Behind the Scenes of the BPMN Extension Mechanism
Principles, Problems and Options for Improvement

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

Keywords: Business Process Modeling, Language Extension, Enterprise Modeling, Meta Modeling, Meta Model Extension, MOF, BPMN.

Abstract: The Business Process Model and Notation (BPMN) is a standard for modeling business processes that is widely used and accepted both in academia and industry due to its well-defined meta model, its large set of concepts and its extensibility. BPMN is one of very few modeling languages that provides an integrated extension mechanism. However, the mechanism is not often implemented in research articles or in professional practice. We suppose, that both syntactical and methodical aspects within the BPMN extension mechanism may cause misunderstandings and uncertainty regarding its implementation. Therefore, we conducted an in-depth analysis of the extension mechanism in order to rationally figure out problematic aspects. These aspects are consolidated and compared to two existing BPMN extension methods. Based on that, a range of further research topics is finally derived.

1 INTRODUCTION AND MOTIVATION

The Business Process Model and Notation (BPMN) is a very common modeling language for business processes and widely used in professional practice (Chinosi and Trombetta, 2012). BPMN is defined by the Object Management Group (OMG) and additionally specified as ISO standard (OMG, 2011b). The modeling language provides a set of generic business process elements, independent from a specific domain. However, both in academia and industry, it is often necessary to extend BPMN with custom concepts in order to represent characteristics of a particular vertical domain (e.g., health care or quality management). Thus, it remains possible to both use the benefits of BPMN as a general purpose process modeling language (e.g., standardization, tool support and awareness) and adapt the language to specific requirements of a problem or a domain. BPMN is one of very few modeling languages that provides a set of generic extension elements within its meta model (see Figure 1). That extension mechanism is intended to enable the definition of extensions and ensure validity of the BPMN core (OMG, 2011b).

However, only very few BPMN extensions base on the mechanism, what hampers model interoperability, comparability and replicability (Braun and Esswein, 2014). We assume, that both missing methodical support and some syntactical obscurities lead to these deficiencies. We therefore conducted an in-depth analysis of the BPMN specification in order to carve out problematic issues. Thus, we aim to foster a discussion within academia regarding to a potential improvement of the BPMN extension mechanism that could finally lead to improvements in practice (e.g., better BPMN tool support for extensibility). This research paper is understood as a research-in-progress article, since we focus the current state of the art and indicate some first points for improvement.

The structure of the article is as follows. Section 2 provides an investigation of implicit extension capabilities with BPMN. In detail, we looked for elements that can be exploited for customization. Section 3 presents the BPMN extension elements in detail. While considering its elements, problematic aspects regarding to syntax and semantics are explicated. The same procedure is applied to the extension mechanism at all in Section 4. Afterwards, a table of "problem notices" is presented and compared to approaches from literature. Thus, a range of further research topics is finally derived.

Braun R..
Behind the Scenes of the BPMN Extension Mechanism - Principles, Problems and Options for Improvement.
DOI: 10.5220/0005329904030410
In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2015), pages 403-410
ISBN: 978-989-758-083-3

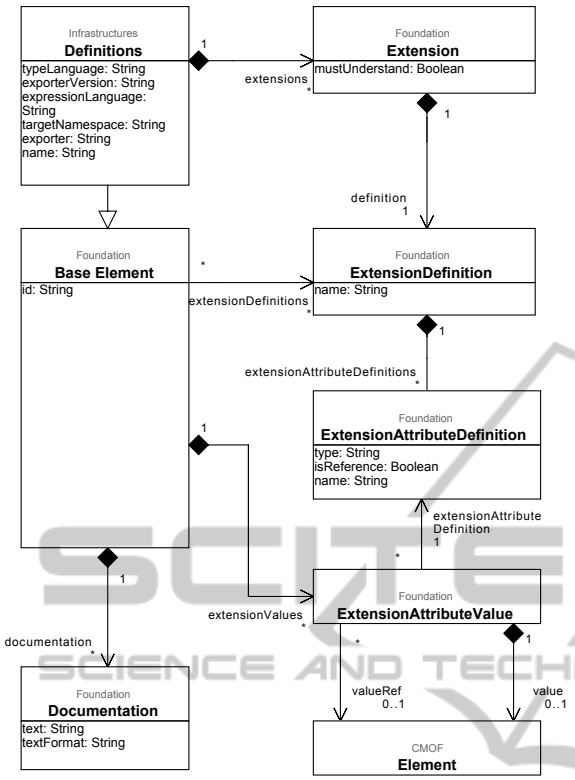

Figure 1: Excerpt of the BPMN meta model containing all relevant classes and relationships of the extension mechanism.

