RationalGRL: A Framework for Rationalizing Goal Models Using Argument Diagrams

Marc van Zee[1], Diana Marosin[2], Floris Bex[3], and Sepideh Ghanavati[4]

[1] Computer Science and Communication, University of Luxembourg, Luxembourg
[2] Luxembourg Institute of Science and Technology, Luxembourg
[3] Information and Computing Sciences, Utrecht University, The Netherlands
[4] ICIS, Radboud University, The Netherlands

Abstract. Goal modeling languages, such as *i** and the Goal-oriented Requirements Language (GRL), capture and analyze high-level goals and their relationships with lower level goals and tasks. However, in such models, the rationalization behind these goals and tasks and the selection of alternatives are usually left implicit. Rationalization consists of arguments for and against certain goals and solutions, which allow checking whether a particular goal model is a correct rendering of the relevant stakeholders' opinions and discussions. To better integrate goal models and their rationalization, we develop the RationalGRL framework, in which argument diagrams can be mapped to goal models. Moreover, we integrate the result of the evaluation of arguments and their counter-arguments with GRL initial satisfaction values. We develop an interface between the argument web tools OVA and TOAST and the Eclipse-based tool for GRL called jUCMNav. We demonstrate our methodology with a case study from the Schiphol Group.

1 Introduction

Goal modeling is an important part of all development processes, particularly in the area of complex reactive and distributed systems [17]. The Goal-oriented Requirements Language (GRL) is part of the User Requirements Notation (URN), which is an ITU-T standard [10]. GRL [1] consists of several intentional elements (such as softgoals, goals, tasks and resources) and links between them. It aims at modeling high-level business and system goals, subgoals and tasks and analyzing the alternative ways of achieving these goals and subgoals. GRL also includes an element called *belief*, which helps capturing the rationales behind softgoals and goals. While GRL models, and goal models in general [20], are useful tools for motivating architectural choices in enterprise and software architecture, they miss important parts of the architecture design rationalization. Although, it is possible to rationalize GRL models using *belief* elements, these beliefs are single and static statements, which do not capture the dynamic goal modeling process, nor do they capture the discussion process between stakeholders in which high-level softgoals are translated into goals, which are in turn refined into tasks [17]. As a result, GRL models are only the end product of a modeling process, and they do not provide any insight on how the models were created, i.e., what reasons were used to choose certain elements in the model and to reject the others and what evidence was given as the basis of this reasoning. There are, thus, several questions that are not answered in GRL: Why is a goal created? Why are some goals evaluated positively and some negatively? Do we have any evidence for the fact that performing a certain task contributes to a goal?

To address these questions, much work has been done to integrate the design rationales and formal argumentation with goal models [8, 7, 6, 11]. In past, we also developed a formal logical argumentation framework for reasoning with arguments and evidence, and we provided a metamodel that maps elements of the argumentation framework to the intentional elements (IEs) in GRL [19, 18].

In this paper, we integrate various existing and newly developed interfaces and algorithms into the RationalGRL framework. This framework facilitates argument construction and analysis on the one hand and the rationalization and evaluation of goal models on the other hand. More specifically, RationalGRL framework combines an existing argument diagramming tool[5] [4] based on a formal theory of practical (i.e. goal-driven) argumentation [2, 12, 18] with a standardized goal modeling language and its tool support [1].

The core of RationalGRL is a concrete set of mapping rules from the formal argumentation framework to a GRL model. The mapping rules allow for the automatic translation of arguments and evidence about goals to GRL models. Furthermore, the formal semantics [5] of arguments and counterarguments underlying the argumentation theory helps determining whether the elements of a GRL model are acceptable given the potential contradictory evidence and stakeholders' opinions. In other word, we can compute the initial satisfaction level of IEs in GRL based on the acceptability status of arguments for or against IEs. RationalGRL framework is implemented as an online tool [6]. We also validate RationalGRL with a real-world case study.

Following our research approach and goals, the rest of this paper is structured as follows. In Section 3, we introduce the RationalGRL framework and its mapping rules and algorithms. In Section 4, we evaluate the framework via a case study of the Schiphol Group. First, we model the discussions about a change in the set of architecture principles of the Schiphol Group. Second, we evaluate the framework and the resulted models with enterprise architects of the Group. In Section 5 we present the current literature and the related work. We conclude and present lines for future work in Section 6.

2 Case Study Description: Schiphol Group

Schiphol Group is the owner of an international airport and it operates on both national and international scales. Schiphol Group started using enterprise architecture principles to drive their architecture program in 2003. Principles are

[5] `http://ova.arg-tech.org/`

[6] All implementation details/sources as well as the case study descriptions and models can be found on Github: `http://github.com/RationalArchitecture/RationalGRL`.

