Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration: Application to Telecommunications Service Design

Vanea Chiprianov, Yvon Kermarrec, Siegfried Rouvrais, Jacques Simonin

SoSym manuscript No.
(will be inserted by the editor)

Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration

Application to Telecommunications Service Design

Vanea CHIPRIANOV[1,2], Yvon KERMARREC[1,2], Siegfried ROUVRAIS[1,3], Jacques SIMONIN[1,2]

[1] Institut Mines-Telecom, Telecom Bretagne, Technopole Brest Iroise CS 83818 29238 Brest Cedex 3, Universite europeenne de Bretagne, France
[2] UMR CNRS 6285 Lab-STICC
[3] IRISA

The date of receipt and acceptance will be inserted by the editor

Abstract The competitive market forces organizations to be agile and flexible so as to react robustly to complex events. Modeling helps managing this complexity. However, in order to model an enterprise, many stakeholders, with different expertise, must work together and take decisions. These decisions and their rationale are not always captured explicitly, in a standard, formal manner. The main problem is to persuade stakeholders to capture them. This article synthesizes an approach for capturing and using the rationale behind enterprise modeling decisions. The approach is implemented through a Domain Specific Modeling Language, defined as an extension of a standard Enterprise Architecture Modeling Language. It promotes coordination, enables presenting different stakeholders' points of view, facilities participation and collaboration in modeling activities - activities focused here on Enterprise Architecture viewpoints. To present its benefits, such as rapid prototyping, the approach is applied to large organizations in the context of Telecommunications service design. It is exemplified on modeling and capturing decisions on a conference service.

Key words Enterprise Architecture Framework, Enterprise Architecture Modeling Language, Domain Specific Modeling Language, Language Extension, Metamodel, Design Rationale, Telecommunications Service.

1 Introduction

In order to manage an enterprise, a strategy is to be defined, and an organization chart of its departments must be set up. This organization needs collaboration between departments to manage the complexity of enterprise evolution (as driven by the strategy). The development of large and complex organizations involves many people, stakeholders, each with their own viewpoint. These viewpoints make up the architecture of an enterprise. An architecture will integrate business and functional aspects, as well as technical ones.

The complexity can be faced by modeling techniques. Viewpoints can be modeled through Enterprise Modeling approaches that address enterprise complexity. For this, enterprise stakeholders must take decisions on different issues (e.g., how to align a model with an enterprise strategy), using specific criteria and justifying them with shareable arguments (i.e. decision rationale). These decisions and their rationale are not always captured explicitly, in a standard, formal manner. Hence, many of them are never communicated to other stakeholders. They are most often forgotten once the people that took them leave the enterprise or project. This may result in a loss of knowledge, high difficulty in maintaining and evolving existing systems, encumbrance of communication and common understanding, poor collaboration, lack of traceability, low quality.

This article synthesizes an approach for capturing and using the rationale behind enterprise modeling decisions (e.g., what concepts to include in a model) as a Modeling Language, which is integrated with the system Modeling Languages. This Rationale Modeling Language is transversal to all system Modeling Languages intended for viewpoints, is independent of them, and in general, of the domain. It can be used for any domain that implies decisions and requires their capture.

Addressing enterprise system complexity through Enterprise Modeling, Enterprise Architecture frameworks, is discussed in Sect. 2. To facilitate collaboration between the many stakeholders involved in the development, maintenance and evolvability of the Information System of the enterprise and enable reuse of their knowledge, Design Rationale is introduced, as a key approach,

in Sect. 3. To persuade stakeholders to capture Design Rationale, a Domain Specific Modeling Language is advanced. To define this language, a dedicated process and a Model Driven Approach is presented in Sect. 4. To reduce implementation costs and increase integration with recognized standards of Enterprise Architecture Frameworks, the Design Rationale Domain Specific Modeling Language is developed as an extension of an Enterprise Architecture Modeling Language. Its definition and tools are introduced in Sect. 5. To show its reliability, it is applied for an example of a Telecom conference service, in Sect. 6. The article finishes with conclusions and perspectives, in Sect. 8, after related works in Sect. 7.

2 Enterprise Architecture Frameworks and Modeling Languages

2.1 Architecture and Viewpoints

To document, understand and master the complexity of an enterprise Information Systems, architectures are an important means. For the term *architecture*, the IEEE Standard 1471-2000 is adopted here, which defines it as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principle guiding its design and evolution" [24].

Separation of concerns, specifications from multiple viewpoints, are another important means. In [27], a *viewpoint* is defined as a "work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns". A *view* is defined as a "work product expressing the architecture of a system from the perspective of specific system concerns". A *concern* is an "interest in a system relevant to one or more stakeholders". A *stakeholder* is an "individual, team, organization, or classes thereof, having an interest in a system".

2.2 Enterprise Architecture Frameworks

To describe the architecture of an enterprise, Enterprise Architecture is useful. There are several definitions of Enterprise Architecture, e.g., [11], [31]. Related to the variety and difference of definitions in the Enterprise Architecture discipline, [44] considers that there is a lack of theoretical foundation, definitions or a common understanding within the authors who publish in this context. To avoid contributing to this problem, definitions are adopted here for the main concepts. For *Enterprise Architecture (EA)*, [31] define it as "a coherent whole of principles, methods and models that are used in the design and realization of the enterprise's organizational structure, business processes, information systems, and infrastructure". *Enterprise Modeling (EM)* describes the EA from various viewpoints in detail to allow specifying and implementing the systems [11]. Thus, EM combines two means of mastering complexity, architecture and separation of concerns.

According to ISO 15704 [25] there are two types of enterprise architectures:

1. *System architectures* (also called "Type 1"), that deal with the design of a system. They represent a system (e.g. a system of the Information System of an enterprise) in terms of its structure and behavior.
2. *Enterprise-reference projects* (also called "Type 2"), that deal with the organization of the development and implementation of a project such as an enterprise integration or other enterprise development program. They are actually frameworks aiming at structuring concepts and activities/tasks necessary to design and build an enterprise system. The system design (Type 1) must be coherent with that of other enterprise systems and especially be aligned with the enterprise strategy (Type 2). According to a survey of EAs [11], most of enterprise architectures are Type 2, frameworks.

An *EA framework* is a structure expressed in terms of diagrams, text and formal rules that relates the components of a conceptual entity to each other [26]. It defines and organizes the generic concepts that are required to enable the creation of enterprise models for industrial businesses. Its main purpose is to provide an organizing mechanism so that concepts, problems and knowledge about enterprise interoperability (both inter and intra) can be represented in a more structured way. EA frameworks have been reviewed for example by [38], compared for example by [50].

