Goal-Oriented Requirements Engineering and Enterprise Architecture: Two Case Studies and Some Lessons Learned

Wilco Engelsman[1,2] and Roel Wieringa[2]

[1] BiZZdesign
w.engelsman@bizzdesign.nl
[2] University of Twente
roelw@cs.utwente.nl

Abstract. An enterprise-architecture (EA) is a high-level representation of the enterprise, used for managing the relation between business and IT. [**Problem**] Ideally, all elements of an enterprise architecture can be traced to business goals ad vice versa, but in practice, this is not the case. In this experience paper we explore the use of goal-oriented requirements engineering (GORE) techniques to improve this bidirectional traceability. [**Principal ideas/results**] We collected GORE techniques from KAOS, i*, Tropos, BMM and TOGAF and integrated them in a language called ARMOR. This was used by enterprise architects in case study. It turned out that the language was too complex for the architects to understand as intended. Based on this we redefined ARMOR to contain only a minimum number of goal-oriented concepts, and this was tested in a second case study. This second case study suggests that the minimal version is still useful for traceability management in practice. [**Contribution**] We have identified a core set of concepts of goal-oriented requirements engineering, that can be used in the practice of enterprise architecture. Our analysis provides hypotheses into GORE that will be tested in future case studies.

1 Introduction

In large companies the gap between business and IT is usually bridged by designing and maintaining a so-called *enterprise architecture* (EA), which is a high-level representation of the enterprise, used for managing the relation between business and IT. A full-scale EA consists (i) an architecture of the business, in terms of products, services and processes, (ii) an application architecture in terms of of application components, functions and services, (iii) an infrastructure architecture in terms of servers, mainframes, network, and (iv) the relationships between these different architectures [19].

Enterprise architectures are typically modelled in larger organizations (say starting from 500 employees) and are used to coordinate IT projects and to manage the cost of IT. Increasingly, they are also used to increase flexibility of the organization and to justify the contribution of IT to business goals. This requires traceability of business goals to IT architecture (to quickly identify the impact on IT of changes in business goals) and of IT architecture to business goals (to justify the contribution of an IT component to a business goal). This requires a goal-oriented addition to the current crop of EA modelling languages. In this experience paper, we explore the addition of goal-oriented requirements engineering (GORE) to enterprise architecture modelling in order to realize this bidirectional traceability. An important constraint is that we want the resulting language to be usable and useful for enterprise architects in practice. Usability means at least tool support and understandability for the architects; utility means that the resulting language and tool can indeed be used to realize traceability in practical cases.

2 Related Work

The Business Rules Group has published a model that relates the business goals and EA, called the *Business Motivation Model* (BMM),[1] which is now an OMG standard. The Open Group TOGAF standard also assume a close link between EA and business goals [19].

However, little research has been done to date to extend architecture modelling with goal modelling. Clements & Bass [4] extend software architecture modelling with GORE, but abolish all notational conventions of GORE techniques and return to the basics of bulleted lists of possible goals and possible stakeholders. Stirna et al. [16] describe a participative approach to enterprise modelling that includes relating goals to enterprise models. Jureta & Faulkner [9] sketch a goal-oriented language, that links goals and a number of other intentional structures to actors, but not to enterprise architecture models. Horkhoff & Yu [8] present a method to evaluate achievement of goals by enterprise models, all represented in i*. None of these methods presents a technique to relate business goals to EA validated in practice with enterprise architects.

An important obstacle to applying GORE in practice is the complexity of the notation. Matulevičius and Heymans [11] concluded that i* and KAOS contain constructs not used in practice and contain different constructs representing the same thing. After an ontological analysis they concluded that the i* goal and soft goal are essentially the same concept, just as the means-end relation and the contribution relation [12]. Moody et al. [13,14] identified many opportunities for clarification and simplification of the i* notation. Carvallo et al [3] recommended that practitioners should not and need not learn the entire syntax of i*. Our paper is not about notations but about usability and utility of GORE concepts in EA practice; the Archimate 1.0 language on top of which ARMOR is defined, was already understood and used by the architects who participated in our case studies.

