terms of EA elements such as services, processes and applications. These architecture elements define *what* the enterprise must do to address the business needs, goals, requirements and use-cases. At the same time, these RE elements motivate and justify *why* the EA is defined the way it is.

For example, an organisation may experience dropping sales, unsatisfied customers and a high workload for the customer support department. In order to address these problems, the organisation decides to introduce a new business service to support on-line portfolio management. This solution leads to a new design problem: the creation of new or the adaptation of existing business processes that support this service. Similarly, these processes may require IT support, which leads to the development of new application services.

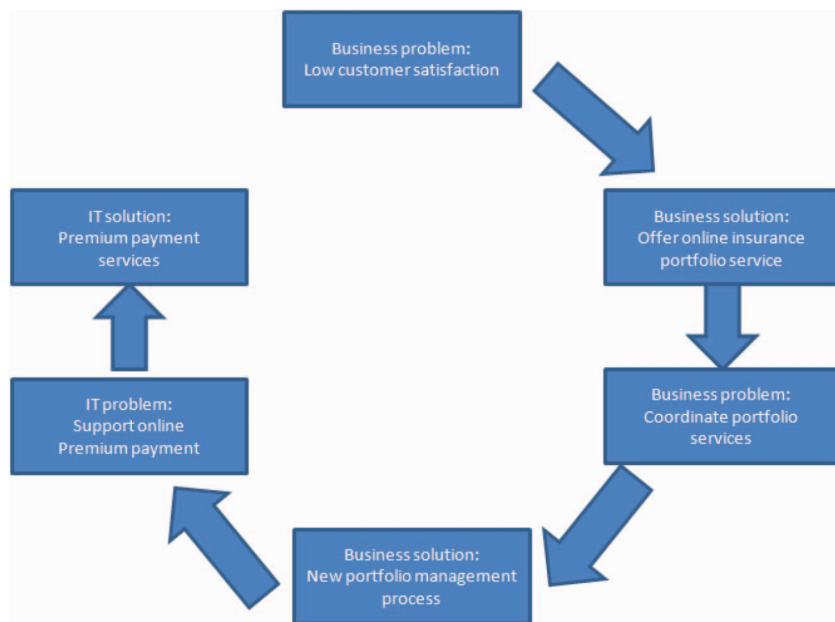


Figure 3. Problem chains.

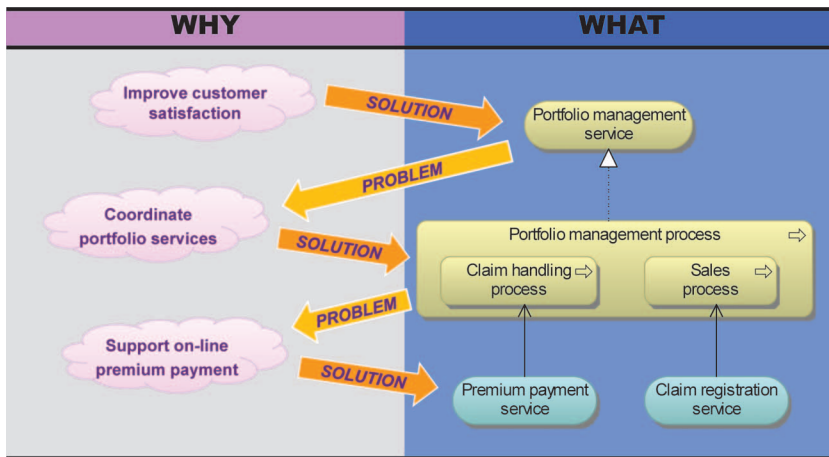


Figure 4. Problem chains relating business requirements to EA.

This way of thinking enables traceability. EA elements can be traced back to the goals and requirements that motivated their introduction. Reversely, RE elements can be traced forward to the services, processes and applications that implement these elements. This traceability is needed to successfully analyse and manage the impact of changes to an enterprise. For example, in case certain business goals are affected by changes in legislation, it becomes now possible to determine precisely which products and services are affected by these changes.

4. Requirements modelling

Besides its alignment to ArchiMate, we want the ARMOR language to align with concepts and ideas from existing languages for requirements modelling, wherever possible. Our intention is not to introduce a new language per se, 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 (Dardenne *et al.* 1993), the i* framework (Yu 1997), and the KAOS notation from Op't Land and Proper (2007).

4.1. Language requirements

The following list gives an overview of our 'requirements' for defining a language that supports the modelling of goals and requirements on EAs:

- (1) Re-use of concepts and ideas from existing languages and methods for goal modelling.
- (2) Alignment with ArchiMate.
- (3) *Traceability*. Adaptation to change is an important requirement for EAs. 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) Facilitate *documentation*, *communication* and *reasoning* about requirements.

- (5) *KISS (keep it small and simple)*. ARMOR should be based on a small set of generic concepts that allows one to model the motivation aspect of EAs in an intuitive way. This should also help in obtaining a language that is easy to learn, understand and apply.
- (6) *Extensible*. It should be possible to extend ARMOR with specialised concepts and associated analysis techniques. This would allow users to choose between a basic and advanced versions of ARMOR.

Requirements 4–6 are considered as ‘soft’ requirements. In practice, this means that they are used as guidelines for making decisions rather than hard criteria with pre-defined norms that can be validated afterwards.

Our goal is to capture the business context leading to EA, in other words to capture the high-level goals that lead to the architectural designs. GORE facilitates capturing the business context (van Lamsweerde 2001) and provides a structured way to refine these goals into requirements that can be assigned to elements from the architecture. GORE itself also provides a number of advantages (van Lamsweerde 2001), like improved traceability, evaluation of alternatives and reasoning about conflicts. We opted for a goal-oriented RE language, based upon these goals and possible advantages. Therefore, the subsequent sections will only focus on goal-oriented RE languages.

4.2. 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 (Business Rules Group 2008) and has been adopted as an OMG standard in 2005 (Object Management Group. Business Motivation Model Version 1. <http://www.omg.org/spec/BMM/1.0/>).

