Extensibility of Enterprise Modelling Languages

Dissertation

zur Erlangung des akademischen Grades

Dr. rer. pol.

vorgelegt an der

Technischen Universität Dresden

Fakultät Wirtschaftswissenschaften

von

Dipl.-Wirt.-Inf. Richard Braun

Vorgelegt: 09.09.2016
Verteidigt: 09.11.2016

Gutachter:
Prof. Dr. Werner Esswein
Prof. Dr. Susanne Strahringer

Richard Braun

Extensibility of Enterprise Modelling Languages

Technische Universität Dresden
Fakultät Wirtschaftswissenschaften
Lehrstuhl für Wirtschaftsinformatik,
insb. Systementwicklung

Preface

Einführung

Während meiner Tätigkeit am Lehrstuhl für Wirtschaftsinformatik, insb. Systementwicklung der TU Dresden arbeitete ich innerhalb des Sonderforschungsbereichs *SFB Transregio 96* in einem Projekt zur Beschreibung und Bewertung innovativer Technologien im Werkzeugmaschinenbau. Während des Projektes ergab sich die Notwendigkeit, die den Technologien zugrunde liegenden Verfahren zu modellieren. Die Wahl fiel dabei auf die BPMN, den führenden Standard für die Geschäftsprozessmodellierung. Die Domänenspezifik, Komplexität und Vielschichtigkeit der zu modellierenden Verfahren erforderte jedoch die Anpassung der BPMN, um alle notwendigen Aspekte innerhalb der Sprache abzubilden und z. B. Ressourcen-Konzepte auf Metamodell-Ebene spezifizieren zu können.

Überraschenderweise stellte sich die folgende Implementierung als schwierig heraus, da so gut wie keine nutzbaren Vorarbeiten zur systematischen Erweiterung von Modellierungssprachen allgemein und der BPMN im Speziellen existieren. Auch bietet die BPMN selber nur einen sehr rudimentären, unscharf definierten und teils widersprüchlichen Erweiterungsmechanismus. Fast alle Sprachen zur Unternehmensmodellierung verzichten gänzlich auf die Betrachtung möglicher Erweiterungen. Dementsprechend werden die meisten Spracherweiterungen ad hoc definiert oder modifizieren die originale Grammatik gar. Auch sind die meisten Spracherweiterungen werkzeuggebunden und liegen nicht in einem austauschfähigen Format vor. Eine entsprechende Wiederverwendung oder Adaption wird daher ebenso erschwert wie die systematische und transparente Konstruktion von Erweiterungen ganz allgemein. Diese methodische Unreife steht im Gegensatz zum tatsächlichen Bedarf der situativen, problemspezifischen oder domänenspezifischen Ausgestaltung oder Verfeinerung von eher generisch definierten Unternehmensmodellierungssprachen wie BPMN, EPK oder ArchiMate.

Motiviert durch diese Notwendigkeit befasse ich mich seit 2014 mit der Erweiterbarkeit von Unternehmensmodellierungssprachen, insbesondere am Beispiel der BPMN. In der Forschungsthematik kumulieren mehrere relevante Aspekte. Zuvorderst bilden semi-formale Modellierungssprachen prinzipiell ausdrucksstarke Instrumente, um die heutige Komplexität vielschichtiger Informationssysteme überhaupt beschreibbar und zu einem gewissen Grad beherrschbar zu machen. Sie stehen dabei symbolisch für die immanent wichtige Schnittstelle zwischen maschinell-

technischen Anwendungssystemen und sozialen Akteuren innerhalb von Organisationen.

Um der Komplexität des Problembereichs nicht mit einem schwer handhabbaren multi-pluralistischen Satz von verschiedenen Modellierungsmethoden zu begegnen, proklamiere ich die Fokussierung auf einen Kern von hinreichend generischen Standardsprachen in der Unternehmensmodellierung und deren systematische und einheitlich definierte Erweiterung, insofern ein spezifischer Bedarf dies erfordert. Dieser "*Standard+X*"-Ansatz ist unter anderem durch den Erfolg von erweiterbarer Open-Source-Software inspiriert. Mit Hilfe wohl-definierter Erweiterungsmöglichkeiten erlaubt dieser Software-Typ spezifische Erweiterungen und somit nutzergruppengerechte Individualisierungen. Einmal entwickelte Erweiterungen können zusätzlich auf marktplatz-ähnlichen Portalen angeboten und somit von anderen Nutzern wiederverwendet werden.

Forschungsschwerpunkte

Die Forschungsarbeiten zu Spracherweiterungen innerhalb der letzten zweieinhalb Jahre können als dreigeteilt angesehen werden.

Der erste Schwerpunkt setzt sich mit den zu entwickelnden BPMN-Erweiterungen auseinander und stellt deren methodische Implikationen im Rahmen der bestehenden Sprachstandards dar. Dies umfasst zum einen ganz konkrete Spracherweiterungen wie z. B. BPMN4CP, eine BPMN-Erweiterung zur multi-perspektivischen Modellierung von klinischen Behandlungspfaden. Zum anderen betrifft dieser Teil auch modellierungsmethodische Konsequenzen, um parallel sowohl die zugrunde liegende Sprache (d. h. das BPMN-Metamodell) als auch die Methode zur Erweiterungsentwicklung zu verbessern und somit den festgestellten Unzulänglichkeiten zu begegnen. Dieser Teil intendiert daher primär eine rigorose, aber pragmatisch-orientierte Implementierung in derzeit gegebenen Modellierungsumgebungen, um dem aktuellen Stand der Technik gerecht zu werden.

Der zweite Schwerpunkt adressiert die Untersuchung von sprachunabhängigen Fragen der Erweiterbarkeit, welche sich entweder während der Bearbeitung des ersten Teils ergeben haben oder aus dessen Ergebnissen induktiv geschlossen wurden. Der Forschungsschwerpunkt fokussiert dabei insbesondere eine Konsolidierung bestehender Terminologien, die Beschreibung generisch anwendbarer Erweiterungsmechanismen sowie die nutzerorientierte Analyse eines potentiellen Erweiterungsbedarfs. Dieser Teil bereitet somit die Entwicklung einer generischen Erweiterungsmethode grundlegend vor.

Hierzu zählt auch die fundamentale Auseinandersetzung mit Unternehmensmodellierungssprachen generell, da nur eine ganzheitliche, widerspruchsfreie und integrierte Sprachdefinition Erweiterungen überhaupt erst ermöglichen und gelingen lassen kann. Dies betrifft beispielsweise die Spezifikation der intendierten Semantik einer Sprache. Die Thematisierung von Erweiterungen führte somit in letzter Konsequenz zu sehr elementaren Untersuchungen, die ob ihrer Komplexität und Vielschichtigkeit nicht vollständig bearbeitet und gelöst werden konnten. Die vorliegende Arbeit versteht sich somit insbesondere hinsichtlich dieses Schwerpunktes explizit auch als

die vorstrukturierte Grundlage für weitere Forschungsarbeiten, welche im Rahmen eines Habilitationsprojektes vertieft und ausgebaut werden sollen.

Danksagung

Herrn Prof. Dr. Werner Esswein danke ich für die eingeräumte Freiheit bei der wissenschaftlichen Schwerpunktsetzung, die vorbehaltlose Unterstützung aller Forschungsaktivitäten sowie die übertragene Verantwortung in der akademischen Lehre und in Forschungsprojekten. Frau Prof. Dr. Susanne Strahringer gebührt mein ausdrücklicher Dank für die unkomplizierte Übernahme des Zweitgutachtens sowie die kurzfristige und zügige Begutachtung der Arbeit.

Meinen Kollegen am Lehrstuhl für Systementwicklung danke ich für die jederzeit angenehme, kollegiale und positive Arbeitsatmosphäre. Herrn Dr. Hannes Schlieter, Herrn Martin Burwitz und Herrn Martin Benedict danke ich für die sehr ergiebige und fortwährende Zusammenarbeit in Bezug auf die Anwendung von Spracherweiterungen im Kontext von Informationssystemen im Gesundheitswesen. Aus unserer Zusammenarbeit entstanden mehrere Publikationen, welche sowohl innerhalb der Forschungsgemeinschaft als auch in Praxisprojekten Anklang fanden und thematisch weiter vertieft werden sollen.

Meiner Kollegin Frau Jeannette Stark danke ich für die vielfältigen Diskussionen in der Endphase unserer beiden Promotionen sowie die begonnene Zusammenarbeit in Bezug auf grundlegende Forschungsfragen der konkreten Syntax. Herrn Dr. Jens Weller danke ich für die Einführung in Grundzüge der Metamodellierung zu Beginn meiner Tätigkeit als Studentische Hilfskraft am Lehrstuhl. Frau Lisa Gerstenberger danke ich herzlich für die fortwährende Unterstützung bei der Abrechnung vieler Dienstreisen sowie ihre Hilfe beim Verständnis der universitären Administration.

Darüber hinaus möchte ich explizit den unzähligen anonymen Gutachtern meiner Forschungspapiere danken. Die in der Regel sehr hilfreiche Kritik sowie die Diskussion eigener Forschungsarbeiten auf Konferenzen förderten die kritische Reflexion der eigenen Arbeiten, lieferten wichtige Impulse und prägten somit den inhaltlichen Fortschritt der Arbeit sowie meinen wissenschaftlichen Werdegang.

Den Kunden meiner Firma *digiturax* danke ich für das teilweise seit mehr als zehn Jahren entgegengebrachte Vertrauen. Es war mir somit bereits im Studium als auch während der Promotion möglich, gewonnene theoretische Kenntnisse praktisch anzuwenden und zu reflektieren. Dieser Perspektivwechsel ist insbesondere in der Wirtschaftsinformatik-Forschung außerordentlich wichtig und kostbar.

Abschließend möchte ich mich herzlich bei meiner Familie bedanken. Meiner über alles geliebten Freundin Minja danke ich für ihr grenzenloses Verständnis in den letzten Monaten. *Minja, volim te – do neba i nazad!* Der größte Dank gilt meinen Eltern, die mich bei meinen wissenschaftlichen und beruflichen Vorhaben stets unterstützt haben und mich im letzten Jahr motiviert haben, das Promotionsvorhaben zu finalisieren und nicht dem Drang einer kontinuierlichen Verfeinerung und Weiterentwicklung zu unterliegen. Ich widme ihnen dieses Werk von ganzem Herzen.

Dresden, September 2016 Richard Braun

Contents

Part I Motivation and Introduction

1 Research Design . . . 3
1.1 Context and Motivation . . . 3
1.2 Research Problems . . . 4
1.2.1 Extensibility in the Context of BPMN and MOF . . . 4
1.2.2 Extensibility for EMLs in General . . . 6
1.2.3 EML Definitions in General . . . 7
1.3 Research Objectives . . . 10
1.4 Research Approach . . . 12
1.4.1 Research Papers and Consolidation Essay . . . 12
1.4.2 Application of Design Science . . . 13

2 Organisation of the Consolidation Essay . . . 17

Part II Fundamentals

3 Relevant Publications . . . 21
3.1 Publication POEM-2014 . . . 22
3.2 Publication MODELSWARD-2015-A . . . 24
3.3 Publication MODELSWARD-2015-B . . . 26
3.4 Publication ICEIS-2015 . . . 27
3.5 Publication MKWI-2016 . . . 29

4 Terminological and Conceptual Foundations . . . 31
4.1 Peculiarities of Enterprise Modelling Languages . . . 31
4.2 Discussion on Standard Enterprise Modelling Languages . . . 33
4.2.1 Pro Standardisation . . . 33
4.2.2 Contra Standardisation . . . 34
4.2.3 Towards Extensible Standard EMLs . . . 34
4.3 Terminology . . . 34
4.3.1 Extension . . . 35
4.3.2 Pseudo Reduction . . . 38
4.3.3 Hybrid . . . 38
4.3.4 Semantic Extensions and Modifications . . . 39

4.4 Summary and Conclusion 39

5 Extension Types of Enterprise Modelling Languages 41
5.1 Criteria for EML Classification 41
5.1.1 Formalisation 41
5.1.2 Focus 42
5.1.3 Domain-Specificity 42
5.2 Framework Architecture 43
5.3 Types of Enterprise Modelling Languages 43
5.4 Consequences for EML Extension Types 46
5.4.1 Formalisation 46
5.4.2 Views 47
5.4.3 Domain-Specificity 47
5.5 Language Extension Types 48
5.5.1 Accents – Semantic Extensions 48
5.5.2 Dialects – Syntactic and Semantic Extensions 49
5.6 Consolidation and Conclusion 49

Part III Extension Mechanisms

6 Relevant Publications 53
6.1 Publication WIT-2014 53
6.2 Publication MKWI-2014 55
6.3 Publication BIBM-2014 57
6.4 Publication WI-2015 59
6.5 Publication CCIS-2015 61
6.6 Publication MEDI-2015 63
6.7 Publication CBI-2015 65

7 Introduction 67
7.1 State of Affairs 67
7.2 Classification and Specification 68

8 Mechanisms 71
8.1 Annotations 72
8.1.1 Leveraging Principles from Software Engineering 72
8.1.2 Decorators 72
8.1.3 Plugins 73
8.1.4 Aspects 73
8.1.5 Add-Ons 74
8.2 Hooking (Under-Specification) 74
8.2.1 Motivation 74
8.2.2 Adaptation and Application 75
8.3 Profiling and Stereotypes 75
8.3.1 Motivation 75
8.3.2 Adaptation and Application 76
8.4 Multilevel Modelling 76

8.4.1 Motivation: Issues within Fixed Level Architectures 76
8.4.2 Existing Multilevel Modelling Approaches 77
8.4.3 Core Question: Specialisation or Instantiation 78
8.4.4 Principle of Adaptation for EML Extensions 80
8.4.5 Adaptation Procedure 81
8.4.6 Required Redesign 82
8.4.7 Demarcation from Other Approaches and Limitations 82
8.4.8 Pragmatics 83
8.5 Simple Generalisation/Specialisation 83
8.5.1 Motivation 83
8.5.2 Architecture and Application 84
8.5.3 Pragmatics 84
8.5.4 Restrictions and Limitations 84
8.6 Semantic Extension Techniques 84
8.6.1 Motivation 84
8.6.2 Architecture and Application 85
8.6.3 Pragmatics 85
8.6.4 Restrictions and Limitations 85

9 Repository 87
9.1 Overview 87
9.2 Comparison 89
9.3 Combination of Mechanisms 90

Part IV Semantics-Driven Justification of Extension Need

10 Relevant Publications 95
10.1 Publication DESRIST-2015 96
10.2 Publication REBPM-2014 98
10.3 Publication BIBM-2015 100
10.4 Publication EEWC-2015 102
10.5 Publication IDS-2015 104
10.6 Publication HICSS-2016 106
10.7 Publication ZEUS-2016 108
10.8 Publication AQEMO-2016 109
10.9 Publication MODELSWARD-2016 110

11 Relevant Unpublished Papers 113
11.1 Paper UNPUB-MOF4EM-2016 114

12 Motivation and Introduction 117
12.1 Related Work 117
12.2 Pragmatics and Semantics First 118
12.2.1 Methodical Consideration of Pragmatics and Semantics in EML Extensions 120
12.2.2 Consequences for Extension Design 121
12.2.3 Assumptions and Limitations 121

13 Structure for Extension Procedure 123

14 Use Case Analysis 127
14.1 Motivation and Fundamentals 127
14.2 Related Work 127
14.3 Input, Method, Output 128

15 Requirements Analysis 131
15.1 Motivation and Fundamentals 131
15.2 Related Work 132
15.3 Input, Method and Output 133
15.3.1 Requirements Classification 133
15.3.2 Derivation of Requirements from Use Cases 133
15.3.3 Output 135

16 Concept Analysis 137
16.1 Motivation and Fundamentals 137
16.1.1 Relevance of Semantics and Current Issues 137
16.1.2 Consequences for Extension Method 138
16.1.3 Ontologies for Semantics Representation 138
16.1.4 From Requirements to Semantics Specification 139
16.2 Ontological Constructs from Conceptual Requirements 139
16.3 UEML for the Representation of Material Semantics 139
16.3.1 Introduction 139
16.3.2 Architecture 140
16.3.3 Application and Adaptation 142
16.4 Semantic Comparison with UEML 144
16.4.1 Related Work 144
16.4.2 Correspondence Types in UEML 145
16.4.3 Proposal for Correspondence Typology 146
16.5 Justifying and Modelling Intended Semantic Constructs 151
16.5.1 Invariant Semantics 152
16.5.2 Variance of Type Semantics 153
16.5.3 Relevance 154
16.5.4 Instances as Types Intuitively 154
16.5.5 Output 154
16.6 Ontological Constructs from Capability-Related Requirements 154
16.6.1 DMM for the Representation of Formal Semantics 155
16.6.2 Outline and Discussion of a Possible Implementation 156
16.7 Perspectives and User-Related Requirements 157
16.7.1 Perspectives 157
16.7.2 User-Related Requirements 158
16.8 Output 158

17 Correspondence Analysis ... 161
17.1 Material Semantic Constructs ... 161
17.1.1 Equivalent Scenes ... 162
17.1.2 Similar Scenes ... 164
17.1.3 Different Scenes ... 166
17.1.4 Instances ... 168
17.1.5 Consolidation, Application, and Remarks ... 168
17.2 Formal Semantic Constructs and Perspectives ... 169
17.2.1 Formal Semantic Constructs ... 169
17.2.2 Perspectives ... 169
17.3 Output ... 170

18 Extension Preparation and Subsequent Stages ... 171
18.1 Pragmatics-Driven Pre-Selection ... 171
18.1.1 Formalisation Dimension ... 171
18.1.2 View Dimension ... 173
18.1.3 Domain-Specificity Dimension ... 173
18.2 Mechanism Selection ... 174
18.3 Mechanism Application and Subsequent Steps ... 175
18.3.1 Extension Definition in General ... 176
18.3.2 Extension Definition in BPMN ... 176
18.3.3 Extension Labelling ... 177

19 Conclusion ... 179

Part V Conclusion and Further Research

20 Contributions ... 185
20.1 Research Artefacts, Studies and Consolidations ... 185
20.2 Reflection of Research Objectives ... 186

21 Implications for Further Research ... 189
21.1 Pragmatics ... 189
21.2 Semantics ... 189
21.3 Syntax ... 190
21.4 Method ... 191

List Of Figures ... 193

List Of Tables ... 193

References ... 197

Appendix ... 209
1.1 Overview of Extension Mechanisms ... 209
1.1.1 Decorators ... 209
1.1.2 Plugins ... 210

1.1.3 Aspects 211
1.1.4 Add-Ons 212
1.1.5 Hooking 213
1.1.6 Profiling 214
1.1.7 Multilevel Modelling 215
1.1.8 Generalisation/Specialisation 216
1.1.9 Semantic Extension 217

Part I

Motivation and Introduction

1

Research Design

1.1 Context and Motivation

Enterprises are multifarious, heterogeneous socio-technical Information Systems (IS) whose components are interrelated within a complex nexus of interdependencies on different abstraction levels [1, 2, 3]. Enterprise Modelling (EM) aims to support the conceptualisation, abstraction, and final representation of relevant enterprise-related aspects by creating conceptual models [4, 5]. EM hence serves as an auspicious approach for managing present-day business complexity in the light of increasing interdependencies between and within IS [6]. In particular, enterprise models support inter-subjective communication, documentation, organisational engineering, systems engineering, as well as various operative tasks and automatisation purposes [7, 8, 9].

Enterprise models are instances of Enterprise Modelling Languages (EML). EMLs are conceptual modelling languages that provide concepts for modelling characteristic aspects for the analysis and design of IS and application systems [10, 11, 12, 6]. EMLs serve as semi-formal languages and have a precisely defined formal syntax and mostly informal semantics [13, 14]. Several official language standards and de facto language standards have evolved over the last decade, e.g. BPMN [15, 16], ArchiMate [17, 10] or EPC [18, 19].