[1] http://www.businessrulesgroup.org/bmm.shtml

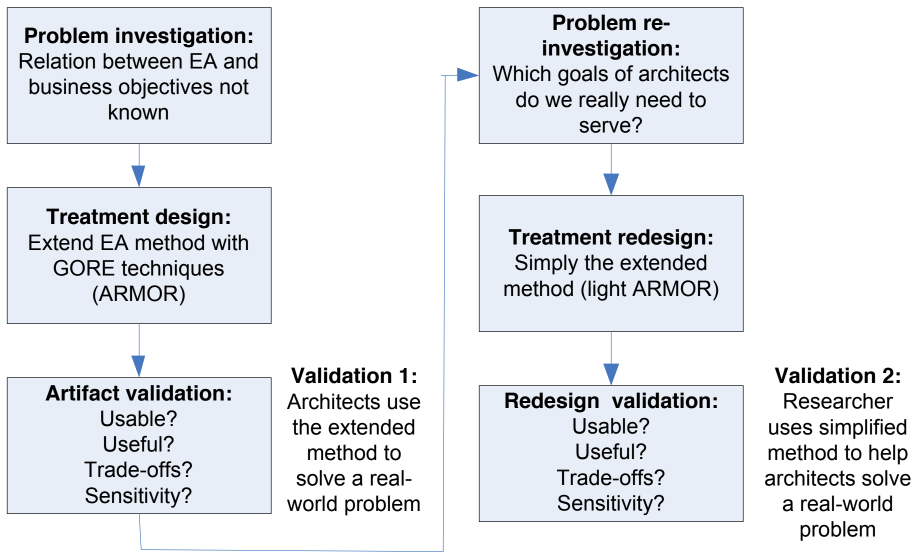
Problem investigation: Relation between EA and business objectives not known
Treatment design: Extend EA method with GORE techniques (ARMOR)
Artifact validation: Usable? Useful? Trade-offs? Sensitivity?
Validation 1: Architects use the extended method to solve a real-world problem
Problem re-investigation: Which goals of architects do we really need to serve?
Treatment redesign: Simply the extended method (light ARMOR)
Redesign validation: Usable? Useful? Trade-offs? Sensitivity?
Validation 2: Researcher uses simplified method to help architects solve a real-world problem

Fig. 1. Design research methodology of this paper

3 Research Methodology

We used a design research methodology in which we alternate over an engineering cycle, where we design an artifact, and a research cycle, where we investigate the properties of this artifact and of the problems it is intended to solve [7,20] Figure 1 shows that we executed the engineering cycle twice. In the first iteration, we investigated the problem to be solved, designed a method called ARMOR to treat the problem (section 4), supported by a tool for editing and traceability analysis[2] and validated the artifact (section 5). In the second, we stripped ARMOR to its essentials, called Light ARMOR (section 6), and validated this lightweight version and supporting tool (section 7).

ARMOR is an extension of an EA modelling language called Archimate 1.0 [18] with goal-oriented requirements engineering (GORE) techniques [5]. We call this a *treatment* rather than a solution because it would be simplistic to assume that any real-world problem can be totally solved, just as it would be simplistic to assume that any medical problem could be totally eliminated by a medicine.

ARMOR combined concepts from all well-known GORE languages, which is why this research also provides insights into GORE concepts in general. To validate ARMOR, the first author taught the method to enterprise architects of a large government organization, who then used it to perform an EA design project. This is a form of *technical action research* (TAR), in which an artifact is validated by actually using it to solve a real-world problem. This TAR project itself has the structure of an engineering cycle performed by the enterprise architects (figure 2).

[2] http://www.bizzdesign.nl/download/downloads-trial-software

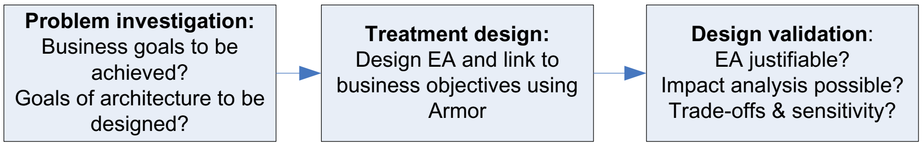
Problem investigation:
Business goals to be achieved?
Goals of architecture to be designed?
Treatment design:
Design EA and link to business objectives using Armor
Design validation:
EA justifiable?
Impact analysis possible?
Trade-offs & sensitivity?

Fig. 2. Structure of validations 1 and 2

These insights from case study 1 led to an improved problem understanding and in a second engineering cycle we simplified ARMOR in the light of the lessons learned. Light ARMOR was then used by the first author to design an EA for another client, acting as consultant. This is validation 2 in figure 1. This is a second TAR project, but this time with the researcher (Engelsman) as actor, rather than the client itself, as in validation 1.

The lessons learned from validation 2 were used to answer the researchers' validation questions about Light ARMOR. These answers were then generalized to GORE concepts in general, when used in similar contexts (section 8).