2 IMPLICIT EXTENSIBILITY

Leveraging a modeling language for specific purposes not always requires the definition of lightweight or heavyweight extensions, since modeling languages sometimes provide concepts that are intended for custom specification. Thus, we first consider such concepts within BPMN in order to pose existing capabilities beyond the extension mechanism. Sometimes, it might be more favourable to use the following options instead of taking a sledgehammer to crack a nut (extension mechanism).

2.1 Artifacts and External Relationships

Artifacts are used to provide additional information about a process. BPMN provides three *Artifacts* by default: *Groups*, *Text Annotations* and *Associations* (OMG, 2011a, p. 67). BPMN explicitly allows modelers and modeling tools to add additional *Artifacts* for specific purposes (OMG, 2011a, p. 28). This opportunity allows augmentation of the BPMN (Atkinson et al., 2013) in the sense of specific domain concepts or constructs for operations on BPMN models such as model transformations. As stated, new types can be added to a BPMN diagram, if sequence flow rules and message flow connection rules are respected (OMG, 2011a, p. 66). From a language point of view, rules and constraints of BPMN are not affected by this simple extension option. Moreover, the definition of a new information view of the languages is facilitated. Additionally, BPMN allows the integration of "a new shape representing a kind of Artifact" (OMG, 2011a, p. 8).

Problem Notice 1. Despite the flexibility of the *Artifact* approach, there are some unconsidered issues. First, it remains unclear what level of complexity a custom *Artifact* can have. This especially addresses the issue of attributed *Artifact* types and relations between them. Actually, *Artifacts* are supposed to be very simple objects getting their semantics primarily from their assigned name. Second, it also remains open, whether custom *Artifacts* should be integrated on meta level (M2) or model level (M1). While the first one increases model exchangeability, the latter is easier for integration within modeling tools.

BPMN also allows the integration of *Artifacts* and "elements expressed in any other addressable domain model" (OMG, 2011a, p. 62). This is enabled by the *External Relationship* element which represents associations between *Artifacts* an external model elements. This simple mapping mechanism is intended to facilitate traceability, model derivation and model integration (OMG, 2011a, p. 63).

In consequence, BPMN provides two basal procedures for domain-specific adaptions: First, new *Artifacts* can be designed individually. Strictly speaking, each custom *Artifact* also leads to a new variation of the BPMN meta model since new *Artifact* types need to be integrated. Thus, such an extension would lead to a minor, element-wise specialization of the BPMN meta model. However, this is not clearly explicated by BPMN as we stated above. Second, the *External Relationship* mechanisms provides the opportunity to integrate BPMN models and models of other modeling languages. Hence, the BPMN meta model remains unmodified, but techniques like model weaving need to be applied in order to realize model integration (Del Fabro and Valduriez, 2009).

2.2 Adapting the Concrete Syntax

Generally, BPMN is quite open for the customization of the concrete syntax: "An extension SHALL NOT change the specified shape of a defined graphical element or marker (e.g., changing a square into a tri-

angle, or changing rounded corners into squared corners, etc.)" (OMG, 2011a, p. 8). Further, adaptions of the concrete syntax can also be used for the explication of specific semantics: For instance, coloring elements can emphasize some categories. It is also allowed to add markers to elements in order to emphasize specific sub types. As stated above, *Artifacts* can be layouted individually.

3 EXPLICIT EXTENSIBILITY

BPMN provides an extension by addition mechanism that ensures the validity of the BPMN core elements while specifying additional constructs (OMG, 2011a, p. 44). Figure 1 depicts the relevant meta model excerpt. Due to the fact, that the majority of BPMN extensions is not designed in conformity to the BPMN meta model (Braun and Esswein, 2014), we provide an analysis of BPMN extension capabilities by analyzing both syntactical and semantical aspects of the four BPMN extension classes in detail. We also consider relevant consequences for XML based model interchange. Based on a rational discourse, some problematic (or even not clearly defined) aspects are stated within problem notices. Finally, these statements are summarized in a table and possible points for improvement will be suggested.