The complexity of enterprise-related issues requires a permanent review of the communicative usefulness of existing EMLs [20]. Fixed and all-encompassing standard EMLs are hence rather illusory and also impractical in terms of a complete a priori design [21, 22]. Instead, it is more reasonable to consider flexible modelling approaches. Therefore, some authors proclaim the integration and composition of modular meta model components (according to [23, 24]), while other authors suggest the design, specification, and arrangement of dedicated and precisely fitting Domain-Specific Modelling Languages (DSMLs [25]). In contrast to these modular approaches, some research has been conducted around punctual extensions or customisations of standard languages like BPMN [21, 26, 27]. The stated approaches differ in terms of invasiveness, the forethought of particular mechanisms, as well as academic and industrial prevalence [28].

Within this thesis, the EML extension approach is followed. The approach stands for the situational extension of a commonly used language like BPMN for a particular industry, domain or business problem [27]. The approach is motivated by the intended exploitation of the benefits of standard languages, such as well-defined syntax, general prevalence, tool support, and model interoperability (cf. [22]). Investi-

gating the situational adaptation of prevalent EMLs further appears to be promising in terms of avoiding exuberant method pluralism in the EM domain (according to [29]) and may also contribute to the evolution of EMLs in general [30].

A review of the current state of the art reveals the emergence of extensions, adaptations, or variants of common EMLs [31, 32, 33, 28, 19]. In the light of the notable need for research on extending EMLs [34, 21, 22, 33, 35], this thesis tackles several issues of EML extensibility using the motivating example of the Business Process Management and Notation (BPMN).

1.2 Research Problems

Our research was initially driven by extending the BPMN within research projects in the field of Product Engineering [36, 37] and Clinical Process Management [27]. It became apparent that the design of standard-compliant BPMN extensions was not straightforward and error-prone, as the official BPMN specification reveals several methodical and architectural shortcomings which are closely related and partially caused by deficiencies of the underlying meta modelling language, the Meta Object Facility (MOF [38]).

Although MOF itself serves as a commonly applied and widely disseminated language for the specification of meta models, it suffers from limited capabilities for the definition of EMLs and lacks in the provision of appropriate extension mechanisms. Consequently, respective methodical guidance needed for conducting specific extensions in BPMN is also missing [39].

Both BPMN and MOF are officially defined standards of the Object Management Group (OMG), though the addressed issues are not limited to the field of OMG-related standards, but rather prominent for all EMLs [28]. Despite the generally perceived relevance of EML extensions and adaptations in academia and professional practice [7], there are numerous research challenges that have not yet been tackled [40, 28, 39, 19]. Some issue causes further research in regard to fundamental but indeed under-investigated topics such as language pragmatics and language semantics [40, 9].

Considerations on EML extensibility are hence triggered by the specific case of BPMN on the one side and further implications in the context of MOF as well as general extensibility of EMLs on the other side. An extensive review of the literature corpus as well as in-depth analyses of multiple EML specifications (e.g. [33, 28, 39, 41, 42]) reveal several research gaps, which are consolidated within three main parts below: Extensibility in the context of BPMN and MOF (Sect. 1.2.1), extensibility for EMLs in general (Sect. 1.2.2), and general aspects in regard to EML definitions within the MOF environment (Sect. 1.2.3).

1.2.1 Extensibility in the Context of BPMN and MOF

The stated issues consequently manifest in particular EMLs like BPMN, which serves as dominant process modelling language in academia and industry [15, 16]. BPMN provides a large vocabulary of generic and partly under-specified language constructs aiming to facilitate cross-domain and industry-independent process modelling.

Research Problem 1: Specific Extension Need in BPMN

The involvement in large-scale research projects creates a need for implementing different concepts within BPMN in order to fulfil project-specific tasks. Within the research project *SFB Transregio 96*, it is necessary to integrate different types of resources and machine-related attributes into engineering processes [37]. Several research projects in the field of Telemedicine and Clinical Process Management provoke the integration of additional perspectives in BPMN as well as different concepts for modelling Clinical Pathways (CPs [43]). The required concepts as well as the required perspectives are not part of the BPMN meta model.

Research Problem 2: Syntactical Specification of BPMN Extensions

BPMN is frequently adapted to different degrees. Numerous extensions, variants, or adaptations have evolved over the last years [33]. Although BPMN is one of the very few languages that provides an extension mechanism, it reveals several shortcomings [39, 28].

First, a lack of specifity has to be bemoaned. Four extension meta classes are defined: *Extension*, *ExtensionDefinition*, *ExtensionAttributeDefinition*, and *ExtensionAttributeValue*. These meta classes semantically indicate a particular annotation of meta types. However, it remains unclear whether the stated classes allow type extensions or only attribute extensions of original BPMN meta classes. Due to this missing clarification, it is also difficult to define complex extension models with multiple relations and different relation types (e.g. aggregations or compositions). This issue is closely related to the below discussed abstraction concerns (cf. [39]).

Second, neither BPMN nor the BPMN extension mechanism provides any opportunity to define perspectives or other instruments for user-specific views and complexity reduction (cf. [44]). BPMN explicitly states that it does not serve as a dataflow language, for instance [15, p. 22], but actual need for integrating non-process concepts and even non-process transformations has been observed (cf. [33]). Due to the prominent role of business process modelling within EM and its generally integrated character, a strict demarcation from other purposes is often difficult.

Third, it is currently not possible to specify the concrete syntax of BPMN and there are no capabilities for integrating the introduced concrete syntax with the abstract syntax. Consequently, there is a lack of exchange specification for extensions, although a re-usage of some MOF-based formats is promising (e.g. MOF2XMI transformations).

Research Problem 3: Abstraction Conflicts in BPMN Extension Mechanism

The current BPMN extension mechanism further reveals another syntactical issue that has to be addressed. Any extension application yields an actual intermediate level between the meta model level and the model level (referred as M1.5 level [39]), since the defined extension has to be instantiated at first in order to become part of the language. The same issue can be observed within Profiling in UML [45]. However, it is more feasible to position additional meta classes on the same classification level, causing a relocation of extension meta types to the meta meta model level in order

to define (and instantiate) extensions for the meta model level. This issue is tightly coupled with general abstraction issues in the context of MOF.

BPMN reveals further abstraction conflicts in regard to missing specification of extended *BaseElement* classes as well as the integration of the *CMOF::Element* class from the meta meta model level, which breaks the four level architecture of the OMG again [39].

Research Problem 4: Missing BPMN Extension Method

The consideration of extensibility in BPMN is limited to a few syntactical operations and omits semantic and pragmatic aspects completely. There is missing guidance in terms of both domain conceptualisation and semantic justification; this leads to a particular gap between domain analysis and artefact design, although procedure models and methodical guidance serve as immanently important elements for modelling methods [46, 47].

To the best of our knowledge, Stroppi et al. [48] exclusively address this topic by introducing a transformation-based BPMN extension method. Their method builds on the standard BPMN extension mechanism and provides a step-wise extension design procedure. However, the method does not consider the underlying conceptual shortcomings of BPMN and solely focuses on the abstract syntax, omitting any aspects from EM or material semantics [48].

1.2.2 Extensibility for EMLs in General

Research Problem 5: Inconsistent Terminology

Reflecting the literature on EML extensions further reveals an inconsistent terminology within the research community. Extensions, modifications, or specifications of EMLs are referred to differently. For instance, some authors refer to *extensions* [49, 34, 22, 33], while other authors give the impression of having a rather broad understanding of modifying an EML, which could lead to *adaptations* [9], *customisations* [50], *variants*, or *variations* [51, 21, 19]. Also the concept of *dialects* can be found in literature [21, 22, 19].

Furthermore, conceptual principles and specific technical implementations are often amalgamated, implicating insufficient separation (cf. the extension technique of Profiling [42]). However, definitional clarification of key terms within a field of research serves as an essential feature for research in general and for design-oriented research in particular [52, 53, 54]. It is necessary to establish a terminological and conceptual base for further analysis.

Research Problem 6: Missing Guidance for Extension Justification

There is also a lack of methodical support for the systematic analysis and design of EML extensions [28], which covers several related issues. Due to a certain overemphasis of syntactical tasks [40], the actually more interesting tasks of identifying and justifying extension need based on intended pragmatics and derived semantic constructs are largely omitted. It is therefore unavoidable to consider the respective contextual situation. Very few papers address this topic to any degree [55, 25, 56].

The stated shortcomings are closely related to immanent EML issues in regard to the specification of semantics and an appropriate consideration of pragmatics [40, 57]. Consequently, it seems to be reasonable to investigate these aspects from scratch. The analysis of EML use cases and particularly resulting requirements on the semantics may facilitate the selection of appropriate syntactical extension mechanisms in later design stages [45].

Research Problem 7: General Lack of Reference Methods and Mechanisms

The provision of an integrated EML extension method could further facilitate the standardised specification of extensions for a particular EML, fostering their dissemination and reuse. Recent research reveals that the majority of EML extensions are defined by informal statements and underlying design procedures are rarely externalised in a comprehensive manner [33, p. 50], [19, p. 11]. Both aspects are amplified by the absence of generally applicable meta-model-based extension mechanisms, the lack of integrated extension methods, and shortcomings of existing methods like in BPMN [58, 59].

Extensibility is not explicitly considered in leading meta modelling languages and a general lack of reference mechanisms and procedures has to be concluded [39]. Atkinson et al. [26] state that "no existing modelling framework offers the ideal mix of features needed to support the full range of modelling language extension requirements found in the enterprise computing domain" [26, p. 49]. The absence of a consistent methodical and architectural base for EML extensions further hampers the design of powerful modelling tools and impedes effective reuse in the sense of adapting extension techniques to several domains.

1.2.3 EML Definitions in General

Research Problem 8: Abstraction Conflicts and Missing Integration in MOF-based Languages

Analysing BPMN extensibility reveals several shortcomings and inefficiencies on the meta meta model level, which is defined by the MOF. Concerning the *abstract syntax*, the separation of abstraction level is broken, as MOF imports and uses the UML Infrastructure Library that is originally positioned on the meta model level within the UML specification. This cycle definition causes further problems, e.g. in regard to the possible application of the Profile mechanism. It should therefore be clarified and fixed.

Additionally, the *concrete syntax* of a modelling language cannot be defined in a generic manner on the meta modelling level, although the Diagram Definition (DD) standard is predestined for this purpose. Confusingly, DD and its contained packages (namely DG) are originally positioned on the model level, which should be discussed, too [45, p. 109], [42, p. 139].

Research Problem 9: Limited Capabilities of MOF for Enterprise Modelling

The above mentioned research problems cumulate in the limited expressiveness of MOF for EM in general. Basically, MOF is originally not designed with a focus on EML definition, but rather as an instrument for technically driven tasks and tool integration. However, MOF inevitably gets into the focus of EM, as it serves as a well-disseminated standard for the BPMN definition. Especially from the perspective of BPMN, MOF needs to provide more efficient and less contradictory capabilities for different aspects.

In its current version, MOF amalgamates conceptual and technical concepts [60]. It is also not possible to explicitly define perspectives or comparable instruments for complexity reduction. MOF only provides the general and less specific instrument of Packages. Extensibility is limited to package-based extensions, rather simple Tag-based annotations, and an unclear role of the Profiling mechanism [61, 45].

There is also no guidance on defining the concrete syntax of languages. Respective specifications (e.g. in the case of BPMN) are hence limited to textual descriptions and the representation of symbols and icons. This hampers tool-independent exchange of meta models as well as the exchange of their extensions. It could also cause variance on the shape of model elements. Finally, appropriate integration between abstract and concrete syntax is also missing.

Research Problem 10: Limited Consideration of Semantics on the Meta Model Layer

Considering the definition of EMLs further introduces the traditional yet unsolved challenge of defining semantics on the meta model level. Reflecting the specifications of BPMN and MOF, the complete lack of specification instruments for the definition of semantics has to be bemoaned. Semantics are usually specified in a textual and unsystematic manner (cf. [62, pp. 67-69], [63, p. 19], [64, p. 108], [65, pp. 690, 706], [66, p. 485]). There is also no immanent differentiation between material and formal semantics, which causes a severe issue in the context of BPMN and process modelling in general [67, 8]. Also ambiguity aspects in regard to material semantics are not considered (e.g. lexical ambiguity or different epistemological positions). Accordingly, semantic description techniques can neither be applied for EML extensions nor re-used for extension justification. A deeper analysis of these aspects is therefore generally necessary.

Related Work

Very few research works investigate extensibility of EMLs. Relevant works are briefly considered below in order to differentiate them from this thesis. More detailed analyses of these considered related work can be found in the published research papers that are referred to in the consolidation parts of this work.

Bjeković et al. [21, 22] discuss flexibility of standard EMLs and thereby investigate the role of pragmatics and semantics on a fundamental level. The authors primarily aim at understanding and explaining the main drivers and factors behind the actual usage of EMLs in order to identify particular differences from the intended

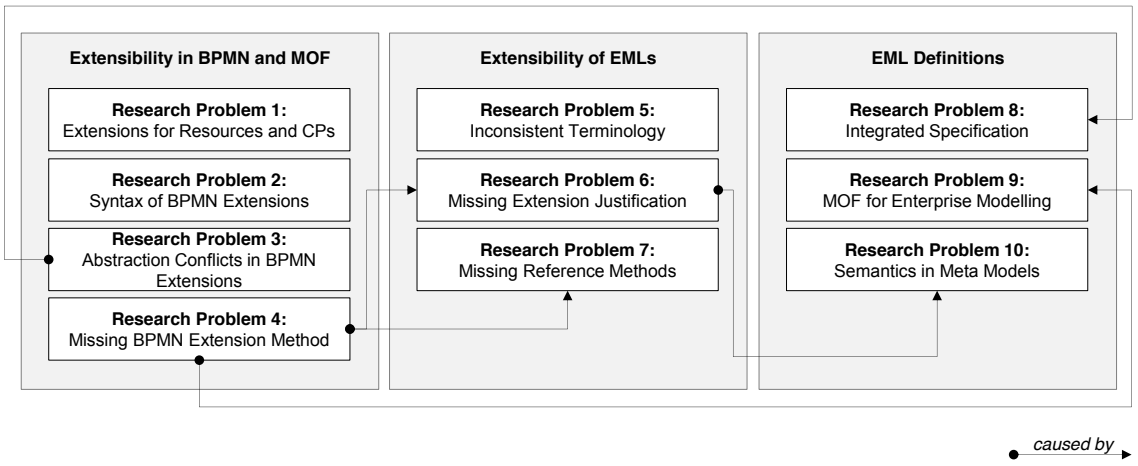

Fig. 1.1. Consolidated research problems

usage of an EML. Also the most efficient degree of standardisation is covered [22]. This approach is generally similar to the research directions proposed in this thesis, but the research of Bjeković et al. is rather theoretically driven, omitting any practical application in BPMN and respective limitations from MOF, for instance.

Karagiannis [68] and Visić et al. [69] discuss flexible modelling languages and agile modelling methods. Agility and flexibility of modelling methods is an outstanding aspect that corresponds to the premises of this thesis. However, the stated authors rather work in the field of the ADOxx modelling suite, which implicates a strong tooling focus, while OMG-related modelling standards are not considered.

Similar to that, Frank [25] investigates the analysis and design of DSMLs in order to provide perfectly fitting languages for business tasks. A strong user focus as well as a special consideration of the particular modelling context is similar to this thesis, but the author rather focusses on dedicated language design instead of the extension of standard EMLs.

Stroppi et al. [48] introduce a transformation-based extension method for the well-guided design of BPMN extensions in compliance with the BPMN meta model. The approach starts with a domain model that is translated to a valid BPMN extension model and a corresponding exchange specification [48]. Stroppi et al. [48] primarily focus on the abstract syntax and specific tool implementation, while semantic issues or peculiarities of EMLs are not discussed.

Atkinson et al. [26] briefly discuss extensibility of EMLs by proposing basal extension purposes. However, their work has not been continued in recent years and the authors mainly focus on a technical level of EMLs.

Patig [30, 70] analyses and investigates the evolution of modelling languages, namely Petri Nets and Entity Relationship Models. Thereby, particular variation mechanisms as well as evolution trends are proposed in a rather large context. However, the focus lies not on EM, but rather on simple conceptual modelling languages.

Finally, there are some authors which conduct specific investigations on the extensibility of single languages. Kopp et al. [34] provide an extensive analysis of BPEL extensions and Pardillo [31] examines lightweight Profile-based extensions within UML. However, both works rather appear as isolated research papers which are not integrated within larger research on extensibility.

1.3 Research Objectives

Figure 1.1 summarises the stated research problems and outlines their relations. Despite the perceived need for appropriate methodical support, research on EML flexibility and EML extensions is limited and most EMLs lack in the provision of respective mechanisms. This thesis is hence driven by two main premises. A particular EML should *"be extensible"*, in the sense of providing a sufficiently specified, consistent, and comprehensive syntactical and semantic environment that enables language extensions. In contrast to this rather language-oriented perspective, the premise *"make useful extensions"* covers the systematically guided design and evolvement of extensions in accordance with user needs. Based on the above outlined research problems, the following research objectives are pursued by this thesis.

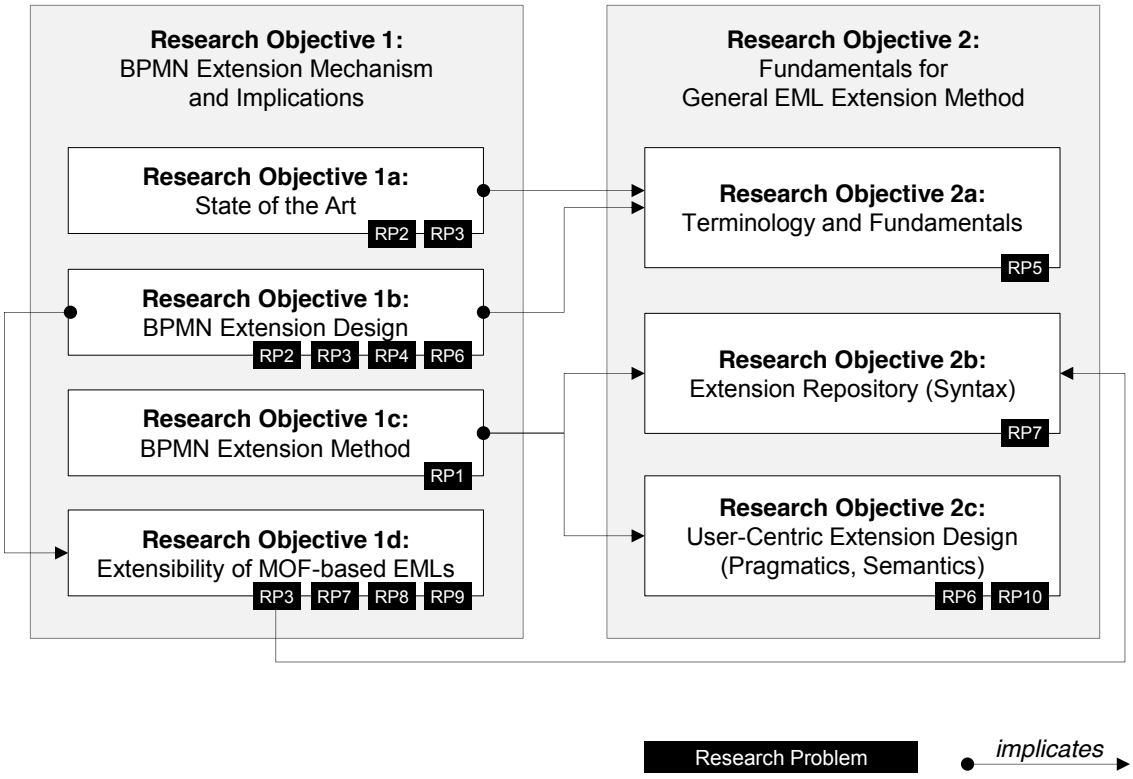

Fig. 1.2. Research objectives and associated research problems

Research Objective 1

Research Objective 1 covers the step-wise, incremental *improvement of the BPMN extension mechanism* that includes the methodically guided design of particular BPMN extensions as well as the derivation of generalisable *artefacts for other MOF-based languages*. This research objective explicitly respects the current state of the art within the OMG environment, i.e. the current BPMN specification as well as its particularly existing methodical extensions [48]. The approach is motivated by practical reasons and the required relevance of design-oriented research [71]. It is intended to enhance existing standards and commonly applied methods instead of

defining artefacts completely from scratch. This includes particular workarounds that may exist in the area of BPMN and MOF. The concerned idea of working with a certain state of the art is reconsidered within the *Standard+X* approach. The first research objective is consequently divided into the following sub-goals:

- **Research Objective 1a**: Investigating and summarising the *state of the art* in regard to BPMN extensibility and EML extensibility in general.
- **Research Objective 1b**: Design of a *BPMN extension for resource modelling in Product Engineering* and a *BPMN extension for Clinical Process Management*.
- **Research Objective 1c**: Design of a *BPMN extension method* with a special consideration of reusing OMG standards and already defined method fragments.
- **Research Objective 1d**: Inductive derivation of insights and artefacts for general *extensibility of MOF-based EMLs*.