4 Definition of ARMOR

Table 1 lists the major GORE concepts and shows how we have used them in ARMOR. The following list summarizes the motivation for the construction of ARMOR. More detail is provided elsewhere [5].

– Goals belong to *stakeholders,* and different stakeholders may have conflicting goals. This is important in practice but is left undefined in most GORE languages, although the i* concept of intentional actor has some similarity with our stakeholder concept. We have adopted the stakeholder concept of TOGAF [19].
– BMM, i*, and KAOS all define a *goal* as an end (or desire or intention) of a stakeholder but differ in defining this goal as a property of the system or of its environment. We define goal as some end a stakeholder desires to achieve and leave open what it is a property of.
– We follow i* in distinguishing *hard* and *soft* goals but make the requirement "clear satisfaction criteria" explicit by requiring measurability.
– Goal *decomposition* is in terms of conjunction of subgoals. It is called "refinement" in KAOS. Tropos uses the concept of satisficing. i* and BMM have rather vague definitions.
– The *contribution* relation is defined most clearly in Tropos and is taken to mean influence, positive or negative.
– The *means-end* relation is used in i* to identify tasks to realize goals and in KAOS to identify operations to realize goals. In ARMOR we define it as relating a goal (the end) to some artifact (the means) that realizes the goal. This artefact can be anything, such as a goal, requirement or an element from the architecture.

Table 1. Overview of GORE and ARMOR constructs

GORE construct	**ARMOR construct**
“Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments” i* [21].	A *stakeholder* is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture ARMOR [5]. adopted from TOGAF [19].
”Goals are desired system properties that have been expressed by some stakeholder(s)” KAOS [10]. ‘Goals are the intentions of a stakeholder” i* [21].	A *goal* is some end that a stakeholder wants to achieve [5].
“Hard Goals are the intentions of a stakeholder” i* [21].	A *hard goal* is a goal with measurable indicators [5].
“Soft Goals are goals without clear satisfaction criteria” i* [21].	A *soft goal* is a goal without measurable indicators [5].
“An element that is linked to its component nodes” i* [21]. “An end that includes an other end” BMM [2]. ”The parent is satisficed if all of the offspring are satisficed” Tropos [1]. “The conjunction of all the subgoals must be a sufficient condition entailing the goal” KAOS [10].	A goal can be *decomposed* into two or more concrete sub-goals, such that the goal is achieved if and only if all its sub-goals are achieved.
“The contribution of a design on a qualitative goal ...” KAOS [10]. “Link elements to a soft goal to analyze its contribution” i* [21]. “Contribution analysis identifies goals that can contribute positively or negatively in the fulfillment of the goal to be analyzed...” Tropos [1].	A goal G1 *contributes* to another goal G2 if satisfaction of G1 influences the satisfaction of G2 positively or negatively [5].
“These links indicate a relationship between an end, and a means for attaining it i* [21]”. ”Relationship linking a requirement to operations KAOS [10]”.	A *means-end* relation relates a goal (the end) to some artefact (the means) that realizes the goal [5].
“Goals are conflicting if under some boundary condition the goals cannot be achieved altogether” KAOS [10]”.	A *conflict* relation exists between two goals if under some boundary conditions they cannot be achieved together [5].
“Goal assigned to an agent of the software being studied. KAOS [10]”. “A quantitative statement of business need that must be met by a particular architecture or work package” TOGAF [19] .	A *requirement* is some end that must be realized by a single component of the architecture [5].
“Concerns are the key interests that are crucially important to the stakeholders in the system, and determine the acceptability of the system” TOGAF [19].	A *concern* is some key interest that is crucially important to certain stakeholders in a system, and determines the acceptability of the system [5].
“An Assessment is a judgment about some Influencer that affects the organization’s ability to employ its Means or achieve its Ends BMM [2]”.	An *assessment* is the outcome of the analysis of some concern [5].

- Only KAOS defines the *conflict* relation. However we believe it to be so different from the contribution relation that we include it, adopting the KAOS definition.
- KAOS is also the only GORE language that explicitly defines the *requirement* concept. It is defined as a concrete goal that has been assigned to a single actor. TOGAF defines requirement as a business need allocated to an architecture. The ARMOR definition combines these two definitions.
- The concepts of *concern* and *assessment* are not part of GORE but of the EA literature. We therefore included these concepts, taking our clues from BMM and TOGAF.

ARMOR has a notation that extends the EA language Archimate 1.0 [18], and tool support in the form of an editor. The editor supports the creation of integrated goal models and EA models. The tool also provides functionality to trace requirements to EA and vice versa. The resulting language is called ArchiMate 2.0. ArchiMate 1.0 is an Open Group Standard[3]. ArchiMate 2.0 is currently under review by The Open Group for acceptance to update ArchiMate 1.0. The notation is described and motivated elsewhere [5,15] and does not concern us here.