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 5 depicts an overview of the BMM. 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.

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 organisational environments and their information systems. The central notion is the *intentional*

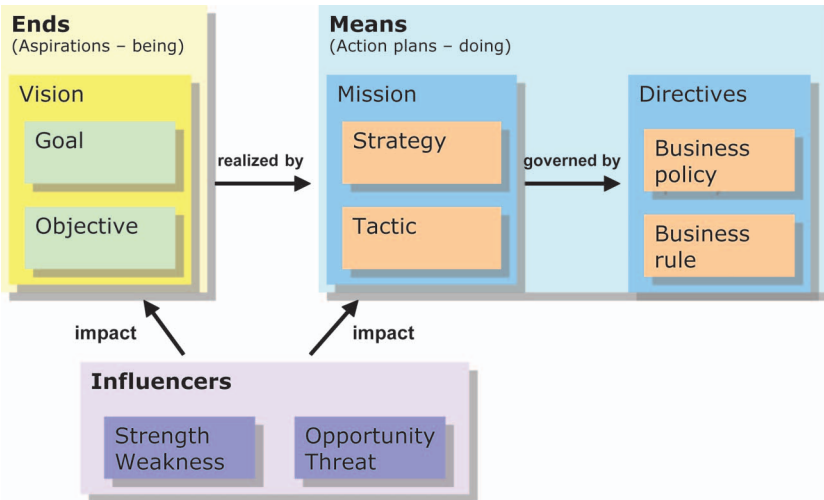


Figure 5. BMM overview.

actor. Actors within an organisation 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 SD model and the strategic rationale (SR) model. An SD model describes the dependencies among actors in an organisational context. A dependency models an agreement between two actors, where one actor (the depender) depends on another (the depensee) 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 SR 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 (Yu *et al.* 2006), for example, to assess the ability, workability, viability and believability of goals and tasks.

4.4. KAOS

KAOS is a methodology for RE (Respect-IT 2007). 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. Van Lamsweerde (2000) 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 RE 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 cannot 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 (van Lamsweerde 2000, van Lamsweerde *et al.* 1998).

4.5. Observations

The following observations aim at guiding the decisions about the concepts that should be supported by ARMOR.

The BMM cannot 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) RE. Elements of the BMM, such as goals and strategies, and also SWOT 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 (Yu *et al.* 2006). 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 (Dardenne *et al.* 1993) 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). 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.

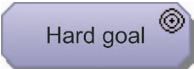

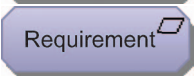
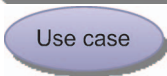
In these steps, guidelines like ‘KISS’ and suitability of the proposed concepts for the EA domain are taken into account. Furthermore, a ‘minimal’ set of generic concepts is strived for in order to keep ARMOR broadly applicable and to facilitate modifications and extensions later on when more experience has been gained with the use of ARMOR.

The definition of ARMOR is divided into two domains: the requirements domain and the stakeholder domain.

5.1. Requirements domain

Table 1 depicts the requirements concepts that are supported by ARMOR, including their notation.

Table 1. ARMOR – requirements domain.

Concept	Notation	Concept	Notation
Hard goal		Soft goal	
Requirement		Use-case	

5.1.1. Goals and requirements

The key concept is the concept of goal, which is supported by BMM, i* and KAOS. Their definitions have in common that a goal represents 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 generalisation 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 ‘operationalising’ 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 specialises in terms of strategies, tactics, objectives, etc. For a similar reason, we want to distinguish between goals that can and cannot (yet) be assigned to actors.

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.

Based on the aforementioned analyses, we define the concepts of goal and requirement as follows:

- *A goal models some end that a stakeholder wants to achieve.* The desired end can be anything, e.g. some effect in or state of the problem domain, a produced value, tasks, or a realised system property. Hard goals are quantifiable goals with norms that specify when a goal is achieved. Soft goals are qualitative, i.e. not quantified, and in general more abstract. Typically, soft goals have to be refined into more concrete goals to make them quantifiable.
- *A requirement models some end that must be realised by a single actor.* A requirement can be considered as a specialisation of a goal that is delimited in scope and functionality, such that it can be assigned to a single actor.
- The modelling of (business) use cases is strongly related to the modelling of goals and requirements. Therefore, ARMOR also supports use-cases, similar to UML use-cases, including the include and extend relation. Use cases are 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 (Jacobson 1995, Object Management Group. Unified Modelling Standard Version 2. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML). 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. Because a use-case is associated with a single system that implements it, a use-case is considered as a type of requirement in ARMOR.

5.1.2. Relations in goal refinement

Table 2 depicts the relations that are supported with ARMOR, including their notations.

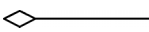
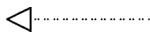
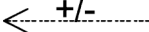



BMM, i* and KAOS all support the refinement of goals into sub-goals. Moreover, BMM and i* distinguish two types of refinement relations: means-end relationships and decomposition relations. It is important to understand the distinction between means-end and decomposition. Existing languages like i* and KAOS do not distinguish correctly between these relations. Decomposition decomposes a goal into more concrete sub goals in such a way that a part-whole relationship exists between the goal and its decomposition. A means-end relation is much more a *causal effect* relation. The achievement of a means leads to the realisation of the end. If there are multiple means identified, the realisation of just one will facilitate the realisation of the end.

Therefore, we provide the following definitions:

- A goal decomposition decomposes an abstract goal to multiple more concrete sub-goals, such that the abstract goal is achieved if and only if all sub-goals are achieved. Typically, goal decomposition is used to define a goal more precisely, resulting in goal trees with measurable indicators, e.g. key performance indicators, at the leaves of the trees. A decomposition answers the WHAT question in goal refinement. Figure 6 illustrates a decomposition. It creates a definition of what decreasing the support staff actually means.
- A means-end relation relates a goal (*the end*) to some artefact (*the means*) that realises the goal. This artefact can be another goal or requirement, or can be an architectural element, such a service, process or application. While goal decomposition is used for the more precise definition of goals, the means-end relation is typically used to illustrate causal effect relations between goals and other artefacts. The means-end relation answers the HOW and WHY questions in goal refinement.