Research Objective 2

Research Objective 2 covers the elaboration of fundamentals for a generic EML extension method, as some aspects of BPMN-related extensibility concern EMLs in general. Respective findings hence indicate more general insights on a generic and methodical layer. This objective is rather normative and implicates the consideration of further implementations in the context of EMLs. It also covers fundamental and general topics and is hence rather theoretically driven [71]. The second research objective is divided into the following sub-goals:

- **Research Objective 2a**: Consolidation of *key terminology* in the context of EML extensibility and elaboration of an *extension framework* in order to characterise the type of EMLs and their extensions. This seems to be necessary in order to understand different types of EML extensions that are intended by users in order to appropriately match use cases and syntactical extension mechanisms.
- **Research Objective 2b**: Elaboration and specification of a *mechanism repository* in order to facilitate generic extension design. This implies a special consideration of syntactical operations and design alternatives.
- **Research Objective 2c**: Outlining an EML extension method with a special focus on *user-centric extension design*. This implies an explicit consideration of *EML pragmatics* and *EML semantics*, which is important for mainly two reasons. First, the expectations and requirements of prospective language extension users should fall within the focus in order to provide useful and adequate extensions and avoid technically driven overemphasis of pure syntax operations. Second, the language engineer needs to be supported methodically from the beginning in order to explicitly consider user expectations or domain peculiarities.

Overview

Especially Research Objective 2c is featured by a notable level of complexity and multi-facetedness in terms of EML semantics. Thus, it is explicitly intended to elaborate respective fundamentals first and outline possible realisations which need to be intensified and extended in further research.

Figure 1.2 presents all research objectives, the covered research problems, and the relations between the research objectives. In particular, Research Objectives 1a and 1b serve as preparation for the solution of Research Objective 2a, as they are expected to provide several insights on the current state of the art. Research Objective 1c is expected to provide fundamentals for Research Objectives 2b and 2c, as it addresses syntactical, semantic, and procedural aspects that might be inductively abstracted to EML extension design in general. Further, Research Objective 1d facilitates the examination of Research Objective 2b to a certain degree, as it intends the provision of generic EML concepts. The derivation of them is fostered by tackling Research Objective 1b.

1.4 Research Approach

1.4.1 Research Papers and Consolidation Essay

The stated research objectives have been tackled within a cumulative research project that is composed of **22 single research papers** and a so-called **consolidation essay**. The consolidation essay is mainly represented within this document by its three main components in the Parts II, III and IV.

Fig. 1.3. Legend for structuring research papers and the consolidation essay

All relevant papers as well as the essential parts of the consolidation essay are summarised in Fig. 1.4 and Fig. 1.5. The used notation is represented in Fig. 1.3. A particular research work is identified by its publication ID and the main contribution is depicted in the middle of the rectangle. Following [53] and [72], the type of contribution is stated in square brackets. We divide specific artefact types (*extension*, *method fragment*, *method*), rather abstract *frameworks*, as well as *overviews*, which indicate a state of the art analysis or some kind of consolidation. If a research work provides a main contribution (e.g. by integrating different prior results into one main artefact), then the rectangle is grey and highlighted by a thick borderline. Different contributions are covered by a particular research theme that represents the tackled research objectives. The specific relation between single publications is expressed by three relationship types. *Input* stands for the re-usage of artefacts, for instance, and their evolvement and progression. *Relates to* represents a particular coupling between publications, while a specific input relation is missing. *Instantiation* represents the application of generic principles to a specific case. For instance, the application of the generic Profiling technique to BPMN [45, 42] or the application of the *SemFrame* framework to BPMN extensions [57, 9].

Research Objective 1

Research Objective 1 and corresponding sub-goals are mainly tackled within published papers as outlined in Fig. 1.4. Several papers focus on Research Objective 1a

by considering the state of the art in terms of BPMN extension design [33, 73], as well as the critical analysis of the BPMN extension mechanism in general [39]. Research Objective 1b is tackled by designing a BPMN extension for resources in Product Engineering [36, 74] and by the iterative design of BPMN4CP, an extension for multi-perspective hospital modelling [27, 75, 76, 77]. Revisions of BPMN4CP thereby cause implications for a general BPMN extension procedure [27, 76] and further evolve several architectural extensions of the MOF [77].

Research Objective 1c is considered by different publications which focus on syntax [59, 44, 42, 45, 78], semantics [9], and procedural aspects [55]. Motivated by the concrete example of BPMN, two papers refer to MOF-based extensibility in general and, hence, tackle Research Objective 1d [45, 78].

Research Objective 2

Research Objective 2 and respective sub-goals are tackled by a combination of single papers [79, 41, 57, 67, 28, 80] and insights or artefacts that are inductively inferred from BPMN-related research. These papers provide the foundation for tackling Research Objectives 2a, 2b and 2c within this consolidation essay as outlined in the middle of Fig. 1.5.

The consolidation essay intends the integrated and multi-perspective consideration and analysis of EML extensibility in order to prepare the evolution of a generic EML extension method. *The consolidation essay, hence, not only serves as a simple summary of already completed research, but also advances this research in specific aspects.* The underlying papers for Research Objectives 2a, 2b and 2c are elaborated in Sect. 2 in order to introduce the structure of this consolidation essay.

1.4.2 Application of Design Science

All research works as well as their final aggregation follow the Design Science Research (DSR) paradigm, aiming at elaborating and designing innovative and relevant artefacts in a rigorous manner [81, 82, 71, 83, 84]. More precisely, several research works follow the recently introduced requirements-driven DSR approach [79]. Requirements-driven DSR proclaims the consequent usage of different requirement types for the analysis, design, construction, and evolution of artefacts in order to facilitate design decisions, enhance their evaluation and support comprehensibility as well as procedural transparency [79].

For instance, requirements are generally elaborated in order to explicate domain features [33, 27]. In addition, a well-defined and fine-grained requirements set formulates respective revision need for the extension with different perspectives [76]. Requirements are further utilised for specifying the need of generic extension mechanisms on the meta meta model layer and their language-specific instantiation [45, 42]. Research on single artefacts was partially conducted as an iterative design process, whose iterations were caused by the identification of updated or expanded requirements which came from application in real-world projects or progress in research. In particular, the BPMN4CP extension went through two iteration cycles, causing additional meta model extensions [27, 75, 77, 76].

DSR generally requires the rigorous evaluation of artefacts. However, this task is difficult to realise in a straightforward manner since modelling languages or frameworks serve as rather normative artefacts (cf. [85, 79]) whose effects cannot be measured precisely within complex socio-technical systems like IS. Against the backdrop of this issue, a mild form of evaluation was conducted by describing the applicability and usefulness of artefacts by demonstrations (according to [83]).

Artefacts that were designed in order to tackle Research Objective 1 were therefore usually evaluated by applying them to representative use cases, partly in the context of real-world projects [36, 74, 75, 27, 77]. For some other artefacts, the proof-of-concept principle was conducted (e.g. in case of the Profiling extension mechanism [45, 42]).

Artefacts that were designed in regard to Research Objective 2 have not been evaluated yet due to their essential and consolidating character that intends to integrate several research streams in order to elaborate a solid foundation for a generic and integrated EML design method. Nevertheless, further research on that is indeed necessary (cf. Part V). For instance, it is promising to investigate whether EML extensions designed with the EML method are more appropriate for their users. Such topics are excluded from this work. Instead, required fundamentals, method fragments, and techniques are elaborated in the context of BPMN and MOF-based languages.

Research on BPMN and EML extensibility generally ranges within the two extrema of practical applicability (e.g. BPMN and MOF standards with all peculiarities) and methodical rigour, attempting to reach a possibly clean, integrated and consistent solution (e.g. re-considerations of semantic specification dimensions). Respective investigations are driven by critical reflections and in-depth analyses of existing artefacts. For instance, the extension mechanisms of BPMN, several parts of the MOF, as well as the DD standard are critically analysed in a fine-grained manner [45, 42, 77].

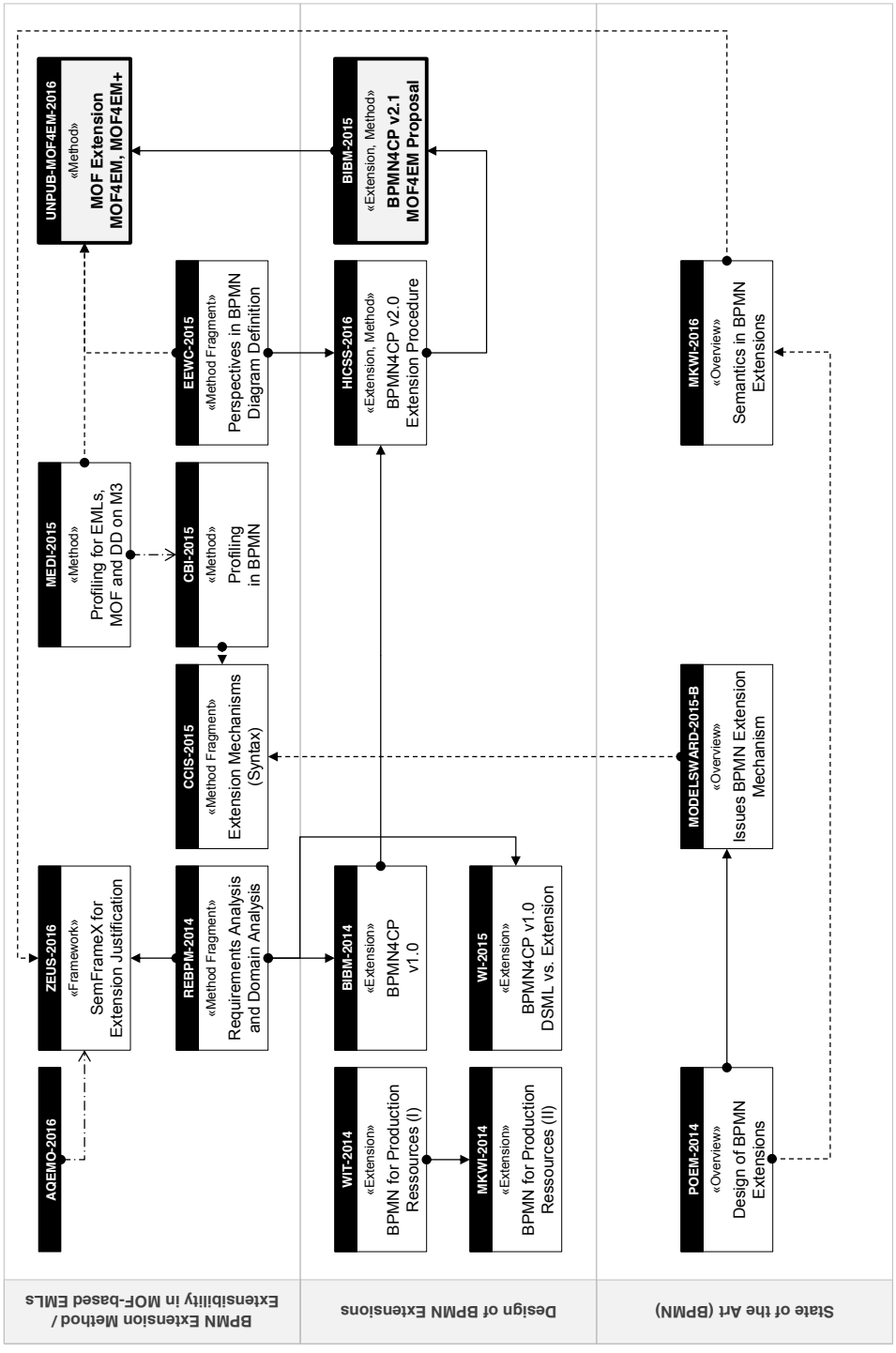

Fig. 1.4. Research design, relevant publications and considered Parts of the consolidation essay in the context of Research Objective 1

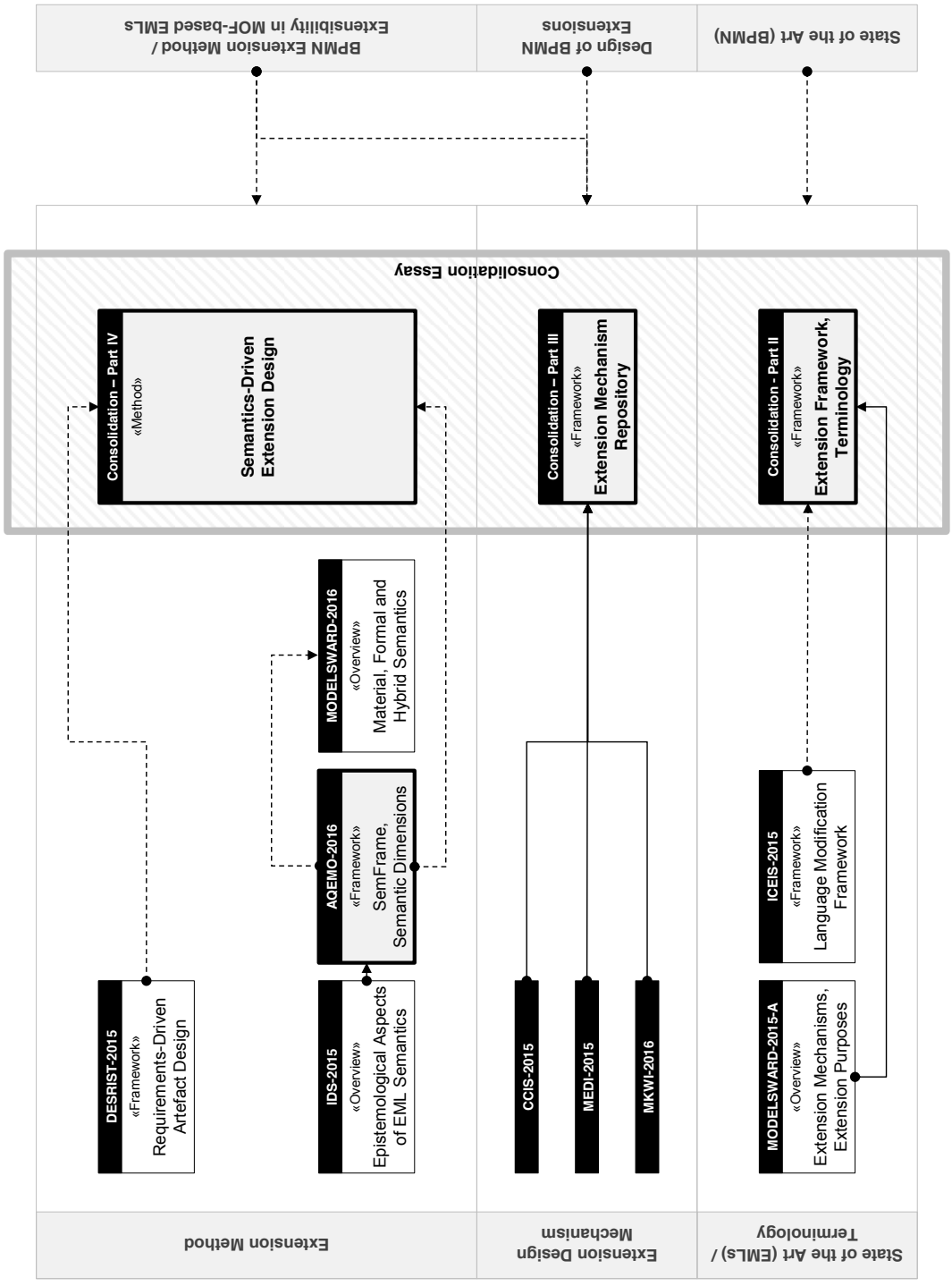

Fig. 1.5. Research design, relevant publications and considered Parts of the consolidation essay in the context of Research Objective 2

2

Organisation of the Consolidation Essay

As mentioned in Sect. 1.4, the consolidation essay aims to tackle three objectives. The essay is consequently divided into three Parts. Each of them bases on several papers that are referred to or partly applied within each respective Part (cf. Fig. 1.5). Each paper is assigned to exactly one Part according to its most relevant contribution or impact.

Part II provides fundamentals for EML extensibility by elaborating terminological and conceptual foundations and by proposing different types of EMLs and EML extensions in order to facilitate problem understanding. This Part bases on research papers which investigate the state of the art in terms of extensibility in general [28, 80], BPMN extensibility [33, 73], and current issues with the BPMN extension mechanism [39].

Part III elaborates a generic repository of different extension mechanisms by summarising recently applied mechanisms within a specification framework. This Part bases on several publications regarding insights from designing specific BPMN extensions [36, 74, 27, 75], a critical analysis of the current BPMN environment [59], as well as papers on the Profiling technique [45, 42].

Part IV introduces semantics-driven extension design, which proclaims a strong focus on intended pragmatics and intended semantics. Due to the notable lack of research on these topics, several fundamentals are additionally provided in order to consolidate relevant research, e.g. pragmatics, types of semantics, and respective description instruments. This Part bases on a publication on the methodical design in DSR projects [79], papers on methodical BPMN extension design [55, 76, 77], a research paper on BPMN extension justification [9], publications on EML specifications with MOF [44, 78], as well as a rather fundamental consideration of semantics from an epistemological point of view [41] and two research papers on different semantic dimensions in EMLs [57, 67].

Part V summarises the entire work and discusses the achievement of the research objectives by consolidating their contributions. Accordingly, detailed implications for further research are given in order to provide a solid base for following research tasks.