2.3 Enterprise Architecture Modeling Languages

Applying EA frameworks to create Type 1 architectures (i.e. unambiguously specifying and describing system components and their relations) requires coherent Enterprise Architecture Modeling Languages [29]. A *Modeling Language (ML)* is "a graphical or textual language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [7] [23]. An *Enterprise Architecture Modeling Language (EAML)* is a high level of abstraction language, aiming at representing enterprise architectural structure, characteristics and properties at early stage of design [11]. Although general purpose modeling languages like UML have been applied to model EAs, e.g. [21], they are not specially adapted for this task. The most notable example of an EAML is ArchiMate [48].

EA benefits (e.g., [11] [31] [29]) have been reviewed and categorized by [42] into hard (e.g. improved interoperability and integration of enterprise constituting systems), intangible (e.g. providing a holistic view of the

enterprise), indirect (e.g. understanding the wealth of interconnections with an enterprise's customers, and other partners) and strategic (e.g. increasing the insight and overview required to successfully align the business and technology platforms). The evidence of the contribution of EA to the achievement of organizational goals is reviewed by [8]. To achieve these goals, alignment between viewpoints and enterprise strategy is needed [47].

EAMLs are used to model an enterprise and its products/services/systems from several viewpoints. To better differentiate them from other MLs, they are called here *system MLs*. Collaboration between stakeholders with different viewpoints is one of the main issues introduced by EAMLs.

3 Collaborative Methods for Designers

Collaboration is a process where two or more stakeholders work jointly or together, at the same time or asynchronously, to create or achieve the same goal, especially in an intellectual endeavor, by sharing knowledge, learning and building consensus (according to the Merriam Webster and Cambridge dictionaries).

Benefits of collaboration for an enterprise include: improved access to information and people across the enterprise, on-demand availability of data for accelerated decision making, enterprise-wide sharing of knowledge and resources, reduced error rates.

3.1 Design Rationale for Collaboration and Enterprise Architecture

In spite of the variety of collaboration methods, an important and common issue of all is information seeking. One of the most absent piece of information is about design decisions. *Design Rationale (DR)* is the justification behind decisions, the reasoning that goes into determining the design of the artifact [19]. So, capturing DR and enabling its retrieval would reduce significantly the information seeking time, increasing the performance of collaboration.

DR also supports collaboration by [19]:
- promoting coordination in design teams,
- exposing differing points of view,
- facilitating participation and collaboration,
- building consensus.

Specifically for EA, DR is useful to select the best design from a group of design alternatives, such as various architecture styles. These design choices are interrelated, and maximizing certain attributes of the design may minimize to an unacceptable level other attributes [20] (e.g. performance vs. security). Therefore trade-offs are to be managed, in this multi-criteria decision making problem [9].

Another specific interest of DR in the context of EA is related to two types of stakeholders. First are the EA designers, who work with Type 2 EA architectures. They take certain decisions and recommendations which should be followed by the second type of stakeholders, the EA users. These work with Type 1 EA architectures. One of their difficulties is to understand, appropriate and use in their activity the decisions and recommendations taken by EA conceivers [46]. As the DR DSML allows capturing and retrieving decisions and their rationale, it will help reduce this gap.

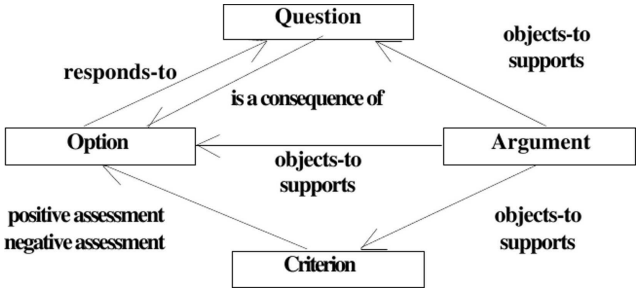


Fig. 1 The QOC schema, from [19].

3.2 The Process of Using Design Rationale

The main activities involving DR are:
1. *Capture*: the process of eliciting rationale from designers (e.g., directly from designers, or through an automatic manner) and recording it. Designers may be quite reluctant about capturing rationale [19]. One of the most important factors of this resistance seems to be the intrusiveness of the capture process (e.g., the disruption in designers' thinking). To solve the capture problem, one promising direction explores reducing its intrusiveness by integrating rationale artifacts with the system models [52].
2. *Formalization*: the process of transforming rationale into the desired representation form. There are many ways in which DR argumentation may be structured. One school of thought uses a group of DR schemas (e.g. Fig. 1, from [19]) having Issue-Based Information System (IBIS), Questions Options Criteria (QOC), and Decision Representation Language (DRL) as its most prominent members.
3. *Retrieval*: the process of getting recorded rationale to the people who need it, providing access to it. Knowledge is thus capitalized upon and transmitted between projects.

3.3 Towards a Design Rationale Domain Specific Modeling Language

To address the DR capture issue, for EAs, this article proposes formalizing the rationale as a ML which is integrated with the system MLs. This DR ML is transversal to all system MLs intended for viewpoints, is independent of them, and in general, of the domain. It can

be used for any domain that implies decisions and requires their capture. The DR ML is based on argumentative formalization of DR, and contains concepts and constructs from the representation schema. One, or a combination of, schemas like IBIS, QOC, DRL can be used. Designers use it during the modeling activities, at the moment which is natural to them. This is expected to reduce the DR capture intrusiveness. For DR retrieval, existing navigation or query-based systems can be integrated.

Such a DR ML, because it is transversal to EAMLs, is well suited to share the knowledge coming from collaborative decisions and their rationale at the enterprise level. Being an ML, it is easy to use and understand, encouraging the capture and allowing the capitalization upon of shared knowledge by several enterprise stakeholders.

4 Introduction to Domain Specific Modeling Languages and Meta-modeling for Language Definition and Extension

DR is a concern, a domain; the DR ML can be considered a Domain Specific Modeling Language.

4.1 Domain Specific Modeling Languages

Domain Specific Modeling Languages can be used in Enterprise Modeling to make explicit each view [36]. A Domain Specific Modeling Language is an ML (concept already defined) and a Domain Specific Language. A *Domain Specific Language (DSL)* is "a language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain" [51]. A *Domain Specific Modeling Language (DSML)* is therefore taken in this article to be a graphical or textual language that offers, through appropriate notations and abstractions, expressive power focused on a particular problem domain, to visualize, specify, construct and document the artifacts of a software-intensive system.

DSMLs allow experts to express, validate, modify tasks specific to their domain. Empirical studies, e.g. questionnaires [35], confirm that DSMLs are superior to more general purpose languages in all cognitive dimensions. Thus quality, productivity, reliability, maintainability, re-usability, flexibility can be enhanced [32].