Figure 7 illustrates a means-end relation. In this figure, reducing the amount of customer calls is believed to realise decreasing the support staff.

Table 2. Goal refinement relations.

Relation	Notation	Relation	Notation
Decomposition		Means-end	
Contribution		Conflict	
Include		Extend	

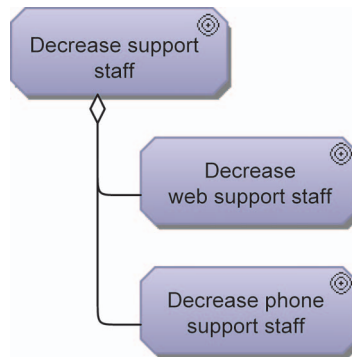


Figure 6. A decomposition.

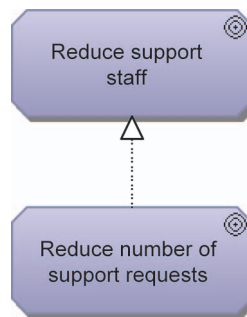


Figure 7. Means-end.

By reducing the amount of customer calls we can reduce the support staff. The stakeholders believe there is a *cause effect* between these two goals. This causal effect is based on some goal theory, or assumptions, from the stakeholder that if customers stop calling, you do not need staff answering them.

5.1.2.1. Conflicts, obstacles and contributions. Both i* and KAOS allow one to model that some goal or situation has 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 is 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 Section 5.2).

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

- A *contribution relation* from some goal G1 to another goal G2 to represent that G1 contributes (influences) the satisfaction of G2 positively or negatively. Typically, the contribution relation is used 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, different qualification ranges may be used, such as the range 0 ... 10 or the range ++, +, +/-, -, -- mentioned above.
- A *conflict relation* 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.

5.1.2.2. 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.



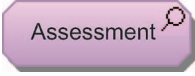
Since it is considered useful to make such assumptions explicit, ARMOR supports the general notion of ‘assumption’, however an assumption is not recorded as a concept but as a property of a goal.

5.2. Stakeholder domain

Table 3 depicts the concepts of the stakeholder domain, which are used to model the origin and owners of goals and requirements. The concepts have the following interpretation:

- A *stakeholder* represents an individual, team, or organisation (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture. This definition is adopted from TOGAF (The Open Group. TOGAFTM Version 9. <http://www.opengroup.org/togaf>). A stakeholder typically associates value to certain aspects of the enterprise, and thus also its reflection in the enterprise’s architecture. Examples of stakeholders are not only the board of directors, shareholders, customers, business and application architects, but also legislative authorities.
- A *concern* represents some key interest that is crucially important to certain stakeholders in a system, and determines the acceptability of the system.

Table 3. ARMOR – stakeholder domain.

Concept	Notation
Stakeholder	
Concern	
Assessment	

A concern may pertain to any aspect of the system’s functioning, development or operation, including considerations such as performance, reliability, security, distribution, and evolvability. This definition is also adopted from TOGAF (The Open Group. TOGAF™ Version 9. <http://www.opengroup.org/togaf>).

- An *assessment* represents the outcome of the analysis of some concern. This outcome may trigger a change to the EA, which is addressed by the definition of new or adapted business goals.

The association relation of ArchiMate is (re-)used to relate stakeholders to concerns and concerns to assessments. A stakeholder can have one or more concerns, and a concern may be shared by multiple stakeholders. An assessment typically assesses a single concern, but could involve multiple concerns. A concern may be analysed through different assessments.

5.3. Meta-model

Figure 8 depicts the abstract syntax, or meta-model, of ARMOR. The coloured classes represent the new concepts introduced by ARMOR, the white classes represent

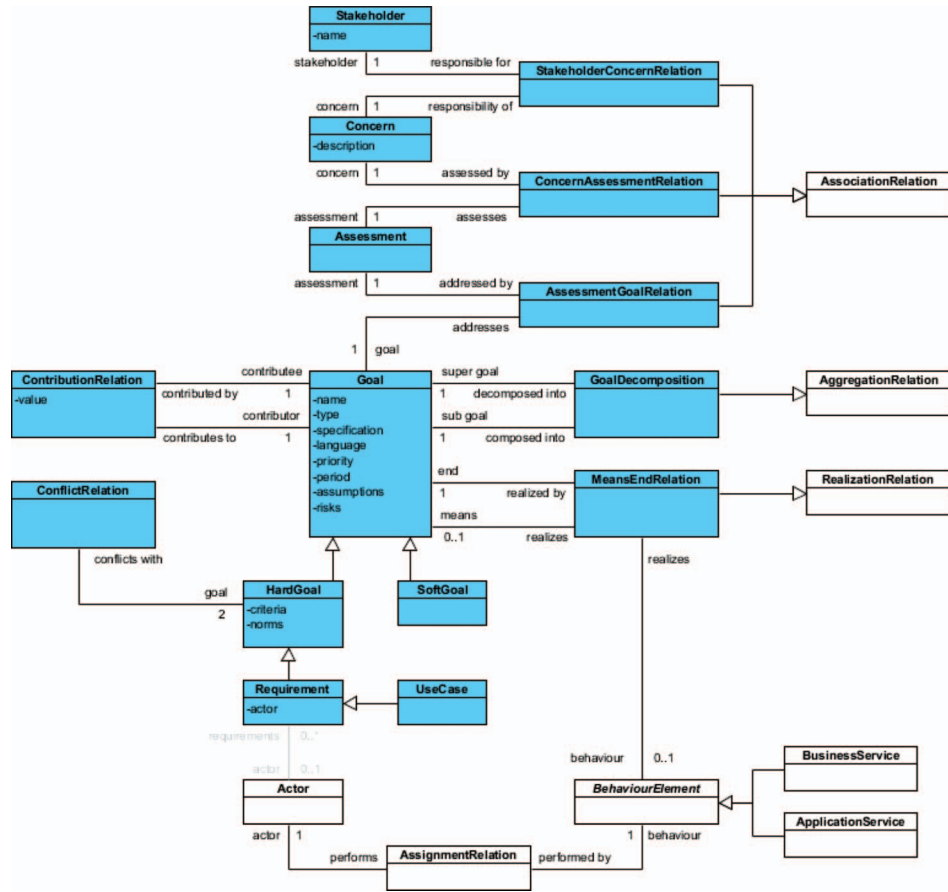


Figure 8. ARMOR meta-model.

