Towards Meta Model Provenance: A Goal-Driven Approach to Document the Provenance of Meta Models

Sybren de Kinderen[1(✉)], Monika Kaczmarek-Heß[1], Qin Ma[2], and Iván S. Razo-Zapata[3]

[1] University of Duisburg-Essen, Essen, Germany
{sybren.dekinderen,monika.kaczmarek}@uni-due.de
[2] University of Luxembourg, Esch-sur-Alzette, Luxembourg
qin.ma@uni.lu
[3] Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg
ivan.razo-zapata@list.lu

Abstract. This paper introduces the notion of meta model provenance. Meta model provenance helps to understand the origins of meta model elements, such as language concepts, attributes, or constraints. Thus, it should answer questions such as: where did this language concept come from? under which assumptions was it introduced? Among others, meta model provenance is intended to support the controlled evolution of languages and informed language (re-)design. In this paper, we focus on a goal-driven meta model provenance approach. This is one specific operationalization of the meta model provenance concept, which shows how goal models help to understand the origins of the elements of a conceptual modeling language. To illustrate our goal-driven provenance approach, we use a scenario from the electricity domain.

Keywords: Meta model provenance · Modeling language design · Goal-driven meta model provenance

1 Introduction

With the proliferation of modeling languages and sharing of created models [1,2], questions regarding origins of a language specification emerge, e.g., where did this language concept come from? for what purpose it was specified, under what particular assumptions, and for what use scenarios? Existing approaches to design modeling methods postulate an informed language design process, within which, among others, use scenarios or requirements [3] are defined that subsequently drive the definition of the language. However the dissemination of modeling languages typically boils down to its abstract syntax, usually in the form of a meta model (i.e., a set of concepts, their attributes, relations and constraints), concrete syntax and semantics [4]. Thus, information regarding the

G. Poels et al. (Eds.): PoEM 2017, LNBIP 305, pp. 49–64, 2017.
https://doi.org/10.1007/978-3-319-70241-4_4

language's intension and domain constraints/requirements are either lost, or are captured in separate documentation that is not always available to an interested audience.

Such a language specification with lacking documentation can result in an ad hoc language (re-)design, expressing itself most prominently in an uncontrolled evolution of modeling languages. Consider the Enterprise Architecture modeling language ArchiMate [5]. For the core ArchiMate language, a set of requirements and assumptions were defined [6], such as conceptual parsimony (meaning, to keep the amount of concepts in the language to a minimum). However, these language requirements were never part of the core language specification. Largely as a result of this, later extensions to ArchiMate have been defined that ignore these language requirements. The resulting extensions are therefore not fully in line with the ideas behind the original language. In the case of the ArchiMate's motivation extension, this has resulted in concepts that can confuse language users and/or concepts that are obsolete [7]. For example, the motivation extension has both the concepts of a driver and a goal, whereas according to the principle of conceptual parsimony having the concept of a goal would arguably suffice.

In this paper, we focus on information regarding the origins of elements of language specification. We term this information as "provenance". The term provenance refers to the origin, history or pedigree of some artifact [8]. In the Information Systems (IS) domain, the concept of data provenance is widely used to denote origin or history of data [8–11]. Similarly, while not under the banner of provenance, the conceptual modeling community, particularly Thalheim [12], emphasizes analysis of conceptual models and modeling languages using a set of questions based on the classical rhetorical frame introduced by Hermagoras of Temnos[1]. Various modeling language design methods, such as [3,13–15], concretize a subset of these questions, focusing for example on stakeholder (who) and scenario (why) analyses. Yet, in the light of provenance as we find it in the IS discipline, both such questions and the language design approaches fall short. Particularly, in terms of how this information concretely relates to a language specification, in which form this information can be captured, and (related) what particular value this information as a language's provenance has.

Therefore, motivated by the shortcomings of the current state of documenting the information on origins of language specification, we propose a concept of a *meta model provenance* recording the intention standing behind the configuration of a language specification. To capture the origins of a modeling language, there is a need to define (1) the provenance concept that would account for peculiarities of language design as well as (2) corresponding methods that would help trace that provenance. Thus, we define the concept of meta model provenance considering the following aspects: (1) its scope, i.e., what it describes, (2) the way it should be represented and the representation's formality level, and finally, (3) the way it should be visualized and disseminated. We argue that capturing explicitly meta model provenance contributes to the *understandability* of the

[1] Who, what, when, where, why, in what way, by what means?

meta model and its current configuration. Thus, it supports *controlled language evolution*. Furthermore provenance contributes to *informed language (re-)design*, i.e., (re-)designing a language based on domain "rules", use scenarios, (prospective) language users, and purposes of language users.

In this paper we first sketch the core idea of the meta model provenance concept and show its overall design. Thereafter, we focus on goal-driven meta model provenance, whereby we use goal models to capture the intension of the elements of an abstract syntax. We use a case scenario from the electricity domain to illustrate the goal-driven approach.

