Towards the State of the Art of Extending Enterprise Modeling Languages

Richard Braun
TU Dresden, Chair for Wirtschaftsinformatik, esp. System Development, 01062 Dresden, Germany

Keywords: Enterprise Modeling, Modeling Extension, Meta Modeling, Extensibility, Meta Model Enhancement.

Abstract: In the previous decade, more and more de facto standards of enterprise modeling languages (EML) evolved. The establishment of EMLs leads naturally to an increasing number of EML extensions in order to integrate requirements and needs from specific problems or domains in an EML. Thus, EML extensibility is proposed as a relevant topic within both the field of meta modeling and enterprise modeling. We therefore conducted an analysis of existing meta modeling languages and well known EML languages in order to derive the current state of the art in terms of EML extensibility. In addition to that, classification schemes for extension purposes and extension mechanisms are presented. Finally, topics for further research are proclaimed in order to facilitate more research on language extensibility.

1 INTRODUCTION AND MOTIVATION

Generally, Enterprise Modeling Languages (EML) are of primary importance for conceptual and technical complexity management within enterprises (Frank, 1999). For instance, EMLs are used within enterprise architecture frameworks to facilitate the management of different views (perspectives), aspects and several levels of abstraction in an integrated manner (Frank, 2002; Braun and Winter, 2005). EMLs can also provide the fundament for model-driven engineering approaches (Atkinson and Kuhne, 2003). In the course of the last two decades, several EMLs such as BPMN (Chinosi and Trombetta, 2012), ARIS (Scheer and Nüttgens, 2000) or ArchiMate (Lankhorst et al., 2009) evolved. Also, the general-purpose modeling language UML is often applied in the context of enterprise modeling.

As it is well known from software engineering (e.g., in the field of ERP systems (Botta-Genoulaz et al., 2005)), the prevalence of these standards leads consequently to an increase of situations where EMLs need to be customized or extended in order of satisfy specific requirements coming from the peculiarities of a specific industry, business or - more generally - domain-specific problem. The increasing number of EML extensions provides evidence for this assumption (Braun and Esswein, 2014a; Pardillo, 2010). Thus, it is necessary to address the issue of extending EMLs in detail due to the following reasons:

Domain-specific Configuration. There are numerous reasons and scenarios that require the extension of an EML. For instance, enhancing an EML with analytical concepts (e.g., for performance measurement), adding a new perspective to the EML (e.g., resource concepts in process models) or due to reasons of interdisciplinarity (e.g., adapting process models in manufacturing (Braun and Esswein, 2014b)). Further, EMLs in the field of enterprise architecture management are typically very abstract and under-specified, wherefore domain-specific extension and customization become necessary (Malavolta et al., 2013). This situation also occurs when the general-purpose modeling language UML needs to be extended for specific platforms or domains (Pardillo, 2010). Extensibility of EML is extremely relevant in the enterprise modeling domain to the vast amount of stakeholders and their perspectives to the enterprise (Atkinson et al., 2013).

Managing Language Complexity and Method Pluralisms. The design of universal languages, that cover all possible purposes within enterprise modeling or within one domain is barely shapeable and only theoretically feasible, since different stakeholders use different languages (Becker, 2014). Besides, there is a traditional trade-off between language complexity and language understanding (Malavolta et al., 2013; zur Muehlen and Recker, 2008). Hence, it seems to be more reasonable to proclaim the usage of a small set of popular EMLs and extend them domain-specifically (Loos et al., 2013). Thus, it might be possible to unify the EML language area and provide precise mechanisms for their extension instead of designing “Yet Another Modeling Language”, which possibly has even huge redundant overlaps with basal concepts from standard EMLs.

Interoperability and Tool Support. There are also several aspects from a technical point of view that strengthen the relevance of EML extensibility. For instance, modeling tools need to be extensible in order to facilitate EML extensions in a standard manner while ensuring a valid language core (see the proposed plugin mechanism stated in (Atkinson et al., 2013)). Also, the exchange of EML extensions between two modeling tools should be possible in order to allow interoperability and data exchange. Thus, not only machine-to-machine communication is facilitated, but also communication between human stakeholders.

Language Evaluation. Extensibility of EMLs comprises also the opportunity of improving and evolving an EML. For instance, commonly occurring extensions can be seen as evidence for changing the EML in some aspects. For example, Decker et al. (2007) proposed a choreography extension of BPMN that was widely integrated in BPMN 2.0 (Decker et al., 2007).