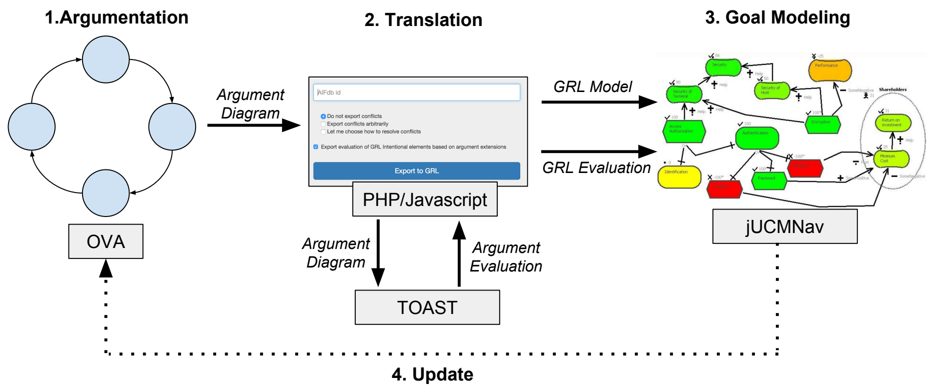
generally defined as "a family of guidelines (...) for design" [9] or "general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way an organization fulfils its mission" [16]. In 2003, the principle Adhere to the Corporate Data Model was advocating the use of a company-wide defined data model, such that it provided a high level insight on all the data that were used in the processes and applications. In 2007, the principles were evaluated by a team of five architects and it was concluded that the principle was not very successful, and was conceptually conflicting with another architecture principle Package selection before custom development.

In this paper, we use these principles as the base of our case study and we provide *a posteriori* analysis of the discussions and evidence that were used in forming the architecture principles and present the goal models generated using the RationalGRL framework. In total, we formalized around 60 arguments and 30 inferences/attack relation. Due to space constraints, we only provide a small subset of the models in this paper.

In Section 3, we introduce the RationalGRL and use the case study to illustrate different parts of the framework. We, further, model the case study in more detail and we evaluate the correctness and feasibility of RationalGRL in Section 4.

3 The RationalGRL Framework

The main components of the RationalGRL framework are shown in Figure 1. The four main parts of the framework, Argumentation, Translation, Goal Modeling, and Update, are numbered and depicted in **bold**. For each component, the technology used to implement it, is marked in a filled rectangle. The last step (*Update*) is out of the scope of this paper. We, now, briefly explain the process of how a goal model is developed in the RationalGRL framework. We discuss the details of the steps in the following subsections.



(● Do not export conflicts) Export conflicts arbitrarily / Let me choose how to resolve conflicts

✓ Export evaluation of GRL Intentional elements based on argument extensions

Fig. 1: Overview of the RationalGRL Framework

In *Step 1 - Argumentation*, stakeholders discuss the requirements of their organization. In this process, stakeholders put forward arguments for or against certain elements of the model (e.g. goals, tasks,..). Arguments about why certain tasks can contribute to the fulfillment of goals and an evidence to support a claim are also part of this process. Furthermore, stakeholders can challenge claims by forming counterarguments. The complete set of claims, arguments and counterarguments can be represented in an argument diagram. We present a formal representation of argument diagrams in Section 3.1.

In *Step 2 - Translation*, the argument diagram is translated to a goal model, in our case GRL. In addition to the structure of arguments and counterarguments, this step also provides means to translate the evaluation of arguments in the argument diagrams to the initial satisfaction values of the GRL intentional elements, which can be positive or negative. A positive evaluation indicates that the element is supported by one or more arguments, while a negative evaluation indicates that the element is not a good alternative in GRL.

In *Step 3 - Goal Modeling*, the goal model that is generated by the Translation process, is evaluated by the stakeholders. These models can be used as a discussion means to investigate whether the goals in the model are in line with the original requirements of the stakeholders. This allows a better rationalization of the goal modeling process, with a clear traceability from the goals of the organization to the arguments and evidence that were used in the discussions.

Step 4 - Update involves translating GRL models with its analysis back into an argument diagram. This falls outside the scope of the current paper.

3.1 Step 1 - Argumentation

In requirements engineering and design rationale, people often discuss multiple design alternatives and the goals and principles related to these alternatives. Such reasoning about which goals to pursue and which actions to take is commonly referred to as *practical reasoning*, and has been studied extensively in formal argumentation, most notably by Atkinson *et al.* [2], who define the following basic argument structure for reasoning from goals to actions.

$$\begin{array}{l} \text{I have goal } G \\ \text{Doing actions } A \text{ will contribute to goal } G \\ \textit{Therefore} \text{ I should do actions } A. \end{array} \tag{1}$$

This basic argument can be further extended to capture subgoals (*i.e.*, realizing goals $G_1, \ldots, G_n$ will allow me to realize goal G_i). Like all argumentation, practical reasoning is defeasible, in that conclusions which are at one point acceptable can later be rejected because of new information or claims. For example, there might be unwanted side effects to performing action A, or there may be alternative actions that also lead to the realization of one's goals.