the reused concepts from ArchiMate. Concepts from the stakeholder domain are: stakeholder, concern, assessment and their relations: stakeholder concern relation, concern assessment relation and assessment goal relation. These relations are all based upon the existing association relation from ArchiMate. In the requirements domain we find the goal concept, which is specialised into hard goal and soft goal. The hard goal is further specialised into requirement. A use case is a specialisation of a requirement. The contribution and conflict relation are two new relations. The means-end relation is a specialisation of the ArchiMate realisation relation. The goal decomposition relation is a specialisation of the ArchiMate aggregation relation.

The idea is to use ARMOR in combination with ArchiMate. The realisation of requirements through ArchiMate concepts corresponds with the way KAOS relates requirements to behaviour, actors and data. Behavioural elements realise the requirements; actors are responsible for performing this behaviour and data are processed through performing this behaviour, see Figure 9.

6. Application of ARMOR: patient registration

We will demonstrate the application of ARMOR through a real life example in the healthcare industry. Through the remainder of this example we will call the hospital where this project took place Hospital X. This particular hospital specialises in movement and posture disorders.

6.1. Case description

Hospital X strives to ensure satisfied patients, responsible and satisfied employees, safe and efficient care, steering based on quality and service and finally accomplishment and growth. Hospital X elaborated this in its mission, which is included in the policy plan; this mission has four core elements:

- Patient focus, deliver care based on the needs and wishes of the patient;
- Excellence, develop treatments based upon the latest scientific breakthroughs;
- Innovation through research and development;
- Entrepreneurial through expanding and seizing business opportunities.

Hospital X experienced a number of problems, the major concern being that patients had trouble with the lengthy registration process, which was basically in conflict with their patient focus goals. Secondary concerns were incomplete patient information, incomplete information delivery and no insight in whether patients should be billed or not based upon their insurance status. Last but not least, the new electronic patient record (EPR) act forces hospitals to record patient information in

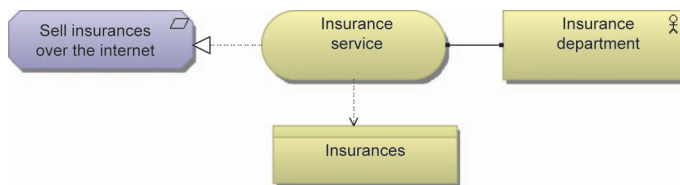


Figure 9. Realising requirements.

a national database to improve information delivery between care givers. Hospital X also identified a number of business opportunities, healthcare 2.0 being the most important one. Healthcare 2.0 is patient centred working using the most modern (communication) technologies available.

The project goal was to elicit high-level business requirements and analyse these using the modelling language presented in this article. We used two workshops to elicit and validate the high-level requirements and *designed* a to-be architecture based on these results. This part of the architecture was then discussed and approved by the senior information architect at the hospital.

Through this example we will demonstrate that ARMOR facilitates EA design through relating stakeholder goals to elements from the EA through goal refinement, thus demonstrating the need for that particular architecture. We will demonstrate that: (i) ARMOR captures the relevant business context, comprising assessments of stakeholder concerns and information found in business plans; thus facilitating improved traceability from high-level goals to architecture requirements and architecture elements, (ii) that this traceability facilitates adapting to change easier, through following the relationships from high-level strategic goals to the architecture realising it, (iii) ARMOR also facilitates improved detection of conflicting interests and solutions and (iv) using ARMOR it becomes possible to evaluate alternatives against soft goals from the organisation. Owing to space limitations it is impossible to be anything near complete; therefore, we will only provide models emphasising on these aspects.

6.2. Goal modelling using views

This example uses three kinds of modelling views. The first view is the stakeholder view. In this view we model the relevant stakeholders of this project, their concerns, assessments of these concerns and the first high-level goals. The second view we use is a goal refinement view. Here we select a stakeholder concern and refine the initial goals into requirements for the EA. The final view is a requirements implementation view. Here we show which architectural elements realise the requirements. This example is structured according to these views. We will provide at least one example of each view.

6.2.1. Stakeholder view

This stakeholder view is limited to the board of Hospital X, as shown in Figure 10. The board is concerned with innovation, patient focus, excellence and entrepreneurship. These concerns were identified in the policy plan of Hospital X. Two major assessments were identified at Hospital X. To address the innovation concern all new products and services should fit within healthcare 2.0. Healthcare 2.0 means patient centred working and using modern technology to bring the healthcare provider and the patient closer together and give the patient a sense that he is more involved in the care process. A second area of concern is the current patient registration system. This system is troublesome for the patient; he has to wait in line before he can register himself as a patient at the hospital. Patient registration is a lengthy process, and has a negative impact on the patient, where he is preoccupied with his illness and should not be bothered with a lengthy registration process.