All in all, we argue that extensibility of EMLs is an important topic within enterprise modeling; mainly due to reasons of separation of concern, interoperability, avoidance of redundancy and finally better communication at all. Within this article we aim to consider the current state of the art in terms of extensibility of EML in order to set up a foundation for further research and the discussion of benefits and shortcomings within the research community. Therefore, we investigate extension mechanisms of both existing EMLs and some meta modeling languages such as MOF, MEMO and E3. Based on the theoretical analysis of both the literature and specifications of several EMLs, we finally outline some classifications for extension mechanisms and extension techniques.

The remainder of this article is as follows: Section 2 provides fundamental aspects and defines the term extension. Afterwards, Section 3 examines meta modeling languages regarding their explicit provision of extension mechanisms. Consequently, Section 4 considers the extensibility of modeling languages. Section 5 provides an classification of extension purposes and mechanisms. The paper ends with an explication of proposed further research topics in this area of discourse.

2 FUNDAMENTALS

2.1 Definition

Generally, the author follows the language-based meta modeling definition (Kühne, 2006; Strahringer, 1998) and thus differentiates between four levels of abstraction what is well known from OMG architecture: The meta meta modeling level (M3), the meta modeling level (M2), the model level (M1) and the object level (M1). In detail, an EML is understood as part of an enterprise modeling method (Greiffenberg, 2004). Within this research-in-progress article, we focus on the language grammar (syntax) and explicitly on any kind of extensibility of both the abstract and the concrete syntax (see Figure 1). As stated in Section 1, we both consider several EMLs on level M2 and general extensibility possibilities on level M3.

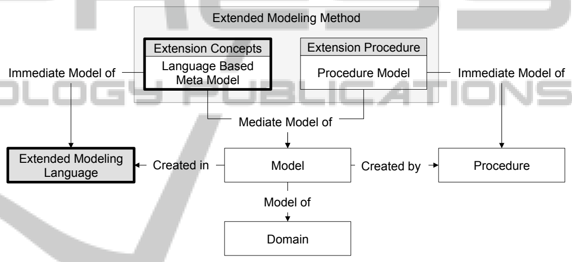

Figure 1: Elements and instances of an extended modeling method (referring to (Greiffenberg, 2004)). The focus of this research paper lies merely on syntactical aspects of modeling languages (thick border lines).

Currently, there is a lack of a common EML extension definition. Weisemöller and Schürr (2008) define an extension in the context of MOF as follows: Let m_1, m_2 be meta models defined in MOF and p_1, p_2 their outermost packages. M_1 is called an extension of m_1 if m_1 != m_2 and there exists a package p_{ext} such that p_2 is the result of merging p_{ext} into p_1.

However, that definitional approach is limited to the package-related MOF. More general, an **extension** can be understood as an enhancement of an EML with functionality according to the (domain-specific) needs or requirements of language users. Generally, an extension is neither useful nor functional on its own and depends on the extended language (host language). Further, a **standard-conform extension** is understood as an extension that either follows the well-defined extension mechanism of an EML or at least do not contradict both syntax and semantics of the language (Kopp et al., 2011). Thus, it is also possible to cover meta model changes as EML extension, although a well-defined extension mechanism is missing.

2.2 Related Work

To the best of our knowledge, only very research articles explicitly address the issue of EML extensibility so far. **Atkinson et al. (2013)** emphasize the relevance of the topic for enterprise modeling due to the vast amount of domain-specific views and stakeholders. Also, the authors discuss some pros and cons of existing extension approaches (in-built, meta model customization and model annotation). The authors state the evident lack of suitable tool support for all of these approaches since modeling tools only (if indeed) provided hardwired, dedicated meta model changeability. Thus, Atkinson et al. (2013) proclaim the consideration of the following design principles regarding EML extensibility: Separation of concerns; low coupling between host language and extensions as well as high cohesion within an extension. Consequently, the authors introduce the OCA multi-level modeling approach in the context of EML extensions. This approach allows arbitrary ontological extensions of a language while run-time. However, their research is very engineering-driven and mainly focusses on the concrete syntax and the definition of user-specfic views (Atkinson et al., 2013).

Braun and Esswein (2014) conduct a systematic review of existing BPMN extensions and provide a framework for the analysis of extensions. Further, their analysis include the consideration of the applied methodology that leads to any proposed extension. The authors state that only very few extensions are conform to the BPMN standard and they also constitute the need for an integrated methodological support for language design and construction (Braun and Esswein, 2014a).

Moreover, **Kopp et al. (2011)** and **Pardillo (2010)** examine BPEL extension respectively UML extensions and profiles. Both approaches less focus on enterprise modeling and merely provide statistical analysis of published extensions.