The main disadvantage of DSLs is the additional cost of designing, implementing and maintaining it. DSL development is hard, requiring both domain and language development expertise, which few people have [41]. Meta-tools greatly reduce these additional costs. Another way of reducing them is by designing DSLs by re-using a base language.

4.2 The Meta-modeling Approach for Language Definition and Extension

To reduce the main disadvantage of DSLs, development costs, a model-driven approach has been chosen. DSL development approaches, in general, propose processes with several phases. For example [10] consists of 7 phases: *Decision*, *Domain analysis*, *Design*, *Implementation*, *Testing*, *Deployment*, *Maintenance*. This article concentrates on the *Design* and *Implementation* phases, as these are the essential phases for defining a DSL.

In the Meta-modeling for Language Definition approach [18], the abstract and concrete syntaxes are described using Meta-Models (MM). One way of describing operational semantics is by generating code from the new language to existing (programming) languages, and so producing an executable form of the model. The mapping between abstract and concrete syntax, and respectively between abstract syntax and semantics may be described using Model Transformations (MT).

To further reduce DSL development costs, a language re-use mechanism, namely extension, is used here. A *profile* is a generic extension mechanism for customizing base languages with constructs that are specific to particular domains, platforms. The Meta-modeling for Language Extension approach consists in extending (inheritance) the initial MMs describing the abstract and concrete syntaxes, and extending the semantics.

The DR DSML is designed and implemented as a profile, an extension, to an EAML. Noteworthy is the fact that the DR DSML definition and development approach, presented in the next section, is independent of the chosen EAML. However, to exemplify it, the ArchiMate language is chosen.

5 Extending Enterprise Architecture Modeling Languages with Collaborative Capabilities

Among EA frameworks, the focus here is The Open Group Architecture Framework (TOGAF) [49]. TOGAF provides methods for assisting in the acceptance, production, use and maintenance of an EA. At the core of TOGAF is the Architecture Development Method, consisting of eight phases and providing one of the most complete [45] guiding step-by-step processes for creating an EA. TOGAF is sometimes [50] considered one of the strongest frameworks on the business and technical layers, providing much detail for them.

One well recognized EAML is ArchiMate [48]. It shares [4] important concepts with TOGAF, and thus "the two are usable together". It models three phases of TOGAF Architecture Development Method into three layers: Business, Application and Technology. The general structure of models within the different layers is similar. The same types of concepts and relations are used, although their exact nature and granularity differ between layers. A viewpoint, according to ArchiMate [48],

is a means to focus on particular aspects of the architecture. Some viewpoints have a scope that is limited to a single layer; others link multiple layers. ArchiMate regulates interoperability between layers by defining possible dependencies. Due to its proximity with TOGAF, and its maturity, ArchiMate is retained here as the EAML for which the DR DSML is designed as an extension.

To define the DR DSML, the development approach proposed by [10] is followed (cf. Sect. 4.2). To *decide* whether the DR DSML should be defined, the need for collaboration in modeling EAs should be considered (cf. Sect. 3). As the *domain* to be *analyzed* is that of DR, the information that characterizes it has already been captured for DR formalization, for example by schemas (e.g. IBIS, QOC, DRL). The QOC [37] schema (cf. Fig. 1) is chosen, due to previous successful use, e.g. [52]. Noteworthy is that the remainder of the approach applies the same, if any other DR schema were chosen.

5.1 The Design of the Design Rationale Domain Specific Modeling Language

To *design* the DR DSML, the Meta-modeling for Language Definition approach (cf. Sect. 4.2) is followed. Its abstract syntax is defined as a Meta-Model (MM) inspired from the QOC schema. The MM, described in MOF, is presented in Fig. 2. During decision making, designers have to asses several proposals for solving an issue, comparing them against several criteria and finally choosing one resolution. The MM captures this by modeling an *Assessment* as containing one *Proposal* to one *Issue* (but, in general, one *Proposal* can be attached to several *Issues*, and one *Issue* usually has several *Proposals* - cf. the association between them). This proposal is assessed against one *Criterion* or more (which differ and depend on the type of *Issue* - the same *Criterion* can be attached to several *Issues*, and one *Issue* may have one *Criterion* or more - cf. the association between them). The *Proposal* could be chosen as *Resolution* or not. To connect the DR concepts with the EAML (ArchiMate) concepts, the *Annotation* is introduced as inheriting from *ArchimateElement*. In this way, the DR concepts, in turn inheriting from *Annotation*, are extending the EAML definition (i.e. the DR DSML is designed as an EAML profile), and can be processed by the EAML associated tools. One or more *Annotations* can be associated to one or more *ArchimateElements* (cf. the association between them in the MM, Fig. 2). *ArchimateElement* is a root concept in the ArchiMate MM (not in the standard definition, but in the Archi [2] implementation used for editor building).

Still for its *design*, the concrete syntax is defined. The graphical representations have been chosen following for the most [52]. They are presented in Fig. 3, which presents the palette of the graphical editor.

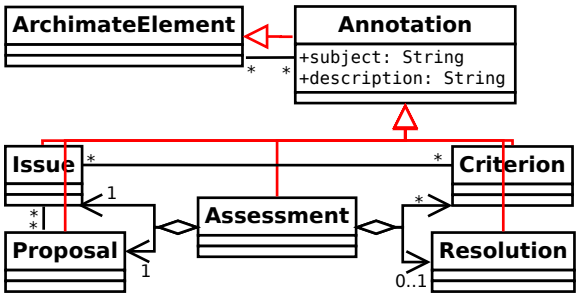


Fig. 2 The abstract syntax of the DR DSML.

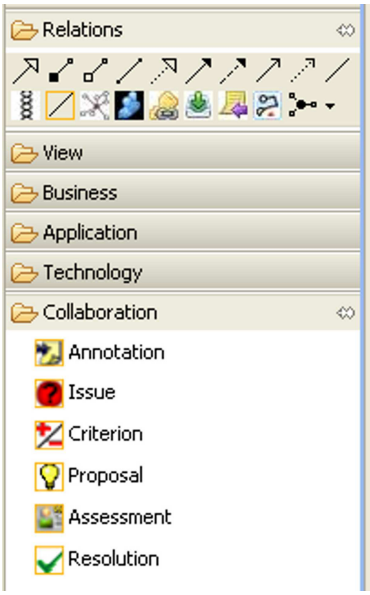


Fig. 3 The concrete syntax of the DR DSML.