This paper is structured as follows. First, we introduce different aspects covered by Domain-Specific Modeling Language (DSML) design methods, and subsequently discuss how these are reflected in different DSMLs. Next, we introduce the concept of meta model provenance and motivate its role in the design and application of DSMLs. Then, we introduce a goal-driven meta model provenance approach. Subsequently, the proposed approach is illustrated with a scenario from the electricity domain. The paper concludes with final remarks.

2 The Design of Domain-Specific Modeling Languages

Typically DSML design methods take domain characteristics as a point of departure, particularly in the form of the purposes of a DSML, prospective use scenarios, and domain constraints, cf., [3,13–15]. Further, they propose to document the design decisions along the way. This all leads to what we term"informed language design", meaning that the elements of a DSML are systematically reconstructed from the domain that they represent.

As a synthesis of different DSML design methods we observe that the following selected aspects inform the DSML design (note that, due to space considerations we cannot discuss each aspect in detail): (1) *Use scenarios*, either in terms of (a succession of) tasks to be supported by a DSML [3], or in the form of user stories [14]. (2) *Purposes*, which are similar to scenarios, but without the specific temporal ordering. Purposes are reflected in selected DSML design guidelines from [15], and in the requirements engineering approach from [16]. (3) Having a *feedback loop* between initial versions of a DSML and its prospective users, in terms of both agile process [13,14], and the use of mock up diagrams to elicit end user feedback [3]. (4) *Capturing domain features*, prominently in terms of feature diagrams [14,17]. Here "features" capture the variability of a domain, in terms of features common to the domain (e.g., for the "browsing" domain of [17], "get" and "post"), and features that differ across the domain (e.g., for "browsing": each browser can have a different set of plug-ins).

However, even if the above mentioned aspects are important for the design of a DSML, the resulting DSMLs usually lack a concrete relation to them. Either the aspects are (1) documented, but the documentation exists separate from the DSML specification, it is not publicly available, or the documentation is written in a natural language that is different from the DSML specification, or (2) the documentation is not at all provided. Indeed, we have observed such a lack of

traceability not only in state-of-the-art enterprise modeling approaches such as ArchiMate [5] or in some DSMLs being part of Multi-perspective Enterprise Modeling (MEMO) [4], but also in de facto standards such as Unified Modeling Language (UML) [18].

During the design of ArchiMate requirements for modeling, analysis and visualization have been identified [6], and have helped to shape the core ArchiMate language specification. However, these requirements are not explicitly traceable to the final language specification. This, consequently, increases the risk of developing extensions to ArchiMate that do not meet the initially defined requirements anymore (as with the motivation extension, discussed in the introduction).

Furthermore, recent languages being part of MEMO have been designed following the DSML design method by Frank [3]. As a result the relevant use scenarios, general and specific requirements, as well as justification of designed decisions have been documented, e.g., [19]. However, this information is not always available to a wider audience as (1) it might be unclear where this information is stored, (2) it is disseminated only in German, as in [19].

UML presents similar issues in the sense that its initial general requirements do not say much about how the final language specification was decided upon [18]. As a result, there are concepts in UML whose rationale is not clear, which leads to different (mis)interpretations (e.g., purely in terms of semantics an aggregation relationship can be substituted by an association relationship with corresponding cardinalities [18]).

To sum up: during language design there exists initial documentation about the language purposes, their requirements and the intended use of a language. However, that information is not linked to the resulting language specifications, neither is it always available to audiences interested in extending those languages. This introduces the danger of *uncontrolled evolution*, which can already be observed in standard languages such as ArchiMate. Therefore, we argue that whenever the designed domain-specific modeling approach is to be used beyond the specific group of people involved in its creation, it is imperative that one captures the provenance of the language specification in terms of its origins: the purposes standing behind a language, its use scenarios, and domain constraints. Thereby, one fosters controlled language evolution, and the informed (re-)design of conceptual modeling languages.

3 Meta Model Provenance

In this section, we first characterize meta model provenance based upon a synthesis of language design literature and literature on data (visualization) provenance (in Sect. 3.1). Following this, we focus on our exemplary provenance approach: goal-driven meta model provenance (in Sect. 3.2).

3.1 Characterizing Meta Model Provenance

For our research purposes, the provenance of a language specification is of importance. As most languages are defined by using a meta model, we further denote it as **meta model provenance** and focus on provenance of meta model elements such as concepts, their attributes, relations, and constraints.

As per the introduction, the term "provenance" is used in a wide array of domains [20,21]. Two of these are relevant to us: data provenance and data visualization provenance. Data (visualization) provenance approaches focus on a static world view when defining the cornerstones of provenance. This is in contrast to capturing the provenance of dynamics, as prominently happening within workflow provenance [22]. Hence, in their way of thinking, data (visualization) provenance approaches are close to what we intend to capture. Thus, in characterizing meta model provenance, we turn to key questions as defined in data (visualization) provenance cf. [20,21]: (1) what is the scope of the meta model provenance, in terms of the type of information that can be captured? (2) how it is represented and what level of formality is used? and (3) how can provenance be visualized?

Accordingly, we structure our discussion on the proposed idea of meta model provenance in line with these three aspects.