3 EXTENSIBILITY IN META MODELING LANGUAGES

Analyzing EML extensibility also requires the consideration of existing meta meta modeling approaches (languages defined on the M3 level), since these meta languages might provide some advises or mechanisms for extending languages (Frank, 2013).

3.1 MOF

Meta Object Facility (MOF) is the prevalent standard for the definition of meta data and hence it is a popular instrument for meta modeling, for example in the field of model-driven engineering (Kühne, 2006). Consequently, MOF is located on the topmost layer and its primary responsibility is to define a language for the specification of a meta model (OMG, 2014b). MOF can be divided into an essential version and a complete version of the language (EMOF and CMOF). Capabilities of MOF are organized in four main packages: *Common*, *Reflection*, *Identifier* and *Extension*. This package structure already indicates the relevance of extensibility and MOF explicitly aims to provide a simple definition of meta model extensions (OMG, 2014b, p. 5). Basically this is facilitated by package merging and the shared usage of the *Common Core* package (OMG, 2014b, p. 7).

However, MOF only provides very simple means for extending model elements by providing **name-value-pairs** within the *Extension* package (OMG, 2014b, p. 3). Thus, tags on *Elements* of the meta model can be realized. Actually, this is the only concept within the EMOF *Extension* package. The CMOF *Extension* package merges the EMOF *Extension* package and adds the extension owner concept.

Next to this simple tagging approach, there seems to be the possibility of a **profile mechanism**. This is not directly explicated in the MOF specification, but rather interpretable from the specification of *UML Infrastructure Library* which states that its "profile mechanism may be used with any meta model that is created from MOF" (OMG, 2011c, p. 185). This statement indicates, that each language defined by the MOF can make use of the profile mechanism that is commonly used in UML modeling (Pardillo, 2010). The only condition for that is a required reference of the profile to the respective meta model (OMG, 2011c, p. 176).

The mentioned statement arises the question, why MOF does not define the profile mechanism on level M3. The answer comes from the pretty strange architecture of MOF: MOF merges packages that are defined within the *UML Infrastructure Library* on level M2. Strictly speaking, the *Infrastructure Library* acts both as a meta meta model (M3) and as meta model (M2), what leads to a obvious conflict of abstraction. Even when accepting these crude architecture, there still remains the problem that MOF actual does not have any valid syntactical access to the *Profile* package. MOF only merges the packages *InfrastructureLibrary::Core::Basic* and *InfrastructureLibrary::Core::Constructs*; but not the *Profiles* package. Also, it is confusing that the *Profiles* packages is associated with the *Core* package (OMG, 2011c, p. 173), although it is a separate package outside the *Core* (OMG, 2011c, p. 12). Obviously, there is some confusion and the syntax contradicts the above mentioned indications that the profile mechanism can be used by any meta model created from MOF (see (OMG, 2011c, p. 185))!

At this point, both MOF and UML suffer from an obscure differentiation between the abstraction layers. It seems to be that the *Infrastructure Library* defines the UML and MOF is only responsible for some elementary concepts and tool support. Due to the focus on meta data definition and the mixture of meta modeling and tool concepts, MOF provokes some problems in the enterprise modeling domain (Frank, 2008).

All in all, we can conclude that MOF provides a simple name-value-pair extension of each element on M3 level. Further, there is confusion on the stated profile mechanism. Based on the MOF meta model we do not see any applicability of that mechanism for MOF defined meta models at all. This is also strengthened by the fact that none of the MOF based languages uses profiles (see Section 4).

3.2 MEMO

MEMO (Multi-Perspective Enterprise Modeling) is a framework for enterprise modeling and the integration of several views, perspectives and aspects of enterprises (Frank, 2002). The framework provides the meta modeling language MEMO MML for the integrated specification of domain-specific modeling languages (Frank, 2008, p. 22) and also provides a procedural description of required design and construction steps.

The MEMO framework does not provide a dedicated extension mechanism, as its capabilities are designed for the precise definition of domain-specific languages, which spares extensibility (Frank, 2002, p. 3), (Frank, 2008, p. 30). Consequently, MEMO proclaims that each adaption of a language has to be conducted on the M2 level (meta model) by language engineers that are usually part of the domain-specific context (Frank, 2013, p. 2). Nevertheless, the MEMO author emphasizes the general necessity of application-specific language extensibility (Frank, 2013, p. 15).