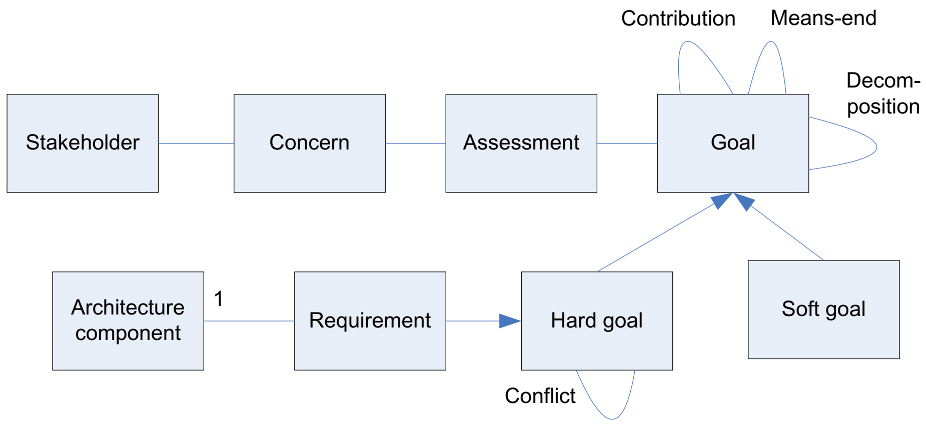

Fig. 3. ARMOR's metamodel. The arrow represents specialization. Cardinalities are not shown in the figure.

Figure 3 shows the core part of ARMOR's metamodel. Cardinalities are not shown so as not to clutter up the diagram, except the cardinality from requirement to architecture component, which is many-one. The diagram shows that stakeholders have concerns, that they assess in a certain way, which leads to goals, that are hard or soft; hard goals can be requirements, and each requirement is allocated to exactly one architecture component. Goals can be decomposed, can have contribution and means-end relations, and they can conflict. The complete meta-model of ARMOR has been described elsewhere [5].

[3] `http://www3.opengroup.org/subjectareas/enterprise/archimate`

5 Case Study 1

To validate ARMOR we first wanted to test usability by enterprise architects. The further question of utility can only be answered once we have a usable language. However, we did want to know whether ARMOR misses potentially useful constructs. We therefore identified the following research questions.

Q1. What constructs of ARMOR do enterprise architects use in practice?
Q2. Why (for which purpose) do they use these concepts and relations?
Q3. Is this the intended use of the constructs?
Q4. Which construct not in ARMOR are considered by architects useful additions to ARMOR?

The only way to answer these questions is to have practicing enterprise architects use ARMOR and observe how they do it. Since ARMOR will not be transferred to a practical context unless we do the transfer, we needed to perform an action case study, where we first transferred knowledge of ARMOR to a company and then observed ARMOR use.

5.1 Case Description and Research Design

The case study took place at a large governmental organization in the Netherlands that we will call Organization 1. The organization is responsible for state pensions and child support payments by the Dutch Government. The budget available for these payments is around thirty billion euros, consisting entirely of taxpayer money. The company employs around 3000 civil servants distributed over several locations in the country. Relevant stakeholders include enterprise architects and information analysts, who are looking for a technique that can show the value of their designs to business stakeholders. Relevant stakeholders also include information managers, who are looking for a technique that would enable them to analyze the effect of changing organization goals on the EA.

Organization 1 contacted BiZZdesign if they could help with improving traceability between the business objectives and the enterprise-architecture. BiZZdesign offered to provide ARMOR with tool support, which the organization accepted.

The first author (Engelsman) provided a one-day training on ARMOR to six enterprise architects of Organization 1. The architects of Organization 1 then proceeded to create ARMOR models of business goals and their links to the existing EA. They did this on their own, by investigating business documents of Organization 1 and by conducting workshops. No help was provided. However, the first author visited Organization 1 every two weeks to review the models made by the architects and to provide advice. On those occasions the first author also made notes of discussions among the architects.

To summarize, the treatment applied to the case consisted of (1) a one-day training and (2) bi-weekly advice. Data collection took place by collecting documents produced by the architects and by making notes during discussions among architects. There was no possibility to collect observations by other means, such as questionnaires or interviews, as the enterprise architects were too busy for that.

5.2 Observations and Explanations

We extracted the following observations from the data.