3.1 Extension

The element *Extension* binds and imports the entire extension definition and its attributes to a BPMN model definition. By doing so, all extension elements become accessible for BPMN elements (OMG, 2011b, p. 58). If the semantics of an *Extension* need to be understood by a BPMN adopter in order to process a BPMN model correctly, the *mustUnderstood* attribute is set to true. Otherwise, the *Extension* remains optional to BPMN definitions. Any BPMN definition can be associated to multiple *Extensions*. Further, an *Extension* consists of exactly one *Extension Definition*, that defines its actual content.

Semantics. The *Extension* class is only the outermost definition layer, without any concrete domain-specific information.

XML. On XML level, the reference to the *Extension Definition* is simply represented by the *QName* type. At this point, the XML specification disappears from the scope of BPMN and refers to an external XML specification.

3.2 Extension Definition

An *Extension Definition* is a named group of new attributes that can be used by BPMN elements. An *Extension Definition* is not inevitably a new element, since it can also be intended as a single additional attribute of a BPMN element. The only attribute of the class is the *name* attribute that denotes the group. The particular characteristics of the new element or attribute are defined by *Extension Attribute Definitions*. A *Base Element* can reference an *Extension Definition* multiple times. On the contrary, it is not intended to specify, *what* specific BPMN element can access the *Extension Definition*. Thus, actually all extensions are accessible for *all* BPMN elements since each element is a sub class of *Base Element*. Also, a navigation from the *Extension Definition* to the *Base Element* is not possible.

Semantics. Name and container of the designed extension. Due to the fact, that the *Extension* is associated with exactly one *Extension Definition*, an extension actually can be seen as exactly one new element in the sense of a concept having attributes or a concept that only stands for an additional attribute of some BPMN element (a differentiation is only possible in context of the application). This new element then can be referenced by several BPMN standard elements what emphasizes the additional character of the extension mechanism. However, this provokes some problems:

Problem Notice 2. Due to this constellation, it is quite inconvenient to add complex extensions, since it is necessary to first design all *Extension Definitions* singularly and than add the relationships between them subsequently. Therefore, it remains unclear, whether an *Extension Definition* can also have an *Extension Attribute Definition* that is typed by another new *Extension Definition*. Actually, that should be possible, since the type is specified as String. However, the specification should make this more clearer, because a domain extension with - for instance - a set of five elements requires a preloading of all five *Extension Definitions* in order to make the corresponding *Extension Attribute Definitions* available. Moreover, it would be better, to explicate the conceptual interrelations between the new elements in one model. Currently, each of the exemplarily five elements would cause a particular extension model and their conceptual interrelations would be barely obvious. Further, it is currently not possible to depict more complex relations between single extension elements such as aggregations or inheritances what hampers expressiveness and the representation of domain rules.

Problem Notice 3. In the current extension mechanism, each BPMN element can access all *Extension Definitions*, since all BPMN types are sub classes of *Base Element*. This leads both to missing separation of concern and to semantic irregularities, as also elements that are not intended to be related to an *Extension Definition*, can be related to them technically. In order to avoid such situations, it needs to be specified, to which element an *Extension Definition* can be associated. This should also be considered on XML interchange level. Otherwise, the stated under-specification could lead to semantically incorrect models.

Problem Notice 4. In the current version, BPMN does not make a conceptual difference between element-wise extensions (in the sense of definable classes with own attributes that can be referenced by other classes) and attribute-wise extensions (in the sense of adding some attributes to standard BPMN classes). While this should be syntactically correct, it might provoke some conceptual confusion whether the definition of new elements is allowed at all.

XML. BPMN specification states that this "type is not applicable when the XML schema interchange is used, since XSD Complex Types already satisfy this requirement" (OMG, 2011b, p. 58). Hence, BPMN excludes XML specification of *Extension Definitions* from its meta model and just refers to under-specified *XSD Complex Types*. There is actually no deeper control of the extension structure (syntax) on XML level.