In previous work [18], we argued that the dynamic discussions about the exact structure and evaluation of a goal model can be captured using practical reasoning, and we developed a formal logical argumentation language for such reasoning based on the $ASPIC^+$ framework [12]. The framework allows us to define our own inference rules for practical reasoning inspired by [2]. The language of RationalGRL for representing the elements of argumentation is directly derived from the logical language proposed in [18]. Here, we present the logical language as a BNF grammar:

$$\begin{aligned} &p ::= [e]s \mid [e]s \rightarrow [g]s \mid [g]s \mid [g]s \rightarrow [g]s' \mid [g]s \rightarrow [NOT][g]s' \\ &e ::= EVIDENCE \\ &g ::= SOFTGOAL \mid GOAL \mid TASK \\ &o ::= e \mid g \end{aligned}$$

Here, p denotes the claims in the argument, s is a sentence in natural language and e, g and o denote the (modal) operators on sentences in the language. The first clause of the BNF grammar states that an argument can have five different forms, and the remaining clauses indicate for which modal operator the variables e, g, and o can be substituted in the first clause. The $\rightarrow$ symbol means "contributes to" as used in practical reasoning argument (1).

The types of inferences and conflicts in our argumentation language are shown in Table 1, rules 1-3. The first rule corresponds to the practical reasoning inference (1) discussed earlier. For instance, from the premise $[SOFTGOAL]$ increase profit and the premise $[TASK]$ use open source software $\rightarrow$ $[SOFTGOAL]$ increase profit, we can conclude (using practical reasoning) $[TASK]$ use open source software. The second and third inference are defeasible forms of the classic modus ponens rule – note that evidence can only be used in the premises of a rule. Rule 4-6 can be used to define which elements are in conflict. The first two rules represent the symmetrical conflict between a claim and its negation. The third rule is an asymmetrical conflict. Note that evidence cannot be attacked.

Table 1: Arguments inference rules and conflicts

Rule	Argument Type	Premise 1	Premise 2	Conclusion
1.	Practical reasoning	$[g]s$	$[g]s' \rightarrow [g]s$	$[g]s'$
2.	Default inference	$[o]s$	$[o]s \rightarrow [o]s'$	$[g]s'$
3.	Default inference	$[o]s$	$[o]s \rightarrow [NOT][g]s'$	$[NOT][g]s'$
4.	Default conflict	$[g]s$		$[NOT][g]s$
5.	Default conflict	$[NOT][g]s$		$[g]s$
6.	Default conflict	$[o]s$		$[g]s'$

Arguments can be rendered as graphs using OVA[7], a web-based tool that supports constructing and storing of argument diagrams in the Argument Web [4][8]. Since it supports all inference and conflict relations that we propose in Table 1, we can directly implement the formal argumentation framework in OVA, simply by annotating arguments with the correct modality.

[7] http://ova.arg-tech.org/

[8] http://aifdb.org/

As an example of an argument in the RationalGRL framework, consider Figure 2. There is an argument for the goal Lower diversity in applications, which is constructed using practical reasoning from two premises: [SOFTGOAL] Lower cost and [GOAL] Lower diversity in applications → [SOFTGOAL] Lower cost. There is also an argument against lowering diversity in applications based on a default inference. Since these two arguments have conflicting conclusion, they attack each other (see Table 1). Moreover, the argument for the goal Use open source solutions is attacked by the evidence that these solutions are insecure.

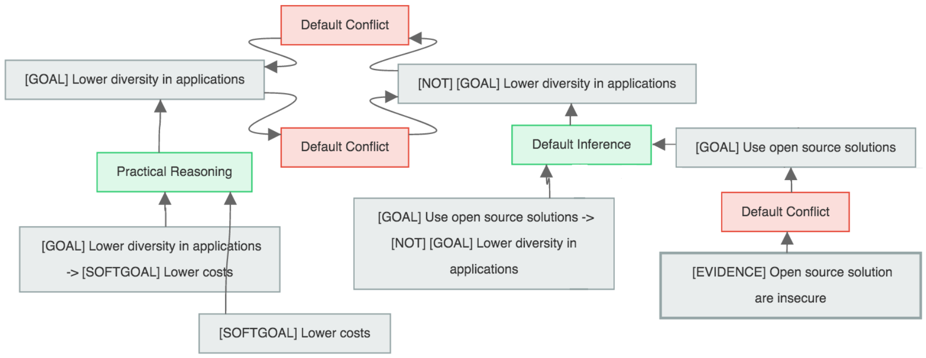

Default Conflict
[GOAL] Lower diversity in applications
[NOT] [GOAL] Lower diversity in applications
Default Conflict
Practical Reasoning
Default Inference
[GOAL] Use open source solutions
Default Conflict
[GOAL] Lower diversity in applications -> [SOFTGOAL] Lower costs
[GOAL] Use open source solutions -> [NOT] [GOAL] Lower diversity in applications
[EVIDENCE] Open source solution are insecure
[SOFTGOAL] Lower costs