- The architects used the *stakeholder* concept as intended, to record the existence of some entity that has a stake in the development of the organization. The (obvious) explanation is that the stakeholder concept is widely known in businesses, and has a meaning well-captured by the TOGAF definition that we adopted.
- The architects also used the *goal* concept as intended. This too is a concept well-known in the practice and theory of business management. However, they did not see why the distinction between soft goals and hard goals would be relevant in their models. This is explained by their way of working: The architects started out identifying relevant business goals and then proceeded, later on in their work, to decompose these into key performance indicators (KPIs). So initially, all goals are soft; eventually, all goals are decomposed into hard goals. For example, the soft goal to maintain quality of service was decomposed into the goals to maintain timeliness of service requests and to maintain legality of service, which are hard goals because measurement procedures were defined for them: the maximum amount of time for a service request, and for every decision a reference to the law on which the decision is based, must be documented. They did not see the point of making this transition explicit by using a different symbol for soft and hard goals.
- The *decomposition* relation was used as intended: to refine a goal into more concrete sub-goals, in such a way that achievement of the conjunction of the sub-goals implies the achievement of the higher level goal. For example, the goal to decrease cost was decomposed into the sub-goals to decrease cost of internal services, to decrease cost of external services and to decrease cost of IT.
- The *contribution* relation was used by the architects as intended, namely to indicate that achievement of one goal influences the achievement of another goal. For example, the goal to increase automatic service delivery contributed positively to the goal of decreasing cost of external services.
- The *means-end* relation is constrained in the ARMOR tool to be an influence relation from a system requirement to a goal. This was understood by the architects and they used it in this way. But they did not understand why a separate means-end relation was included to represent this, where a contribution relation expresses in their view exactly the same thing: Influence.
- The *conflict* relation was not used by the architects in this case. The architects explained that in this case there simply were no conflicts between different stakeholder goals. In addition, they did not see any difference between a conflict and a negative contribution.

- In ARMOR, a *requirement* is a goal that must be achieved by a single component of the architecture. This definition was not quite understood by the architects, and they often formulated requirements that were not goals of a single architecture component. An example of this is the "requirement" that the use of marketing techniques must be improved. This is a business goal, not a system requirement.
- The architects had difficulty understanding the difference between *concerns* and goals. The intention of the concept is that it be used for areas of concern for the stakeholder, such as sales, cost or profit. Instead, architects in our case used it to denote stable goal-like statements, such as the goal to achieve excellent service delivery, or to achieve a result-oriented working environment. Even after explaining the difference in one of our bi-weekly meetings, they kept using it the same way. An explanation of this could be that the concern concept is too general to be of use. What concerned the architects in our case was goals; so they used it to express goal-related concerns.
- The architects found it difficult to understand the difference between concern, goal and *assessment*. They sometimes used the assessment concept to store the contextual reasons for having a goal. For example, the goal of cost-reduction was annotated with an "assessment", that is a contextual reason, namely that the Dutch government faces the need for large budget cuts due to the financial crisis and the aging population.

5.3 Answers to Research Questions

Q1. *What constructs were used?* All constructs except the *conflict* relation were used by the architects in this case. The conflict relation was not used because the architects stated that there were no conflicting goals in this case. There is not much we can conclude from this: Surely there are some cases where there are no conflicting goals, and we believe this is one of them; but there are other cases where there *are* conflicting goals. At the very least we can conclude that the idea of conflicting goals (goals that cannot always be all satisfied at the same time) was understood by the architects.

Q2. *Why (for which purpose) do they use these concepts and relations?*

Q3. *Is this the intended use of the constructs?* The constructs of *stakeholder, goal, decomposition* and *contribution* were used as intended. The concept of *requirement* was not used as intended, but rather was used as if it were the same concept as that of a goal. That is, requirements were not always allocated to one architecture component.

The *means-end* relationship was used as intended, namely as relation from requirement to goal, because the tool did not allow any other use. The architects did not see a relevant difference with the contribution relation.

Finally, the concepts of *concern* and *assessment* were not understood by the architects.

Q4. *Which potentially useful constructs do architects miss in ARMOR?* The architects found it useful to express contextual reasons for a goal, and used the *assessment* construct to do this.

5.4 Validity

Our observations may have been influenced by the fact that the first author also designed the language; this may have impacted the training positively (exceptionally inspiring explanations) or negatively (too much knowledge taken for granted). It may also have motivated the architects to have a socially desirable opinion about ARMOR. However, the architects had to do a real-world project with limited resources and as they are paying for this consultancy in money, and spending time on using ARMOR, they have no reason to present their experiences more favorably to the designer of ARMOR than they are.