The scope of meta model provenance. Different research initiatives have been undertaken with the aim to standardize the scope of data (visualization) provenance. One important effort is the so called W7 model [23,24]. The W7 model represents different components of provenance, namely *what*, *when*, *where*, *how*, *who*, *which* and *why*, and their relationships to each other. In the W7 model, data provenance is conceptualized as consisting of various events that happen during the lifetime of the data from its creation to destruction, cf., [23]. Each of the 7 Ws has its subtypes, for instance, *what* has among others subtypes including creation, modification and publication, and ownership, whereas *how* can be classified into single and complex actions [10].

Likewise, but without using the term provenance per se, in the conceptual modeling domain Thalheim [12] uses W-questions as a baseline for developing a "Theory" of modeling. Thereby, he first characterizes modeling by four core characteristics: wherefore (purpose), whereof (origin), wherewith (language), and worthiness (value). Subsequently, he details each of the core characteristics by means of further Ws. For example, in the case of wherefore (purpose): how, why, whereto, when, for which reason [12].

Although the proposed dimensions from both data provenance as well as the theory of conceptual modeling by [12] constitute a comprehensive framework regarding different types of provenance information, it is unclear how – concretely – this information complements conceptual modeling languages, and what purpose the capturing of this information should particularly serve. Also, the respective W frameworks provide few hints regarding the way in which the information should be stored, retrieved, and visualized.

Turning now to the aspects informing the DSML design methods from Sect. 2, we observe that each of them provides an answer to a subset of W-questions.

For instance, scenario- and goal-driven methods answer questions like *for whom?* and *for which purpose(s)?*, whereas methods that capture domain features help to partly answer *what?* questions. However, while providing a valuable input, the aspects mentioned by the DSML design methods as such are insufficient for our purposes since they have not be designed with the idea of provenance in mind. Thus, as discussed in Sect. 2, the provenance information resulting from these design methods is insufficiently documented and/or insufficiently related to a language specification.

Representation and the used level of formality. Provenance can be represented at different levels of formality, depending on the analysis task to be supported [20,22]. Here the formality ranges from plain text annotations, up to formalisms that can be automatically processed, such as the query inversion described in [20] to retrieve data lineage in a database. The manner in which provenance is represented has implications for the costs of recording it, the set of analysis tasks it supports, and for its ease of use [20,22]. For one, due to their unrestrictive nature *plain text annotations* offer rich expressiveness, but at the same time introduce the challenge of storing and retrieving the relevant data [20]. In contrast, purely formal provenance approaches offer rich possibilities for automated analyses, but often are difficult to come to grips with for average end users [22]. As a semi-formal approach, conceptual modeling combines the respective strengths of a purely plain text approach, and a purely formalized approach. Like plain text annotation approaches it offers possibilities for rich expressiveness, yet at the same time, it offers a subset of computer-supported analysis capabilities from purely formal provenance approaches.

Provenance visualization. Depending on its audience and intended use, provenance information can be visualized in different ways. In line with [20,22] this can be an informal way, e.g., by means of plain text notes, by means of query results represented in the syntax of a query language, or by means of visual graphs. Each visualization of provenance information comes with some advantages and some drawbacks. Like the level of formality, a plain text description can be relatively easily prepared and requires no prior knowledge (only the knowledge of the natural language it was expressed in is required). On the other hand, using formal techniques, such as query languages, automated reasoning, requires prior training and/or an engineering background.

3.2 A Goal-Driven Approach to Document Meta Model Provenance

In what follows, we show how meta model provenance can be operationalized by focusing on the intention standing behind the current language specification. Thus, we focus on the following subset of W-questions: (1) **for what reasons**? This question concerns the goals and/or information needs of involved stakeholders that are going to be fulfilled; and (2) **what?** This question concerns elements of the (meta) model that are affected by the purposes of a language, or have been defined because of it.

In line with [16], we consider the goals of both direct users of a language (such as language designers, or information analysts), and those that receive results out of the use of a language (such as CxOs being interested in certain analysis results). The goal-driven approach to document meta model provenance should be able to: express why different concepts have been included in the language specification and which application scenarios the different meta model elements are to support. Also, by relying on goal models one can check to what extent models that are instantiated from the meta model comply with domain goals.

Requirements. To provide transparency with respect to assumptions and desired features for our provenance approach (thus in a sense providing rudimentary provenance of our approach), we first explicate the requirements for our goal-driven meta model provenance approach. Thereafter, we discuss the provenance approach itself.

R1: *Having a semi-formal provenance approach.* With our goal-driven provenance approach we are, on the one hand, interested in expressing domain goals, the stakeholders behind these goals, and relations between goals. Thus, we are interested in having an approach with a rich expressiveness. On the other hand, we are interested in performing computational analyses with our approach, prominently to check to what extent a model generated with the meta model is in line with the domain goals. This implies computational reasoning capabilities for the approach. Thus, all in all we are interested in a semi-formal provenance approach: sufficiently rich in expressiveness to capture various facets of intentionality, yet sufficiently computational to support reasoning.