Fig. 2: Example RationalGRL argument diagram

ASPIC$^+$, on which the RationalGRL argumentation language is based, is an instantiation of Dung's [5] abstract framework. Therefore, we can use the standard argumentation semantics to compute whether we are justified in accepting certain goals given the stakeholder's arguments. In other words, given a set of arguments and their attack relations, as presented in Figure 2, we can determine which of the arguments are acceptable and which are defeated by other arguments. Informally, all arguments that are not attacked by any argument are *IN* (colored green), arguments that are attacked by an argument that is *IN* are *OUT* (colored red), and arguments that are only attacked by arguments that are *OUT* are *IN* as well. Otherwise, an argument is *UNDECIDED* (colored grey).

In our example, the argument [EVIDENCE] Open source solutions are insecure is *IN*, which makes the argument for [NOT][GOAL] Lower diversity in applications *OUT*, which then makes the argument for [GOAL] Lower diversity in applications *IN*. An evaluation of the arguments is presented in Figure 3.

In summary, this subsection described how we can intuitively represent and visualize a formal argumentation framework using a browser-based tool. Moreover, we briefly explained how to compute the acceptability of arguments. These two notions are used in Step 2 - Translation to generate GRL models and to support the initial satisfaction value of GRL IEs.

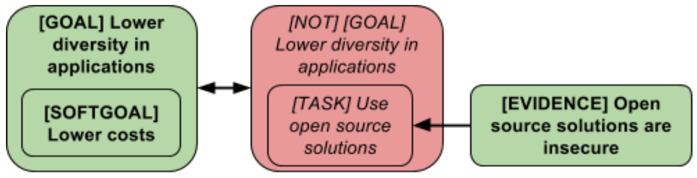

[GOAL] Lower diversity in applications
[SOFTGOAL] Lower costs
[NOT] [GOAL] Lower diversity in applications
[TASK] Use open source solutions
[EVIDENCE] Open source solutions are insecure

Fig. 3: Acceptability status of main arguments in Figure 2.

3.2 Step 2 - Translation

The Translation process translates the argument diagrams that we introduced in the previous subsection (see Figure 2 and 3) into GRL models with initial evaluations of the IEs. The process consists of two translations/mappings: The first mapping generates the intentional elements and their relationships of the GRL model from the argument diagrams, and the second mapping generates the satisfaction values of the GRL IEs from the acceptability status of their underlying arguments. The translation of the argument diagram of Figure 2 is shown in Figure 4.

We assume familiarity with the basics of goal modeling and GRL, and we refer to Amyot [1] for a more detailed overview. The task(⬡) Use open source solutions contributes negatively to the goal (⬭) Lower diversity in applications, which in turn contributes positively to the softgoal (⬭) Lower costs. The belief (⬯) Open source solutions are insecure is linked to the task. A GRL model is evaluated by assigning initial satisfaction values to a subset of its IEs, and by propagating these values to the remaining intentional elements of the model through the links that connect them [1].

The Translation process assigns evaluation values to GRL intentional elements using the evaluation values of the arguments that are IN or OUT. For arguments that are $UNDECIDED$, GRL computes the evaluation values using the bottom-up propagation algorithms.

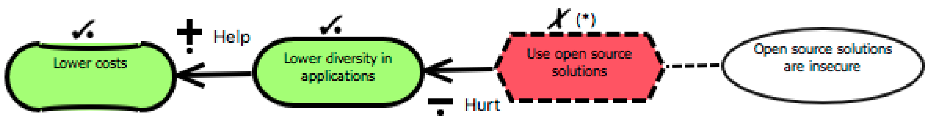

Lower costs
+ Help
Lower diversity in applications
(*)
Use open source solutions
Hurt
Open source solutions are insecure