Also, the observer (Engelsman) may have let his desire to design a usable and useful language influence his observations. This may have impacted the observations where architects where observed to use the ARMOR constructs as intended, but not the observations where the architects were observed to misunderstand the constructs of ARMOR. We regard at least those latter observations as credible.

Finally, could we generalize from this case to other cases? Generalization from case studies cannot use statistical inference but can use reasoning by analogy [6,17]. This means that we should explain our observations in terms of some general characteristics of the case, and provide a plausible argument that in cases with the same general characteristics, the same observations will be made.

Our observations all relate to understandability, and this relates to the cognitive competencies of the enterprise architects in Organization 1. The architects in Organization 1 had to be able to design and understand a distributed enterprise architecture for an organization of 3000 employees. Each of them had at least 2 years of experience as enterprise architect, and the organization operated its EA process at a maturity level comparable with level 2 of the US Department of Comments Architecture Capability Maturity Model[4]. All of this may explain why they used the constructs of stakeholder, goal, decomposition and contribution as intended, and we expect that in other organizations, similar to Organization 1 in the aspects just mentioned, architects will understand and use these constructs as intended too. But we also expect that in many of those organizations, the constructs of hard and soft goal, requirement (as defined in ARMOR), concern and assessment will not be understood and be used in a way not intended by the designers of ARMOR, that the means-end relation will be considered superfluous and that negative contribution will not be distinguished from conflicts. This generalization is a hypothesis that must be validated in replications of this case study. We do not claim that it will be found to be true for all future case studies. However we do expect to encounter in the future cases similar to this one. This was a sufficiently strong reason for us to redesign the language.

[4] `http://ocio.os.doc.gov/ITPolicyandPrograms/Enterprise_Architecture/PROD01_004935`

6 Redesign

Figure 4 shows the metamodel of a stripped down version of ARMOR that we call Light ARMOR. We dropped the constructs of concern, assessment, hard and soft goal and means-end from the language as these were not understood, or the relevance not understood, by the architects. To facilitate recording contextual reasons for a goal (the construct missed by the architects in Organization 1), the *Goal* construct was extended with a text attribute in which this reason could be recorded in free text.

The construct of *Contribution* was replaced by that of *Influence* so that we can avoid the locution "negative contribution", which we ourselves find as confusing as the concept of negative income. A goal G1 *influences* another goal G2 if satisfaction of G1 has an effect on the satisfaction of G2. So influence is a causal relation.

We did keep the notion of *Conflict* as the inability to satisfy two goals simultaneously can be a case of causal prevention ("negative contribution") but it may also be a case of logical inconsistency, legal exclusion, ethical incompatibility, or plain monetary conflict (satisfying the goals jointly exceeds the budget). The concept of conflict is complex and awaits future exploration; but we find it too important to drop from the language just because it has not been used in one case.

Finally, requirements are a special case of goals, just as before, but we dropped the idea that we require a separate modeling concept for it. A requirement is just a goal assigned to a component of the architecture.

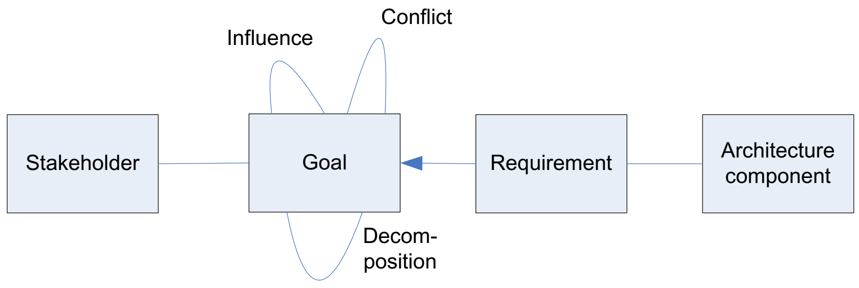

Fig. 4. Meta-model of Light ARMOR

7 Case Study 2

In addition to learning about the understandability of Light ARMOR, we would now like to learn about the utility of the language. Did our drastic reduction in the number of constructs impact the ability of enterprise architects to use the language (and supporting tool) to trace business goals to architecture components and vice versa?

The best way to find an answer to this question is to have enterprise architects use Light ARMOR to model the goals of an enterprise architecture, and then actually let them do the backward and forward tracing. This turned out not to be possible on short notice, and so we chose another form of action research, namely one in which the researchers themselves use their technique to solve a customer problem. In case study 2, the first author used Light ARMOR to solve an organizational problem following the engineering cycle of figure 2 and then used this experience to answer some validation questions about the design of Light ARMOR (figure 1). The research questions of case study 2 are, then:

- Q1 Is Light ARMOR understandable to architects?
- Q2 Can Light ARMOR be used to trace back and forth between business goals and enterprise architecture components?