R2: *The visualization should fit the users of this information, and the desired use scenario.* Thus, a minimal learning effort should be accounted for. As a result, one should also decide on the way this information should be visualized for the involved stakeholders so that the desired use cases can be supported. Here our assumption is that goal models are at least accessible to the primary stakeholder of a language: a language designer. Thereby, based on the existing modeling expertise we assume that a language designer has familiarity with goal modeling or, if not, that the effort to learn goal modeling is low.

R3: *Having a close relation between language provenance and language specification.* As already discussed in Sect. 2, keeping the documentation separate from the language specification makes the specification less accessible to an interested audience. In addition, post hoc tracking of provenance introduces difficulties such as forgetting the provenance over time, or by introducing an additional "rationalization" bias by making up provenance reasons in retrospect. Therefore, capturing the provenance should ideally be done at the same time as, and together with, the definition of the language specification.

R4: *Supporting different levels of granularity.* We should capture information regarding both the intentional elements, and the origin of elements of the language specification at different levels of granularity. If we desire to keep the language specification close to the corresponding provenance information, it follows that we should cover comprehensively relations between different elements

of the language specification and elements of the language provenance. Concretely, this implies that provenance should be tracked both at (i) a fine-grained level, i.e., on the level of attributes, roles, relations and cardinalities, and at (ii) the level of meta types, constraints, or even clusters of those.

R5: *Having software tool support.* As stated, we are interested in a semi-formal approach, in part because this provides us with computational reasoning support. In particular, we are interested in compliance checking of language specification elements with provenance elements, as well as compliance of the models instantiated from the meta model with the domain goals. To support us in this task, a software tool is required that provides reasoning capabilities.

The GRL4DSML Approach for Goal-Driven Meta Model Provenance. Guided by the above requirements we now introduce the GRL4DSML approach for capturing the provenance of the language specification using goal models. GRL4DSML consists of two parts: (1) a meta model, for conceptually expressing elements of an abstract syntax, intentional elements, and a relationship between abstract syntax elements and intentional elements; and (2) a process, for using the GRL4DSML meta model to capture language provenance in a goal-driven manner.

The GRL4DSML meta model is shown in Fig. 1. It integrates concepts from both GORE (Goal-Oriented Requirement Engineering) and DSML design, and defines connections among these concepts. On the GORE side, we use a core subset of the Goal-oriented Requirements Language (GRL) [25] meta model. GRL offers the following main concepts. Goals are objectives that a stakeholder (modeled as an actor) would like to achieve. To achieve goals, actors may employ resources (physical or informational entities) and perform tasks. We take GRL as the baseline because it is standardized by the Telecommunication Standardization Sector (ITU-T) and has a mature tool support in terms of jUCMNav[2].

On the DSML design side, a core set of meta modeling concepts is extracted and included. Meta modeling is a means to specify DSMLs, where the abstract syntax of a DSML includes the following elements: (1) a meta model that defines the constructs of the DSML, and (2) a set of well-formedness constraints. In the example we use the MEMO Meta Modeling Language (MML [26]). It offers meta types, their attributes and relations, as well as possibility to define OCL constraints. MML, when compared to 'traditional' meta modeling languages (e.g., the Meta Object Facility of UML), provides additional language constructs such as intrinsic attributes and relations. Such intrinsicness means that the attribute/relation is instantiated only at the instance level and not at the type level, and is visualized with a white letter "i" on a black background.

We enumerate the following main constructs in the GRL4DSML meta model: "meta types", "attributes", and "relations". Recall that the purpose of GRL4DSML is to guide the DSML design with goals of DSML stakeholders and to document such a provenance. To this end, we require the traceability from

[2] http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome.