Each publication is characterised by a set of the attributes in order to provide a concise and comprehensive overview that facilitates its integration within the entire thesis. Therefore, the following attributes are introduced:

- *Publication ID*: A publication is identified by a unique identifier that is created by the acronym of its publication medium and the year of publication.
- *General Information*: This category covers basic information, i.e. title, authors, year of publication, publication medium (e.g. proceeding or journal) as well as the number of pages.
- *Rankings*: Further, the ranking of a medium is quoted in the form of the VHB-JOURQUAL-3 ranking [86] and the WKWI-2008 ranking [87]. Due to a lack of common and unified information on particular acceptance rates, possible rank upgrades or downgrades within the WKWI-2008 ranking are omitted [87, p. 162]. It should be further noted that several publications cannot be ranked with the stated metrics, as they are part of adjacent research disciplines like Computer Science (e.g. [39, 59]) or Bioinformatics (e.g. [27, 77]).
- *Contribution of the Authors*: According to the authors of a publication, the work share of each author is given in percentage values. Thereby, author contributions are represented by the titles or topics of the regarded paper sections an author has written. In case of rather conceptual contributions (e.g. in regard to the research strategy or focus of a publication), an author contribution is referred as "research conception". If a publication has only one author or the contribution of the second author is limited to conceptual work, then further detailing is omitted.
- *Abstract*: The research paper is summarised by its abstract.
- *Summary of Contents*: This category aims to provide a structured and aggregated overview of the publication in order to briefly summarise its contents and contributions to the entire thesis. Therefore, the main motivation behind a particular paper is stated. The underlying research methods as well as potentially re-used results from previous papers (referred as input) are further stated. In addition, the main contributions of a paper are explicated. The contributions are differentiated in regard to their specifity (meta level, meta meta level, general implications, consolidations etc.). The general context of a paper is stated in order to facilitate associations to particular research problems (BPMN, EMLs, EM in general).

The original version of each publication is attached within a separate document. In this document, all research papers are ordered according to their integration within this consolidation essay as presented in the table of contents.

Part II

Fundamentals

3

Relevant Publications

3.1 Publication POEM-2014

General Information:

Title:	Classification of Domain-Specific BPMN Extensions
Authors:	Richard Braun, Werner Esswein
Year:	2014
Medium:	Frank, U., Loucopoulos, P., Pastor, O., Petrounias, I. (Eds.): The Practice of Enterprise Modeling, 7th IFIP WG 8.1 Working Conference
Series:	Lecture Notes in Business Information Processing, Vol. 197
Pages:	42-57
Reference:	[33]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions of Authors:

Richard Braun:	90% (introduction and motivation, BPMN extension analysis framework, BPMN extension classification, implications, conclusion)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Lack of methodical guidance on BPMN extension design; missing extension overview and comparison
Input and Method:	Literature review, content analysis of 30 extension definitions
Contributions:	State of affairs in regard to BPMN extensions: • Majority of extensions is not compliant to BPMN standard • Methodical shortcomings

Implications:

- Determined need for integrated BPMN extension method
- Classification framework: quantitative and qualitative criteria
- Extension types (abstract syntax, concrete syntax, semantics)

Abstract:

BPMN is a standard for modeling business processes and provides meta model concepts for the design of extensions. Thus, domain-specific extensions of the BPMN are facilitated. This research article provides an overview of BPMN extension development by the descriptive analysis and classification of 30 BPMN extensions. An extensive literature review was conducted in order to find published extensions. Further, a classification framework was designed to enable a comprehensive analysis of each extension. The analysis showed, that four out of five extensions are not compliant with the BPMN standard. Also, we found several methodological shortcomings that should be tackled in further research.

3.2 Publication MODELSWARD-2015-A

General Information:

Title:	Towards the State of the Art of Extending Enterprise Modeling Languages
Authors:	Richard Braun
Year:	2015
Medium:	Hammoudi, S., Pires, L.F., Desfray, P., Filipe, J. (Eds.): Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development
Pages:	394-402
Reference:	[28]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions of Authors:

Richard Braun:	100%

Summary of Contents:

Context:	EMLs
Motivation:	Missing conceptualisation and overview on EML extensions
Input and Method:	Consolidation of the current state of affairs
Contributions:	Motivation and implications for extensible EMLs: • Investigated lack of consideration within meta models and meta modelling languages • Classification of extension purposes and extension mechanisms

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.

3.3 Publication MODELSWARD-2015-B

General Information:

Title:	Behind the Scenes of the BPMN Extension Mechanism – Principles, Problems and Options for Improvement
Authors:	Richard Braun
Year:	2015
Medium:	Hammoudi, S., Pires, L.F., Desfray, P., Filipe, J. (Eds.): Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development
Pages:	403-410
Reference:	[39]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	BPMN
Motivation:	Critical analysis of BPMN extension mechanism
Input and Method:	Critical in-depth analysis of BPMN meta model and extension mechanism
Contributions:	Elaboration of nine problem statements and implicit requirements for meta meta model changes (i.e. MOF revision)

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.

3.4 Publication ICEIS-2015

General Information:

Title:	A Generic Framework for Modifying and Extending Enterprise Modeling Languages
Authors:	Richard Braun, Werner Esswein
Year:	2015
Medium:	Hammoudi, S., Maciaszek, L.A., Teniente, E. (Eds.): Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 2
Pages:	277-286
Reference:	[80]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	C

Contributions of Authors:

Richard Braun:	90% (introduction and motivation, fundamentals, framework architecture, conclusion)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	EMLs
Motivation:	Lack of methodical support for EML modifications
Input and Method:	Consolidation of the current state of affairs
Contributions:	Framework for meta model modifications and their consequences: • Dimensions: Abstract syntax, concrete syntax, semantics • Operations: Add, remove, specify, redefine

Abstract:

Conceptual modeling languages are of great importance within information systems management. During the last decade, a small set of commonly used enterprise modeling languages established and gained broad acceptance in both academia and practice (e.g., BPMN). Due to their dissemination, these languages often need to be extended or adapted for domain-specific or technical requirements. Since most modeling languages provide rather poor extension mechanisms, it is necessary to modify a language meta model directly. However, there is lack of integrated methodical support for these modifications. Within this position paper, we therefore proclaim a

generic framework for modifying enterprise modeling languages on the meta model level. The framework is divided into the main parts of a modeling language (abstract syntax, concrete syntax, semantics) and respective operations (add, remove, specify and redefine).

3.5 Publication MKWI-2016

General Information:

Title:	Semantics in the Context of BPMN Extensions – State of Affairs and Research Challenges
Authors:	Richard Braun, Werner Esswein
Year:	2016
Medium:	Nissen, V., Stelzer, D., Straßburger, S., Fischer, D. (Eds.): Tagungsband Multikonferenz Wirtschaftsinformatik 2016
Pages:	1119-1130
Reference:	[73]

Rankings:

VHB-JOURQUAL-3:	D
WKWI-2008:	C

Contributions:

Richard Braun:	90% (introduction and motivation, semantics and language extensibility, semantic review of BPMN extensions, implications and possible approaches, conclusion)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Semantic analysis of BPMN extensions
Input and Method:	Literature review, content analysis of 36 extension definitions
Contributions:	In regard to BPMN: • Descriptive analysis of BPMN extensions in regard to semantics • Lack of semantic considerations In regard to EMLs in general: • Semantic extension types • Research challenges for EML semantics

Abstract:

This research article addresses the issue of meta model semantics in extensions of enterprise modeling languages by conducting an extensive analysis of 36 extensions of

the process modeling language BPMN. The study reveals remarkable shortcomings regarding to semantic considerations both during extension design and extension specification. Only 50% of the analyzed extensions provide a clear description of the required domain concepts and only the minority conducts some kind of ontological comparison with original BPMN elements. Extension semantics largely arise directly from syntactical additions, but some extensions also introduce additional semantics without syntactical changes. Positively, semantic redefinitions are very rare. However, 80% of the analyzed extensions do not provide any semantic specification of the extension elements. We therefore outline several aspects for further improvement.

4

Terminological and Conceptual Foundations

Within this Section, the terminological and conceptual foundations for a prospective extension method are elaborated. We consolidate current research in the context of EML extensions and propose definitions for key terms. It should be understood as conceptual-deductive work (referring to [88]), aiming to consolidate and organise commonly used concepts in order to prepare the design of a prospectively useful framework. It primarily contributes to the knowledge base and prepares the design of artefacts in the sense of extension methods (cf. [81, 71]).

4.1 Peculiarities of Enterprise Modelling Languages

Enterprises are multifarious, heterogeneous socio-technical IS, whose components are interrelated within a complex nexus of interdependencies on different abstraction levels [1, 2, 3]. EM intends to conceptualise, abstract, and represent parts and aspects of enterprises by creating conceptual models in order to foster communication between involved stakeholders and enable an integration of static, procedural, and functional dimensions [89, 5, 4, 11, 6].

EM intends to understand, improve and control enterprises [90, p. 124] and should work as conceptual infrastructure for a high level of integration [12]. EM serves as a capable and auspicious approach for managing present-day business complexity in the light of increasing interdependencies between and within IS. It further aims to make relevant things explicit and to provide a mechanism for working with these models [6]. While enterprise models are usually designed for documentation, intersubjective communication purposes and managerial decision making [3, p. 26], they are also useful for operative support and automation [7, 8], which indicates a parallel consideration of material semantics and formal semantics [73].

This Section aims to work out the particularities of EMLs in order to clarify its specific position within the class of conceptual modelling languages and emphasise the inherent need for extension. This deeper consideration is necessary due to missing consensus on elementary EM concepts, since EM is a field under ongoing research [3]. EMLs are conceptual modelling languages that provide concepts for modelling characteristic aspects of enterprise systems like organisational architectures, business processes or resources [10, 12]. EMLs are usually semi-formal languages, having a precisely defined formal syntax and mostly informal semantics [13, 14]. In comparison to design-oriented languages or domain-specific programming languages [91, 92], EMLs feature at least four prominent specificities:

Dynamics: Modern enterprises and businesses are complex socio-technical systems having knowledge-intensive, project-specific value creation processes. They are consequently characterised by ongoing dynamics and a certain level of inherent volatility, which requires agility and constant flexibility (e.g. [93, 94]). Frequent changes may also influence the way of communication within and about enterprises, which in turn provokes a new impetus for supplementing EMLs with context-specific concepts [3, pp. 23-26]. Consequently, the languages used to model enterprises become the subject of frequent change, as EMLs provide both representational and linguistic functions [95, 40]. Karagiannis [68] states that "models are means of representing the relevant knowledge pertaining to different facts", establishing an enterprise-related terminological box [68, p. 6].

Perspectives: Modern businesses are further characterised by a high degree of inter-disciplinarily, cross-company cooperation and project teams with different professional backgrounds [12]. Consequently, enterprises face the issue of establishing a common communication base between different users and stakeholder groups [7], having different purposes [6], conceptual perspectives [96] and also different modelling languages [97]. In EM, the systems under study have an inherent complexity, covering multiple views, facets, and perspectives. Understandings of enterprise-related issues and concepts between stakeholders may change or evolve over time (referring to [68, p. 6]). The underlying (perhaps time-dependent) multi-perspectivity requires some kind of EML adaptivity, since all-encompassing standard EMLs are rather illusory and impractical in terms of complete a-priori designs [22, p. 9], [35, p. 24]. Perspectives are generally accepted techniques for stakeholder-specific complexity reduction and conceptual integration in EMLs [12, 96, 6].

Semantics: Semantics require special attention as EMLs typically address real world things to a certain degree, which causes the issue of ambiguity due to individual mental conceptualisations and their consequences to model understandings [98, 99, 100, 22, 101]. Despite different personal interpretations, the understanding of rather generic BPMN concepts like Tasks or Resources may be different according to specific domains. This is especially relevant for standard EMLs like BPMN [33, 27]. Additionally, also formal semantics in the sense of automatically interpretable models have to be considered as demonstrated in Bork & Fill [8]. A detailed discussion on the ambivalent role of semantics in the context of EMLs can be found in recent literature [41, 57, 67]. Particular investigations of BPMN and its extensions can be found [73].

Concrete syntax: Concrete syntax plays an outstanding role and EMLs usually serve as diagrammatic modelling languages [46]. In contrast to textual modelling languages, the intent is to facilitate the visually supported transfer of model information [102]. This can be supported by the integration of commonly used domain symbols of a stakeholder group, for instance [25]. Consequently, not only the abstract syntax, but also the concrete syntax has to be taken into account when examining EML extensions.

4.2 Discussion on Standard Enterprise Modelling Languages

The *Standard+X* approach bases on the idea of extending a well-known standard EML with specific concepts and ensuring the validity of the EML core by applying inherent extension mechanisms. Designing an EML extension in conformity to this extension mechanism should also ensure exchangeability and reusability of them. The extension approach is primarily useful for standard languages or commonly used languages with a high degree of dissemination. Languages with a limited scope like DSMLs are rather a subject for language revisions and the need for non-invasive modification is inherently supposed to be lower. Below, the importance of standard EMLs is discussed controversially by elaborating pros and cons of standard creation in general. The recent discussion on specifying an EPC standard (e.g. [103]) indicates the necessity of such a discussion. However, a general discussion of EML standards has been omitted so far.

4.2.1 Pro Standardisation

Standardisation of EMLs facilitates four aspects, which are considered below. First and foremost, the *communication* between human actors and also between machine actors can be improved remarkably by the commonly accepted usage of a standardised EMLs. This covers the vocabulary of the language (abstract syntax), their graphical representation (concrete syntax), as well as the meaning of language constructs (semantics). Moreover, different stakeholder perspectives can be integrated [22, p. 4] and the language is generally designed for reusability across domains and application scenarios [22, pp. 4ff], [68, p. 5]. Standardised communication between machines refers to *interoperability* between modelling tools. Establishing and applying an EML standard may diminish tool dependency and could also avoid the expensive and error-prone transformation between dedicated language implementations.

Another important aspect is the level of *quality, reusability, and reference building*. Standards foster the agreement and consensus processes within particular communities (cf. [104, 105]) and may hence facilitate the collection and consolidation of established and well-proven language concepts, rules, and approaches within one EML (cf. [40]). Such a concentration of knowledge is promising in terms of EML evolution and EML quality (e.g. [70]). For instance, the UML was largely motivated by the integration of a number of commonly used object-oriented design languages [106, 107] and parts of early BPMN extension become part of the BPMN standard itself (cf. [108, 15]). Focussing on a limited set of standard EMLs further avoids exuberant method pluralism (according to [29]) and standardising could also avoid redundant design of generic concepts and constructs in dedicated approaches (e.g. the recreation of control-flows as covered in [43] and [27]).

Finally, we proclaim standards as an important artefact type and general *object of research in design-oriented research*. As stated in Braun et al. [79], design-oriented research in the IS discipline struggle with certain demarcation issues from professional engineering. More precisely, it seems to be at least debatable whether research institutions having traditionally very limited budgets and resources may produce highly innovative and especially practically applicable artefacts (in comparison to large business companies with notable research budgets). It seems to be

promising to focus on topics which are highly relevant and important on the one side, but not that attractive for business companies on the other side. Consolidating a certain state of affairs and deriving particular standards could be such a research niche that is worthy for consideration.

4.2.2 Contra Standardisation

Despite the number of benefits, there remain some issues that have to be considered. The first issue addresses the paradigmatic determination of particular structures and concepts, which could impede the integration of innovative approaches within an EML (e.g. Multilevel Modelling [12]). The normative character of EML standards could also impede the linguistic function of a language in the sense of subjective conceptualisations about a particular area of discourse [109, 110, 40]. The fixation of a particular EML version could also lead to the temporal acceptance of shortcomings, which are barely correctable due to the long revision intervals [111]. For instance, some OMG standards like MOF or BPMN reveal some architectural shortcomings or inconsistencies [85, 112, 39, 42], but respective revisions or requests for revisions are missing so far.

4.2.3 Towards Extensible Standard EMLs

We proclaim the general definition and usage of EML standards due to the stated benefits. However, several peculiarities of EML standards as well as the reflection of the stated contra arguments cause the following necessities. Most EMLs require mechanisms for situation, context-specific or domain-specific adaptation in order to both provide concepts for multiple industries as well as capabilities for the definition of fine-grained integration rules of particular domains [3, 40, 28]. It is important to emphasise that the respective extension mechanisms must be provided by the EML itself in order to enable the flexible design of EML extensions, which can enhance the dissemination of a language in turn (referring to [113]). We further argue that each extension of an EML has to be justified based on an analysis of the intended semantics, which come from the required pragmatics of the EML users.

Except the above stated dedicated design or combination of DSMLs [12], it is possible to differentiate between two main approaches: *Meta model composition* [23, 24] and *language extensions* [21, 26, 40, 33]. Meta model compositions refer to the combination and integration of meta models or meta model fragments, which is very useful for integrating PSMLs. For instance, an EML for resource modelling is integrated with a EML for process modelling purposes.

With respect to also rather fine-grained extensions (e.g. semantic extensions [73]), we prefer the standard extension type and refer to this as the *Standard+X* approach. This approach covers all types of extending an EML, i.e. both syntactical and semantic extensions to differing extents. Each kind of extension guarantees conformity to the original EML meta model and neglects any invasive meta model modification.

4.3 Terminology

Extending EMLs naturally arouses the association of extending programmes in the field of Software Engineering, which is a longstanding research discipline in compar-

ison to EM. Basically, an extension in the field of Software Engineering is understood as a programme that can be adapted to new, not precisely predictable tasks, without altering the original software core (according to [114] and [115]). An appropriate adaptation of this term to the field of conceptual modelling languages or EMLs is missing so far. Current discussions on flexible or extensible EMLs struggle within ambiguous definitions about the outcome of adaptation or extension processes. Variants, dialects, or lightweight versions are only some of the used terms (e.g. [22, 19, 69]).

We therefore propose an explicit definition of extensions below. We also differentiate some adjacent concepts that are often amalgamated in the context of EML extensions, i.e. modifications and reductions. Within this discussion, the language under consideration is referred as original language [26].

4.3.1 Extension

Kopp et al. [34] define an extension twofold. In the narrower sense, an extension *enhances the expressiveness and functionality* of a conceptual modelling language by introducing additional types and attributes for the representation of purpose-specific concepts. The authors emphasise that an isolated extension is neither useful nor functional on its own [34, p. 6]. In a wider sense, an extension must *conform to the extension mechanism of a language*.

We basically follow this definition, but suggest a slight relaxation in regard to the kind of existence dependency in order to comprise all component-like extensions or meta-model bridging approaches (cf. [22, 45, 24]). Thereby, extensions should be always *non-invasive*, which means that the original meta model must not be modified or reduced [50, p. 123]. This enables backward compatibility as well as an exchange of models created with the original language. In addition, the original semantic character of the language should be preserved [80].

The demand for non-invasiveness further *forbids any kind of mandatory extensions*. Mandatory extensions cover the introduction of additional constructs (or additional constraints), which have to be met by original constructs. Mandatory extensions need to be excluded and treated as modifications, since fully-compliant models cannot be realised per se [50, p. 126]. Examples for mandatory extensions are introduced associations from original meta classes to extension meta classes with a lower cardinality of greater than zero. Also the mandatory inheritance of properties from introduced generalisations would represent such a mandatory extension. For reasons of clarity, additional generalisations are therefore excluded.

Mandatory:	Additional expressiveness
	Non-invasive
	Non-restrictive
	Backward compatible
Optional:	Conformity to extension interface (if defined)

Table 4.1. Characteristics of extensions

Additional expressiveness can be realised by additional concepts on the same level of abstraction (horizontal) or on a different level of abstraction (specialisation

for refinement; referring to [50]). Horizontal extensions aim at detailing and can be divided into extensions within the same area of discourse and extensions with concepts from other domains or areas of discourse (cf. [26]). Generally, any extension enhances the expressiveness of a original language to a certain extent [69, p. 4].

Below, different types of top-level extensions are briefly introduced according to [50].

Generalisation

Extension by generalisation means that the extended language serves as a superset of different languages, i.e. the original language and additional languages. This is typically conducted for joining different independent languages [50, p. 124]. In its strictest form, generalisation could also constitute integration of additional super types in the case of a very simple and basal language (according to [80, p. 280]).

Each generalisation extension can be treated as a special type of a disjoint specialisation that will be mentioned below [50]. With respect to the avoidance of mandatory and hence invasive extensions, generalisations are only treated as valid and completely compliant extensions, if the direct or indirect properties of the new super types are optional and not mandatorily restricted (related work, e.g. [50], remains imprecise on this aspect). In this case, a generalisation would be classified as a modification, since the type of an original class is obligatorily modified, which indicates that original models would become invalid.