3.3 Extension Attribute Definitions

As stated above, the *Extension Attribute Definition* class defines the actual characteristics of an extension. An *Extension Attribute Definition* is specified by the name and the type of the attribute. The corresponding type must be given as string reference to the identifier of the corresponding class. The *isReference* attribute indicates whether the attribute value is set directly or by reference to the referred element.

Semantics. A list of attributes of a new element or attribute.

XML. BPMN specification states that "this type is not applicable when the XML schema interchange is used; since the XSD mechanisms for supporting Any-Attribute and Any type already satisfy this requirement" (OMG, 2011b, p. 59). This statement also indicates that the XML based definition of an extension is out of the syntactical scope of BPMN. Moreover, implementing the specification depends totally on the language engineer. In this case, it might be better to keep it within some syntactical rules of BPMN in order to avoid semantically incorrect models (see above).

3.4 Extension Attribute Value

The *Extension Attribute Values* class can be used for the specification of concrete attribute values. If the corresponding *isReference* attribute of the *Extension Attribute Definition* is set of false, the attribute value is given directly. Otherwise, a reference to the targeted class needs to be given (OMG, 2011b, p. 59). A specific meta model *Element* is addressed in both cases. The stated *Element* class is one of the most generic classes within the Complete Meta Object Facility (CMOF) on level M3. On that meta meta model level, *Packages*, *Features* and *Classifiers* (OMG, 2014) inherit from the *Element* class. Classifiers are specified into *Classes* and *Data Types*. Thus, both complex types (e.g., BPMN elements) and primitive data types (e.g., Strings) can be referenced by *Extension Attribute Values*.

Semantics. A list of typed attributes of a new element.

Problem Statement 5. While understanding the semantics of the *Element* reference, the syntax is not accurate at this point, since the BPMN specification integrates an element that is specified on level M3 (within CMOF) and not on level M2 as BPMN is (see Figure 2). In the current version, the extension meta model violates the separation of abstraction layers (type instance relation). Also, BPMN does not provide clear evidences on the relation between basic concepts and the corresponding meta meta level (M3). Actually, each abstraction level is realized by type instance relations. Nonetheless, this is quite clear for the most classes (e.g., namespaces or associations), it is confusing regarding to the extension classes, since MOF already provides a basal extension mechanism and we cannot detect any correspondence to that in BPMN (OMG, 2014, p. 23).

3.5 Mechanism in General

As stated above, BPMN defines elements that should facilitate the design of vertical or domain-specific extensions. By providing elements on meta level M2 the instantiated extensions are actually located on model level M1 due to the "instance of" relation. However, an application of the extension requires another instantiation step regarding to the creation of the actual extension instances on level M1. This instantiation sequence collides with the strict four layer architecture of the OMG. Respectively, a new "intermediate layer" is created, which contains the standard meta

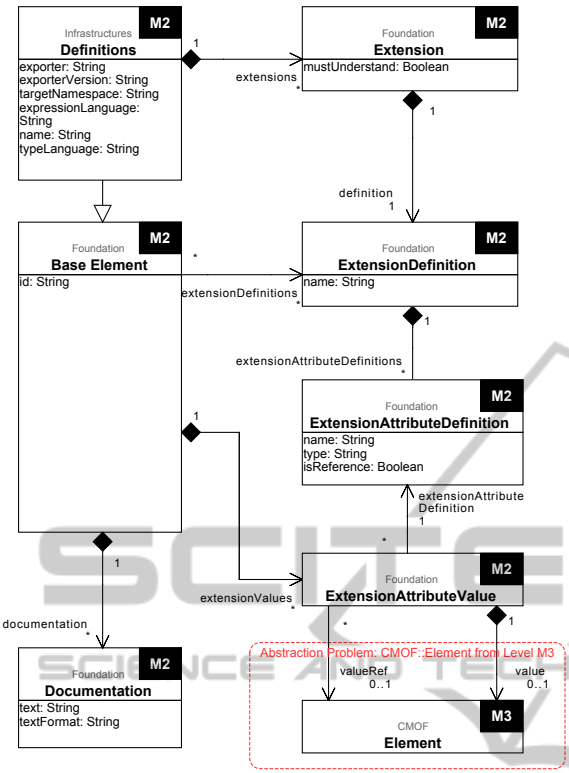

Figure 2: Abstraction problem: The class *CMOF::Element* originally comes from level M3, while all other classes are from the BPMN definition on level M2.

model and its new extension meta model elements (could be referred as "M1.5', see Figure 3). Nevertheless, the BPMN mechanism is principally feasible, its application should be clearly explained, since the *CMOF::Element* relation speaks against this assumption (see above).