5.2 The Implementation of the Design Rationale Domain Specific Modeling Language

For *implementation*, the semantics of a DR DSML are special, consisting in storing and retrieving the DR. Currently, the DR is stored in the model, and can be retrieved by direct visualization. Other retrieving systems can be integrated as off-the-shelf components.

Still related to *implementation* are language-specific tools. The most common one is the editor. Meta-tools targeted to the generation of graphical editors used in this project are the Eclipse Graphical Editing Framework (GEF), the Eclipse Modeling Framework (EMF). An existing open source ArchiMate graphical editor, Archi, developed as an Eclipse plug-in, has been selected and reused, expanded (more details about the plug-in extension process in [15]) with the Java code needed to describe the concrete syntax of the DR DSML extension.

The resulted editor presents the classical divisions of an Eclipse-based editor. Archi has a model navigator and an outline of the graphical model. Its central window presents views (defined as tabs) of the graphical model. The palette was extended with DR specific concepts (cf. Fig. 3), from which the designer can select,

drag and drop the desired concepts. The designer can associate any of these DR concepts to any of the ArchiMate model elements (cf. the DR MM in Fig. 2). The editor enforces the constraints defined in the DR MM on the models. For example, an *Assessment* must have exactly one *Issue*, and the editor does not allow the designer to associate e.g. two issues to the same assessment. Also, the editor allows that *Proposals* are associated only to *Issues*, and not, e.g. with *Criteria*. In this way, model validation is done in order to ensure the integrity and lack of ambiguity of the DR concepts.

By integrating the DR DSML with the system ML, the EAML, both in its language definition and associated tools, the designers' reluctance to capture raising issues is to be decreased, thus reducing the DR capture issue. To better show the integration and use of the DR DSML, an example is presented in the next section.

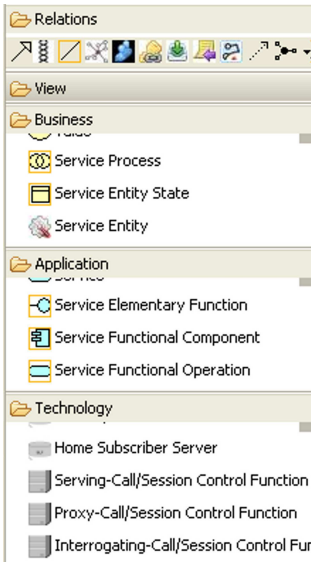


Fig. 6 Excerpt from the Telecom ArchiMate extension concrete syntax.

6 Application to Telecom Service Design

The example presented here comes from Telecommunications, but the use of the DR DSML is independent of the domain. It could have been an example from the automotive industry or health-care. The example is modeled using an EAML extension for Telecommunications. This extension is briefly introduced before the example.

6.1 An Enterprise Architecture Modeling Language Extension for Telecom Service Design

To define the Telecom DSML, the development approach proposed by [10] is followed (cf. Sect. 4.2). The *decision* to define it is sustained by requirements of Service Providers for Service Creation, including: domain specificity, rapid prototyping [22]. For *domain analysis*, informal discussions, MMs from literature (e.g. [5]) and modeling of service platforms (e.g. IMS [1]) were used to build initial Telecom MMs.

6.1.1 The Design of the Telecom Enterprise Architecture Modeling Language Extension

The initial Telecom MMs were used to extend the abstract syntax of the three layers of ArchiMate, following the approach introduced in [16]. This approach introduces three principles for expert-based MM extension. By applying these three principles, the initial ArchiMate MMs were extended with Telecom-specific concepts.

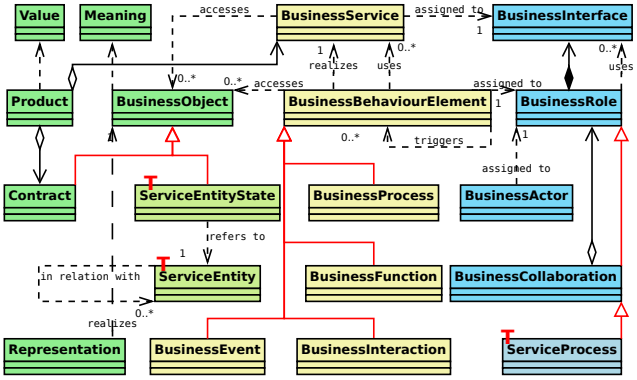
The extensions for the Business, Application and Technology layers of the ArchiMate abstract syntax were presented in [16], [14] and respectively [17]. The 3 MMs are grouped here for an overview. The entities specific to Telecom are marked in all figures with a red **T**. The Telecom-specific concepts of the Business (cf. Fig. 4(a)) and Application (cf. Fig. 4(b)) MMs are derived from the MMs proposed by [5]. The Telecom-specific concepts of the Technology (cf. Fig. 5) MM are modeled on the IP Multimedia Subsystem (IMS) [1]. IMS was chosen because it provides interfaces to traditional switch networks and to IP-based networks, the main technological platforms for implementing telecom services.

Still in the *design* phase, the concrete syntax was defined. For each concept and relation introduced in the MMs, a graphical representation was defined. An excerpt is presented in Fig. 6. For example, the three Telecommunications-specific concepts introduced in the Business MM: *ServiceEntityState*, *ServiceEntity* and *ServiceProcess* (cf. Fig. 4(a)) have graphical representations in the Business group from Fig. 6. In their definition, closeness to concepts from ArchiMate was emphasized, so that concepts that inherited those from ArchiMate have inherited their representation as well, with a mark of distinction (e.g. an orange border).

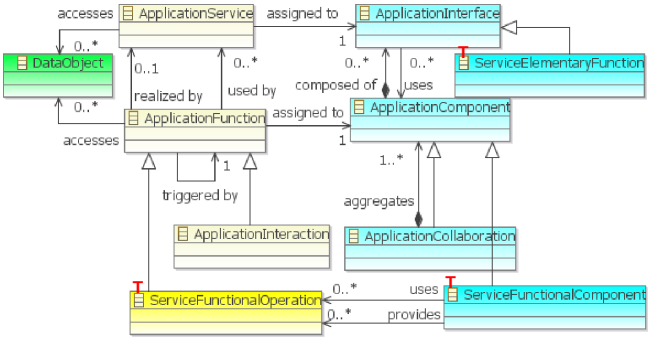
6.1.2 The Implementation of the Telecom Enterprise Architecture Modeling Language Extension

To *implement* the semantics of the Telecom extension, template-based code generation (model-to-text transformation) was chosen. This was due to the maturity of this approach, and the availability of powerful, mature, free tools. In this approach, the transformation of the input language is captured in rules called templates. The templates are defined using the Xpand template description language, and its editor and compiler, the free Eclipse plug-in OpenArchitectureWare. The static semantics is implemented, resulting in Java classes, with attributes, method signatures. For the behavioral semantics, method call is implemented, resulting in the possibility of translating sequence-like diagrams into code.