Specialisation

In case of specialisation, the extended language serves as a superset of exactly one original language [50, p. 124]. Specialisation serves as a compliant derivation and can be divided into several types, which are briefly considered below.

Disjoint: A disjoint specialisation is characterised by the introduction of optional, unrelated and hence independent constructs. This leads to a rather low-coupled union of an original language with additional concepts or an additional language, which comprises language composition approaches (e.g. [24]). Disjoint specialisations extensions can be primarily used to conduct horizontal extensions in order to integrate concepts from other domains, for instance (cf. [26]). Except in the case of generalisation (see above), the level of abstraction is therefore usually the same. It has to be stated again that the demand for optionality excludes mandatory properties (e.g. a minimum cardinality equal to or greater than one), which implicate dependencies of additional concepts. Otherwise, such extensions are treated as modifications.

Conservative: A conservative specialisation does not introduce completely unrelated concepts, but rather intends a refinement of already existing ones. It is therefore useful for the (domain-specific) specialisation of rather under-specified or generic concepts. Consequently, the level of abstraction of the additional concepts differs and a vertical extension evolves.

Additive: According to Atkinson & Kühne [50], an additive specialisation combines the two mentioned types and covers both the introduction of so far unrelated concepts (disjoint) and specialising concepts (conservative). For instance,

most BPMN extensions are implemented in this way, proposing both vertical and horizontal extensions [33].

Modification

In contrast to the explicitly emphasised non-invasiveness, several authors give the impression of having a rather broad understanding of extensions in the sense of general language customisations [50, 22, 68]. Consequently, extensions seem to be synonymous with variants (e.g. [40, 19]) or general language modifications [116].

Mandatory:	Additional expressiveness Invasive Restrictive Not backward compatible Ad hoc alteration: No conformity to extension interface

Table 4.2. Characteristics of modifications

We therefore define modifications as *invasive alterations* of the original language, which implies missing backward compatibility. Not all models of the original language can be modelled with a modified language, as any modification determines non-compliant or partially-compliant consequences, which refers to a certain loss of original expressiveness [50, p. 121]. Removing a construct or changing its value range to a more restrictive character could cause modification, for instance [116, p. 2011].

With respect to model configuration management [47], any modification causes a revision of the original meta model indicating a new version of it. Modifications therefore lead to language variants [80].

Overwriting: Overwriting covers the *redefinition or alteration of original meta model constructs*. This encompasses the alteration of owned and associated properties, range values, or even multiplicity values, for instance. Redefinitions can be also realised by simple renaming of original constructs implicating altered semantic references [80, p. 280]. Technically, each overwriting operation is composed of at least one removing operation and at least one adding operation to introduce the desired concepts. In contrast to modification by reduction, the general shape or structure of the original meta model remains stable, but its particular nodes (or node values) are modified. Overwriting also encompasses each intensification and limitation of conditions defined in the original language. As stated above, existence dependencies are hence permitted (e.g. in regard to associated concepts). More precisely, any change of original concepts that cause the exclusion of valid original models is referred to as overwriting modification. Exemplarily overwriting operations include the following (cf. [80, pp. 280ff]):

- Limitation of multiplicity values (e.g.: "0..*" is limited to "1..*" or "1..5" to "2..2")
- Additional superclass that owns mandatory properties
- Introduction of mandatory properties to an original class
- Renaming of elements

Reduction

Literature on flexible EMLs also considers language reduction as an appropriate means for better EML application [117, 118, 119, 120, 22]. Reducing the vocabulary of EMLs is often motivated by a particularly perceived language overload and the existence of rarely used concepts (cf. [119, 121, 122]), which in turn manifests the trade-off between language expressiveness and language comprehensibility [7]. Technically, reduction is realised by omitting constructs of an EML meta model, leading to a smaller meta model. The meta modelling pruning technique of Bea et al. [118] can be applied for this purpose.

Destructive: A destructive reduction refers to the removal of at *least one mandatory* construct from the original language. This implicates a lower level of expression on the one side and a missing backward compatibility on the other side, since not all models created with the extended language can be modelled with the original language (non-compliant or partially-compliant derivation as discussed in [50, p. 121]).

Compliant: A compliant reduction refers to the removal of solely optional constructs. The derived extension therefore constitutes a subset of the original language and each model created with the extension can be created from the original language. However, at least one model of the original language cannot be created with the extension (partially-conformant derivation [50, p. 121]). This might be useful for complexity reducing language subsets [117].

4.3.2 Pseudo Reduction

Modifications should be prevented in order to correspond to the above mentioned benefits of non-invasiveness of EML extensions. However, the need for stakeholder-specific subsets of very comprehensive EMLs like BPMN – that is even exacerbated by the sheer amount of extensions (cf. [33]) – requires some means for de facto reduction in order to create sub-views or filters on the entire vocabularies. We therefore propose a pseudo reduction, called *reduction by extension*.

The introduction of *additional perspectives* enables the creation of situational, complexity reducing subsets of exactly those concepts, which are useful for a particular stakeholder group [44]. Importantly, underlying constructs remain unaffected and respective conditions have to be preserved within the views. These filters may contain additional concepts, if required (e.g. [76]).

Technically, this approach requires respective filter mechanisms. More precisely, view and perspective concepts need to be available on the meta model level, indicating an EML-specific definition on the meta meta model level (cf. MEMO MML [60] and E3 [47]). Despite the promising benefits of this solution, prevalent meta modelling languages like MOF struggle with the provision of respective meta concepts and the necessary groundwork as to be laid on the level of EML design.

4.3.3 Hybrid

In case of combining extensions and modifications, modifying alterations are dominant and prevent a classification as non-invasiveness extension.

4.3.4 Semantic Extensions and Modifications

As stated in Braun & Esswein [73], current literature over-emphasises syntactical aspects in the context of EMLs and EML extensions. However, it is also imaginable to conduct semantic adjustments in order to create semantic BPMN extensions, which keep the original language syntax unaffected [80]. It is therefore necessary to consider the semantic level of EMLs as an explicit place for "semantics only" extensions. In particular, the respective semantic constructs and semantic mappings between syntactical constructs and these semantic constructs have to be addressed as follows [73].

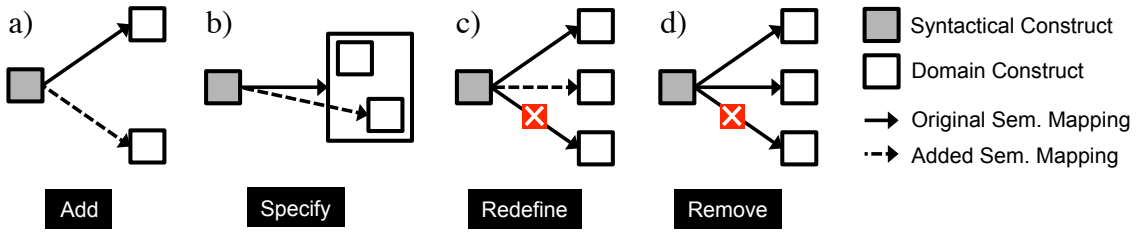

Fig. 4.1. Types of semantic alterations [73, p. 1124]

Semantic Extension

In case of semantic extension, original semantic mappings and domain constructs remain unaffected, but additional semantic mappings and constructs are introduced as depicted in Fig. 4.1. Additional semantic mappings are introduced in case of *adding semantics*, implicating a larger space of application of a particular syntactical construct. In case of *specifying semantics*, domain concepts are specified and explicitly referred to by introduced semantic mappings. This means that a particular concept is understood in a more precise, perhaps domain-specific meaning. This could be the case within specific and delimited stakeholder groups having common domain understanding of rather generically specified syntactical constructs.

Semantic Modification

Semantic modifications cover the removal of semantic mappings implicating the limitation of the intended meaning of syntactical constructs. In case of *redefining semantics*, original semantic mappings are removed and finally replaced by new mappings, leading to a modification of the original meaning. In case of *removing semantics*, original semantic mappings are removed without the assignment of new ones. This implicates a loss of precision in terms of interpreting the syntax of an original language.

4.4 Summary and Conclusion

This Section provides a consideration and definition of key terms within the context of EML extensions and characterises different types of language modification.

Further, the *Standard+X* approach was elaborated and discussed in detail. This Section hence acts as conceptual base for the subsequent topics, which consider particular syntactical extension techniques or the semantic justification of extensions, for instance.

5

Extension Types of Enterprise Modelling Languages

The stated language extension types rationally cause the deeper consideration of the underlying EML type in order to cope with potential consequences for EML extension design. This seems to be necessary, since EML differ in regard of their expressiveness, level of domain-dependence, and their particular application area. It is therefore sensible to consider the topic of EML extensions from a rather language-oriented point of view in order to differentiate types of EML extensions for different EMLs. Surprisingly, a dedicated analysis and classification of EMLs has rarely been addressed in literature so far. Atkinson & Kühne [50] present a framework for the differentiation based on domain-specificity and abstraction (in the sense of problem-oriented and solution-oriented). Frank [123] divides domain-specific languages in regard of their enterprise-specific implementation, which motivates a multilevel based approach (referred as Reference DSMLs and Local DSMLs). In respect of apposite extension mechanisms, we therefore consolidate existing works and propose a useful EML classification schema with respective differentiation criteria in order to facilitate the appropriate selection of extension mechanisms.

5.1 Criteria for EML Classification

The characterisation of EMLs needs to be conducted by a certain set of attributes in order to estimate consequences for extension design. We therefore proclaim three attributes: *Domain-Specificity*, *Formalisation*, and *Focus*.

5.1.1 Formalisation

In this work, formalisation is understood as the type of semantic concepts in the area of tension between problem space and the solution space that is finally derived or designed for managing and solving particular tasks.

The *problem space* regards to parts of the real world or area of discourse and aims to represent (or reconstruct [41]) the "what" perspective on a particular problem. Abstraction in the problem space actually refers to the basal kind of modelling, encompassing the selection of omitted and abundant attributes (cf. [124, 41]). Finding the right level of abstraction is a challenging task and a crucial aspect for the differentiation between GPMLs and DSMLs.

The *solution space* covers concepts which are explicitly designed or derived for solving a particular task. Dependent on the type of tasks, the kind of respectively

designed models could differentiate between rather real-world oriented conceptual models (e.g. process models for process re-engineering), rather technical-oriented conceptual models (e.g. intermediate models in MDA approaches), or derived software code. This could further imply a certain difference between material semantics (human beings are model readers), formal semantics (models should be invariantly interpreted by technical actors), or respective hybrid implementations (cf. [67]). The solution space hence reflects the "how" perspective on enterprises and implies differing levels of formalisation from the business domain to the technical system (cf. [5, 50, 125]).

5.1.2 Focus

Analysis and management of enterprise systems is typically organised within Enterprise Architecture Frameworks (EAF), which are composed of several layers and perspectives [4, 5, 125]. While the above stated abstraction attribute addresses the tension between problem space and solution space, the focus attribute refers to those parts and aspects that are primarily covered by an EML. We therefore refer to the framework of Frank [5] that proposes a matrix consisting of perspectives (abscissa) and aspects (ordinate).

Perspectives represent specific professional backgrounds that correspond to cognitive dispositions, technical languages, specific intentions, objectives, or capabilities [126, pp. 10ff]. The perspective concept corresponds to the concept of abstraction, if perspectives are understood in a strictly ascending manner, which implicates that a lower perspective has a higher formalisation degree; cf. the perspectives *Technology*, *Application*, or *Business* in ArchiMate [10] or similar perspectives in Frank [5] and Adam & Esswein [127], as well as the summary in Weller & Esswein [125].

Perspectives typically encompass multiple aspects. Aspects can be seen as specific views on enterprises that work like filters on a set of concepts in order to reduce complexity [47]. With respect to the meaning of aspects in Aspect-Oriented Programming [128], the term "views" is used in the following discourse for reasons of clarity. The point of intersection between (probably multiple) perspectives and (probably multiple) aspects is referred to as focus and represents a selective view on enterprises [12, p. 948]. In regard to the characterisation of EMLs, a focus may also span multiple perspectives if necessary. The criterion should emphasise the specific intention of an EML and stresses its differentiation in comparison to other languages.

For instance, BPMN focusses on the process view of enterprises and enables modelling both the IS perspective and the application system perspective of enterprises [125, 129]. RiskM enables modelling the risk view on multiple perspectives [96, p. 601]. The modelling languages KAOS and i* cover the requirements view with respective foci between early and late requirements [130, 131, 132]. Other languages provide multiple foci of particular industries, e.g. the healthcare sector (CPmod [43]).

5.1.3 Domain-Specificity

This criterion represents the kind of specificity in regard of a particular area of discourse or industry, which is more broadly referred as domain. Domain-Specific

Modelling Languages (DSMLs) refer to specific domains and support their concise modelling by using common domain terminology and notation that is reconstructed from the particular fields of application [50, 133, 134, 135, 136, 25].

DSMLs usually serve as small languages with a limited set of concepts, precise integrity constraints, and a concrete application focus [137, 138, 60]. Consequently, the way of using and interpreting constructs is rather standardised, as DSMLs base on invariant semantics implicating a certain level of consensus within the possibly limited number of users (cf. [25]). In regard of the dichotomy between highly specialised DSMLs (e.g. for specific businesses) and rather generic DSMLs (e.g. for entire industries), Frank [123] proposes a multilevel based approach for the derivation of different DSML types based on a specific reference DSML on top that enables a flexible language architecture and renounces the rigid four-level architecture of MOF.

The opposite of domain-specificity is generality (e.g. [69]). General Purpose Modelling Languages (GPMLs) provide generic, domain-independent and generally usable concepts that can be applied across different domains, industries, and businesses.

5.2 Framework Architecture

Unfortunately, current research lacks in the provision of EML classification frameworks and a precise positioning of EMLs is hence difficult. We therefore propose a framework based on the presented attributes. The classification framework was designed in alignment with the Dresden Architecture Framework (DAF) and existing EAFs (cf. [5, 127, 125, 10]). Basically, the DAF – as a consolidation of existing EAFs with a special focus on language integration and extensibility – was adapted as follows.

The levels of system design (e.g. application level) were adapted to represent the formalisation level of an EML. It is assumed that each EML has a notable but differing level of semi-formality (cf. discussions on semantics [14, 139, 100, 8, 67]).

The view level was reused in order to represent particular views on enterprises. The enterprise-specific levels were omitted and replaced by the domain-specificity levels as depicted in Fig. 5.1. All characteristics are respectively represented by one dimension enabling an integrated specification of EMLs. The intersection points and areas shape the above-introduced foci and represent the scope of an EML.

For instance, BPMN is a domain-independent, generic EML for process modelling. BPMN both supports the creation of less formalised models (e.g. for pure documentation) and semi-formal models (e.g. partly automatable workflows). As depicted on the left side of Fig. 5.1, BPMN can hence be located on the *Generic* level (dimension *Domain-Specificity*), within the *Process* view, and affects the *Business Level* and the *Application Level* of the *Formalisation* dimension.

5.3 Types of Enterprise Modelling Languages

The presented framework facilitates the location of existing EMLs as well as the more fine-grained characterisation and differentiation between them. Basically, the

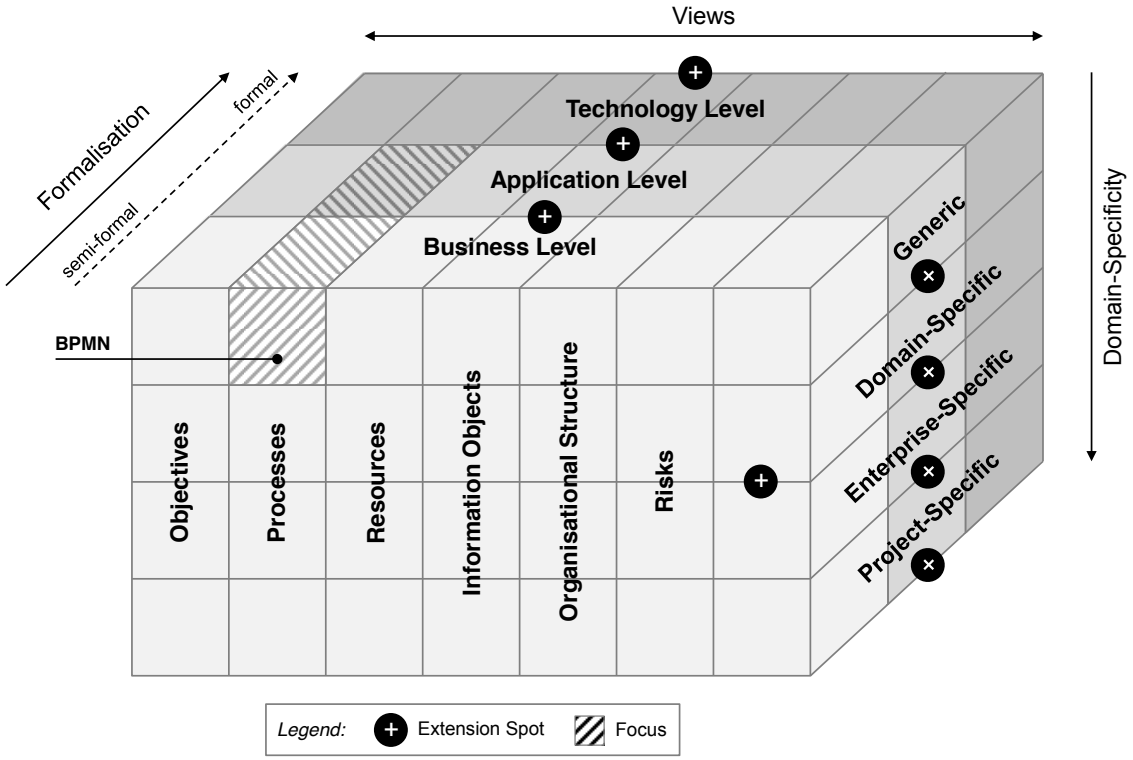

Fig. 5.1. Extension framework with a localisation of the BPMN

following three general EML types can be identified: General Enterprise Modelling Languages (GEML), Purpose-Specific Modelling Languages (PSML), and Domain-Specific Modelling Languages (DSML). Thereby, the rather formal *Technology Level* of the Formalisation dimension is intentionally included, since EMLs may also cover formal aspects as outlined in Sect. 4.1. As expected, most EML address rather non-technical aspects [6].

GEMLs support the integrated, multi-perspective analysis of entire enterprise systems and provide rather abstract and generic enterprise concepts. Examples for GEMLs are ArchiMate [10], ARIS [18], or MEMO, although MEMO can be rather seen as a design framework for an integrated derivation of languages [60, 12]. GEMLs are under-specified and ontologically incomplete on purpose, which may indicate a specification of concepts according to a particular domain, modelling task, or perspective (according to [25, p. 136]). For instance, ArchiMate was intended to remain as simple and generic as possible [10], and such languages are often intentionally designed to express an enterprise at a high level of abstraction [35, 7]. GEMLs differ from General Purpose Modelling Languages (GPMLs) like UML in regard of the general focus on enterprise-related concepts that can be found in different views of the presented classification framework. From the perspective of GPMLs, GEMLs can be classified as domain-specific languages covering the domain of enterprises. GEMLs are usually problem-oriented and semi-formal, but the derivation or existence of well-formalised, solution-oriented concepts is not impossible. GEMLs further cover multiple views of enterprises in an integrated manner.

PSMLs can be seen as specialisations of GEMLs that are limited to a certain view on enterprises. The prefix "purpose" highlights this view in regard to its intended ap-

plication in a rather general sense, as views are also concepts within EML definition (cf. [47]). Concepts of PSMLs remain generic and independent of any domain and can also cover different abstraction levels. BPMN serves as a prominent example for a process-specific PSML [15, 16]. RiskM is an example for a risk-specific PSML and languages likes i* or KAOS represent requirements-specific PSMLs [130, 131, 96].