Fig. 4: Generated GRL Model from argument diagram of Figure 2

Mapping 1: Argument Diagrams to GRL Models. The rules for mapping argument diagrams to GRL models are presented in Table 2. Rule 1-3 are direct mappings. Rule 4 translates an evidence from the argument diagram to a belief element in the GRL model. The reason for this is that GRL does not contain elements to reason about evidence. In the argument diagram, evidence may be used to attack existing arguments (e.g., argument [EVIDENCE] Open source solutions are insecure of Figure 2). In that case, since the evidence is not translated to the GRL model, we connect it as a Belief link to the element it attacks (e.g., belief Open source solutions are insecure in in Figure 4). Rule 5 and 6 directly translate the meaning of $\rightarrow$ (contributes to) to the GRL model. For instance, the argument [GOAL] Lower diversity in applications $\rightarrow$ [SOFTGOAL] Lower costs of Figure 2 is translated into a positive contribution between the two elements in Figure 4, and the argument [GOAL] Use open source solutions $\rightarrow$ [NOT][GOAL] Lower diversity in applications is translated into a negative contribution. Rule 7 adds negative contributions between conflicting elements (except evidence, since they are not part of GRL). Since the negative contributions for the conflicts in Figure 2 are represented already by the previous rules, this rule has no effect in our example. Finally, rule 8 adds a belief link between the generated belief elements of rule 4 and the element it attacks.

Table 2: Mapping rules from argument diagrams to GRL models

Rule	Argument Type in the Argument Diagrams	Corespondent in GRL models
1.	[TASK] T	Add Task T
2.	[GOAL] G	Add Goal G
3.	[SOFTGOAL] G	Add Softgoal G
4.	[EVIDENCE] E	Add Belief statement E
5.	a $\rightarrow$ b	Include positive contribution a to b
6.	a $\rightarrow$ [NOT] b	Include negative contribution a to b
7.	Conflict between a and b a,b of type (SOFT)GOAL/TASK)	Include negative contribution a to b
8.	Conflict between a and b a of type (EVIDENCE) b of type (SOFT)GOAL/TASK)	Include Belief Link from a to b

Mapping 2: Argument Evaluation to GRL Evaluation. Recall from the previous section that arguments in an argument diagram can be IN, OUT, or $UNDECIDED$. The mapping from argument evaluations to the GRL intentional elements evaluations is straightforward and is shown in Table 3. All arguments that have a corresponding intentional element in the GRL model receive the evaluation value corresponding to their argument evaluation. For instance, If an argument [GOAL] Lower costs is IN in the argument diagram, then it receives the satisfaction value Satisfied.

Implementation Details We implement the online Translation Tool[9] using PHP. The tool takes the following inputs:

– *AIFdb id.* Argument diagrams created in OVA (see Section 3.1) can be saved to the Argument Web and their identifier is called an AIFdb id.

[9] `http://www.marcvanzee.nl/RationalGRL/`

Table 3: Argument Evaluation to GRL Evaluation Mapping Rules

Acceptability status of argument A	GRL evaluation value of A (if exists in model)
A is IN	Satisfied
A is $UNDEC$	None
A is OUT	Denied

– *Export conflicts preferences.* Attacks between arguments in the argument diagrams can be bi-directional, but relations between GRL elements can only be one-directional. Therefore, we require conflicts to be resolved, which can be either done automatically, not at all, or manually. When the conflicts are done manually, the user has to decide for each conflict which of the supporting arguments is more important.
– *Export evaluations.* When this is not checked, the tool will only export the GRL models.

The translation tool requests a JSON object of the argument diagram from the Argument Web using the AIFdb id. In order to translate the arguments to GRL IEs and create GRL models, the tool parses the arguments and relations and then uses the mappings of Table 2. This is exported in XML format, which can directly be imported in jUCMNav[10].

To obtain the evaluations of the arguments, the tool requests another JSON object from TOAST, a tool for evaluating the Dung semantics[11]. TOAST also provides interface with the AIFdb, thus, obtaining this object is straightforward. The JSON object contains the evaluation of the arguments, and the tool uses this to set the evaluation of the GRL IEs using the mapping rules of Table 3.

3.3 Step 3 - Goal Modeling

Once the GRL models have been generated from the argument diagram, it can be visualized, reviewed and can provide further discussions on whether it is in line with the initial requirements. By analyzing the GRL models, it is possible to detect inconsistencies or identify alternatives that should be reconsidered. For instance, the GRL model of Figure 4 shows that even though a task is not possible, the goal and softgoal are still satisfied. This additional information can be used as an input in the discussion and support the introduction of different measures to satisfy the initial goals.

[10] Note that in order to import these GRL models correctly, one has to install Graphviz Dot (`http://wwww.graphviz.org/content/dot-language`) and provide the path to it in the AutoLayout Preferences of the jUCMNav Preferences. In this case, the layout of the GRL model will be generated automatically.

[11] http://toast.arg-tech.org/

4 Evaluation

In this section, we formalize the case study in RationalGRL and we evaluate the models with enterprise architects from Schiphol Group in order to verify whether our models correctly capture the situation at hand, and whether our methodology can improve the current practice in enterprise architecture and can provide additional insights and understandings for architects.