Related to *implementation* are language-specific tools. These are usually the editor and the compiler de-



(a) ArchiMate business MM extended by a Telecom profile (superscript T). More details in [16].



(b) ArchiMate application MM extended by a Telecom profile (superscript T). More details in [14].

Fig. 4 ArchiMate business and application MMs extended by a Telecom profile.

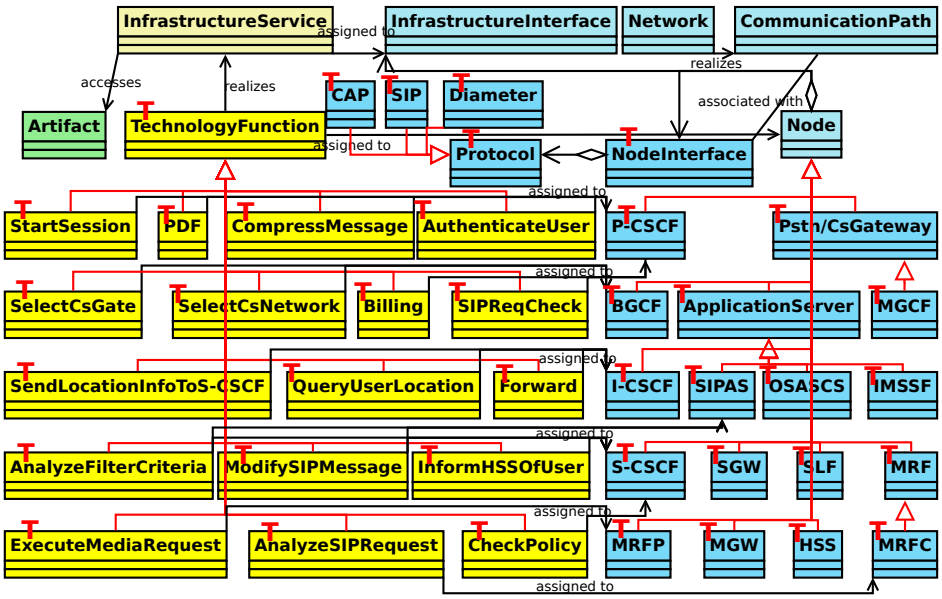


Fig. 5 ArchiMate technology MM extended by a Telecom profile (superscript T). More details in [17].

scribing the operational semantics. As for the DR DSML, Archi has been extended to include the concrete syntax of the Telecom ArchiMate extension. For compiler implementation, a meta-tool, OpenArchitectureWare (OAW), has been used. The templates for translating the Telecom extension of ArchiMate into Java, written in Xpand, describe the configuration of a compiler. The generated skeletons can be further detailed using Java libraries that abstract Telecom protocols, like JAIN. As the Metamodeling for Language Definition approach has been used, *maintaining* and evolving both DSML definitions benefits from the high automation degree introduced by MMs and meta-tools.

6.2 A Collaboration Example for Designing a Telecom Conferencing Service

To exemplify the DR and Telecom EAML extensions and their associated software tools, they are applied to the description of a Telecom service, a conference system. A conferencing service is a virtual meeting, done with the help of a set of telecommunications technologies (e.g., telephone, video, web), which allows two or more geographically remote locations to interact in real-time via two-way video and/or audio and/or text transmissions simultaneously. Notable examples include WebEx™, Skype™ for software solutions, Polycom®, Tandberg™ for hardware or complete solutions.

Because of space considerations, the focus of this example for the Application ArchiMate layer is limited to the phase of entering the conference (for a more complete version cf. [14]). This is the most difficult/important phase, when the signaling is established (cf. Fig. 7). The concepts of Application: 'ServiceFunctionalOperation', 'ServiceFunctionalComponent', and relations of 'triggering', 'assignment' used in Fig. 7 are those defined by ArchiMate and the Telecom MM extension (cf. Fig. 4(b) and Fig. 6). The *Client part of the conference*

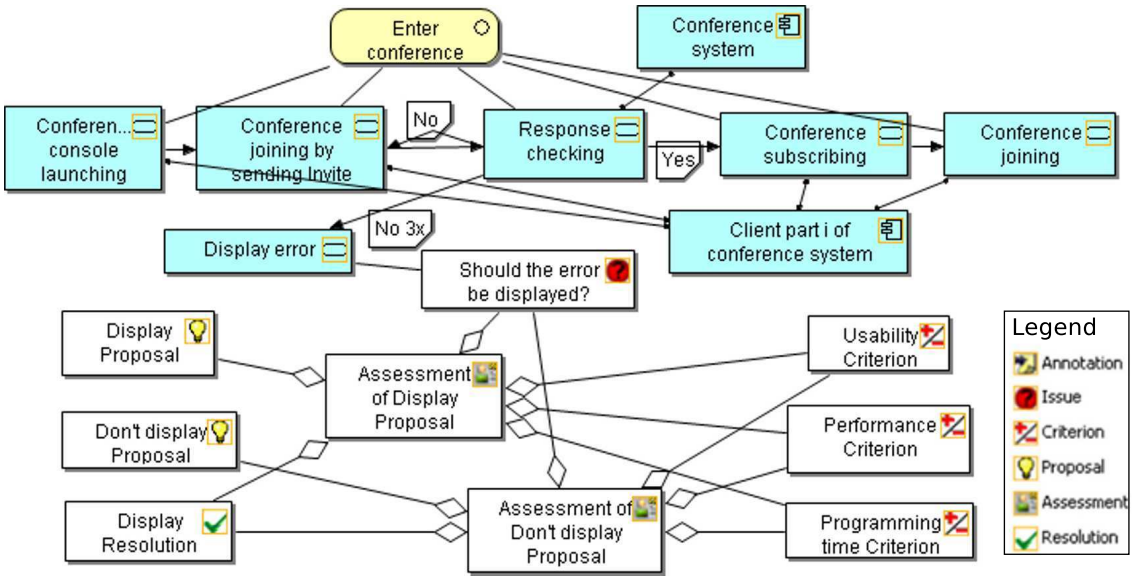


Fig. 7 Example of Design Rationale ArchiMate extension use with a conference service example developed at the Application layer of the Telecom ArchiMate extension.

system launches the console and tries to join a participant to the conference. If after three tries the connection is refused by the *Conference system*, an error message is displayed. If everything is alright, the participant is subscribed and joined to the conference.