DSMLs provide specific concepts of a domain that is aligned with particular domain terminology, implicating a lower level of abstraction [25]. The degree of view coverage may differ according to the intended application of a DSML. Some DSMLs cover multiple views of enterprises (e.g. CPmod [43]), while other DSMLs only focus on one or two aspects. In contrast to, for instance, Frank [126, 25], we understood domains explicitly in the sense of concrete industries (e.g. healthcare or manufacturing). DSMLs in the understanding of Frank [126, 25] – e.g. ResML [140], RiskML [96], or ControlML [141] – are rather seen as PSMLs due to their independence of specific industries. We therefore aim to correspond to the peculiarities of EM in regard to their practical application within concrete industries and daily businesses.

	Dimensions				
Type	**Abstraction**	**Views**	**Domain-Specificity**	**Description**	**Examples**
GPML	Various	Unspecified	Generic	Highly generic meta concepts for conceptual modelling.	UML [106, 107]
GEML	Business Level, Application Level	Multiple	Generic	Fundamental meta concepts for the domain of EM, e.g. Actor, Event, Goal or Process (cf. [101]).	ArchiMate [10], ARIS [18], MEMO [60]
PSML	Business Level, Application Level	Selective	Generic	Fundamental meta concepts for a particular aspect of EM or a specific view within EM (e.g. Task, Decision, Event and Participant within process modelling [16]).	Petri Nets [142], BPMN [15], EPC [19], RiskML [96], ResML [140], GoalML [143], ITML [144]
DSML	Various	Various	Domain	Meta concepts, which are typical for multiple aspects of a particular domain or industry (e.g. healthcare or manufacturing) and their valid application is limited to that domain (cf. de facto invariant semantics [25]).	CPmod [43]
Local DSML	Various	Various	Enterprise, Project	Refined or specified concepts of a DSML for solving issues of a specific enterprise or project, which leads to a limited scope of a DSML [123].	cf. [123]

Table 5.1. Language types

Table 5.1 summarises the role of the introduced EML types against the background of the classification dimensions and differentiates them from further types of conceptual modelling languages. The outlined classification framework serves as a base for the presentation of respective consequences in regard to EML extension types below, which determines the final syntactical implementation.

5.4 Consequences for EML Extension Types

The proposed EML classification framework facilitates the guided derivation of EML extensions. As mentioned above, an EML can be located on a *cohesive* three-dimensional corpus within the framework, which features its language characteristics. Expanding this corpus over adjacent framework elements manifests an extension, which leads to a larger set of covered concepts within the language. Such an expansion can also be caused by a refinement of single elements, although each expansion is only possible at the outlined extension points. The view dimension can be expanded infinitely. In contrast, the abstraction dimension and the domain-specificity dimension can only be extended within the respective outer limits. Below, extension consequences for each dimension are briefly considered.

5.4.1 Formalisation

Extending the formalisation dimension refers to the attachment of additional concepts with a different level of semantic variance between formal and informal semantics (e.g. [139]). This aspect explicitly refers to the difference between human model readers and technical model interpreters (in contrast to semantic variance of domain-specificity). Typically, a semi-formal modelling language like BPMN is extended with concepts for formal model interpretation. This reduction of interpretation variance is often the result of three main causes: *Behavioural formal semantics*, enhanced *static formal semantics*, and *model analysis* [67].

Behavioural formal semantics covers the automatic interpretation of models in the sense of invariantly defined model state transitions, which enables workflow execution, for instance [63, p. 25], [145, p. 489], [146, p. 435], [8, p. 3403], [15, p. 435]. Formal semantics enable prescriptive modelling, while material semantics rather support descriptive modelling [14].

Static formal semantics refers to valid and consistent model states by introducing additional constraints, for instance [67, p. 417]. Static formal semantics can be seen as intensification of particularly defined syntactical rules [40, p. 439]. While behavioural formal semantics cover transitions, static formal semantics focus on a particular model state (e.g. [145]). Both types have in common that they are part of the conceptual scope of language.

In contrast, the class of **model operations** refers to situations where an EML gets extended for the purpose of analysing models or facilitating some kind of work on these models [67, p. 417]. Extensions for such tasks implicate a separation between the actual conceptual scope of a language (e.g. process-related concepts) and respective analytical and hence rather technical capabilities (e.g. for transformations or calculations). Mixing both concept types is critical in regard to separation of concern and tool independence (cf. the discussion on MOF shortcomings in [60]). Especially MOF-based languages suffer from a noticeable mixture of both types (e.g. ItemDefinition and Import in BPMN [15]). It is hence advisable to avoid the amalgamation of constructs and rather introduce an appropriate analytical layer as an additional perspective in order to ensure loose coupling.

All in all, abstraction extensions primarily affect the semantics of EML and imply respectively required syntactical extensions in order to support and enable these extensions. It is hence extremely important to consider additional semantic mappings

and semantic constructs and their consequences to the original EML meta model at first (e.g. [73]). Although some authors propose the investigation of hybrid forms between semi-formal and formal semantics [67], it is further important to ensure a unified and homogenous semantic level. Semantic differences should be treated appropriately. This is closely related to the general necessity of more sophisticated and multi-faceted investigation of semantics in EML [57].

5.4.2 Views

Extending the view dimension refers to an extension of the conceptual, non-technical scope of the EML. The *vocabulary* of a language is expanded in order to enable the formulation of a broader set of sentences, i.e. conceptual models. The meta model is hence supplemented with additional concepts [147]. The semantic level of any extension is thereby equal or at least similar. Two types of view extensions are differentiated according to Atkinson et al. [26]: *enhancement* and *augmentation*.

Enhancement covers the introduction of additional concepts from the same area of discourse [26]. A process modelling language is extended with additional process-related concepts, for instance. Enhancement implicates refinements and expansions of a particular view within the presented framework.

Augmentation covers the introduction of additional concepts from another area of discourse that is related with a particular view over integrating concepts (referring to [26]). For instance, a process modelling language is extended with concepts from risk modelling. The integration and combination of different views could also lead to a composition-like extensions (cf. [24]). However, it is important to find appropriate mechanisms for meta model component integration, homogenous semantic levels, and appropriate means of handling the potentially increasing complexity of the vocabulary. Consequently, augmentation refers to the combination of different views.

5.4.3 Domain-Specificity

On the one side, generic EM concepts are often reused across domains and businesses (cf. [101, 68]). On the other side, these rather generic concepts need to be specified, adapted, or extended due to situational or domain-specific requirements and needs [56]. Extending the domain-specificity dimension hence causes the specialisation of rather generic language constructs in regard to particularities of industries, enterprises, departments, projects, or even situations (cf. [40, 12]). This includes the integration of respective domain terminology and the *reduction of semantic variance* between prospective *language users*. An increased domain-specificity may solely involve the specification of semantics [73] of under-specified concepts (e.g. Pools and Lanes in BPMN).

In contrast to the abstraction dimension, extensions in the domain-specificity dimension do not address the degree of formalisation, but the level of common understanding of language concepts and the range of the language. Thereby, the integration of specific rules and constraints within the language meta model reduces the degree of modelling freedom on the one side, but aims to enhance modelling accuracy on the other side. The understanding of "semantic variance" hence differs between the stated dimensions. Consequently, specialisation techniques are required on both

syntactical and semantic levels. It is further advisable to investigate the parallelism between highly generic concepts of the host language and respectively fine-grained and specific concepts for a particular context on the other side (cf. [147, 148, 123]). So far, generic languages like UML or ArchiMate provide basic specialisation means (cf. [31, 28]). Some authors further proclaim the need for specialising DSMLs in order to respond to context- or situation-specific requirements and support DSML evolution [135, p. 15], [25, pp. 136, 141], [69, p. 293].

5.5 Language Extension Types

Various terms and labels exist in the research community as outlined in Sect. 1.2.2. It is therefore promising to find more general notations for EML extensions types, which particularly consist of multiple extension techniques as stated in Part III. The consideration of modelling languages generally provokes the consideration of language adaptation in linguistics. Two types should be reflected: *accents* and *dialects*. Both types are briefly discussed below. In combination with the introduced extension framework, they should support clear and precise communication about (especially complex) EML extensions in order to emphasise their intention and purpose.

5.5.1 Accents – Semantic Extensions

Characteristics

Basically, an accent is defined as characteristic *pronunciation* of a particular group of people relative to the group of people speaking a particular language [149]. Speaking with an accent means that the way of speaking a language (usually the native language) is mistakenly transferred to a language. While the grammar and the vocabularies of a language are accurately used, the manner of *expressing* it differs, which could lead to misunderstandings within conversations with non-accent speaking people. An accent often relates to a *particular group of people* having something characteristic in common, e.g. their geographic origin or a particular social class [150].

Adaptation

The accurate use of grammar and vocabulary can be adapted to the correct use of the EML syntax. Consequently, the syntax is not extended. However, semantics of an EML are extended within a particular group of EML users. An EML accent therefore refers to all those EML extensions which solely consist of *semantic extensions*. An EML dialect can be seen as a specific way an EML is spoken and understood in a particular domain or within a specific group of EML stakeholders. Semantically underspecified constructs are therefore specified according to the peculiarities of domains (cf. [73]). Also the formalisations of concepts with variant semantics could establish an accent. Therefore, accents refer to the Formalisation dimension and the Domain-Specificity dimension of the proposed framework (cf. Sect. 5.2). Accents are interesting from a technical point of view as they avoid any kind of tool modifications and just address the meaning of syntactical constructs.

5.5.2 Dialects – Syntactic and Semantic Extensions

Characteristics

A dialect is defined as "a particular form of a language, which is peculiar to a *specific region or social group*" [149]. Dialects can be seen as language varieties, differing in terms of syntax, lexis, or phonology [151], i.e. the vocabulary that is provided by the language as well as its explication. In the field of Software Engineering, a dialect is understood as a modified version of a programming language [152].

Adaptation

Due to the larger extent of variation, an EML dialect is hence understood as an extension of the original *syntax* of an EML or as the extension of both *syntax and semantics*. EML dialects are therefore characterised by a higher level of complexity in regard to the number of used and specified concepts. However, the introduction of additional syntactical constructs is characteristic for EML dialects. Consequently, dialects could be created by extensions in each framework dimension (cf. Sect. 5.2). With respect to the framework, three major dialect types can be identified: Annotations, Specialisations, and Hybrids.

Annotation Dialects primarily refer to extensions of the view dimension indicating view-across extensions (e.g. in case of enhancement). Compositionally created EML extensions (referring to [24]) are special annotation dialect implementations.

Specialisation Dialects cover primarily syntactical specialisations of rather underspecified modelling languages. This type is typically conducted for a specialisation of GEMLs or PSMLs (e.g. for the healthcare domain [27] or industrial engineering [153]). Specialisations introduce additional syntactical constructs in order to enable a lower level of abstraction that is more appropriate to a particular domain or more adequate for a group of stakeholders. This means that existing meta model classes are specified by generalisations and the underlying concept thus become more concrete and specific, while the general vocabulary and constrains remain unaffected.

Hybrid Dialects cover both above introduced dialect types and combine them to complex EML extensions.

5.6 Consolidation and Conclusion

The above introduced language extension framework and the extension types should finally be consolidated and summarised. The consolidation should facilitate two tasks: It should support the selection of the most appropriate EML extension type based on semantic and pragmatic considerations (cf. Part IV), and it should further support the selection of the most adequate syntax extension mechanism (cf. Part III).

Table 5.2 consolidates the previous Sections and summarises relevant dimensions, extension types, and the resulting language extension type that is justified by the particular *extension corpus*. The extension corpus is created by the set of extended framework dimensions and can be seen as those framework elements that are affected by an extension.

Dimension	**Abstraction**	**View**	**Domain-Specificity**
Extension Type	*Behavioural-formal Semantics*: • Execution *Static-formal Semantics*: • Validation, Conditions *Model Operations*: • Analysis, Transformation	*Enhancement*: • Refinement, same area of discourse *Augmentation*: • Additional area of discourse	Higher degree of *specialisation* or lower *level of invariance*
Language Ext. Type	Accent (semantics only) Specialisation Dialect	Annotation Dialect	Accent (semantics only) Specialisation Dialect
	Hybrid Dialect		

Table 5.2. Language extension types

Part III

Extension Mechanisms

6

Relevant Publications

6.1 Publication WIT-2014

General Information:

Title:	Extending BPMN for Modeling Resource Aspects in the Domain of Machine Tools
Authors:	Richard Braun, Werner Esswein
Year:	2014
Medium:	WIT Transactions on Engineering Sciences, Vol. 87
Pages:	450-458
Reference:	[36]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions of Authors:

Richard Braun:	90% (introduction and motivation, research design, analysis, design, example, conclusion and further research)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Lack of machine-related resource concepts in BPMN
Input and Method:	BPMN extension method of Stroppi et al. [48]
Contributions:	BPMN extension: • Extension for resource modelling (less specified due to page restrictions)

Extended BPMN extension method:

- Initial domain analysis stage investigates capabilities of BPMN and justifies extension decision

Abstract:

Engineering processes that use machine tools face the possible problem of inaccuracy and a loss of product quality because of position errors of the machine tool. Current research projects address this problem and provide so-called correction and compensation methods to reduce position errors in machine tools and thus enhance the product quality. However, since these optimization processes are very innovative and sophisticated, their description as well as their integration into engineering processes is not trivial. This research article aims to address this issue by a conceptual modelling approach. The popular Business Process Management Notation (BPMN) is adapted and extended by domain specific concepts to represent resource intensive engineering and optimization processes. The BPMN extension is evolved systematically on the base of the BPMN specification and previous research in the field of BPMN extensibility. Further, a literature review was conducted to identify relevant resource concepts for the domain of machine tools.

6.2 Publication MKWI-2014

General Information:

Title:	Entwicklung einer BPMN-Extension für ressourcenintensive Prozesse im Maschinenbau
Authors:	Richard Braun, Werner Esswein
Year:	2014
Medium:	Kundisch, D., Suhl, L., Beckmann, L. (Eds.): Tagungsband Multikonferenz Wirtschaftsinformatik 2014
Pages:	1574-1586
Reference:	[74]

Rankings:

VHB-JOURQUAL-3:	D
WKWI-2008:	C

Contributions of Authors:

Richard Braun:	90% (introduction and motivation, problem domain, research approach, analysis, design, example, conclusion and outlook)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Lack of machine-related resource concepts in BPMN
Input and Method:	BPMN extension method of Stroppi et al. [48]
Contributions:	BPMN extension: • Extension for resource modelling Extended BPMN extension method: • Domain analysis is operationalised by domain ontology (OWL Lite) • Correspondence types are textually introduced • Reuse of existing modelling languages and extensions

Abstract:

In Grundlagenforschungsprojekten im Bereich des Maschinenbaus werden Verfahren erforscht, die thermisch bedingte Produktionsfehler an Maschinen vermindern sollen. Zur Modellierung dieser Verfahren kann die Prozessmodellierungssprache BPMN verwendet werden, welche aufgrund ihrer generischen Ausrichtung jedoch

Mängel in Bezug auf die spezifische Abbildung von Ressourcenobjekten aufweist. Es wurde daher auf Basis des Design-Science-Ansatzes eine BPMN-Erweiterung für ressourcen-intensive Prozesse im Maschinenbau entwickelt. Im Gegensatz zu anderen Forschungsarbeiten wird die BPMN-Erweiterung systematisch auf Basis einer Domänen-Ontologie, des BPMN-Erweiterungsmechanismus sowie des Vorgehens von Stroppi et al. (2011) konstruiert.

6.3 Publication BIBM-2014

General Information:

Title:	BPMN4CP: Design and Implementation of a BPMN Extension for Clinical Pathways
Authors:	Richard Braun, Hannes Schlieter, Martin Burwitz, Werner Esswein
Year:	2014
Medium:	Zheng, H.J., Dubitzky, W., Hu, X., Hao, J., Berrar, D.P., Cho, K., Wang, Y., Gilbert, D.R. (Eds.): 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)
Pages:	9-16
Reference:	[27]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions of Authors:

Richard Braun:	50% (method, domain analysis, extension design, further research)
Hannes Schlieter:	25% (introduction and motivation, contributions)
Martin Burwitz:	15% (demonstration, tool implementation)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Modelling Clinical Pathways with BPMN
Input and Method:	Extended BPMN extension method, including the BPMN extension method of Stroppi et al. [48]
Contributions:	BPMN extension: • BPMN4CP v1.0 Enhanced BPMN extension method: • Detailed equivalence check based on requirements and derived language concepts • Explicit specification of the concrete syntax

Abstract:

The Business Process Model and Notation (BPMN) is a standard for business process modeling that is very common in professional practice due to its expressiveness, the well defined meta model and the possibility of workflow integration. This research article aims to apply the BPMN for the representation of clinical pathways in order to utilize its benefits in the clinical context. BPMN provides a set of generic process modeling elements what makes it necessary to extend the language by domain-specific concepts from the field of clinical pathways (e.g., evidence indicators). Therefore, the extension method of Stroppi et al. (2011) was applied and extended in order to facilitate a systematic design and development. This research article provides the analysis of requirements and relevant concepts for modeling clinical pathways. Based on a domain ontology, need for extension is identified and the valid BPMN extension meta model is designed by the construction of a conceptional domain model and the corresponding BPMN extension model. The evolved extension BPMN4CP is demonstrated by an example process of wisdom tooth treatment.

6.4 Publication WI-2015

General Information:

Title:	Extending a Business Process Modeling Language for Domain-Specific Adaptation in Healthcare
Authors:	Richard Braun, Hannes Schlieter, Martin Burwitz, Werner Esswein
Year:	2015
Medium:	Thomas, O., Teuteberg, F. (Eds.): Smart Enterprise Engineering: 12. Internationale Tagung Wirtschaftsinformatik
Pages:	468-481
Reference:	[75]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	A

Contributions:

Richard Braun:	60% (BPMN extensibility, design of the extension, discussion, further research)
Hannes Schlieter:	20% (introduction and motivation)
Martin Burwitz:	10% (demonstration)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN, EMLs
Motivation:	Modelling CPs with BPMN and support for decision about selecting a DSML-based approach or an extension-based approach
Input and Method:	Extended BPMN extension method, including the BPMN extension method of Stroppi et al. [48]
Contributions:	BPMN extension: • BPMN4CP v1.0 Methodical implications: • Comparison and criteria for decision support in regard to DSML or extension

Abstract:

It is often required to provide a modeling language that enables the representation of domain-specific problems and concepts. Domain-specific modeling approaches can be applied for that. However, these approaches usually suffer from low dissemination, missing tool support and high design costs. Thus, it might be more reasonable to adapt and extend common standard modeling languages. This research article presents an extension of the common process modeling language BPMN for modeling clinical pathways in the healthcare sector. The extension is designed methodically by application of the extension design method of Stroppi et al. (2011), which was extended regarding to a deeper domain analysis. The domain analysis considers the design of a domain ontology, requirements analysis as well as an equivalence check between domain concept and BPMN concepts. Finally, the evolved extension is compared with the CPmod modeling language of Burwitz et al. (2013) in order to discuss strengths and limitations.