3.3 E3

The E3 meta modeling approach is not very acquainted but provides a well-defined meta modeling language for the integrated definition of EMLs. The E3 framework provides an integrated meta meta model for the definition of EMLs by integrating contextual aspects (e.g., objects and its properties) and presentational aspects (e.g., views and different presentations of a context object (Greiffenberg, 2004)). The E3 framework intends to provide a generic framework for language specification at all and integrates all relevant aspects of language definition. In contrast to other approaches, it also proposes required process steps within language design in order support language engineers. However, extensibility of instantiated languages is not considered. Rather it implies direct adaptation on M2 level (meta model).

3.4 GME

The General Modeling Environment (GME) is a toolkit for the creation of domain-specific modeling languages and the integration of heterogeneous models (Ledeczi et al., 2001). The meta meta language MetaGME is the kernel of the toolkit. MetaGME bases on UML and OCL and provides generic concepts for language definition such as *Models*, *Parts* and so-called *First Class Objects* (Ledeczi et al., 2001). GME provides minor guidance for language creation but lacks in any consideration of language extensibility. Moreover, it is up to the language engineer to alter a meta model.

3.5 Gmodel

The objective of the Gmodel meta modeling language is a compact, modular and extensible design of domain-specific modeling languages that facilitate model integration and interoperability (Bettin and Clark, 2010). The stated extensibility is enabled by the Gmodel architecture that allows multi-level modeling instead of the fixed four layered architecture of OMG. Therefore, Gmodel proposes so-called *Semantic Identities* on M3 level and also allows the integration of typed links on this level.

3.6 Conclusion

Reviewing the stated approaches reveals a lack of suitable and more sophisticated mechanisms of extensibility on the level of meta modeling languages. Only MOF and Gmodel consider extensibility to a minor degree: MOF provides fundamental concepts for the definition of new attributes. Gmodel facilitates vertical extensibility by multi-level modeling that is additionally propose by Atkinson et al. (2013). Generally, the approaches leave language extensions open to the language engineer and his alterations and customizations of a meta model. However, missing guidance and borders cause ad hoc changes depending on single engineers, which is only suitable if there is no noteworthy dissemination of the language and no need for comparison, integration or tool implementation.

4 EXTENSIBILITY IN EML

4.1 UML

Due to the dissemination of the Unified Modeling Language (UML), many extensions and adaptions of the language exist (Weisemöller and Schürr, 2008). They are mainly used for domain-specific view points, model transformation (MDA) and model analysis (Selic, 2007). UML explicitly allows meta model modification for custom purposes (heavy-weight extension), but provides no detailed recommendations on that. On the contrary, the lightweight extension mechanism enables the definition of specific profiles within UML models on level M1. Profiles are limited extensions of referenced meta models with the purpose of adapting the meta model to a specific platform or domain (OMG, 2011c, p. 183). Thus, it can be seen as an “extension by addition” mechanism that generates UML dialects (OMG, 2011c, p. 23). It is important to emphasize that profiles do not change the UML meta model but rather its instances. Actually, an “intermediate meta level (M1.5)” is produced which enhances the vocabulary of UML without changing the meta model. Therefore, several concepts are included within UML meta model: *Stereotypes* (extend meta classes of the UML), *Tag Definitions* (attributes of Stereotypes), *Constraints* and some relationship types. Strict filtering options provide the chance to define tight DSMLs.

The concrete syntax of UML can be extended by the annotation of specific icons to *Stereotypes*. Structured icons are not possible. While the UML specification does not provide procedural guidance for the design of *Profiles*, there are several research papers addressing this issue (Selic, 2007; Lagarde et al., 2008). UML profile mechanism is very popular due its simple applicability and tool support. However, the limited expressiveness of the approach might lead to the definition of complex (and thus expansive) OCL statements (Weisemöller and Schürr, 2008).

4.2 Languages Defined with MOF

As stated in the previous section, MOF is a common meta modeling language specified by the Object Management Group (OMG). Hence, we have analyzed a range of business-oriented OMG languages. None of the analyzed languages uses the MOF extension mechanisms or re-uses the profile mechanism (see Section 3.1)! The majority of the languages does not provide extensibility and only BPMN, Essence and KDM provide appropriate concepts (see Figure 2). CMMN and DMN only emphasize forbidden extension operations within its models but do not provide a conceptual understanding of extensibility. Neither SBVR, VDML nor SysML consider extensibility. IFML states some kind of black and white lists of elements that can be extended or not.