4.1 Modeling the Case Study in RationalGRL

In 2003, the principle Adhere to the Corporate Data Model was advocating for the use of a company-wide defined data model, such that it provided a high level insight on all the data that is used in the processes and applications. The IT department was assigned the task to define this data model. All applications were supposed to be compatible to this data model. The main motivation for adopting this principle was to obtain a clear and standard approach for information handling. This could improve the way in which customers can be served and to lower the costs.

In 2007, the principles were re-evaluated by a team of five architects. It was concluded that the previous principle was not very successful. Some of the arguments used in this discussion are shown in Figure 5. An important issue with the corporate data model - principle was the effort needed to be invested by the ICT department to define this data model. Schiphol Group not only focuses on aviation, but also on retail and security. These domains have different needs when it comes to the data they use and their internal processes. In terms of business objects and "on paper" definitions, the data model was agreed upon, but it was never really implemented.

This situation is reflected in a simplified way in the argument diagram by an attack from the argument [EVIDENCE] CorpDM is not defined on [TASK] ICT department defines CorpDM. Two of the other attacks are direct consequences of this issue. Since there was no corporate data model, the principle could not be used ([EVIDENCE] Principle has been used minimally... use it now), and databases between applications were seldom shared ([EVIDENCE] Databases seldom shared).

In addition, the principle was conceptually conflicting with another architecture principle Package selection before custom development. It was virtually impossible to find third-party applications and vendor packages that comply with the corporate data model. This is reflected in the argument diagram as a bi-directional conflict between the principle and [TASK] Use data models of packages applications instead of CorpDM. This task is a direct consequence of the principle Package selection before custom development.

This set of arguments and the evaluation of the real-life situation made architects realize that the focus should be on the exchange of information between applications, not on how the data is stored and managed centrally. This shift of paradigm resulted in creating a new principle Adhere to the canonical data model.

We translate the final argument diagram of the situation in 2007 after introducing the new principle using RationalGRL framework. The result is presented in

Figure 6. Based on the evaluation of GRL IEs, the new principle as well as goals such as Lower diversity and total cost of ownership, Clear and standard way of interfacing, Few dependencies between applications, and Faster time to market receive a positive evaluation. Moreover, the old principle receives a negative evaluation, together with its related goals and tasks. This conclusions can provide insights on how to prioritize from a set of principles, or how to take better informed design decisions when facing alternative solutions.

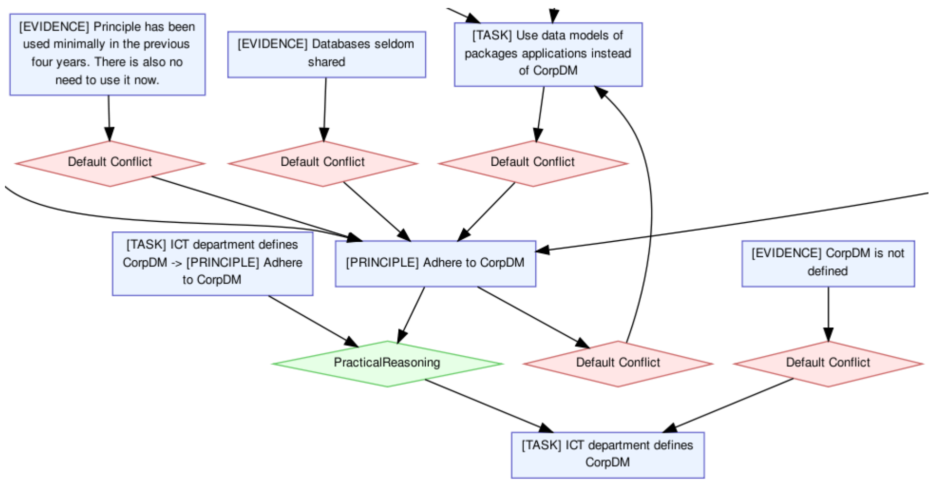

[EVIDENCE] Principle has been used minimally in the previous four years. There is also no need to use it now.
[EVIDENCE] Databases seldom shared
[TASK] Use data models of packages applications instead of CorpDM
Default Conflict
Default Conflict
Default Conflict
[TASK] ICT department defines CorpDM -> [PRINCIPLE] Adhere to CorpDM
[PRINCIPLE] Adhere to CorpDM
[EVIDENCE] CorpDM is not defined
PracticalReasoning
Default Conflict
Default Conflict
[TASK] ICT department defines CorpDM

Fig. 5: Part of the argument diagram of the Schiphol Group principles 2007 (Visualized in the Argument Web)

4.2 Evaluation with Schiphol Group Enterprise Architects

We evaluated our framework and its results with enterprise architects of the Schiphol Group. We first discussed the argument diagrams in order to evaluate whether they represent the situation at 2003 and 2007 correctly. The architects found that argument diagrams are a useful tool to link and reason about arguments. However, they noted that it may be easier to construct the arguments and counterarguments *a postiori* than to do this *a priori*. They felt that it is easier to look back on the process and to extract that relevant arguments, than to do this while the process is still ongoing.