However, displaying an error is not straightforward in a distributed system, and could constitute an issue (cf. *Should the error be displayed?* in Fig. 7). Not only *Usability*, but also system *Performance* and *Programming time* criteria should be considered. The basic proposals of *Displaying* and *Not displaying* are each assessed against all three criteria and the resolution is chosen to *Display* the error. The arguments, the rationale for this decision (no doubt related to giving much more importance, weight, to the *Usability* criterion than to the other two), is captured in the two *Assessments*. Of course, the same decision holds true for all errors, so the assessments can be capitalized upon, retrieved, for similar cases.

To give an idea of what kind of code can be obtained, an excerpt of the result of applying the template-based code generation that describes the semantics (cf. Sect. 6.1.2) of the Telecom ArchiMate extension is presented in Lst. 1. The excerpt shows part of the class *Clientpart1ofconferencesystem*, line 1 from Lst. 1, corresponding to the *Client part i of conference system* 'ServiceFunctionalComponent' from Fig. 4(b). Lines 16, 17, 18 from Lst. 1 are especially interesting because they show the method *joinconferencebysendingInvite* calling the *checkresponse* method of class *Conferencesystem*, corresponding (cf. Fig. 4(b)) to the 'triggering' between the *Conference joining by sending Invite* 'ServiceFunctionalOperation' of the *Client part i of conference system* 'ServiceFunctionalComponent' and the *Response checking* 'ServiceFunctionalOperation' of the *Conference system* 'ServiceFunctionalComponent'.

```
1  public class Clientpart1ofconferencesystem{
     private String name = "
         Clientpart1ofconferencesystem";
     private String id = "78d717c8";
     public Clientpart1ofconferencesystem(){}
     public void setName(String newName){
       this.name = newName;}
     public String getName(){
       return name;}
     public void setId(String newId){
       this.id = newId;}
     public String getId(){
       return id;}
     public void joinconference(){}
     public void launchconferenceconsole(){
       this.joinconferencebysendingInvite();}
     public void joinconferencebysendingInvite(){
       Conferencesystem conferencesystem = new
           Conferencesystem();
       conferencesystem.checkresponse();}
     public void displayerror(){}
     public void subscribetoconference(){
       this.joinconference();}}
```

Listing 1 Excerpt of generated Java code for the conference service at Application layer (cf. Fig. 4(b))

The example shows how the DR DSML can be used with a system DSML to capture, store (together with the system models) and retrieve (by direct visualization) design decisions and their rationale.

6.3 Discussion

Using the DR EAML extension, the designer is capable, when taking a decision, to document it. Thus, the integration of DR capture in the EAML helps capturing rationale. The model becomes more comprehensible, helping the knowledge transfer and collaboration.

Captured DR may be reused. Similar issues may reoccur in different parts of the same project or in different

projects. A resolution taken at a business or application layer will be applied consistently across the entire project or even enterprise. Should other resolutions be taken locally, in particular situations, the assessments would be visible and available at all layers. In this way, rationale, constraints can be transferred across layers, contributing to better decision-making in future models.

Although the integration of the DR DSML with the system ML, the EAML, should decrease the designers' reluctance to capture raising issues, another essential aspect for this is the usability of the tools. Without a user interface that allows and encourages creative freedom, the designers would not consistently use the graphical editor either to capture or retrieve DR. However, a certain degree of formality, structure is necessary, to enable verifications and assistance, thus reducing the number of errors. A more detailed discussion on the trade offs between a higher degree of formality and creative freedom, and achieving a compromise, blending both, has been done for example in [13].

From the perspective of supporting the evolution of a system, especially its technical architecture, this may drive stakeholders to change already taken, major decisions. For example, different architectural styles are eligible for the same kind of service (e.g. client-server or peer-to-peer styles). Most often, non-functional properties are the key elements to guide stakeholders in such a selection process [40]. Investigation if (and how) DR from an earlier architectural style should/could remain available in a newer one is still an open issue in our proposal.

7 Related Work

The related work is classified into four categories: collaborative methods used for Enterprise Modeling, DR and Model Driven Engineering, DR MMs used for EA, and EAML extensions.

Among collaborative methods used for Enterprise Modeling, two are [3], and [12]. The first one uses DEMO as an ML, which captures production and coordination acts of stakeholders in an enterprise. It introduces the possibility of simulating (to ensure validation and verification) an enterprise model through Petri nets. It concludes that a moderate number of elements, intuitive notations and a small set of elements to reduce the cognitive load of the stakeholders are crucial to encourage collaboration among them. Proposing six concepts and notations that have been, for the most part, previously used with success [52], the DR DSML answers adequately these requirements, and in a much more lightweight manner. The second method [12] uses UML and proposes a distributed, complex system for the creation, capture, storage, compilation and retrieval of engineering knowledge in the context of collaborative product design. While offering less options, the DR DSML required much less effort to define. Moreover, neither of these two collaborative methods focuses on DR.

DR has recently begun being a hot research topic in MDE, and even in the larger community of Software Engineering. For example, there have been proposals for integrating DR tools and modeling tools, e.g. for UML [33]. However, DR is not modeled as a DSL, not even as a MM, nor integrated as a UML profile. Another direction investigates linking design decisions to model elements [34], so that consistency checks and reuse of decisions are enabled. In the case of the EAML DR extension advanced in the present article, the linking is ensured by association relations. Yet another direction investigates DR evolution. As any other artifact, DR may evolve, thus impacting other design decisions and their DR. A MM to support design decision and DR evolution is proposed by [39]. However, although there is a MM proposed, DR is not modeled as a DSL. In none of these cases, direct reference of applying DR for EA is made.

However, proposals of modeling DR with MMs and using it for EA do exist. For example, [53] proposes a graphical notation (and the implied MM) for using DR to capture system architecture evolution. Although not presented as such, it is practically a DR DSML. It is used in a case study to model something very similar to a business process. However, the DR DSML is not defined as an EAML extension. Another approach introduces a formal language, an extension of influence diagrams, to support the analysis of EAs, of goals and decision alternatives [28]. However, this language is not integrated into an EAML. Another MM for capturing DR [54] is integrated in a conceptual framework for proactive decision identification, enforcement and decision maker collaboration. The approach is applied to the design of enterprise applications employing Service Oriented Architecture as their primary architecture style. However, they do not use the MM to develop a DSL.

There are several ArchiMate proposed extensions. For example, one [43] introduces an extension for modeling and managing motivation, principles and requirements in TOGAF. Another [30] argues for an extension for modeling TOGAF's implementation and migration phases. However, none of them deals with DR.