		Syntax		Procedure
		Abstract	Concrete	
OMG languages	**UML**	●	⊙	⊙
	BPMN	●	⊙	⊙
	CMMN	–	–	–
	DMW	○	–	–
	SBVR	○	–	–
	KDML	○	–	–
	Essence	●	–	–
	IFML	⊙	–	–
	SysML	○	–	–
	KDM	●	⊙	⊙
	SMM	○	–	–
⋮	**ArchiMate**	⊙	–	–
	ARIS	○	–	–

● (exists) ⊙ (partially) – (does not exist)

Figure 2: Extensibility of abstract and concrete syntax within MOF-based languages and their provision of methodical support for extension design.

Essence introduces the *Extension Element* concept and OCL based *Extension Functions* in order to change attributes of elements (OMG, 2014a, p. 60-65). In case of extension, the extended elements remain oblivious of its modification. This “non-destructive” kind indicates a run-time extension on model level (OMG, 2014a, p. 97). **SMM** defines simple concepts (tag-value-pairs and textual annotations) for the implementation of extension of every SMM element on model level (OMG, 2012).

KDM (Knowledge Discovery Meta-Model) both provides an extension of the language itself (level M2) and its instances (level M1). For the first one, a new package within KDM can be defined with mandatory components (*Model*, *Abstract Element* and *Abstract Relationship*). For the latter, a dedicated mechanism is specified, that provides elements for the definition of extension classes, possible extension value classes and simple key-value-pairs as annotations (OMG, 2011b, p. 39). Although the architecture and the semantics of the stated extension concepts are quite similar to UML, KDM declares its own definition.

4.3 BPMN

BPMN (Business Process Model and Notation) is the current de facto standard in the field of business process modeling (Chinosi and Trombetta, 2012; OMG, 2011a). In contrast to many other languages, BPMN provides an "extension by addition" mechanism containing four class: An *Extension Definition* is a named group of new attributes which can be referenced to any BPMN element. An *Extension Definition* consists of several *Extension Attribute Definitions* that define the particular attributes. Values of these *Extension Attribute Definitions* can be defined by the *Extension Attribute Value*. The class *Extension* binds the entire extension definition and its attributes to a BPMN model definition and makes it available and accessible.

Despite the well-defined syntax, there are some shortcomings of the mechanism: For instance, the XML definition of BPMN does not support extensions completely. Also, each extension is related to the generic *Base Element* of BPMN what hinders separation of concerns. Besides, BPMN lacks in providing methodical guidance for its application. A few authors address this issue: Stroppi et al. (2011) proclaim a MDA based approach for the transformation of conceptual domain models into valid BPMN extension methods (Stroppi et al., 2011). Braun and Schlieter (2014) introduce a structured approach for the analysis of domain concepts that should be added to the BPMN (Braun and Schlieter, 2014).

4.4 ArchiMate

ArchiMate is an EML for enterprise architectures that divides an enterprise into layers, aspects, internal views and external views (van Haren, 2012). In contrast to other EMLs from the architecture domain, ArchiMate provides two extension approaches. First, the "profile specialization mechanism" enables the annotation of attributes to concepts and relationships. Thereby, a Profile is understood as a data structure that is defined independently and connected to concepts and relationships of ArchiMate (similar to model weaving (Del Fabro and Valduriez, 2009)). Thus, ArchiMate permits the integration of external model elements. Moreover, ArchiMate explicates the opportunity to specialize concepts and relationships through inheritances between a general standard element and a domain-specific element (similar to UML Stereotypes). However, ArchiMate does neither provide a well-defined meta model of all extension-relevant concepts nor a procedure model for its application.

4.5 ARIS

ARIS (Architecture of Integrated Information Systems) is multi-perspective enterprise modeling framework that is mainly used in the field of business process management and ERP management (Scheer and Nüttgens, 2000). However, there is no commonly accepted meta model of ARIS, as it is not standardized by a larger institution. Consequently, ARIS does neither provide nor consider extensibility. Existing ARIS extension first re-engineer the ARIS meta model and then customize it ad hoc according to their requirements (Stein et al., 2008).

4.6 Conclusion

It can be summarized, that the minority of the addressed modeling languages provide an extension mechanism. UML, BPMN, Essence, KDM and ArchiMate provide concepts for the definition of new language objects. However, only BPMN and KDM allow heavyweight adaptations of the meta model in a structured manner. The other approaches rather support the lightweight definition of domain-specific elements on model level, without changing the meta model. In contrast to abstract syntax, adaptations of the concrete syntax is barely considered. Also precise methodical guidance for the extension design process is mostly missing.