6.5 Publication CCIS-2015

General Information:

Title:	Meta Model Extensibility of BPMN: Current Limitations and Proposed Improvements
Authors:	Richard Braun
Year:	2015
Medium:	Hammoudi, S., Pires, L.F., Desfray, P., Filipe, J. (Eds.): Model-Driven Engineering and Software Development - Third International Conference, MODELSWARD 2015, Revised Selected Papers
Series:	Communications in Computer and Information Science, Vol. 580
Pages:	230-247
Reference:	[59]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	BPMN
Motivation:	Critical analysis of BPMN extension mechanism and provision of design alternatives
Input and Method:	Adaptation of extension mechanisms from UML (Profiling) and Software Engineering (Hooking, Plugins, Add-Ons)
Contributions:	Specification of extension mechanisms for BPMN: • Profiling • Hooking • Plugins • Add-Ons Meta meta modelling concepts for generic extension design in MOF on M3 level, e.g.: • Extension interface • Extension points • Hooking generalisations

Abstract:

The Business Process Model and Notation (BPMN) is the prevalent conceptual modeling language for business process modeling and process analysis. BPMN benefits from its expressiveness and the well-defined meta model, which is defined by the Meta Object Facility (MOF). The emergence of BPMN entails an increasing demand for language extensions in order to both benefit from the dissemination and apposite concepts. Although BPMN is one of very few languages that explicitly provides capabilities for its extension, the proposed mechanism reveals some shortcomings and inaccuracies concerning model abstractions, specificity and semantical clarity. A list of improvable aspects is hence provided based on an in-depth analysis of the extension mechanism. The analysis has a special focus on the abstract syntax (BPMN meta model). Several techniques for enhanced BPMN extension design are proclaimed by adapting alternative mechanisms for language extensibility: Profiling, under specification (hooking) and annotation (plug-ins and add-ons). The stated mechanisms are partly adapted from other modeling languages (profiling) or the field of Software Engineering (hooking, plug-ins, add-ons). Each approach is described by its core concepts, its application and by some examples. The approaches are finally compared regarding several criteria.

6.6 Publication MEDI-2015

General Information:

Title:	Towards an Integrated Method for the Extension of MOF-Based Modeling Languages
Authors:	Richard Braun, Werner Esswein
Year:	2015
Medium:	Bellatreche, L., Manolopoulos, Y. (Eds.): Model and Data Engineering, 5th International Conference MEDI
Series:	Lecture Notes in Computer Science, Vol. 9344
Pages:	103-115
Reference:	[45]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions:

Richard Braun:	90% (introduction and motivation, fundamentals, related work, requirements, MOF extension capabilities, construction of the extension method, conclusion and further research)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	EMLs
Motivation:	Extensibility of MOF-based languages and limited extension capabilities on M3
Input and Method:	Adaptation of Profiling from UML
Contributions:	Profile-based extension method for MOF-based languages: • Abstract syntax (MOF-based Profiles) • Concrete syntax (DD) • Integration (MOF-to-DD mapping) Generic infrastructure for MOF-based extensions on M3: • Infrastructure on M3 • DD package on M3 • Introduced integration package on M3

Abstract:

During the last years, various MOF-based modeling languages became de-facto standards in their field of application. Due to their common application and dissemination the need for extending these languages also increased in order to integrate domain-specific concepts or facilitate interoperability and tool support. However, only the minority of MOF-based modeling languages provides an extension mechanism and even those defining one, reveal some syntactical issues (e.g., BPMN). Also MOF itself does not provide an integrated and consistent extension mechanism. We therefore proclaim the application of the UML-based profile mechanism for extending the abstract syntax of MOF-based languages while keeping their original meta models unaffected. Further, the application of the Diagram Definition (DD) standard for extending the concrete syntax is outlined and both aspects are integrated. The research article proposes a generic extension method for MOF-based languages based on existing concepts and constructs from the MOF environment. In this context, the article also discusses the positions of the Profiles package and the Diagram Graphics (DG) package within the OMG meta hierarchy.

6.7 Publication CBI-2015

General Information:

Title:	BPMN Extension Profiles – Adapting the Profile Mechanism for Integrated BPMN Extensibility
Authors:	Richard Braun
Year:	2015
Medium:	17th IEEE Conference on Business Informatics - Volume 1
Pages:	133-142
Reference:	[42]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	B

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	BPMN
Motivation:	Structured and well-defined extension method for BPMN in order to compensate current shortcomings
Input and Method:	Demonstration of the MEDI-2015 contribution by instantiation to the BPMN
Contributions:	Profile-based extension method for BPMN: • Implementation of the generic method in BPMN • Transformation rules for translating a conceptual domain model into a BPMN extension profile Implications for MOF-based EML definitions: • Well-defined specification of the concrete syntax by applying DD (e.g. BPMN DG)

Abstract:

BPMN is one of the most prevalent modeling languages within the field of enterprise modeling and constitutes as de facto standard for business process modeling. BPMN provides explicit capabilities for extending the vocabulary of the language. While a range of BPMN extensions evolved during the last years, only very few of them conduct the extension mechanism. Instead, the vast majority of extensions

is designed in an ad-hoc manner and lacks in conformity to the BPMN standard, what hampers interoperability, extension integration and model reuse. We suppose, that some architectural shortcomings and barriers of the extension mechanism itself provoke missing application. For instance, BPMN reveals abstraction conflicts and inaccuracies within the extension design. The extension mechanism also lacks in regard of exactly specifying those elements that are extended. Further, the exchange of both the concrete syntax and serialized extended model data remains vague. We therefore present an adaptation of the well-established profile mechanism from the UML Infrastructure in order to facilitate a straightforward definition of the abstract syntax for BPMN extensions. Additionally, the definition of the concrete syntax is supported by an instantiation of OMG's Diagram Definition (DD) specification in order to provide an integrated extension definition and exchange specification. The migration of existing, standard-conform BPMN extensions is outlined by a transformation process.

7

Introduction

7.1 State of Affairs

EML extensions can have different reasons, as outlined in Sect. 5. The semantically driven need for language extension hence requires the definition and application of extension mechanisms in order to implement the required adaptations appropriately. Currently, there is no commonly used set of well-specified extension mechanisms. There is a general lack of syntax repositories while dedicated research on extensible syntax architectures in the context of EM is missing [28]. Nevertheless, a few works can be found in the general area of conceptual modelling.

Atkinson & Kühne [50] motivate the issue of language customisation in general and provide an introduction and discussion of different mechanisms, namely Stereotyping, Customisation, Domain-Specific Modelling, and Multilevel Modelling. The authors focus rather on engineering-oriented modelling in the context of UML and do not cover the peculiarities of EM. Further, invasive techniques like meta-model customisation or the definition of DSMLs are included, which is not in the scope of this work due to the criteria of non-invasiveness (cf. Sect. 4.3.1).

Pardillo [31] provides a statistical analysis of conducted UML extensions and found that especially Classes, Properties, and Associations are extended by profiles. Due to the high level of domain-independence in UML and the intentionally underspecified character of Classes in UML, most profile-based extensions actually serve as de facto DSMLs (cf. [147, 154]).

Langer et al. [155] provide an approach for the generic application of Profiling to modelling languages in the context of MOF-based languages. The authors therefore introduce EMF Profiles as lightweight extension means for MOF-based modelling languages. This work strongly focusses on the EMF modelling environment [155, p. 4] and requires therefore a dedicated multilevel architecture within EMF, since the profile technique is not located on the meta meta model level (cf. [42, 156]). Moreover, the approach is rather technical-oriented and does not cover the specific features of the EM domain.

The same issue can be observed in the field of mostly textual domain-specific languages (e.g. [157]), which provide sophisticated mechanisms for extending the abstract syntax, but no appropriate means for perspectives, diagrams, or semantics.

The discussed papers solely focus on syntactical analysis and omit a deeper analysis of the underlying context and modelling purpose, which are significant topics for EMLs.

7.2 Classification and Specification

The presented state of affairs leads to the unrewarding situation that EML extensions are either implemented in an ad hoc manner or within rather simple and hence semantically limited extension mechanisms [59, 28]. Implementations efforts are therefore expensive and the level of reuse is rather low. With respect to the overall aim of providing methodical support for extension design, it is hence extremely important to provide (1) a repository of applicable extension mechanisms and (2) support the integration between extension needs and those mechanisms in order to facilitate straightforward extension design.

The intention behind the syntax repository is the guided selection of mechanisms within a potential extension situation. It is therefore advisable to elaborate different criteria in order to characterise mechanisms, differentiate them, and estimate respective technical consequences within the (meta) meta modelling architecture. Below, the set of elaborated criteria for this purpose is presented. Single criteria are thereby adapted from the field of Software Engineering [158, 159].

User Perspective (Semantics and Pragmatics – Outside View):

- *General Scope*: Characteristic idea and scope of the mechanism; consideration of prevailing authors or meta modelling environments.
- *Dissemination and Occurrence*: Assessment of current dissemination of the extension technique indicating a certain level of relevance and prominence in *existing* modelling languages.
- *Pragmatics*: Useful application purposes and typical application scenarios.
- *Appropriate Extension Type*: For which extension types and language extension types could the mechanism be applied and used?
- *Main Benefits*: Summary of main benefits in comparison to alternative approaches.
- *Limitations and Problems*: Possible limitations in regard to its application and occurring problems.

Technical Perspective (Language Specification – Inside View):
Syntactical Macro View:

- *Invasiveness and Architectural Predesign*: Level of invasiveness in the sense of architectural preparations within the original meta model or in regard to the applied meta modelling language.
- *Implementation and Application*: Condensed process of extension definition and application (adapted from [158, 159]).
- *Multiple Application*: Possible or explicitly intended combination with other mechanisms and multiple applications with extensions of the same mechanisms.

Syntactical Mirco View:

- *Extension*: Type of meta modelling constructs that can be extended.
- *Modularity*: Degree of modularity in regard to a potentially isolated instantiation of the extension. Refers to the principles of high cohesion, low coupling and separation of concern [160, 26].

- *Inner Complexity and Design Freedom*: Level of design freedom and opportunity to define complex extension structures. This refers to the set of possibly usable and applicable meta modelling constructs.
- *Interface Structure*: Extension interface definition within the original meta model (if required).

Consequences for Meta Modelling and Meta Meta Modelling:

- *Redesign Consequences (if required)*: As investigated, very few EMLs provide dedicated extension mechanisms and EMLs providing one reveal different shortcomings [28, 39, 59]. It is therefore necessary to consider potential modifications of the underlying meta model in order to conduct a mechanism. Such modifications may comprise alterations on the meta meta model level. This criterion therefore derives respective consequences and changes for the BPMN modelling environment.
- *Meta Meta Modelling*: If required, particular consequences for meta modelling languages are summarised based on the elaborations within the criterion before.

The criteria catalogue is divided into a *user-driven perspective* representing the outside view on a mechanism and a *technical perspective* indicating detailed syntactical characteristics and consequences. Each mechanism is generally analysed and discussed in the Appendix. Then, concrete consequences in regard to the BPMN and the underlying meta modelling language MOF are briefly considered for reasons of practical applicability.

8

Mechanisms

Below, different extension mechanisms are presented, classified, and semantically specified in order to provide a black-box-like view on each mechanism for a purposeful application. The presented mechanisms and thus the entire repository are results of an analogy-driven adaptation from different fields like Software Engineering (e.g. Decorators [161]) or Reference Modelling [59]. Also preliminary works from implementation-oriented conceptual modelling are examined and adapted [42, 59, 161].

First, different patterns and principles from Software Engineering are adapted in order to provide meta model annotations. In particular, Decorators (Sect. 8.1.2), Plugins (Sect. 8.1.3), Aspects (Sect. 8.1.4), and Add-Ons (Sect. 8.1.5) are reflected and discussed. Since these approaches rather address the run-time level of programs, a straightforward one-to-one adaptation is less useful, as we rather focus on the definition level, i.e. meta models. Nevertheless, aspects like non-invasiveness, run-time extensibility, and separation of concern (e.g. [113, 115]) seem to be promising for conceptual adaptation to the meta model level (cf. [45, 156]).

Second, the Software Engineering principle of Hooking as well as techniques from Reference Modelling are adapted and amalgamated in terms of explicit under-specifications in meta models (Sect. 8.2).

Third, the generic adaptation of the well-known Profiling technique from UML is discussed in Sect. 8.3. Profiling, more precisely the application of Stereotypes, is a very popular lightweight mechanism from UML indicating a situational specification of GPMLs that can be transferred to other generic modelling languages [147, 31, 42, 45, 156].

Fourth, Sect. 8.4 analyses the novel Multilevel Modelling approach as a promising architecture for flexible EML extensions across different classification levels [162, 163, 123]. Due to the novelty of the multilevel paradigm itself, several fundamentals are given before.

Fifth, the structured application of the basic Generalisation/Specialisation principle for EML extensions is motivated in Sect. 8.5.

In contrast to the set of primarily syntactical mechanisms, Sect. 8.6 finally introduces semantic extension techniques which solely address semantic extensions, i.e. additional semantic mappings and additional semantic constructs.

8.1 Annotations

8.1.1 Leveraging Principles from Software Engineering

The consideration of unforeseen programme extensions is perceived as an important but non-trivial task within Software Engineering [113, 115]. An extensible programme (or software in a wider sense) is defined as a programme that can be adapted to new tasks without changing the core source code [114]. An extensible programme should provide explicit concepts and mechanisms for customisation, on the one side, but needs to ensure high cohesion within the extension and low coupling between extension and host software, on the other side [160, 164]. Extensions should not be invasive in terms of changing the original host system [115]. We therefore argue to conceptually adapt extension patterns and principles from Software Engineering for the derivation of applicable syntactic extension mechanisms. Consequently, flexible extensions should not be applied to the model layer (by analogy with program code), but to the meta model layer in order to facilitate flexible EML extensions.

In addition to the below discussed approaches, Software Engineering provides further techniques like the Observer Pattern that can be used for execution of actions of dependent instances that are triggered by state changes of other instances (cf. [165]). However, behavioural patterns like the Observer Pattern are rather useful for concrete technical implementations on code level and less useful for conceptual EML extensions as outlined in Sect. 5.2.

8.1.2 Decorators

Motivation

The decorator pattern is a structural pattern for the situational annotation and releasing of capabilities to or from components at runtime [165]. It is motivated by the problem of realising structures that have a large amount of combinatorial variants (e.g. products with multiple options). Implementing those definitions in complex and large-scale generalisation hierarchies is inflexible, partly redundant, and difficult to manage. The decorator pattern therefore aims to ensure that the change of some features at runtime does not require the change of the entire instance at run time. It rather prefers simple binding and releasing of additional features [165]. This is realised by wrapping a component with respectively defined Decorators, enabling a flexible annotatable gateway in front of the actual component. Subclasses of the Decorator can then be used for the injection of additional capabilities [165]. Decorators hence facilitate the localised, situational, and perhaps multiple extensions of components.

Adaptation and Application

Decorators are conceptually adapted in the sense of situational annotations of *context-specific* or *domain-specific attributes and features* of meta classes. In contrast to the below mentioned Aspects, these additional features are in the contextual scope of the area of discourse and can be added to multiple meta classes. For instance, Decorators can be used for the definition of particular roles, complex

classifications, or complex hierarchies. The straightforward binding and releasing of Decorators might be further useful for frequent or dynamic extensions in regard to a special need for situational adaptability.

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.1).

8.1.3 Plugins

Motivation

Plugins are functional extensions of a original system through explicitly defined interfaces [166, 167]. Plugins constitute a supplementing means for the solution of specific business problems within a rather generic environment [167]. In contrast to the most alternative extension means, Plugins can exist on their own, implicating a low level of host dependency and a rather complex inner architecture. A Plugin interface is realised by the definition of several *extension points*. Extension points in programmes can be understood as dedicated code spots, where the executing programme asks a central registration unit for respectively installed plug-ins, whose code has to be executed.

Adaptation and Application

A meta model Plugin is understood as consistent, coherent, and conceivably independent model that enhances the conceptual expressiveness of an EML. Plugins are therefore useful for enhancement or augmentation. In the strongest form, Plugins serve for the annotation of complex meta models, namely other EML meta models (referring to [24]). In contrast to other authors from the field of Software Engineering and also in contrast to some previous work (e.g. [59]), we explicitly exclude any mandatory Plugin extensions, i.e. associations from original meta classes to extended meta classes with a lower cardinality value of at least one or any kind of introduced generalisations (cf. Sect. 8.5).

The excessive application of Plugins might further lead to an over-complication of the original meta model. This could be avoided by the definition of respectively extensible meta model spots (extension spots, cf. [45]). It is therefore suitable to explicitly specify EML interface parts and potential Plugins must conform to these interfaces. Respective interface design could be facilitated by the consideration of techniques from the field of meta model integration, for instance [168, 169, 170].

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.2).

8.1.4 Aspects

Motivation

Aspect-oriented programming (AOP) serves as an innovative technique for the extension of software [171, 128]. AOP intends for the usage of generic functionalities which exist independently of a class structure or program hierarchy. Such functionalities are referred to as cross-cutting concerns (e.g. logging or transaction control mechanisms). These functions are required in different parts of a programme, but an integration of those analytical functions to the class structure of the business logic

would cause class overloading, complication, and violation of separation of concern. AOP therefore aims to identify and separate such functions in order to maintain them in a central spot [171, 128]. Aspects represent capabilities that are required by multiple components (cf. all technical details in [171]). While Decorators primarily support flexible representations of the addressed area of discourse, Aspects rather focus on analytical, domain-independent (cross-cutting) concerns. Their actual application focus hence differs.

Adaptation and Application

Cross-cutting concerns are adapted as *domain-unspecific model information*, which are useful for several parts of the meta model (e.g. for generic model operations). An Aspect is thus understood as a rather complex extension that contains particular extension classes, which could shape a separately existing module. Aspects can be referenced by multiple original meta classes, which allows the combination of multiple additional concepts to one meta class. On a technical level, the annotation of Aspects is minimal and requires very few pre-implementations in an EML [59].

Aspects constitute a special type of Plugin due to three main reasons. First, Aspects extend exactly one original meta element (usually classes). Second, Aspects can be applied to multiple meta classes due to their model-across, cross-cutting features (which serves as the third reason).

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.3).

8.1.5 Add-Ons

Motivation

Add-ons are similar to Plugins and Aspects, but differentiate from both types in some aspects. Add-ons are extensions with *limited capabilities* that support the very selective extension of an original. Add-ons strictly depend on the original host system and cannot be usefully applied on their own.

Adaptation and Application

Add-ons are therefore adapted for the punctual and limited extension of original meta model elements on the same level of abstraction and specifity. Add-ons are therefore primarily applied for reasons of enhancement in order to provide additional attributes or detailed information within one area of discourse. Add-ons are proposed as primarily incremental, spotty, and attribute-wise extensions of original meta classes. The definition of multiple classes spanning extension interfaces remains therefore optional.

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.4).

8.2 Hooking (Under-Specification)

8.2.1 Motivation

The concept of *Hooking* is understood as intentionally leaving parts of a programme open in order to define and specify those parts later in accordance with specific

requirements. Hooks are alternatively referred to as *extension points* [172] or *hot spots* [128], but the terminology hooking is used below for reasons of clarity and in dissociation of comparable approaches. The implementation of Hooks is usually rather optional, which indicates that software remains executable if single Hooks are not filled or specified (referring to [128]).

8.2.2 Adaptation and Application

Hooking is adapted in the sense of providing a certain level of under-specification in some meta model parts for later context-specific concretisation. This intention is generally similar to the Reference Modelling approach, especially to the *instantiation* technique and the *specification* technique [173, 174]. Instantiation is understood as defining explicit placeholders, which are expected to be specified in accordance with valid placeholder values [173]. Specification is understood as concretisation or refinement of less detailed model elements [173]. We therefore differentiate two types of hooking: *Specification and Placeholders*.

Hooking by Specification represents meta model classes that are under-specified on purpose. This might be the case in GEMLs or PSMLs. The concretisation refers to the creation of sub-types with specific attributes indicating a higher level of specifity. The architectural effort on this type of hooking is low.