Based upon these assessments and concerns Hospital X identified a number of change goals, patient registration should happen based upon consumer feedback,

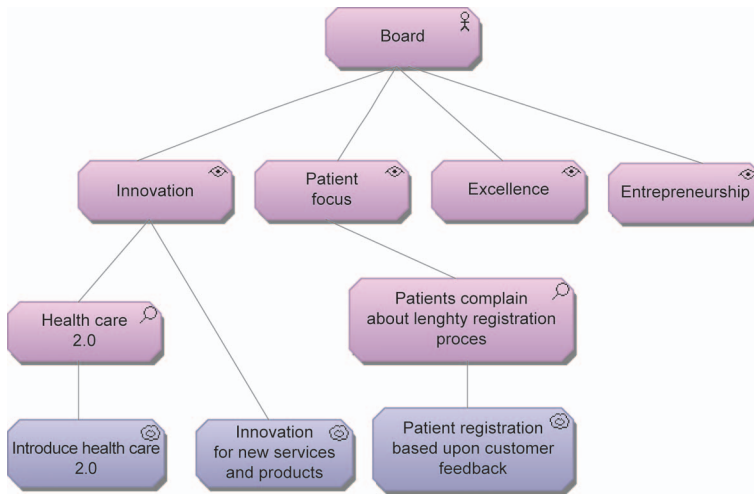


Figure 10. Resulting stakeholder model.

healthcare 2.0 has to be introduced in the organisation and new products and services should be developed on an innovative way. Based upon surveying the customers they decided to improve the patient registration process.

6.2.2. Goal refinement view

In this view the initial goals are structured and refined. To demonstrate the goal refinement view we refine the security concern from the information manager. His main goal was to provide a safe to use service. Safety is by itself a vague statement, therefore we modelled this as a soft goal, it requires a *decomposition* into measurable goals. Safety in this case is decomposed into ‘protect identity of the patient’ and ‘protect patient information’. Protecting the identity of the patient is realised through checking the identity of each individual patient. This is decomposed into automated and manual checks. Protecting personal information is realised through denying access to third parties and this led to the requirement of encrypting the patient information.

6.2.3. Requirements implementation view

Figure 12 depicts a so-called requirements implementation model. Requirements are realised by some form of behaviour from the EA, which falls under the responsibility of an actor and some data is processed. In this view we *operationalise* the requirement for manual checks through a business process ‘patient identification’, which is the *responsibility* of the employee at the registration desk and during this process identification information is used.

7. Tool support and analysis

Any modelling language, especially when applied in realistic situations where models may become quite large, can only be successful if supported by adequate and

professional tooling. A good model editor makes sure that the language is applied correctly and consistently, and may offer facilities for, among others, version control and multi-user editing. A modelling tool may also support the use of viewpoints and views on models. We have implemented the ARMOR language in the architecture modelling tool BiZZdesign Architect (see <http://www.bizzdesign.nl>), as an add-on to the ArchiMate language. This tool provides all of the functionalities described above.

Visualisation and analysis of models can hardly be carried out by hand and requires tools as well. For ARMOR, we have implemented a number of useful analysis techniques for requirements models in BiZZdesign Architect, which we will briefly describe below (illustrated with the Hospital X example).

7.1. Traceability of stakeholder concerns

Figure 11 depicts a viewpoint based upon a stakeholder concern. What we see here is that the board is concerned with innovation. An assessment of this concern reveals that using healthcare 2.0 is an opportunity in new product and service development. Two goals are elicited for this concern and these goals are realised through the first high-level requirement that Hospital X should offer some sort on online registration from home. Figure 12 depicts the first architectural model based upon this requirement. In this way, we can trace from the stakeholders, their concerns and

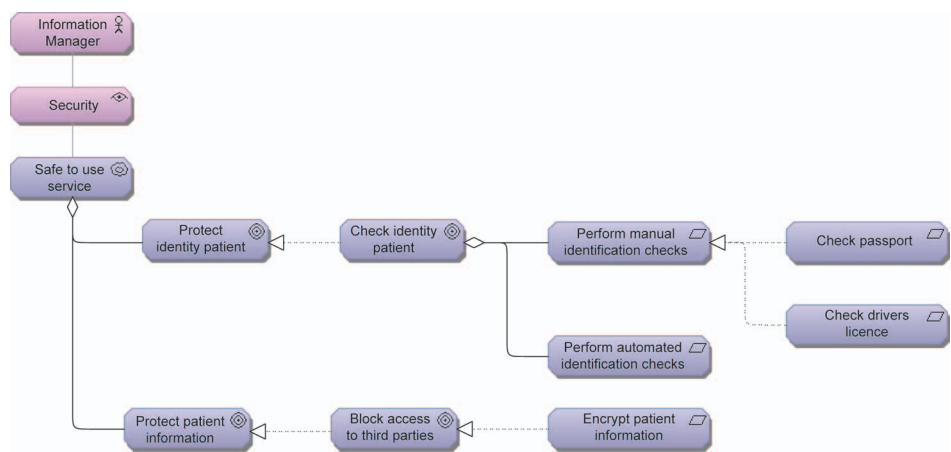


Figure 11. Goal refinement for the security concern.

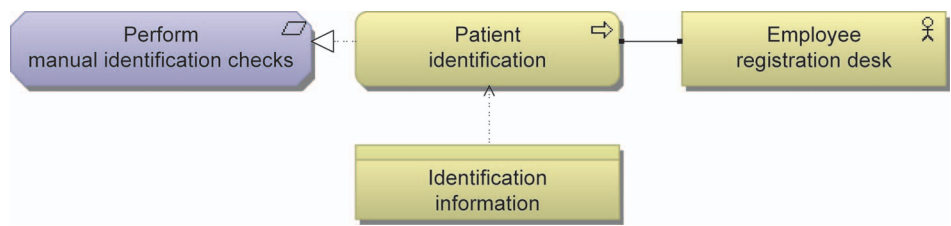


Figure 12. Realising online patient registration.

assessments from these concerns to the high level goals, refined goals, requirements and ultimately the relevant enterprise architectural elements.

7.2. *Impact of change analysis*

Traceability of stakeholder concerns introduces powerful analysis possibilities; one of these is the impact of change analysis. To demonstrate this analysis we use the goal models from Figures 11 and 12. This impact of change analysis is one of the more powerful analysis types. Every organisation interacts with the environment; this environment contributes to the goals of the organisation. One of the high-level goals here is to comply with the data protection act, which is decomposed into the requirement check identity of the patient and compliance with security guidelines. Supposed changes occur in the legislation concerning the data protection act, through these goal refinement links we can trace into the architectural models and analyse which parts of the organisation are affected by these changes. This way we know which services processes, actors and IT systems are affected by a change.