7.1 Case Description and Research Design

The case company, called Organization 2 henceforth, is at a drinking water production facility in the Netherlands. The company is responsible for the production and delivery of fresh drinking water to 1.2 million people and transports 73 billion liters of drinking water each year. It has about 500 employees divided over three divisions, viz. Production, Sales and Environment.

Enterprise-architects and information analysts in Organization 2 are facing rapid change and shrinking budgets and are looking for a technique that will enable them to assess the impact of changing business goals (forward tracing) and to determine the value of the architecture (backward tracing). We were given the opportunity to use Light ARMOR to link business goals to their current enterprise architecture model in a no-fee small consultancy project. This would allow them to see if they would want to use this technique in the future, and gave us the opportunity to perform a first test of Light ARMOR.

We planned and performed the following interactions with Organization 2. The first author interviewed the architect responsible for the EA of Organization 1, and studied primary documents documenting the EA and business goals. He designed a Light ARMOR model of the links with the two, and then interviewed the enterprise architect a second time, asking her, without providing training in Light ARMOR, (1) to explain the Light ARMOR model and (2) to assess whether she could use this model to solve her traceability problem. This provided the enterprise architect with sufficient information to conclude her problem solving cycle (figure 2) and provided the researcher with information to find initial answers to his validation questions (validation 2 in figure 1). The researcher kept a diary of his own modelling process and made a transcript of the interview to be able to answer his own research questions.We emphasize that in this case we interacted with only one enterprise architect of the organization.

7.2 Observations and Explanations

- The major observation recorded in the researcher's diary is that it was often difficult to identify the stakeholders responsible for the goals from the primary documents or from the first interview with the enterprise architect. There are several possible explanations of this, such as that there is so much agreement about goals in Organization 2 that there is no need to record the goal owner; or that there is so much disagreement among the stakeholders that it is too dangerous to record a goal owner.

- The influence relation in this case is truly a causal relationship; including it in a model is an empirical statement that must be true about the world. For example, the goal to perform water filtering influences the goal to achieve clean drinking water. A second example is that the goal to achieve lower operating cost is influenced by the goal to achieve economics of scale with collaborative buying. Like all empirical statements, these influence statements could turn out to be falsified by events in the real world.
- The decomposition relation by contrast is not empirical, but definitional. It was used to create a definition of a term that the stakeholders agreed on. It only expresses an agreement between those stakeholders and not necessarily between other stakeholders. For example, the goal to achieve excellent drinking water quality was decomposed into the goals of sufficient pressure, safe drinking water, odorless drinking water and visually clean drinking water. This is a definition that turns a soft goal into a hard goal.
- The architect judged that Light ARMOR could be used to link business goals to architecture components to realize forward traceability (assessing impact of goal change) and backward traceability (justifying an architecture component). She suggested that this would also be useful to link project goals to business goals, providing a way to scope projects.
- In the opinion of the architect, the conflict relation would be useful in the assessment of project risks. This would however also require a way to document the resolution of these risks.For example record that one of the goals was dropped or that an other way was found to resolve the conflict.
- To test understandability of Light ARMOR we asked the architect to explain the model to us. The architect did not have prior training on GORE or Light ARMOR, but she could readily identify what the models meant.

7.3 Answers to Research Questions

The last observation provides support for the claim that Light ARMOR is understandable for practicing enterprise architects, which answers Q1 for this case.

The positive opinion of the architect about forward and backward traceability provides support for the claim of utility of Light ARMOR, answering Q2. In addition to the use for (1) estimating impact of change and (2) justifying the presence of an architecture component, the enterprise architect suggested using the model for (3) setting project goals and (4) documenting project risks and their mitigation. We will include these possible uses of Light ARMOR in our future research.

7.4 Validity

The major threat to internal validity is that the architect answered our questions in a socially desirable way. There is in this case nothing we can do to mitigate these risks, but in this case too we note that Organization 2 is looking for a way to exercise tighter control over its enterprise architecture in order to respond to changes in goals and a decreasing budget, and, doing so, has little reason to please the researchers. A *negative* response of the architect would have been really informative (and disastrous for the designers of Light ARMOR); the positive response that we actually received is less informative but is still encouraging.