In conclusion, even if collaborative methods for Enterprise Modeling exist, they do not address DR. Even if some preliminary work on modeling DR as MMs/DSMLs and using them for EA exist, such DR MMs are not defined as extensions of EAMLs, thus not addressing the DR capture issue. Even if EAML extensions exist, they do not address collaboration/DR. One essential contribution of our approach consists therefore in synthesizing coherently all these themes.

8 Conclusions and Perspectives

In Enterprise Modeling there are many stakeholders involved, with very different backgrounds, but all of them (should) take decisions that are backed up by arguments.

This article presented a DSML (e.g. Meta-Model, concrete syntax, semantics, graphical editor) for capturing these decisions and the argumentation, rationale, behind them and an example of its usage with the design of a Telecom service. DR being an orthogonal, transversal concern, the DSML is independent of the domain in which the enterprise is active. On the other hand, the DR DSML can be integrated, composed with system DSMLs. The DR DSML was defined as an extension for an EAML, increasing the integration with EA. The Meta-modeling for Language Definition approach (Model Driven Engineering) has been followed for its development, thus reducing development and maintenance costs. The DR DSML promotes coordination among stakeholders, facilitates participation and collaboration, enables building consensus. This results in the enterprise's increased reactivity so as to enhance performance and reduce costs for the enterprise.

To further take advantage of the captured decisions and rationale, DR engines for retrieval can be integrated as a means to implement the DR DSML semantics. Of course, DR covers only one aspect of collaboration, other methods (e.g. multi-writer, real-time, distributed collaborative editors that also include VoIP, chat, white-board, and screen sharing functionality), or dedicated hardware (e.g. multi-touch table, interactive white-boards), or automatic extraction of collaboration information from recordings can be envisioned as applicable to EA. An important development of the DR DSML is integration in a larger Decision Support System. Such a system could help balancing and taking multi-criteria decisions [9] [6]. The current DR DSML could be extended to include concepts like decision-maker, preference, algorithm.

Although exemplified on an EAML Telecom extension, the DR EAML extension is independent of domain. It can be applied to capture decisions and their argumentation in any other field. Also, the DR DSML may be designed as an extension of other Modeling Languages, be they EAMLs or more general purpose ones (e.g. UML).

References

1. 3GPP. TS 23.228 V10.3.1 IP Multimedia Subsystem (IMS) Stage 2 (Release 10), 2010.
2. Archi. `http://archi.cetis.ac.uk/`, accessed on 27.05.2012.
3. J. Barjis. *Collaborative, Participative and Interactive Enterprise Modeling*, volume 24 of *LNBIP*, pages 651–662. Springer, 2009.
4. G. Berrisford and M. Lankhorst. Using ArchiMate with TOGAF–Part 1: Answers to nine general questions about methods. *Via Nova Architectura*, 2009.
5. E. Bertin. *Architecture of communication services in a convergence context (in French)*. PhD thesis, National Institut of Telecommunications and Pierre and Marie Curie University - Paris 6, Paris, 2009.
6. S. Bigaret and P. Meyer. Diviz: an MCDA workflow design, execution and sharing tool. In *25th Mini-EURO Conf. Uncertainty and Robustness in Planning and Decision Making (URPDM)*, Coimbra, 2010.
7. G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide*. Addison-Wesley Professional, Reading, MA, USA, 2005.
8. V. Boucharas, M. van Steenbergen, S. Jansen, and S. Brinkkemper. The Contribution of Enterprise Architecture to the Achievement of Organizational Goals: A Review of the Evidence. In *TEAR*, pages 1–15, 2010.
9. D. Bouyssou, T. Marchant, M. Pirlot, A. Tsoukis, and P. Vincke. *Evaluation and decision models with multiple criteria: Stepping stones for the analyst*. International Series in Operations Research and Management Science, Volume 86. Boston, 1st edition, 2006.
10. I. Ceh, M. Crepinsek, T. Kosar, and M. Mernik. Ontology driven development of domain-specific languages. *Comput. Sci. Inf. Syst.*, 8(2):317–342, 2011.
11. D. Chen, G. Doumeingts, and F. Vernadat. Architectures for enterprise integration and interop: Past, present and future. *Comput. Ind.*, 59:647–659, 2008.
12. Y.-J. Chen, Y.-M. Chen, and H.-C. Chu. Enabling collaborative product design through distributed engineering knowledge management. *Comput. Ind.*, 59:395–409, 2008.
13. V. Chiprianov, Y. Kermarrec, and S. Rouvrais. Metatools for Software Language Engineering: A Flexible Collaborative Modeling Language for Efficient Telecommunications Service Design. In *FlexiTools2010 32nd International Conference on Software Engineering Workshop on Flexible Modeling Tools*, Cape Town, SA, 2010.
14. V. Chiprianov, Y. Kermarrec, and S. Rouvrais. Extending Enterprise Architecture Modeling Languages: Application to Telecommunications Service Creation. In *9th Enterprise Engineering track at 27th ACM Symposium on Applied Computing (SAC)*, volume 2, pages 1661–1666, Trento, Italy, 2011. ISBN 978-1-4503-0857-1.
15. V. Chiprianov, Y. Kermarrec, and S. Rouvrais. On the Extensibility of Plug-ins. In *6th International Conference on Software Engineering Advances (ICSEA)*, pages 557–562, Barcelona, Spain, 2011. ISBN 978-1-61208-165-6.
16. V. Chiprianov, Y. Kermarrec, and S. Rouvrais. Practical Model Extension for Modeling Language Profiles. An Enterprise Architecture Modeling Language Extension for Telecommunications Service Creation. In *French Coloquim in MDE, IDM*, pages 85–91, 2011. ISBN 978-2-917490-15-0.
17. V. Chiprianov, Y. Kermarrec, and S. Rouvrais. Telecommunications Service Creation: Towards Extensions for Enterprise Architecture Modeling Languages. In *6th Intl. Conf. on Software and Data Technologies*, volume 1, pages 23–29, Seville, Spain, 2011.
18. T. Clark, A. Evans, S. Kent, and P. Sammut. The MMF approach to engineering object-oriented design languages. In *Ws. on Language Descriptions, Tools and Applications (LDTA)*, Genova, Italy, 2001.
19. A. Dutoit, R. McCall, I. Mistrk, and B. Paech. Rationale management in software engineering: Concepts and techniques. In A. Dutoit, R. McCall, I. Mistrk, and B. Paech, editors, *Rationale Management in Software Engineering*, pages 1–48. Springer, 2006.
20. D. Falessi, G. Cantone, R. Kazman, and P. Kruchten. Decision-making techniques for software architecture de-