7.3. *Detection of conflicting interests and solutions*

We were able to identify conflicting goals between two different requirements views. The manager care was interested in creating an information exchange service based upon registered patient data. However, this was not allowed because patient data are confidential and this was contained in the information security guidelines within the hospital, see Figure 13.

Because of the importance of complying with information security guidelines, extracting patient information from registration data was dropped as a goal by the manager care.

7.4. *Evaluation of alternatives*

Two main solution alternatives emerged at Hospital X. One possibility was that customers preregister from home via the internet. The second alternative was that they could register at different sign-in stations located throughout the hospital. The latter is also used by Schiphol airport for a speedy check-in procedure. Based upon these alternatives Hospital X selected a number of important soft goals. These alternatives are then evaluated based upon their contribution to said soft goals. Figure 14 the modelled results from this evaluation. The presented alternatives were evaluated against a number of soft goals. In this case stakeholders 'graded' the contributions on friendliness, safety, improved registration, feedback from customers and healthcare 2.0 Based upon these contribution models Hospital X decided to pursue the online registration service.



Figure 13. Conflict detection.

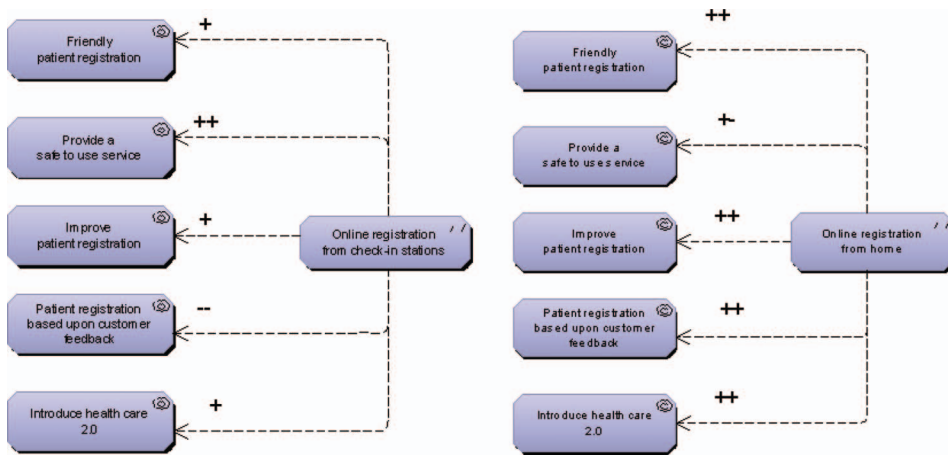


Figure 14. Evaluation of alternatives.

8. Related work

As mentioned in the introduction, requirements management plays a central role in TOGAF's ADM (The Open Group. TOGAF™ Version 9. <http://www.opengroup.org/togaf>). 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 (Mulholland and Macaulay 2006). Like TOGAF, this framework also recognises the importance of requirements for EAs. IAF recognises 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.

i* has also been used as a problem investigation technique for architecture design and business modelling (Yu *et al.* 2006, Samavi *et al.* 2008). This way the motivation for architectural elements is linked to their implementation. Yu *et al.* (2006) illustrate the potential benefit of using BMM and i* in combination to support intentional modelling and analysis of EAs. This work does not consider the integration or alignment of these languages with existing enterprise modelling languages. Gordijn *et al.* (2006) extends intentional modelling with value modelling, by combining the i* framework and the e³ value methodology.

Braun and Winter (2005) discuss relevant meta-models for EAs. They introduce three layers for EA modelling, the strategy layer, the organisational layer and the application layer. This work is mostly relevant for ArchiMate, but they do introduce relevant concepts. These concepts are found in the strategy layer. The strategy layer

models the business units, services, goals and their performance indicators. However, it maps mostly to the ArchiMate business layer whereas they introduce services and business units. Their notion of a goal is also limited to only the strategy layer, whereas we identify goals for the business layer, application layer and technology layer. We also introduce ways how to relate goals to the behavioural elements in the EA, in such a way that already exists in existing RE languages like KAOS (Dardenne *et al.* 1993). We also introduce modelling concepts, which Braun and Winter (2005) lacks.

Kurpjuweit and Winter (2009) and Lagerström *et al.* (2009) propose a way of working similar to the modelling concepts introduced in this article. They take both a stakeholder and goal-oriented approach to derive architectural models based on both stakeholder needs and causal effect relations, which is also similar to an approach we suggested here (Engelsman *et al.* 2009). Our work supplements this work with a modelling language that can be used to capture these results and explicitly show how the derived models implement the stakeholder needs and goals. Our work has a better grounding in GORE theory, thus supporting more concrete concepts and relations, like a distinction between goals and requirements, and a distinction between means-end, decomposition, conflict and contribution.

Concerning tool support for EA, 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 modelling is Enterprise Architect from Sparx Systems (2008). Enterprise Architect is primarily a UML modelling tool focused on software engineering. But it also supports a Zachman Framework extension (Sparx Systems 2008). For modelling the motivation of the Zachman framework Enterprise Architect relies on goal modelling 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 realisation relation between a business service and its use-case. This use-case is associated with a requirement or refined goal. This way ARMOR realises traceability from business goals, through requirements to architectural elements.

Design & Engineering Methodology for Organisations (DEMO) (Dietz 2001) is a methodology for the design, engineering, and implementation of organisations and networks of organisations. Originally meant as an RE approach for information systems, however the authors quickly found out that it was also applicable for business process engineering and workflow management. DEMO is used to not only specify organisations at a behaviour level, but also at the component level, trying to bridge the gap between behaviour specification and design specification. DEMO focuses on specifying the essential models of an organisation. The approach is well validated and growing in popularity.