If BPMN aims to provide an extension on level M2, the problems are getting more severe, since BPMN is defined on level M2 and cannot provide any mechanism of the level it is defined on. Rather, there either needs to be an application of MOF extension capabilities in the form of clear instances or any kind of overwriting of the M3 definitions regarding extensibility.

Problem Notice 6. Currently, it is not clear whether BPMN aims to provide an extension mechanism similar to the lightweight UML profile mechanism (M1) or a heavyweight meta model extension (M2). Evidences for both approaches can be found.

3.6 XML Serialization

As mentioned above, BPMN specification reveals deviations in regard of the XML interchange format. Generally, MOF specifies the XMI metadata interchange format in order to ensure serialized interoperability (OMG, 2014). Also the instances of MOF (meta models on level M2) should make use of XMI. However, BPMN propagates a dedicated XML schema what leads to various conceptual problems (Stroppi et al., 2011, p. 2).

Problem Notice 7. The extension mechanism is not very well included into the BPMN XML schema since the definition of the extension is outsourced to a separate XML schema without any further specification of its structure. Consequently, BPMN also states that if XML schema interchange is used, both *Extension Attribute Definition* and *Extension Attribute Definition Value* cannot be applied (see above). Rather, it is intended to extend the concerning BPMN classes with the XSD *AnyAttribute* and *Any* type. Thus, we observe also a break between the MOF based BPMN meta model and its XSD based serialization! These aspects could lead to uncertainty regarding to the exchange of extension data between two modeling tools, for example.

4 METHODOLOGICAL ASPECTS AND RELATED WORK

4.1 Methodological Aspects

Despite the fact that BPMN provides a well-defined extension interface, a procedure model for the straightforward development of extensions is missing.

Problem Notice 8. BPMN does not provide any methodological guidance. Especially, there is no support for the issue of comparing new domain-specific elements with standard BPMN elements in order to identify reasonable need for extension and ensure the required check whether the elements do not contradict the semantics of other BPMN elements.

Problem Notice 9. Besides to the customization of the concrete syntax of single items, it remains open, how the concrete syntax of extension elements should be exchanged through BPMN Diagram Interchange (OMG, 2014, p. 367). Also, there is no consideration of new diagrams in the sense of views or presentations based on extended elements.

4.2 Related Work

Only very few research articles address extensibility of BPMN. In the following, the approaches of Stroppi et al. (2011) as well as Braun and Schlieter (2014) are discussed against the background of the stated issues.