The observations in this case make it plausible that if we were to repeat such a project in a similar organization (similar size, maturity of EA, experience of enterprise architect, dynamics of changing goals and shrinking budgets), we are likely to get similar results (positive opinion of the architect). This is a hypothesis to be tested in future case studies.

8 Lessons Learned and Further Work

In line with the evaluations reported in related work (section 2), we found that GORE concepts such as means-end relations and the distinction between hard and soft goals could not be used in our two case studies; and the concepts of concern and assessment taken from BMM and TOGAF could not be used either in our two cases. Also, the idea that a requirement exists as a separate modeling concept puzzled the practitioners in case 1. They had difficulty distinguishing between the two.

Stripping these elements away and including the results from case study 2, we conclude that our case studies provide support to the claim that the GORE concepts of *stakeholder, goal, decomposition, influence* and *conflict* are usable in practice and potentially useful for the practitioner. The particular syntax of the language that we used in our case studies did not play a role in these evaluations.

A third lesson we draw from these two case studies is that a stripped down language adding only these elements to an EA language can be useful for maintaining traceability between business goals and enterprise architecture. This is a hypothesis to be tested and possibly further qualified in future case studies.

A fourth and final lesson is that the conflict relation can be confused with the negative contribution relation, but still can be useful to keep because it allows representing project risks and their mitigation. This final hypothesis will be a topic of future case studies.

References

1. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
2. Business Motivation Model: Business motivation model version 1.0. Standard document (2007), `http://www.omg.org/spec/BMM/1.0/PDF` (22.09. 2009)

3. Carvallo, J.P., Franch, X.: On the use of i* for architecting hybrid systems: A method and an evaluation report. In: The Practice of Enterprise Modeling, pp. 38–53 (2009)
4. Clements, P., Bass, L.: Using Business Goals to Inform a Software Architecture. In: 18th IEEE International Requirements Engineering Conference, pp. 69–78. IEEE Computer Society Press (2010)
5. Engelsman, W., Quartel, D.A.C., Jonkers, H., van Sinderen, M.J.: Extending enterprise architecture modelling with business goals and requirements. Enterprise Information Systems 5(1), 9–36 (2011)
6. Forrester, J.: If *p*, then what? thinking in cases. History of the Human Sciences 9(3) (1996)
7. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information system research. MIS Quarterly 28(1), 75–105 (2004)
8. Horkoff, J., Yu, E.: Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences. In: Persson, A., Stirna, J. (eds.) PoEM 2009. LNBIP, vol. 39, pp. 145–160. Springer, Heidelberg (2009)
9. Jureta, I., Faulkner, S.: An Agent-Oriented Meta-model for Enterprise Modelling. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 151–161. Springer, Heidelberg (2005)
10. Lamsweerde, A.: Kaos tutorial. Cediti, September 5 (2003)
11. Matulevičius, R., Heymans, P.: Comparing Goal Modelling Languages: An Experiment. In: Sawyer, P., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 18–32. Springer, Heidelberg (2007)
12. Matulevičius, R., Heymans, P., Opdahl, A.: Comparing grl and kaos using the ueml approach. In: Enterprise Interoperability II, pp. 77–88 (2007)
13. Moody, D.: The physics of notations: Improving the usability and communicability of visual notations in requirements engineering. In: 2009 Fourth International Workshop on Requirements Engineering Visualization (REV), pp. 56–57 (September 2009)
14. Moody, D., Heymans, P., Matulevicius, R.: Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax. In: 17th IEEE International Requirements Engineering Conference, RE 2009, pp. 171–180. IEEE Computer Society Press (2009)
15. Quartel, D.A.C., Engelsman, W., Jonkers, H., van Sinderen, M.J.: A goal-oriented requirements modelling language for enterprise architecture. In: Proceedings of the Thirteenth IEEE International EDOC Enterprise Computing Conference, EDOC 2009, Auckland, New Zealand, pp. 3–13. IEEE Computer Society Press, Los Alamitos (2009)
16. Stirna, J., Persson, A., Sandkuhl, K.: Participative Enterprise Modeling: Experiences and Recommendations. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 546–560. Springer, Heidelberg (2007)
17. Sunstein, C.R.: On analogical reasoning. Harvard Law Review 106, 741–790 (1993)
18. The Open Group: ArchiMate 1.0 Specification. Van Haren Publishing (2009)
19. The Open Group: TOGAF Version 9. Van Haren Publishing (2009)
20. Wieringa, R.J.: Design science as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, Philadelphia, pp. 1–12. ACM, New York (2009)
21. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, pp. 226–235. IEEE Computer Society Press (2002)