5 CLASSIFICATION

5.1 Extension Purposes

The aspect "extension purpose" covers the objective that is related to the aimed extension and reflects the purpose that needs to be fulfilled. So far, only Atkinson et al. (2013) addresses this issue and proclaim "enhancement" and "augmentation". Based on a literature review and the consideration of extension reviews (Braun and Esswein, 2014a; Pardillo, 2010), we have extended their purpose setting as depicted in Figure 3.

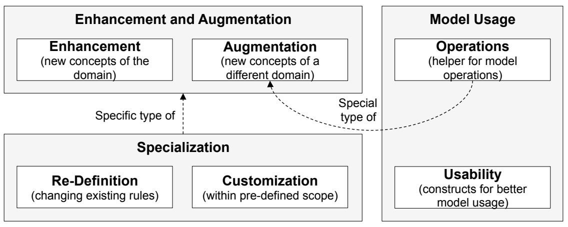
Enhancement and Augmentation
Enhancement
(new concepts of the domain)
Augmentation
(new concepts of a different domain)
Model Usage
Operations
(helper for model operations)
Specific type of
Special type of
Specialization
Re-Definition
(changing existing rules)
Customization
(within pre-defined scope)
Usability
(constructs for better model usage)

Figure 3: Consolidated extension purposes.

Enhancement stands for the extension of a host language with elements from the same domain (e.g., new gateway types within BPMN). Thus, it can be seen as a horizontal EML extension (Atkinson et al., 2013). On the other hand, **augmentation** stands for the vertical extension of a host language with concepts, attributes or rules from an other domain (Atkinson et al., 2013). Both purposes can be integrated (**Enhancement and Augmentation**).

Specialization covers all purposes that aim to characterize an EML for a specific domain, with a very limited number of *new* concepts. Moreover, the abstract syntax remains more or less the same, but the semantical aspects change. Thus, **Re-Definition** stands for the alteration of semantics of the host EML. For instance, by re-defining the interpretation of some EML concepts in the context of a specific domain. **Customization** stands for specifying the concrete syntax or the semantics of EML concepts in predefined boundaries (e.g., by integrating specific terminology of a domain or platform). Accordingly, the elements are only specified and not added. Thus, elements of the host EML remain under-specified and are intended to be adapted to a specific need.

Moreover, **Model Usage** encompasses all extensions that support some kind of working on EML models. We divide this purpose type into operations and usability. **Operations** covers all helper elements that are needed for model analysis or model transformation. Hence, it is a special type of augmentation (to the effect, that the considered domain is understood as an analytical area). Besides, **Usability** reflects all elements that are needed for the support of the model user (e.g., additional annotation elements).

5.2 Extension Mechanisms

An extension mechanism is understood as either an explicit mechanism of an EML for the extension of this language or a more general approach for extension. We have evolved the following mechanisms based on the review of both the literature (Atkinson et al., 2013; Braun and Esswein, 2014a) and existing EML specifications. Figure 4 presents all consolidated mechanisms.

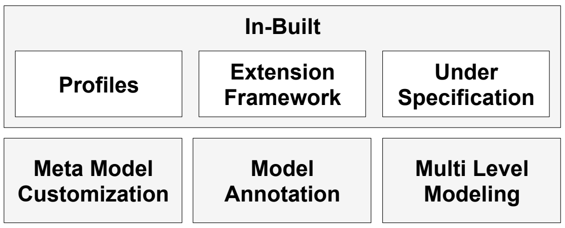

Figure 4: Consolidated extension mechanisms.

The **In-Built** group covers all approaches where the EML meta model contains an inherent mechanism for language extension on the meta level M2. The group can be divided into three sub mechanisms: *Profiles*, *Meta Model Customization* and *Under Specification*. A **Profile** constitutes well-defined language elements defined on level M2 that can be applied for the specification of new elements on level M1 (e.g., UML profiles). In this case, the meta model is not altered at all. Rather, some kind of an intermediate model level is built that allows the instance-specific definition of new elements in level M1. The **Extension Framework** type stands for a similar type, but in this case the proposed extension elements are less strict defined. It provides more a framework for custom extension, but its implementation remains open. For instance, within BPMN it is not clearly specified whether an extension should be implemented in the style of a profile or as a meta model alteration. Finally, **Under Specification** stands for specifications, whose elements are intentionally more generic and provide space for custom specification (e.g., ArchiMate or KDM). This could also affect only single elements (e.g., Lanes in BPMN).

Meta Model Customization can either be explicitly allowed by an EML or applied in an ad hoc manner individually. By reviewing existing meta model specifications, we found that actually no language supports a structured meta model customization. Rather, it remains open to the language engineer. Indeed, some languages indicate what elements can be extended or not, but there is no appropriate procedure for meta model customizations. Of course, altering a meta model always depends on the applied meta meta modeling language (see Section 3.1).