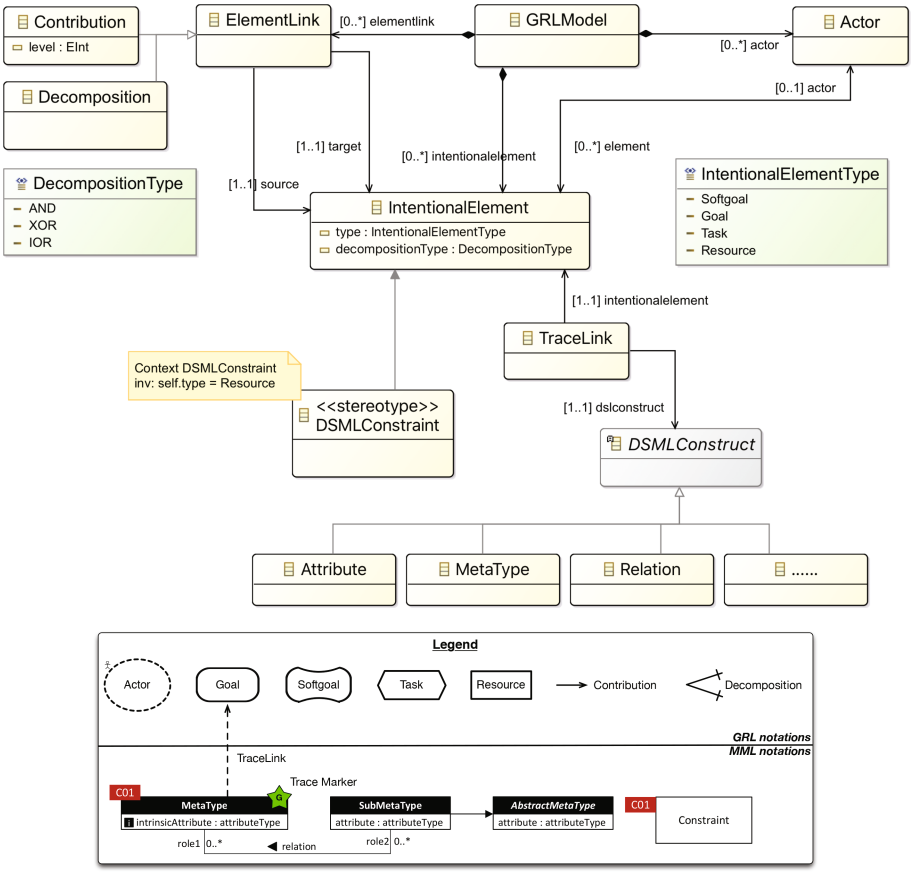


Fig. 1. GRL4DSML design meta model

DSML constructs to goals of stakeholders to indicate that the latter motivates the existence of the former.

Well-formedness constraints associated with a DSML further restrict the shape of instances the DSML can express for the purpose of goal achievement: all instances of the DSML must make these constraints true, and the validity of the constraints guarantees (to an extent) the fulfillment of goals. As a consequence, constraints resemble a special type of informational resources offered by the DSML to contribute to the fulfillment of goals. To capture this in the GRL4DSML meta model, we extend the Resource concept from GRL with a stereotype «DSLConstraint».

The legend at the bottom of Fig. 1 summarizes the visual notation that we use to illustrate the application of GRL4DSML in the case scenario (Sect. 4). For GRL concepts, we reuse GRL's concrete syntax. For meta modeling concepts, we reuse MML's concrete syntax, which is similar to class diagrams. The trace links

between meta modeling concepts (e.g., a meta type) and GRL concepts (e.g., a goal) are illustrated visually by dashed arrows (from the meta type to the goal). As a shortcut, such tracing information can also be depicted by marking the source end of a trace link (i.e., meta type) with a trace marker, which is a green star with the name of the target end (i.e., goal).

The GRL4DSML Process is characterized by four key steps. **Step 1** concerns capturing stakeholder goals in a goal model. Please note that "stakeholder" refers here to both the domain actors as well as a language designer. **Step 2** concerns defining an initial version of the abstract syntax. Such an initial version can either be empty or contain only the main target concept of the DSML and remains at a high level of abstraction. **Step 3** concerns gradually extending the DSML's abstract syntax guided by the goals in the goal model. Note that we only need to focus on the leaf goals in the goal model because the achievement of higher level goals is calculated based on the achievement of leaf goals. **Step 4** concerns evolving the goal model accordingly, because the evolution of the abstract syntax model of the DSML can give rise to new goals that have been overlooked in the first step. Such a co-evolution process continues until no new goals are detected, and all the goals and their achievement are captured by the DSML's abstract syntax.

In the next section, we show how the definition of an exemplary language in terms of an abstract syntax can be combined with a goal model to capture the intention behind a particular language specification.

4 Case Scenario: Substation Automation

This case scenario focuses on designing a DSML for IT infrastructure modeling in the electricity domain. It is part of a larger research project that concerns an assessment of both the technical and economic feasibility of an electricity domain initiative. The idea is that while more and more initiatives in the electricity domain are enabled by IT (IT being, e.g., smart meters), the realization