Next, we translated the argument diagrams to GRL models using the translation procedure and evaluated these GRL models with the architects. The architects confirmed that the models are able to represent correctly part of the problem at hand. However, they also noted that some parts were missing from the models, which implies that beside the documents we gathered and used for modeling, there were additional facts we did not consider. However, this partial representation was found useful and the architects consider the usage of formal methods (such as GRL and argumentation) beneficial for "sanity checks", alongside a better formulation of the principles.

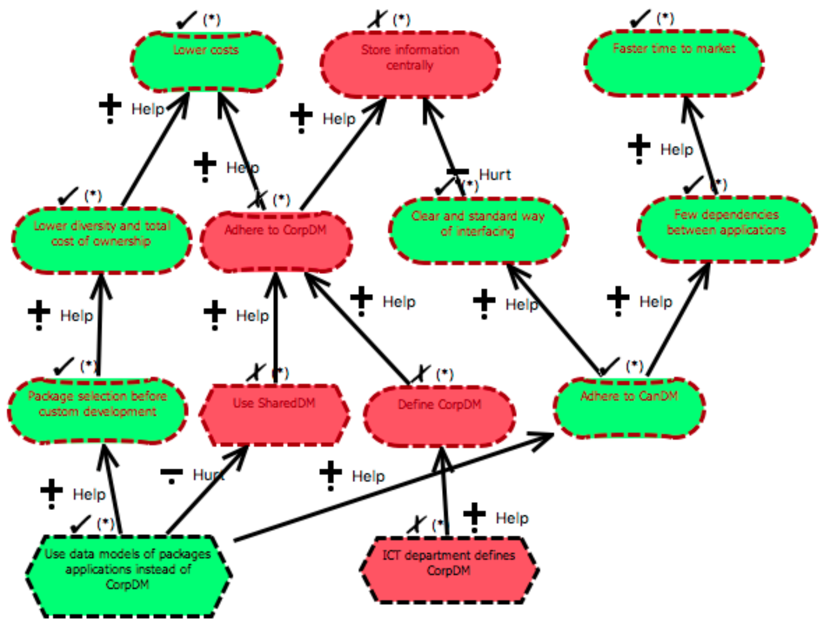

Lower costs
Store information centrally
Faster time to market
Help
Help
Help
Hurt
Help
Lower diversity and total cost of ownership
Adhere to CorpDM
Clear and standard way of interfacing
Few dependencies between applications
Help
Help
Help
Help
Help
Package selection before custom development
Use SharedDM
Define CorpDM
Adhere to CanDM
Help
Hurt
Help
Help
Use data models of packages applications instead of CorpDM
ICT department defines CorpDM

Fig. 6: GRL Diagram of the Schiphol Group principles in 2007

5 Related Work

There are several contributions that relate argumentation-based techniques with goal modeling. The contribution most closely related to ours is the work by Jureta *et al.* [11]. This work proposes "Goal Argumentation Method (GAM)" to guide argumentation and justification of modeling choices during the construction of goal models. One of the elements of GAM is the translation of formal argument models to goal models (similar to ours). In this sense, our RationalGRL framework can be seen as an instantiation and implementation of part of the GAM. One of the main contribution of RationalGRL is that it also takes the acceptability of arguments as determined by the argumentation semantics [5] into account when translating from arguments to goal models. RationalGRL also provides tool support for argumentation, i.e. Argument Web toolset, to which OVA belongs [4], and for goal modeling, i.e. jUCMNav [14]. Finally, RationalGRL is based on the practical reasoning approach of [3], which itself is also a specialization of Dung's [5] abstract approach to argumentation. Thus, the specific critical questions and counterarguments based on these critical question proposed by [3] can easily be incorporated into RationalGRL.

RationalGRL framework is also closely related to frameworks that aim to provide a design rationale (DR) [15], an explicit documentation of the reasons behind decisions made when designing a system or artefact. DR looks at issues, options and arguments for and against the various options in the design of, for example, a software system, and provides direct tool support for building and analyzing DR graphs. One of the main improvements of RationalGRL over DR approaches is that RationalGRL incorporates the formal semantics for both argument acceptability and goal satisfiability, which allow for a partly automated evaluation of goals and the rationales for these goals.

Arguments and requirements engineering approaches have been combined by, among others, Haley *et al.* [8], who use structured arguments to capture and validate the rationales for security requirements. However, they do not use goal models, and thus, there is no explicit trace from arguments to goals and tasks. Furthermore, like [11], the argumentative part of their work does not include formal semantics for determining the acceptability of arguments, and the proposed frameworks are not actually implemented. Murukannaiah *et al.* [13] propose Arg-ACH, an approach to capture inconsistencies between stakeholders' beliefs and goals, and resolve goal conflicts using argumentation techniques.