Model Annotation references the connection and integration of external models with the host language meta model. Such an integration can be realized by techniques like model weaving (Del Fabro and Valduriez, 2009). So far, none of the examined languages supports this concept explicitly. Only BPMN provides similar concepts by its *External Relationship* elements, which supports the integration of external domain models (however, on level M1 (OMG, 2011a, p. 62)).

Last but not least, Atkinson et al. (2013) introduced the novel approach of **Multi Level Modeling** in the context of EML extensions. In contrast to the rigid four layered OMG architecture, the authors propose a flexible multi level architecture based on the orthogonal classification architecture (Atkinson et al., 2009). Although the approach is not very diffused yet, multi level modeling is promising regarding to extensibility since it could solve abstraction difficulties.

6 CONCLUSION AND FURTHER RESEARCH

Within this article, the topic of language extensibility in the field of enterprise modeling languages was introduced by a review of existing extension mechanisms in both meta modeling languages and some EMLs. Based on the examination of these languages and a review of the (scarce) literature on that topic, a classification of extension purposes and extension mechanisms was proposed. Since this article represent research-in-process work, there are several aspects for further investigation:

Syntax. Currently, there is a lack of clear and precise definitions of extension mechanisms in existing EML specifications. Although several approaches exist, actually each of them has some inaccuracies or inconsistencies. Perhaps, this issue can be improved by the definition of a common extension reference model that could both support the revision of meta models concerning extensibility and also facilitate the exchange of model data. Such a reference model might also help to establish a common accepted understanding of EML extensibility and its elements. Therefore, the integrated E3 meta model of Greiffenberg (2004) seems to be a feasible starting point. Further, it is also important to examine the consequences of EML extension to modeling languages within software engineering in order to enable a tight business IT alignment.

Tool Support. The design of EML extension patterns is promising in order to integrate extension purposes and mechanisms and provide a productive base for reuse in extension design. Besides, syntactical issues are always related to a range of technical aspects such as flexible tool implementation, interoperability or possible run-time model alterations (Atkinson et al., 2013). Working on meta models also arises the necessity of suitable revision management (Esswein and Weller, 2007). Especially, when meta model elements are removed (meta model reducing) or overwritten - both aspects are barely examined so far.

Method. Although procedure models are important components of modeling methods (see Figure 1), there is a severe lack of methodological support for the design of valid and sense making extensions. For instance, such procedure models should support the domain analysis phase in order to identify an extension need. Also, they could recommend suitable extension patterns. Perhaps, preliminary studies in the field of situational method engineering can support the design of appropriate extension methods (Brinkkemper et al., 1999).

Acknowledgement. This research was funded by the German Research Foundation (DFG) within the research project “SFB Transregio 96”. The author is grateful for the funding and the research opportunities.

REFERENCES

Atkinson, C., Gerbig, R., and Fritzsche, M. (2013). Modeling language extension in the enterprise systems domain. In *17th IEEE International Enterprise Distributed Object Computing Conference*, pages 49–58.

Atkinson, C., Gutheil, M., and Kennel, B. (2009). A flexible infrastructure for multilevel language engineering. *IEEE Transactions on Software Engineering*, 35(6):742–755.

Atkinson, C. and Kuhne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE Software*, 20(5):36–41.

Becker, J. (2014). Interview with reinhard schütte on “managing large-scale bpm projects”. *Business & Information Systems Engineering*, pages 1–3.

Bettin, J. and Clark, T. (2010). Advanced modelling made simple with the gmodel metalanguage. In *Proceedings of the First International Workshop on Model-Driven Interoperability*, pages 79–88. ACM.

Botta-Genoulaz, V., Millet, P.-A., and Grabot, B. (2005). A survey on the recent research literature on erp systems. *Computers in Industry*, 56(6):510–522.

Braun, C. and Winter, R. (2005). A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform. *Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures*, pages 24–25.

Braun, R. and Esswein, W. (2014a). Classification of domain-specific bpmn extensions. *Lecture Notes of Business Information Processing*, 147:42–57.

Braun, R. and Esswein, W. (2014b). Extending bpmn for modeling resource aspects in the domain of machine tools. *Advanced Materials and Information Technology Processing*, 87:449.

Braun, R. and Schlieter, H. (2014). Requirements-based development of bpmn extensions: The case of clinical pathways. In *IEEE 1st International Workshop on the Interrelations between Requirements Engineering and Business Process Management*, pages 39–44.