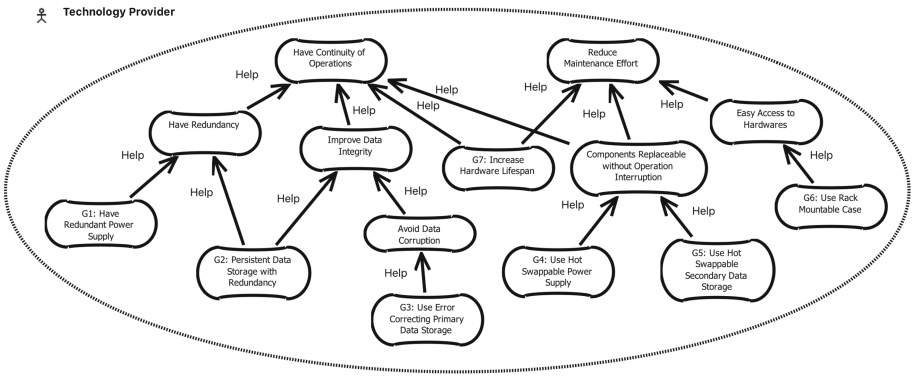


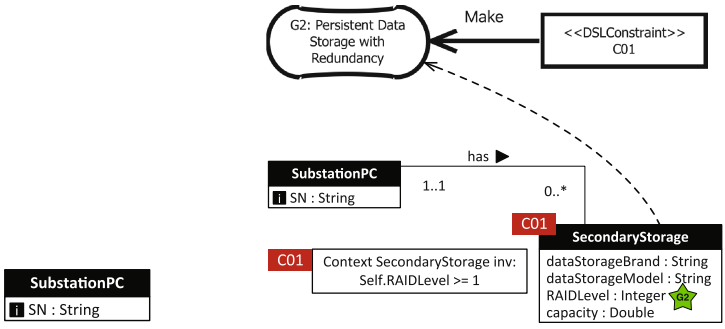
Fig. 2. Goal model capturing the intentions of the technology provider

of such initiatives in the marketplace requires an economic assessment next to the technical one, prominently in terms of the actors involved and what they exchange of value with one another [27]. Since characteristics of an IT infrastructure influence the economical side and vice versa [28], we thus have a need to model IT infrastructure as part of this project.

For scoping reasons, we focus our example on the modeling of substation PCs. In the electricity grid, substations transform high-level voltage, as usually generated by large power plants, to medium to low level voltage suitable for end users (households or small businesses). Substation PCs, then, act as a single point of contact for monitoring the activities in a substation. This concerns both regular activity, such as transmitting metering data, and signaling abnormalities, such as power spikes or disruptions in the electricity flow.

We start with a goal model (Fig. 2) that captures the intentions of the technology provider (**Step 1**). Then, the *DSML Initialization* (**Step 2**) takes place. As shown in Fig. 3a the DSML initially has a single meta type "SubstationPC" to designate the substation PCs being modeled. Each substation PC possesses a unique serial number, modeled by an attribute "SN" of type string.

DSML Extension to Support G2. We elaborate in the following how the DSML is extended to enable the discussion of the fulfillment of the leaf goals in the goal model (**Step 3**). This is done by including the necessary elements in the DSML. For example, consider "G2: Persistent Data Storage with Redundancy". The description of G2 uncovers two concepts: "persistent data storage" and "redundancy". As a response, for capturing the concept of "persistent data storage" the language designer introduces a new meta type "SecondaryStorage" into the abstract syntax. In addition, to characterize a data storage device, a brand name, a model number, and its capacity are also included in the abstract syntax by means of three new attributes for "SecondaryStorage": "dataStorageBrand" and "dataStorageModel" of type string, and "capacity" of type double. Moreover, a new (containment) relation is also introduced, from the "SubstationPC" meta type to the meta type "SecondaryStorage", to indicate that data



(a) Initial version (b) Intermediate version to support G2

Fig. 3. Evolution of Substation PC DSML design

storage devices are components of a substation PC. No restriction is imposed on the cardinalities of the association. Thus, they are set to the most general one, namely "0..*".

For capturing the concept of "redundancy", the language designer adds a property to express redundancy to data storage devices. This is done by referring to the RAID (Redundant Array of Independent Disks) technology, which is a data storage virtualization technology for the purpose of data redundancy. More specifically, yet another new attribute "RAIDLevel" of type integer is introduced to class "SecondaryStorage", to indicate at which level the RAID technology is implemented in a given data storage device.

Figure 3b now shows the modified version of the abstract syntax, including the elements for expressing secondary storage characteristics. The motivation of the current version of the DSML design is documented by tracing the new language constructs in the abstract syntax of the DSML to the corresponding goal in the goal model, in this case goal G2.

Note that the new constructs introduced in the DSML make it possible to express instances (i.e., substation PCs), with which one can discuss about the achievement of G2. However, the actual level of fulfillment of G2 is still to be decided by the value of the "RAIDLevel" attribute in the instances. More specifically, no redundancy is provided if the RAID level is 0. We add a new constraint C01 in the abstract syntax of the DSML to reflect the checking of this value. This constraint is captured in the goal model as a resource offered by the DSML which contributes to the full achievement of G2 (marked by label "make").

DSML Extension to Support G1–G6. By applying **Step 3** to the other leaf goals (except for G7, which is considered only later) in a similar manner, the abstract syntax of the substation PC DSML further evolves to the version captured in Fig. 4. We mark the motivation of each construct directly by the name of the corresponding goal in the abstract syntax, to avoid repeating the goal model diagram.

Goal Model Extension. At this stage, the language designer notices that there are duplications of attributes among the following classes: "PowerSupply", "PrimaryStorage", "SecondaryStorage", and "Case". Namely, all the hardware components use two attributes to capture the brand and model, and two data storage

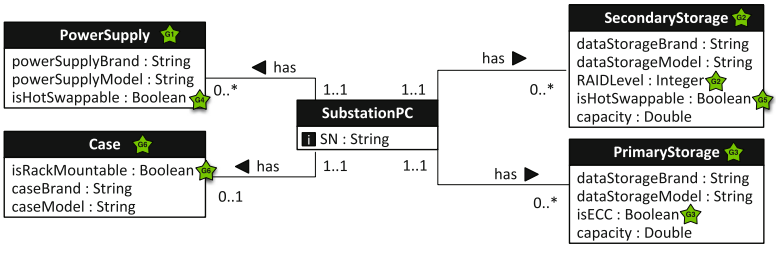


Fig. 4. Substation PC DSML intermediate version to support leaf goals G1−G6

classes share the capacity attribute. Instead of duplicating these attributes from one to another, a better way to implement this would be to introduce generalization/specialization. Such an intention of the language designer is captured by a new goal "G8: Reuse" (**Step 4**), which acts as the motivation of the new abstract meta types "HardwareElement" and "StorageDevice", and the corresponding inheritance relations (**Step 3**).