sign: A comparative survey. *ACM Comput. Surv.*, 43:33:1–33:28, October 2011.
21. A. Fatolahi and F. Shams. An investigation into applying UML to the Zachman framework. *Information Systems Frontiers*, 8:133–143, 2006.
22. J. Hållstrand and D. Martin. Industrial requirements on a service creation environment. In *Proc. of the 2nd Intl Conf. on Intelligence in Broadband Services and Networks: Towards a Pan-European Telecommunication Service Infrastructure*, pages 17–25, London, UK, 1994.
23. Xiao He, Zhiyi Ma, Weizhong Shao, and Ge Li. A metamodel for the notation of graphical modeling languages. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, pages 219 –224, july 2007.
24. IEEE Computer Society. IEEE Recommended Practice for Architectural Description of Software Intensive Systems. IEEE Standard 1471-2000, 2000.
25. ISO. ISO 15704:2000 Industrial automation systems - Requirements for enterprise-reference architectures and methodologies, 2000.
26. ISO. ISO 19439:2006 Enterprise integration - Framework for enterprise modelling, 2006.
27. ISO/IEC. ISO/IEC FDIS 42010. Systems and software engineering Architecture description, 2007.
28. P. Johnson, R. Lagerstrom, P. Narman, and M. Simonsson. Enterprise architecture analysis with extended influence diagrams. *Information Systems Frontiers*, 9:163–180, 2007.
29. H. Jonkers, M. Lankhorst, R. van Buuren, M. Bonsangue, and L. van der Torre. Concepts for modeling enterprise architectures. *Intl. Journal of Cooperative Information Systems*, 13:257–287, 2004.
30. H. Jonkers, H. van den Berg, M. E. Iacob, and D. Quartel. ArchiMate Extension for Modeling the TOGAF Implementation and Migration Phases. Technical report, The Open Group, Catalog number W111, 2010.
31. Henk Jonkers, Marc Lankhorst, Hugo ter Doest, Farhad Arbab, Hans Bosma, and Roel Wieringa. Enterprise architecture: Management tool and blueprint for the organisation. *Information Systems Frontiers*, 8:63–66, 2006.
32. R. B. Kieburtz, L. McKinney, J. M. Bell, J. Hook, A. Kotov, J. Lewis, D. P. Oliva, T. Sheard, I. Smith, and L. Walton. A software engineering experiment in software component generation. In *Proc. of the 18th Intl. Conf. on Software Engineering*, pages 542–552, 1996.
33. P. Konemann. Integrating decision management with UML modeling concepts and tools. In *European Conf. on Software Architecture. WICSA/ECSA. Joint Working IEEE/IFIP*, pages 297 –300, 2009.
34. P. Konemann and O. Zimmermann. Linking design decisions to design models in model-based software development. In *Proc. of the 4th European Conf. on Software Architecture*, ECSA, pages 246–262. Springer, 2010.
35. T. Kosar, N. Oliveira, M. Mernik, V.J.M. Pereira, M. Črepinšek, C.D. Da, and R.P. Henriques. Comparing general-purpose and domain-specific languages: An empirical study. *Computer Science and Information Systems*, 7(2):247–264, 2010.
36. H. Lochmann and A. Hessellund. An integrated view on modeling with multiple domain-specific languages. In *Software Engineering*. ACTA Press, 2009.
37. A. MacLean, R. M. Young, V. M. E. Bellotti, and T. P. Moran. Design rationale. chapter Questions, options, and criteria: elements of design space analysis, pages 53–105. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1996.
38. Z Mahmood. Frameworks and tools for building enterprise information architectures. In *Proc. of 6th Intl. IBIMA Conf on managing Information In Digital Society*, pages 216–226, 2006.
39. I. Malavolta, H. Muccini, and Smrithi R. V. Supporting Architectural Design Decisions Evolution through Model Driven Engineering. In Elena Troubitsyna, editor, *Software Engineering for Resilient Systems*, volume 6968 of *LNCS*, pages 63–77. Springer, 2011.
40. Julien Mallet and Siegfried Rouvrais. Style-based model transformation for early extrafunctional analysis of distributed systems. In *QoSA*, pages 55–70, 2008.
41. M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37:316–344, 2005.
42. E. Niemi. Enterprise architecture benefits: Perceptions from literature and practice. *Internet & Information Systems in the Digital Age: Challenges and Solutions*, pages 161–168, 2006.
43. D. Quartel, W. Engelsman, H. Jonkers, and M. van Sinderen. A Goal-Oriented Requirements Modelling Language for Enterprise Architecture. In *IEEE Intl. Enterprise Distributed Object Computing Conf. (EDOC)*, pages 3 –13, Auckland, New Zealand, 2009.
44. M. Schoenherr. Towards a common terminology in the discipline of enterprise architecture. In George Feuerlicht and Winfried Lamersdorf, editors, *Service-Oriented Computing -ICSOC Ws.*, pages 400–413. Springer, 2009.
45. R. Sessions. Comparison of the Top Four Enterprise Architecture Methodologies. Technical report, ObjectWatch, Inc., 2007.
46. J. Simonin, F. Alizon, J.-P. Deschrevel, Y. Le Traon, J.-M. Jezequel, and B. Nicolas. EA4UP: An Enterprise Architecture-Assisted Telecom Service Development Method. In *12th Intl IEEE Enterprise Distributed Object Computing Conference*, pages 279 –285, 2008.
47. J. Simonin, E. Bertin, Y. Le Traon, J.-M. Jezequel, and N. Crespi. Analysis and improvement of the alignment between business and information system for telecom services. *Intl J. On Advances in Sw*, 4(1):117–128, 2011.
48. The Open Group. ArchiMate 1.0 Specification, 2009.
49. The Open Group. TOGAF Version 9, 2009.
50. L. Urbaczewski and S. Mrdalj. A comparison of enterprise architecture frameworks. *Issues in Information Systems*, 7(2):18–23, 2006.
51. A. van Deursen, P. Klint, and J. Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, 2000.
52. T. Wolf. *Rationale-based Unified Software Engineering Model*. VDM Verlag, Saarbrucken, Germany, 2008.
53. A. Zalewski, S. Kijas, and D. Sokolowska. Capturing Architecture Evolution with Maps of Architectural Decisions 2.0. In I. Crnkovic, V. Gruhn, and M. Book, editors, *Software Architecture*, volume 6903 of *LNCS*, pages 83–96. Springer, 2011.
54. O. Zimmermann, T. Gschwind, J. Kuster, F. Leymann, and N. Schuster. Reusable architectural decision models for enterprise application development. *Sw Architectures, Components, and Applications*, pages 15–32, 2007.