8.1. Architectural principles

An important research development in the field of EA is the use of principles. Because of its importance we will discuss this in a separate section of the related work and provide a more in depth analysis of the topic (Lindstrom 2006, Op't Land

and Proper 2007, van Bommel *et al.* 2007). However, a precise definition of the concept of principles as well as the mechanisms and procedures needed to turn them into an effective regulatory means still lacks (Stelzer 2009). Therefore, this article will only discuss the generic attributes of a principle and how principles can supplement the ARMOR language. Generally speaking, two different types of principles can be distinguished, *normative principles* and *engineering principles*. Normative principles are rules of conduct, or guidelines. They provide a norm that designers have to take into account. A normative principle can be broken; someone can choose not to follow them. Engineering principles describe some law of nature that underlies the working of some artificial device. Engineering principles are based upon causality, if we apply this principle these effects will happen. An example of an engineering principle is that any object, wholly or partially immersed in a fluid, is buoyed up by a force equal to the weight of the fluid displaced by the object. An engineering principle is more a fact and cannot be broken. Architectural principles are *normative principles* based upon causality motivating the working of this principle. In more concrete words, architectural principles use the business drivers as the goal they should realise, for example business goals, architecture goals and IT goals. When an architectural principle is applied, the organisation expects that it has some positive contribution on the business drivers, this is the causality aspect. This is very much similar to the means-end relation this article uses in goal refinement. For example, when an organisation identifies that user productivity is a goal they should pursue, they can identify principles that realise this goal. 'All applications must be easy to use' could be such principle. It is based on the assumption (or fact) that when users do not have to struggle with learning the applications they are more productive, see Figure 15.

Since an architectural principle is a normative principle it constrains the solution space of the designer. A principle applies to all solutions in a particular context and should be stable enough to be reusable.

This article discusses the modelling of goals and requirements for organisations. A goal is a desired state or desired effect that a stakeholder wants to bring about in the problem domain. A requirement is also a desired state or desired effect, but for an actor instead of a stakeholder.

In this context a principle is a *preferred means* to reach a desired state. A principle brings about a desired *proven (or assumed proven)* effect in the problem domain, by constraining the solution space.

Principles and requirements both exist in the solution space; however, a principle is not a requirement. A requirement is associated with a single solution, whereas a principle is associated with all solutions in a particular context. When a principle has to be applied to a single solution it needs to be refined into requirements. This is where the introduced goal refinement techniques from this article can assist.



Figure 15. Causality of a principle.

For example, the principle ‘compliance with law’ is provided by TOGAF as an architectural principle. Applying this principle leads to the desired effect that the legislators will not sanction the organisation. This principle constrains every possible solution an organisation can develop, from business services to IT infrastructure. However, it is still too general to be applied to a single solution. It therefore requires *refinement*, much in the way this article proposed. Compliance with law could be decomposed into more concrete legislation that applies to the organisation and then refined into requirements. It is interesting to notice that with this particular principle an organisation might choose not to follow this principle when the sanctions by the legislator are less expensive than the costs for compliance.

This work therefore extends the current literature about principles with techniques found in GORE.

9. Conclusions and future work

In this article, we have presented a language, called ARMOR, for modelling goals and requirements in EAs. 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 the existing requirements modelling languages and is aligned with the standard enterprise modelling language ArchiMate. This brings existing theory and analysis techniques to the domain of EA modelling. We have demonstrated how to realise traceability of stakeholder concerns to the architectural elements. This traceability is realised through goal refinement and providing means to integrate this into the architecture domain. We are able to model and refine strategic goals and policies found in business plans. Through goal refinement we are able to link this business context to the new architecture elements, thus realising traceability. Through capturing these links it becomes possible to reason about the effects of changing goals on the EA. Through following links in the requirements domain, the EA domain and their integration, we can derive which architectural elements are affected by a change in a high-level goal. We also showed how ARMOR can be used to support stakeholders in reasoning about conflicting interests and solutions. Visualising the effects of conflicting goals helps to understand what the effects are of these conflicts on the EA and leads to dropping or changing certain goals by certain stakeholders. Finally we presented the use of ARMOR to reason about two emerging solution alternatives and to evaluate them based upon the soft goals provided by the stakeholders. Through explicitly modelling these contributions the stakeholders can select the most favourable alternative.

Currently, we apply ARMOR combined with an architecture-driven RE approach in a number of consultancy projects. These projects help to validate and improve ARMOR and the associated approach. For example, through applying ARMOR more in practice we evaluate the appropriateness and completeness of our concepts and derive guidelines for goal decomposition and means-end analysis.

These can be captured in *consistency rules* and *refinement patterns*. Consistency rules guarantee the correctness of requirements model and refinement patterns describe common goal refinements.

Our future work also aims at the formalisation of ARMOR, investigation of the correlation with architectural principles, and the elaboration of various analysis techniques, using existing work such as the work referred to in this article.

We also like to investigate the correlation with architectural principles, business rules, and the elaboration of various analysis techniques, using existing work such as the work referred to in this article. Finally, we wish to extend this work to business process modelling. This way it becomes possible to model the goals for EA and through further goal refinement reach the requirements for business process models.