DSML Extension to Support G7. Finally, the DSML is extended to support also G7 (**Step 3**). This is done by introducing yet another attribute "averagelifespan" of type integer to class "HardwareElement".

Figure 5 presents the final version of the substation PC DSML (at the bottom) and the updated goal model with new goals and constraints (at the top). Note that by enforcing all the constraints in the DSML, the corresponding leaf goals are achieved fully for any substation PC instantiated from it. Please also note that with our approach, the constraints can be switched on and off,

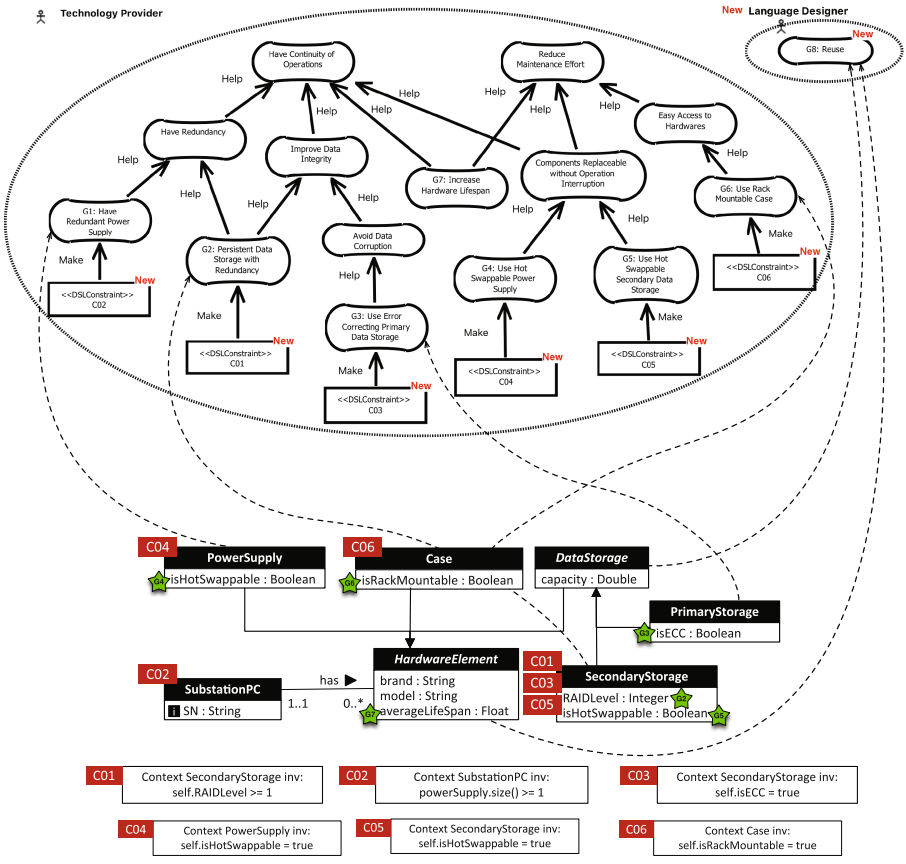


Fig. 5. Updated Goal Model (top) and DSML (bottom), and traceability

depending on the needs of the language user. For example, if at a later stage the technology provider notices that there is also a market for substation PCs without hot swappable power supply, the corresponding constraint C04 can be switched off. The resulting DSML would then also allow the specification of substation PCs without this feature.

5 Conclusions and Outlook

In this paper we have introduced the overall idea of meta model provenance and presented a goal-driven approach to achieve it. We argued that meta model provenance must be able to record the intension behind the elements of a given abstract syntax, to support the controlled evolution and the informed (re-)design of languages.

The goal-driven approach offers a semi-formal representation that helps to capture the scope of a given language as well as the main motivation(s) behind its design decisions, i.e., why concepts have been introduced. Due to its visual expressiveness, it also helps to communicate with stakeholders that are usually involved in the design of a language, i.e., language designers as well as final users. In this way, the idea of meta model provenance aims to complement and support existing modeling methods rather than substitute them.