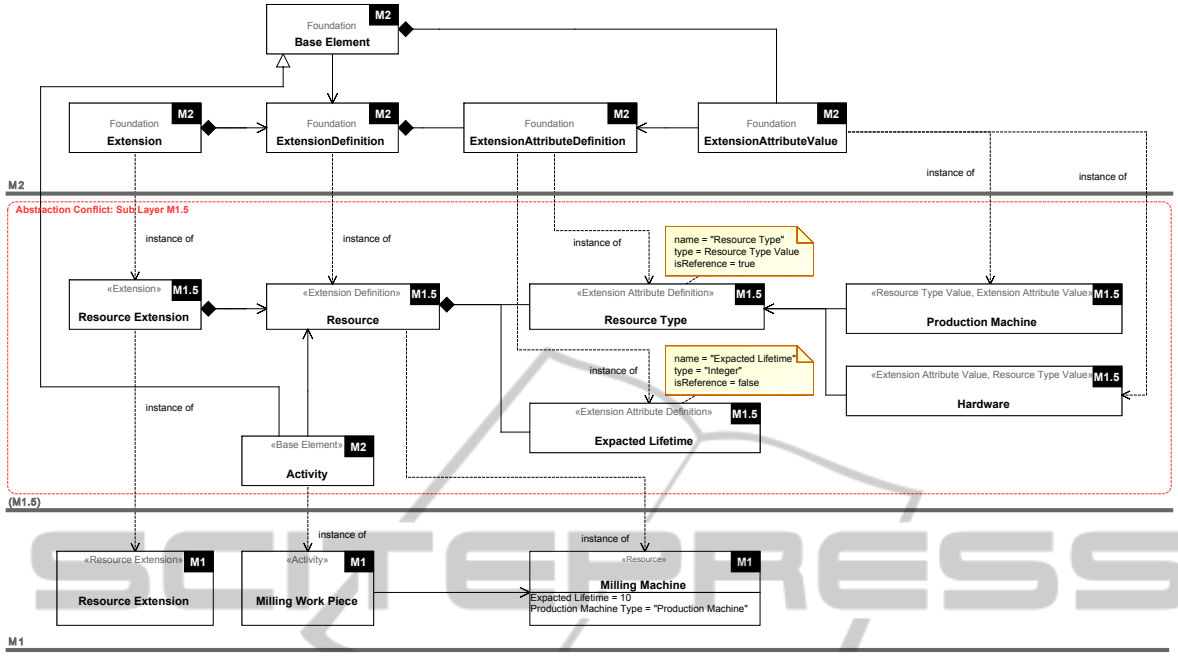

Figure 3: A simple extension example yields that the instantiation of BPMN extension meta model elements leads to a de facto sub layer (denoted "M1.5") that causes irregularities from the perspective of meta model layers. The example depicts the creation of a simple resource extension containing an element with two attributes.

4.2.1 Stroppi et al. (2011)

Stroppi et al. (2011) define a model-transformation based procedure model for the methodical development of valid BPMN extensions models. Their procedure model consists of mainly three steps: First, the domain is conceptualized by defining a Conceptual Domain Model of the Extension (CDME) as UML class diagram. Second, the CDME is transformed into a valid BPMN extension model (BPMN+X) by using UML stereotypes and a set of transformation rules for several model element constellations. Third, the BPMN+X model its transformed into a XML Schema Extension Definition Model, whereby the following matchings are applied: *Extension* to *xsd:schema*, *Extension Definition* to *xsd:group*, *Extension Attribute Definition* to *xsd:complexTypeDefinition* and *Extension Attribute Value* to *xsd:simpleTypeDefinition*. Obviously, the approach of Stroppi et al. (2011) provides a well-defined and consistent approach that tackles the problem of missing methodical support and XSD generation. Nevertheless, we see some minor shortcomings (or better points for improvement) regarding to this approach:

First, the authors state that a BPMN extension is a direct instance of the BPMN meta model. As we have already argued in Section 3.5, this is not the case since the extension model is actually an instance of the new meta model variant on level "M1.5". Thus,

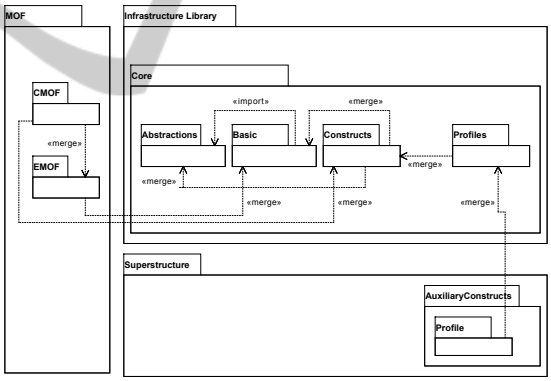

Figure 4: Excerpt of the MOF that indicates that MOF does not merge the *Profile* package. Strictly speaking, it is consequently not possible to apply the profile mechanism.

an extension model is only an indirect instance of the BPMN meta model. Stroppi et al. (2011) implicitly tackle this issue by adapting the profile mechanism from UML to the BPMN extension meta model, what leads to the creation of a $M2_{profile}$ model (in the sense of the their BPMN+X model). Although, this is a suitable technique, the application of the profile mechanism to either MOF or BPMN is actually not permitted by the MOF specification. The (not really obvious) reason is, that MOF does not import or merge the *Profile* package from *Infrastructure Library* (see Figure 4). Only the *UML Superstructure* imports the *Profile* package and is able to apply the