References

- Antón, A.I., 1996. Goal-based requirements analysis. In: *Proceedings of the second international conference on requirements engineering*. Washington, DC: IEEE Computer Society, 136–144.
- Braun, C. and Winter, R., 2005. A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform. In: *Proceedings of the workshop in enterprise modelling and information systems architectures*. EMISA, 64–79.
- Brooks, F., 1986. No silver bullet: essence and accidents of software engineering. *IEEE Computer*, 20 (4), 10–19.
- Business Rules Group, 2008. The business motivation model – business governance in a volatile world [online]. Release 1.3. Available from: <http://www.businessrulesgroup.org/bmm.shtml> [Accessed 27 May 2010].
- Dardenne, A., van Lamsweerde, A., and Fickas, S., 1993. Goal-directed requirements acquisition. *Science of Computer Programming*, 20, 3–50.
- Dijkman, R.M., Quartel, D.A.C., and van Sinderen, M.J., 2008. Consistency in multi-viewpoint design of enterprise information systems. *Information and Software Technology (IST)*, 50 (7–8), 737–752.
- Dietz, J., 2001. DEMO: towards a discipline of organization engineering. *European Journal of Operational Research*, 128 (2), 351–363.
- Engelsman, W., et al. 2009. Architecture-driven requirements engineering. In: E. Proper, F. Harmsen, and J.L.G. Dietz, eds. *Advances in enterprise engineering II*, Volume 28 of lecture notes in business information processing. Berlin. Berlin, Heidelberg: Springer, 134–154.
- Finkelstein, A., et al. 1991. Viewpoint oriented software development: methods and viewpoints in requirements engineering. *Lecture Notes in Computer Science*, 490, 29–54.
- Gordijn, J., Yu, E., and van der Raadt, B., 2006. e-Service Design using i* and e³ value modeling. *IEEE Software*, 23 (3), 26–33.
- ISO, 1998. ISO/IEC 10746-3. *The information technology – open distributed processing – reference model: architecture*. Geneva: ISO.
- ISO, 2007. ISO/IEC 42010. *Systems and software engineering – recommended practice for architectural description of software-intensive systems*. Geneva: ISO.
- Jacobson, I., 1995. The use-case construct in object-oriented software engineering. In: J.M. Carroll, ed. *Scenario-based design: envisioning work and technology in system development*. New York: John Wiley & Sons, 309–338.
- Jonkers, H., et al. 2004. Concepts of modeling enterprise architectures. *International Journal of Cooperative Information Systems*, 13, 257–287.
- Kurpjuweit, S. and Winter, R., 2009. Concern-oriented business architecture engineering. In: *Proceedings of 24th annual ACM symposium on applied computing (SAC)*, Honolulu, Hawaii, New York: ACM, 265–272.
- Lagerström, R., et al. 2009. Enterprise meta modeling methods – combining a stakeholder-oriented and a causality-based approach. In: T. Halpin and et al., eds. *Enterprise, business-process and information systems modeling*. Lecture notes in business information processing, Berlin/Heidelberg: Springer, 381–393.
- Lankhorst, M., et al. 2005. *Enterprise architecture at work: modelling, communication and analysis*. New York: Springer.

- Lindstrom, A., 2006. On the syntax and semantics of architectural principles. *In: Proceedings of the 39th Hawaii international conference on system sciences*. Washington, DC: IEEE Computer Society.
- Mulholland, A. and Macaulay, A.L., 2006. *Architecture and the integrated architecture framework*. Capgemini: White Paper.
- Nuseibeh, B., Kramer, J., and Finkelstein, A., 1994. A framework for expressing the relationships between multiple views in requirements specifications. *IEEE Transactions on Software Engineering*, 20 (10), 760–773.
- Op't Land, M. and Proper, H.A., 2007. Impact of principles on enterprise engineering. *In: H. Österle, J. Schelp, and R. Winter, eds. Proceedings of the 15th European conference on information systems*. St Gallen, Switzerland: University of St Gallen.
- Respect-IT, 2007. *A KAOS tutorial, V1.0* [online]. Available from: <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf> [Accessed 27 May 2010].
- Samavi, R., Yu, E., and Topaloglou, T., 2008. Strategic reasoning about business models: a conceptual modeling approach. *Information Systems and E-Business Management*, 7 (2), 1–28.
- Schoman, K. and Ross, D.T., 1977. Structured analysis for requirements definition. *IEEE Transactions on Software Engineering*, 3 (1), 363–386.
- Sommerville, I., Sawyer, P., and Viller, S., 1998. Viewpoints for requirements elicitation: a practical approach. *In: Proceedings of the third IEEE international conference on requirements engineering (ICRE 98)*. Washington, DC: IEEE Computer Society, 74–81.
- Sowa, J.F. and Zachman, J.A., 1992. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31 (3), 590.
- Sparx Systems, 2008. *MDG technology for Zachman framework user guide* [online]. Available from: <http://www.sparxsystems.com.au/downloads/pdf/ZFUserGuide.pdf> [Accessed 27 May 2010].
- Stelzer, D., 2009. Enterprise architecture principles: literature review and research directions. *In: S. Aier, J. Schelp, and M. Schönherr, eds. Pre-proceedings of the 4th workshop on trends in enterprise architecture research*, Stockholm, Sweden, 23 November 2009, 21–35.
- van Bommel, P., et al. 2007. Architecture principles – a regulative perspective on enterprise architecture. *In: M. Reichert, S. Strecker, and K. Turowski, eds. Enterprise modelling and information systems architectures (EMISA2007)*, no. 119 in lecture notes in informatics. Bonn, Germany: Gesellschaft für Informatik, 47–60.
- van Lamsweerde, A., 2000. Requirements engineering in the year 00: a research perspective. *In: ICSE '00: proceedings of the 22nd international conference on Software engineering*. New York, NY, USA: ACM Press, 5–19.
- van Lamsweerde, A., 2001. Goal oriented requirements engineering: a guided tour. *In: Proceedings of the 5th IEEE international symposium on requirements engineering (RE 2001)*. Washington, DC: IEEE Computer Society, 249–262.
- van Lamsweerde, A., Darimont, R., and Letier, E., 1998. Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*. Special Issue on Inconsistency Management in Software Development, 24 (11), 908–926.
- Wieringa, R., 2004. Requirements engineering: problem analysis and solution specification [extended abstract]. *Lecture Notes in Computer Science*, 3140, 13–16.
- Yu, E., 1997. Towards modelling and reasoning support for early-phase requirements engineering. *In: Proceedings of the 3rd IEEE international symposium on requirements engineering (RE'97)*, 6–8 January. Washington, DC: IEEE Computer Society, 226–235.
- Yu, E., Strohmaier, M., and Deng, X., 2006. Exploring intentional modeling and analysis for enterprise architecture. *In: Proceedings of the EDOC 2006 workshop on trends in enterprise architecture research (TEAR 2006)*. Washington, DC: IEEE Computer Society.