However, to this end more research work needs to be conducted. First of all, despite the benefits offered by our approach, one should also investigate the impact it has on large language design projects. As such, the Return on Provenance Effort (RoPE) should be overall positive. This means that the GRL4DSML process should help controlling the evolution and (re-)design of languages while clearly justifying the effort invested at each step. Second, ways to handle the inherent complexity of the provenance effort should also be investigated as the volume of provenance-related information could potentially dwarf the language specification itself. Third, we should also investigate mechanisms to fully exploit features offered by our goal-driven approach. For instance software tools to fully support compliance checking of languages, as well as (automatically) switching on and off constraints. Finally, governance regarding how much provenance must be exposed to different audiences requires additional research.

References

1. Frank, U., Strecker, S., Fettke, P., vom Brocke, J., Becker, J., Sinz, E.: The research field 'Modeling Business Information Systems'. BISE **6**(1), 39–43 (2014)
2. Clark, T., Frank, U., Kulkarni, V.: Supporting organizational efficiency and agility: models, languages and software systems. Dagstuhl Rep. **6**(5), 31–55 (2016)
3. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering, pp. 133–157. Springer, Heidelberg (2013). 10.1007/978-3-642-36654-3_6
4. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. SoSyM **13**(3), 941–962 (2014)

5. The Open Group: ArchiMate 2.1 Specification: Open Group Standard. The Open Group Series. Van Haren, Zaltbommel (2013)
6. Lankhorst, M.M., Proper, H.A., Jonkers, H.: The anatomy of the ArchiMate language. Int. J. Inf. Syst. Model. Des. (IJISMD) **1**(1), 1–32 (2010)
7. Engelsman, W., Wieringa, R.: Understandability of goal-oriented requirements engineering concepts for enterprise architects. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 105–119. Springer, Cham (2014). 10.1007/978-3-319-07881-6_8
8. Tan, W.C.: Research problems in data provenance. IEEE Data Eng. Bull. **27**, 45–52 (2004)
9. Gupta, A.: Data provenance. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 608–608. Springer US, Boston (2009)
10. Ram, S., Liu, J.: A semantic foundation for provenance management. J. Data Semant. **1**(1), 11–17 (2012)
11. Kramer, F., Thalheim, B.: Metadata as support for data provenance. In: Jaakkola, H., Thalheim, B., Kiyoki, Y., Yoshida, N. (eds.) 26th International Conference on Information Modelling and Knowledge Bases XXVIII (EJC 2016). FAIA, vol. 292, pp. 195–214. IOS Press (2016)
12. Thalheim, B.: The science and art of conceptual modelling. Trans. Large-Scale Data- Knowl.-Center. Syst. **6**, 76–105 (2012)
13. Karagiannis, D.: Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics, PCI 2015, pp. 5–10. ACM, New York (2015)
14. Villanueva Del Pozo, M.J.: An agile model-driven method for involving end-users in DSL development. Ph.D. thesis, Universidad Politecnica de Valencia (2016)
15. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Völkel, S.: Design guidelines for domain specific languages. arXiv preprint arXiv:1409.2378 (2014)
16. De Kinderen, S., Ma, Q.: Requirements engineering for the design of conceptual modeling languages. Appl. Ontol. **10**(1), 7–24 (2015)
17. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. (CSUR) **37**(4), 316–344 (2005)
18. OMG: The OMG Unified Modeling Language, v. 2.5. Technical report (2015)
19. Heise, D.: Unternehmensmodell-basiertes IT-Kostenmanagement als Bestandteil eines integrativen IT-Controllings. Logos, Berlin (2013)
20. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Rec. **34**(3), 31–36 (2005)
21. Ragan, E.D., Endert, A., Sanyal, J., Chen, J.: Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. IEEE Trans. Vis. Comput. Graphics **22**(1), 31–40 (2016)
22. Herschel, M., Hlawatsch, M.: Provenance: on and behind the screens. In: Proceedings of the 2016 International Conference on Management of Data, pp. 2213–2217. ACM (2016)
23. Ram, S., Liu, J.: Understanding the semantics of data provenance to support active conceptual modeling. In: Chen, P.P., Wong, L.Y. (eds.) ACM-L 2006. LNCS, vol. 4512, pp. 17–29. Springer, Heidelberg (2007). 10.1007/978-3-540-77503-4_3
24. Liu, J., Ram, S.: Improving the domain independence of data provenance ontologies: a demonstration using conceptual graphs and the W7 model. JDM **1**(28), 43–62 (2017)
25. ITU-T: User requirements notation (URN)-language definition (November 2008) Recommendation Z.151 (11/08). http://www.itu.int/rec/T-REC-Z.151/en. Last accessed 28 July 2017

26. Frank, U.: The MEMO Meta Modelling Language (MML) and Language Architecture, 2nd edn. ICB-Research Report 43, University of Duisburg-Essen (2011)
27. Niesten, E., Alkemade, F.: How is value created and captured in smart grids? a review of the literature and an analysis of pilot projects. Renew. Sustain. Energy Rev. **53**, 629–638 (2016)
28. Razavian, M., Gordijn, J.: Consonance between economic and IT services: finding the balance between conflicting requirements. In: Fricker, S.A., Schneider, K. (eds.) REFSQ 2015. LNCS, vol. 9013, pp. 148–163. Springer, Cham (2015). 10.1007/978-3-319-16101-3_10