Problem Notice		Stroppi et al. (2011)	Braun and Schlieter (2014)	Possible Improvements and Further Research
1	Artifacts on M1 or M2?	–	–	• Clear statement in specification needed
2	Complex extension models	⊙	(⊙)	• Extending the method of Stroppi et al. (2011) for specific relation types • Perhaps, integration of OCL is helpful for specific dependencies
3	No specification of the extended Base Element sub class	●	(●)	• Update of the extension meta model or explication of conditional statements (OCL) for element specification
4	Clear distinction between element extension and attribute extension	●	(●)	• Update of the extension meta model
5	Conflict of level abstraction *CMOF::Element*	⊙	(⊙)	• Update of the extension meta model
6	Extension instantiation enforces intermediate level (M1.5)	⊙	(⊙)	• Clear statement in specification needed • Option 1: Profile mechanism according to Stroppi et al. (2011) with MOF update • Option 2: Changing level M3 / meta meta model extension capabilities
7	Missing specification of extension XML interchange	⊙	(⊙)	• Generic as well as sufficient specification of a serialization format
8	Lack of methodological guidance	⊙	●	• Consideration of domain analysis (preparation of extension design) • Integrated meta model, adaption of techniques from model comparison and model transformation • Formulation of extension guidelines/ patterns
9	Unclear interchange of concrete extension syntax	–	–	• Generic techniques for concrete extension syntax specification and exchange (re-use of BPMN DI basics!)

● (solution proposed) ⊙ (solution partially proposed) – (neither considered or solved)

Figure 5: Summary of the noticed problems and possible topics fur further investigation.

mechanism. In contrary, MOF provides only a basal, element-wise "name value pair" extension opportunity (OMG, 2014, p. 23).

Second, the method lacks in terms of a detailed analysis and consideration of the domain since it is an engineering driven approach that emphasizes stereotype definitions and transformation rules (Stroppi et al., 2011). Besides, an extension of the CDME model in terms of using further relation types (e.g., navigable associations, aggregations or compensations) should be considered in order to design more complex domain models on a conceptual level. Stroppi et al. (2011) present rules for the transformation of inheritances to basic associations by adding inherited attributes to the sub class. Similar rules need to be defined for further relation types that are needed for expressive domain models.

4.2.2 Braun and Schlieter (2014)

Braun and Schlieter (2014) outline an integrated procedure model for the design of BPMN extensions and extend the approach of Stroppi et al. (2011) by the analysis of the domain and its conceptualization. Therefore, the authors proclaim an initial domain requirements analysis step in order to identify requirements of the general modeling approach. Based on these requirements, a decision on the suitability of BPMN is made. Afterwards a domain ontology should be designed in order to prepare the conceptual domain model and set a base for the so-called equivalence check. This check covers an examination of each required domain concept against standard BPMN concepts and their semantics in order to derive a rational decision whether a concept needs to be added or not (Braun and Schlieter, 2014). However, the proposed method is still under research and more detailed information on the semantical equivalence check is necessary.

5 PROBLEM CONSOLIDATION AND FURTHER RESEARCH

Figure 5 summarizes the analyzed problem state-

ments and compares each of them against the background stated in Section 4.2. Some assessments regarding the approach of Braun and Schlieter (2014) were greyed out since the authors mostly re-use the approach of Stroppi et al. (2011). Also, possible points for improvement or aspects for further research are given. These aspects should be understood as first inspirations for further investigation.

All in all, we see two main directions for improving the current struggle with BPMN extensions. The first one addresses revising and updating parts of the BPMN meta model in order to remedy the examined problems, obscurities and ambiguities. Consequently, this would also require an review of the capabilities of MOF regarding the profile mechanism. However, changing such standards is indeed very hard to implement. Thus, the second option constitutes further development as well as enhancement of the approach of Stroppi et al. (2011) regarding to the mentioned aspects. Beyond, also questions of meta model version management and its respective instances need to be considered in order to ensure reasonable BPMN model comparison. Also, it seems to be promising to examine possible consequences of BPMN extensions relating to executable BPEL extensions (Kopp et al., 2011).

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.

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

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.

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

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.

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.

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

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

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

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.