Brinkkemper, S., Saeki, M., and Harmsen, F. (1999). Metamodelling based assembly techniques for situational method engineering. *Information Systems*, 24(3):209–228.

Chinosi, M. and Trombetta, A. (2012). Bpmn: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134.

Decker, G., Kopp, O., Leymann, F., and Weske, M. (2007). Bpel4chor: Extending bpel for modeling choreographies. In *IEEE International Conference on Web Services*, pages 296–303.

Del Fabro, M. D. and Valduriez, P. (2009). Towards the efficient development of model transformations using model weaving and matching transformations. *Software & Systems Modeling*, 8(3):305–324.

Esswein, W. and Weller, J. (2007). Method modifications in a configuration management environment. *Proceedings of the Fifteenth European Conference on Information Systems*, pages 2002–2013.

Frank, U. (1999). Conceptual modelling as the core of the information systems discipline-perspectives and epistemological challenges. *AMCIS 1999 Proceedings*, page 240.

Frank, U. (2002). Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 1258–1267.

Frank, U. (2008). The memo meta modelling language (mml) and language architecture. ICB Research Report 24, Universität Duisburg-Essen.

Frank, U. (2013). Domain-specific modeling languages: requirements analysis and design guidelines. In *Domain Engineering*, pages 133–157. Springer.

Greiffenberg, S. (2004). *Methodenentwicklung in Wirtschaft und Verwaltung*. Kovac.

Kopp, O., Görlach, K., Karastoyanova, D., Leymann, F., Reiter, M., Schumm, D., Sonntag, M., Strauch, S., Unger, T., Wieland, M., et al. (2011). A classification of bpel extensions. *Journal of Systems Integration*, 2(4):3–28.

Kühne, T. (2006). Matters of (meta-) modeling. *Software and Systems Modeling*, 5(4):369–385.

Lagarde, F., Espinoza, H., Terrier, F., André, C., and Gérard, S. (2008). Leveraging patterns on domain models to improve uml profile definition. In *Fundamental Approaches to Software Engineering*, pages 116–130. Springer.

Lankhorst, M. M., Proper, H. A., and Jonkers, H. (2009). The architecture of the archimate language. In *Enterprise, Business-Process and Information Systems Modeling*, pages 367–380. Springer.

Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., and Volgyesi, P. (2001). The generic modeling environment. In *Workshop on Intelligent Signal Processing, Budapest, Hungary*, volume 17.

Loos, P., Mettler, T., Winter, R., Goeken, M., Frank, U., and Winter, A. (2013). Methodological pluralism in business and information systems engineering? *Business & Information Systems Engineering*, 5(6):453–460.

Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., and Tang, A. (2013). What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering*, 39(6):869–891.

OMG (2011a). *Business Process Model and Notation, Version 2.0*.

OMG (2011b). *Knowledge Discovery Meta-Model, Version 1.3*.

OMG (2011c). *Unified Modeling Language, Infrastructure, Version 2.4.1*. OMG.

OMG (2012). *Structured Metrics Metamodel, Version 1.0*.

OMG (2014a). *Essence - Kernel and Language for Software Engineering Methods, Beta 2*.

OMG (2014b). *Meta Object Facility (MOF) Core Specification, Version 2.4.2*.

Pardillo, J. (2010). A systematic review on the definition of uml profiles. In *Model Driven Engineering Languages and Systems*, pages 407–422. Springer.

Scheer, A.-W. and Nüttgens, M. (2000). *ARIS architecture and reference models for business process management*. Springer.

Selic, B. (2007). A systematic approach to domain-specific language design using uml. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 2–9.

Stein, D.-W. F. S., Lauer, D.-I. F. J., and Ivanov, K. (2008). Aris method extension for business-driven soa. *Wirtschaftsinformatik*, 50(6):436–444.

Strahringer, S. (1998). Ein sprachbasierter metamodellbegriff und seine verallgemeinerung durch das konzept des metaisierungsprinzips. In *CEUR Workshop Proceedings Modellierung*.

Stroppi, L. J. R., Chiotti, O., and Villarreal, P. D. (2011). Extending bpmn 2.0: Method and tool support. In *Business Process Model and Notation*, pages 59–73. Springer.

van Haren (2012). Archimate 2.0 specification.

Weisemöller, I. and Schürr, A. (2008). A comparison of standard compliant ways to define domain specific languages. In *Models in Software Engineering*, pages 47–58. Springer.

zur Muehlen, M. and Recker, J. (2008). How much language is enough? theoretical and practical use of the business process modeling notation. In *Advanced information systems engineering*, pages 465–479.