6 Conclusions and Future Work

In this paper, we developed and implemented a framework to trace back elements of GRL models to arguments and evidence that derived from the discussions between stakeholders. We created a mapping algorithm from a formal argumentation theory to a goal model, which allows us to compute the evaluation values of the GRL IEs based on the formal semantics of the argumentation theory.

There are many directions of future work. There are a large number of different semantics for formal argumentation, that lead to different arguments being acceptable or not. It would be very interesting to explore the effect of these semantics on goal models. Jureta *et al.* develop a methodology for clarification to address issues such as ambiguity, overgenerality, synonymy, and vagueness in arguments. Atkinson *et al.* [2] define a formal set of critical questions that point to typical ways in which a practical argument can be criticized. We believe that critical questions are the right way to implement Jureta's methodology, and our framework would benefit from it. In addition, currently, we have not considered the *Update* step of our framework (Figure 1). That is, the translation from goal models to argument diagrams is still missing. The *Update* step helps analysts change parts of the goal model and analyze its impact on the underlying argument diagram. Finally, the implementation is currently a browser-based mapping from an existing argument diagramming tool to an existing goal modeling tool. By adding an argumentation component to jUCMNav, the development of goal models can be improved significantly.

References

1. Daniel Amyot. Introduction to the user requirements notation: Learning by example. *Computer Networks*, 42(3):285–301, June 2003.

2. Katie Atkinson and Trevor Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10):855–874, 2007.
3. Katie Atkinson and Trevor Bench-Capon. Taking the long view: Looking ahead in practical reasoning. In *Computational Models of Argument: Proceedings of COMMA*, pages 109 – 120, 2014.
4. Floris Bex, John Lawrence, Mark Snaith, and Chris Reed. Implementing the argument web. *Communications of the ACM*, 56(10):66–73, 2013.
5. Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
6. Virginia N.L. Franqueira, Thein Tan Tun, Yijun Yu, Roel Wieringa, and Bashar Nuseibeh. Risk and argument: A risk-based argumentation method for practical security. In *19th IEEE International Requirements Engineering Conference*, USA, June 2011. IEEE.
7. Charles B Haley, Robin Laney, Jonathan D Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *Software Engineering, IEEE Transactions on*, 34(1):133–153, 2008.
8. Charles B Haley, Jonathan D Moffett, Robin Laney, and Bashar Nuseibeh. Arguing security: Validating security requirements using structured argumentation. In *in Proc. of the Third Symposium on RE for Information Security (SREIS'05)*, 2005.
9. Jan A. P. Hoogervorst. Enterprise architecture: Enabling integration, agility and change. *Int. J. Cooperative Inf. Syst.*, 13(3):213–233, 2004.
10. ITU-T. Recommendation Z.151 (11/08): User Requirements Notation (URN) – Language Definition. `http://www.itu.int/rec/T-REC-Z.151/en`, 2008.
11. I.J. Jureta, S. Faulkner, and P.Y. Schobbens. Clear justification of modeling decisions for goal-oriented requirements engineering. *Requirements Engineering*, 13(2):87–115, May 2008.
12. Sanjay Modgil and Henry Prakken. A general account of argumentation with preferences. *Artificial Intelligence*, 195:361–397, 2013.
13. Pradeep K Murukannaiah, Anup K Kalia, Pankaj R Telangy, and Munindar P Singh. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *23rd Int. Requirements Engineering Conf.*, pages 156–165. IEEE, 2015.
14. Gunter Mussbacher and Daniel Amyot. Goal and scenario modeling, analysis, and transformation with jUCMNav. In *ICSE Companion*, pages 431–432, 2009.
15. S. J. Buckingham Shum, A. M. Selvin, M. Sierhuis, J. Conklin, C. B. Haley, and B. Nuseibeh. Hypermedia support for argumentation-based rationale. In *Rationale management in software engineering*, pages 111–132. Springer, 2006.
16. The Open Group. TOGAF 9 - The Open Group Architecture Framework Version 9, 2009.
17. Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proc. 5th IEEE Int. Symposium on RE*, pages 249–262, 2001.
18. Marc van Zee, Floris Bex, and Sepideh Ghanavati. Rationalization of goal models in grl using formal argumentation. In *Proceedings of RE: Next! track at the Requirements Engineering Conference 2015 (RE'15)*, August 2015.
19. Marc van Zee and Sepideh Ghanavati. Capturing Evidence and Rationales with Requirements Engineering and Argumentation-Based Techniques. In *Proc. of the 26th Benelux Conf. on Artificial Intelligence (BNAIC2014)*, November 2014.
20. Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proc. of the 3rd IEEE Int. Symposium on RE*, pages 226–235, 1997.