Hooking by Placeholders refers to the specification of meta model parts which need to be specified on the same level of specifity. The specific structure of some meta models parts is thereby left open and requires detailing on the same level of abstraction. This hooking principle requires an architectural environment in the original meta model in order to define those parts which are changeable, namely *hot spots* and *frozen spots*. Language designers can therefore limit the degree of extensibility and adaptability of an EML. It is further important to state that the degree of under-specification can differ [45].

In case of Hooking by Specification, a concerned meta model element can still be instantiable (with the consequences of a broader semantic variance, for instance). Depending on the extent of the placeholder, Hooking usually requires a mandatory pre-specification, as the original meta models might not be suitable for instantiation.

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.5).

8.3 Profiling and Stereotypes

8.3.1 Motivation

Profiling is a lightweight extension technique for the definition of domain-specific variants of the general-purpose modelling language UML [31]. Profiles are composed of Stereotypes, which extend respectively referred host meta classes by the assignment of Tag Definitions (Attributes) and OCL-based constraints [106, pp. 14, 175]. Semantically, a stereotype-based annotation of additional characteristics should not contradict or redefine original class properties. Profiling therefore serves as a non-invasive extension mechanism that facilitates multiple assignments of additional properties for domain- or context-specific concretisation. Stereotypes heavily rely on the host meta classes and cannot exist on their own.

8.3.2 Adaptation and Application

The Profiling mechanisms can be applied for the concretisation of GEMLs and PSMLs in order to concretise these languages for particularities of specific domains [161]. The EML meta model is thereby specified and becomes more domain-specific in some parts. Profiling is established as a widely used mechanism for UML extensions as demonstrated in Pardillo [31]. Due to the generic character of UML, its Profiles can often be interpreted as de-facto DSMLs [147]. Despite the prominence of Profiling, very few authors analyse its generic application for other modelling languages. Langer et al. [155] introduce a generic profile-based approach in the EMF environment in order to facilitate operations within Model-Driven Engineering. Braun & Esswein [156] motivate a generic application of Profiling within the MOF on the meta meta model layer and outline the multi-facetted application of the underlying non-invasive annotation principle to different meta model types (e.g. views, perspectives and properties). Braun [42] demonstrates the applicability of this approach within BPMN as well as a possible integration into the BPMN syntax extension method of Stroppi et al. [48].

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.6).

8.4 Multilevel Modelling

The above presented extension mechanisms rely on the traditional four-level architecture and particular extensions are applied to the meta model level. Although respective specialisations punctually cause lower levels of abstractions and may indicate de facto intermediate levels with a certain conceptual overloading (cf. [59]), each mechanism is purposefully located on the same meta modelling level as the original meta model. However, limitation to only one classification level seems to be insufficient for EM in general and with regard to the definition of EML accents on lower classification levels in particular. Respective shortcomings of the commonly applied fixed four-level architecture are therefore stated below by exposing major concerns and outlining related solution attempts within the novel Multilevel Modelling paradigm [162, 123]. With respect to the stated novelty and the lack of literature on Multilevel Modelling in the context of EML extensions, the approach will be discussed in more detail than the already introduced mechanisms.

8.4.1 Motivation: Issues within Fixed Level Architectures

Rigid Classification Levels Often Contradict Ontological Classification Levels

Although the concept of meta modelling is generally defined in a generic manner [175], most meta modelling approaches follow a fixed four-level architecture that is composed of a meta meta modelling level (M3), a meta modelling level (M2), a model level (M1), and the level of the area of discourse or respective technical instantiations (M0). The origin of this architecture lies in the structure of the UML and the MOF [106, p. 20]. MOF is the meta modelling language of the OMG and hence also responsible for the definition of EMLs like BPMN. Nevertheless, it has

to be stated that such rigid classification architectures reveal several shortcomings in the context of EM, as the real world and especially the domain of enterprises cannot be classified in four levels per se [123, p. 322], [163, p. 743]. Instead, flexible classification levels are required in some cases due to the complexity and variety of EM-related concepts and the varying number of classification levels [176, p. 196]. Frank [123] uses the prominent examples of product types and products variants for the demonstration of the inherent ambivalence and flexibility of classification in EM [123, p. 330]. Required classification capabilities are not provided by traditional meta modelling approaches (referring to [176]), which merely focus on the relation between classes and instances. This might be expedient for programming tasks, but partly insufficient for conceptualising and modelling IS [177, p. 5]. Particularly required multiple classification sets are therefore often implemented with conceptually unsatisfying workarounds, e.g. by overloading single levels [163, p. 754] or defining de facto intermediate levels combining abstraction characteristics from two levels (cf. [39]). The reached flattening within one level of classification is conceptually questionable, error-prone, and lacks in clarity. It should therefore be replaced by flexible language architectures.

Level-Skipping Instantiation and Dichotomy between Specialisation and Instantiation

A rigid dichotomy between specialisation and instantiation is questionable in regard to the definition of flexible level architectures, e.g. for variants of DSMLs [123, p. 351]. Instead, it is promising to specify level-spanning features and properties in order to enable the instantiation of class features from high abstraction layers on lower abstraction levels [178]. This deferred instantiation of some constructs can be realised by Powertypes [179, 180, 60], Clabjects with respective Potencies [163], or Intrinsic Features, which enable delayed, layer-skipping instantiation and selective specialisation of attributes, operations and associations [60].

Mitigating the Dichotomy Between GPMLs and DSMLs

GPMLs and PSMLs are rather generic and semantically unspecific, which allows certain room for interpretation of many constructs. On the other side, these languages benefit from their high level of reusability and dissemination across domains and industries. In contrast, DSMLs naturally have a higher level of specifity in order to match particular needs adequately [25]. This determined dichotomy between both poles seems to be impractical, since the analysis of enterprises and IS both requires language concepts that are applicable and hence reusable across different domains as well as domain-specific or organisation-specific concepts [163, p. 753], [123, p. 321]. The sheer number and diversity of observed EML extensions amplify this assumption and require an investigation of appropriate domain meta classes or comparable concepts that satisfy both needs.

8.4.2 Existing Multilevel Modelling Approaches

Flexible Meta-Modelling and Execution Language (FMMLx): FMMLx stands for a multi-level architecture for the flexible and fine-graded definition of

different types of DSMLs (e.g. Reference DSMLs, Local DSMLs, Enterprise DSMLs [123]). The approach is motivated by reusing rather generic top-level languages and enables their partial specification according to organisations, domains, or projects that require semantically highly specific concepts [123, p. 336]. Therefore, the Flexible Meta-Modelling and Execution Language (FMMLx) is presented. The approach bases on the following pillars: A flexible number of classification layers, no strict separation between classification levels and a relaxation of the rigid dichotomy between classification and specialisation. As mentioned above, Intrinsic Features are proposed as essential language concepts for this purpose. In contrast to Powertypes and deep classification (cf. [163]), Intrinsic Features comprise not only attributes, but also associations and operations. It is further possible to mark an entire class as intrinsic, which means that all its features (attributes, operations, associations) are intrinsic, too. Each feature can be annotated with a particular specification level in order to support the deferred (or delayed) instantiation on lower classification levels (similar to the Potentcy of Clabjects, for instance). Hence, Intrinsic Features combine characteristics from instantiation and specification [123, p. 334]. Besides the novel conceptual proposal, FMMLx further facilitates seamless integration between models and code and supports the integrated derivation of domain-specific programming languages [123, p. 326]. FMMLx slightly adapts the Executable Meta-Modelling Facility (XMF) of Clark et al. [181] in order to implement Multilevel Modelling technically and facilitate language modifications and extensions at runtime.

Orthogonal Classification Architecture (OCA): Atkinson et al. [176] introduce an orthogonal language architecture that enables the differentiation between *linguistic classification* (vertical, between classification levels) and *ontological classification* (horizontal, within one classification level [182]). Classes therefore function both as classes and objects (referred as Clabjects with a specific Potency [176, p. 197]). In contrast to the FMMLx approach, Atkinson et al. [176] primarily address multiple classifications within the rigid four-layer architecture, aiming to provide flexible modelling tools that enable run-time alterations of the underlying language. This is motivated by cumbersome, time-consuming, and error-prone alterations in current modelling tools and the uniform and level-agnostic kind of multilevel modelling should remedy these shortcomings [176, p. 195]. However, the differentiation between linguistic and ontological classification levels is at least debatable, as each "ontological" concept requires a linguistic representation [177, p. 13]. We therefore prefer FMMLx to OCA and will outline its conceptual adaptation for the purpose of EML extensions below.

8.4.3 Core Question: Specialisation or Instantiation

The above introduced extension mechanisms markedly differ from Multilevel Modelling in regard to the affected classification levels. All presented mechanisms are more or less simple annotations of additional meta classes; either as simple specialisations, highly dependent property-like annotations indicating some kind of de-facto specialisations (Profiling), or type annotations with differing inner complexity (Aspects, Add-Ons, Plugins). The classification level (namely M2 within the MOF architecture) remains fixed. With regard to the introduced EML extension frame-

work, such fixation seems to be inappropriate for some extension purposes and more flexibility in EMLs is generally needed as discussed above.

Adapting Multilevel Modelling as extension techniques causes the precise differentiation between specialisation and instantiation in order to find an appropriate measure for creating additional classification levels. However, clear differentiation between specialisation and instantiation is often only possible in software-oriented modelling and becomes demanding in regard to meta modelling and language engineering. This is mainly caused by the fact that both types are usually referred to as "is a" relations in natural language and underlying conceptual peculiarities are difficult to describe and, hence, difficult to delineate [177, p. 1].

It is therefore essential to basically analyse the difference between classification/instantiation and specialisation/generalisation, especially in regard to the relevant area of discourse that is not limited to its representation in a (software) system with only two technical classification levels. It is hence promising to derive respective guidelines in order to methodically support the selection of respective mechanisms. Frank [177] elaborates detailed characteristics of both types and from scratch and proposes some guidelines for the differentiation between them in language engineering. The main contributions of his paper are excerpted below with respect to the prospectively designed extension method.

Aspect	**Generalisation/Specialisation**	**Classification/Instantiation**
General aspects [177, p. 3]	A is superclass, B is subclass	C is a class
	A and B are classes	D is an instance (i.e. not a class!)
	Substitutability: Each instance of A can be replaced by an instance of B without notice (compatibility, keeping system integrity)	Instances of D are not possible, since D is represented as instance
Degree of determination	*Not determined*: Properties can be added (monotonic extension through subclasses, A specifies only a part of the properties of B)	*Determined*: A particular instance must conform to the set of valid instances that is defined by the class (C specifies all properties of D)
Concepts in logic	*Subordination*	*Subsumption*
Reusing characteristics	*Inheritance*	*Not possible*
Definition of special characteristics	Additional *class properties*	*Instantiation* of class features and class properties

Table 8.1. Comparison of generalisation/specialisation and classification/instantiation

Table 8.1 summarises the main differences between both types. It is further important to become aware of the different understanding of the relation between classes and instances in logic (based on set theory) and programming, as it motivates deep classification and, hence, Multilevel Modelling to a certain degree [177, p. 5].

In logic, a class is defined *extensionally*, i.e. the extension (set) of potential instances of a class. Instantiation is then rather understood as selecting one instance that fulfils particular predicates. This implicates that one instance can be valid to different predicate sets and hence be instantiated by multiple classes. While this is highly flexible, it is difficult in regard to abstraction quality and system analysis. In programming, a class is defined *intentionally*, i.e. as kind of a template with

pre-defined properties. Instances are then created from this template and implicitly conform to the intended and expected property ranges. However, an instance can only be part of the instance set of exactly one class, which in turn requires redundancy and the creation of artificial class structures [177, p. 5].

Based on the stated elaborations, Frank [177] attempts a three-step approach for the differentiation between generalisation and classification. At first, it is necessary to specify all properties of the meta classes which are under investigation [177, p. 14]. Thereby, the type of each property has to be determined according to Table 8.2 in order to analyse whether a property is inherited or instantiated from a particular meta class [177, p. 15].

Property Type	**Description**
Class feature	Characterisations of a class that cannot be applied to any instance. Class features describe a particular inherent state of a class that is not transferable to instances. Class features can be divided into *life-cycle features* and *derivable features*.
– **Life-cycle features**	E.g. class name, creation date etc.
– **Derivable features**	Depend on the corresponding set of instances (e.g. number of instances).
Class properties	Characterise and differentiate specific instances. They are used for the intentional definition of a class and a particular class property corresponds to a feature of a particular instance.
Class invariants	Constraints that apply to class properties. Invariants are valid for all instances and do not differentiate between them. A particular feature value must conform to these "globally defined" constraints.

Table 8.2. Property types

Two aspects mainly determine the differentiation between instantiation and specialisation: The way of reusing characteristics in a concretisation as well as the definition of specific characterisations in a concretisation [177, p. 19ff]. This leads to the set of guidelines indicating criteria-based differentiation that is outlined in Table 8.3.

Request for Concretisation	**Implication**
Representation of class features (life-cycle and derived)	Instantiation only
Reuse of properties as they are	Specialisation only
Specification of further characteristics	
– Arbitrary	Both
– Restricted (determination, cf. Table 8.1)	Instantiation only

Table 8.3. Requests for concretisation

8.4.4 Principle of Adaptation for EML Extensions

Despite the missing consideration in literature, the adaptation of the multilevel paradigm for EML extensions is indeed very promising, especially in regard to the domain-specificity dimension. By definition, GEMLs and PSMLs provide generic and particularly under-specified meta classes, which are promising in terms of domain-specific concretisation, either per generalisation/specialisation or classification/instantiation. In the context of BPMN, all process-related classes (e.g. Task,

Event, Lanes) as well as under-specified classes (e.g. Data Objects, Participants, or Resources) can be perceived as generic and semantically under-specified. A particular instantiation of those classes to domain-specific meta classes on a lower level of classification can enable an injection of specific domain classes that can be instantiated again.

8.4.5 Adaptation Procedure

Multilevel Modelling should be used as an extension approach for the integration of (meta) classes that are instances of meta classes from the original meta model, on the one side, and meta classes for instances on the other side as well. Those instances are thereby perceived as located on the same classification level as the instances of original meta classes. Specifically, the application of the multilevel paradigm for EML extensions should be conducted as follows. At first, respective candidates for multilevel meta classes have to be identified. These candidates then have to be assessed in terms of their multilevel requirements. If these candidates are perceived as multilevel meta classes, then respective syntactical implementations have to be applied. Below, the procedure is discussed in detail.

Step 1 – Candidate Meta Classes: The initial analysis of the extension demand brings out several types of language extension. From scratch, the following types serve as candidates for Multilevel Modelling:

1. Extension meta classes that constitute as *specialisations* of original meta classes.
2. *Instances* of original meta classes that may own *type characteristics*, too.
3. Somehow determined extension constructs that need *further investigation* in order to specify it as subclass or instance.

It is then necessary to conduct an analysis of the extension construct property types in order to elaborate a treatment as multilevel meta class within the extension. The following scenarios are thinkable:

- An extension construct can be classified as *specialisation as well as instantiation* to a certain degree. Properties are therefore both instantiated and reused from an original meta class.
- An extension construct serves as *instantiation* of an original meta class, but provides also type characteristics for further instantiation.
- An extension construct serves as *specialisation* of an original meta class and determines further instantiation of classes that are perceived as types, probably indicating delayed instantiation of particular properties, for instance.

Step 2 – Estimate Multilevel Capabilities: The particular assessment as instance of an original meta class is supported by the following criteria:

- On the level of the original meta class, some class features are represented (i.e. life-cycle class features and derived class features).
- Some or all properties of the original meta class are not simply re-used (indicator for specialisation), but rather instantiated.
- Determination and restriction: Instance appears as valid set of property values of an original meta class (i.e. within the valid, intended and determined range).

An assessment of both types of multilevel candidates in regard to their meta class capabilities should further be applied according to Frank [177] and the discussed criteria in Sect. 8.4.3.

Step 3 – Conduct Syntactical Implementations: After identifying multilevel classes, it is then necessary to specify their characteristics and especially define Intrinsic Features as an annotation of original meta classes as well within the extension constructs. The latter aspect requires a forethought of the intended further levels of classification, thereby causing a consideration of instantiation as specialisation as discussed above.

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.7).

8.4.6 Required Redesign

Self-evidently, the integration of Multilevel Modelling in existing MOF-based languages requires pervasive architectural modifications, since both BPMN and MOF are not capable of multilevel features. Respective concepts are entirely missing. Especially two major changes are therefore required (referring to [123, p. 333]):

- Flexible, recursive language architecture: The MOF architecture needs to be revised in regard to flexible classification levels and deferred instantiation. This implicates the abandonment of the strict separation of levels. For instance, FMMLx enables the creation of (meta) models (in the sense of language defining models) that are composed of classes from different classification levels.
- It is further necessary to implement Intrinsic Features in order to enable the deferred instantiation of properties or entire classes (class methods are not considered in detail).

Procedurally, the original meta model as well as the entire modelling environment need to be prepared for multilevel-based extensions. More specifically, meta classes from BPMN need to be marked with Intrinsic Features and potential extension classes need to be assessed in regard to instantiation/classification. Accordingly, respective properties or entire classes have to be annotated by an "is intrinsic" property as well as the intended level of instantiation. Afterwards, additional classification levels need to be integrated through partial specialisation of the original meta model, leading to a "multilevelled" meta model that can be instantiated in order to apply the extended BPMN meta model.

8.4.7 Demarcation from Other Approaches and Limitations

In regard to the Profiling techniques, extension by Multilevel Modelling differs in terms of the fact that Profiling enables not real generalisation/specialisation but rather an attributive annotation of meta classes. Stereotypes allow semantic extensions through additional properties (more specifically, additional features during instantiation), but no real specialisation in the sense of dedicated and independent types. Additionally, Multilevel Modelling also differs from all other techniques in regard to the unique capability of creating additional classification levels and breaking the fixed four level architecture. Therefore, it is essentially important to assess

whether instantiation/classification or generalisation/specialisation is more appropriate (cf. Sect. 8.4.3).

Multilevel Modelling represents a novel and breakthrough modelling paradigm that is accompanied by major conceptual and, hence, technical innovations and modifications. Frank [123] therefore emphasises requirements for modelling tools in particular. Massive re-engineering of existing meta modelling languages like MOF is required for the representation of multilevel concepts. This provokes the question whether it is more feasible to integrate BPMN in FMMLx first, as proposals for MOF revisions are hardly implementable. An extension by Multilevel Modelling further requires an annotation of Intrinsic Features to the original meta model, which could implicate semantic redefinitions.

Additionally, there are some fundamental issues in regard to Multilevel Modelling. For instance, the substitution principle is broken in FMML and further research is needed in order to find the right number of classification levels.

8.4.8 Pragmatics

Basically, the principle of flexible or localised DSML architectures PSMLs can be adapted in order to create vertical refinements of the vocabulary of GEMLs and PSMLs, which enables the creation of EML accents and EML dialects for a particular domain (industry), enterprise, project, or situation. Also, extensions at model runtime are possible (cf. [123]). Multilevel Modelling hence serves as an opportunity to reduce and potentially solve identified issues with intermediate layers in terms of language dialect building (cf. [40, 39]). Further, it can be used for enhancement and augmentation on lower level of classifications, i.e. if additional modelling levels are extended with respective classes.

8.5 Simple Generalisation/Specialisation

8.5.1 Motivation

The elaboration of Multilevel Modelling reveals the importance of differing generalisation/specialisation and classification/instantiation for extension situations. Multilevel Modelling has been adapted as a convenient method for the implementation of classification/instantiation extensions in EMLs. Contrarily, a basic generalisation/specialisation cannot be found, although Profiling serves as a means for de facto implementations of subclasses [42]. However, neither the definition of real subclasses nor multiple generalisations is possible, as Stereotypes rather provide lightweight extension means on the attribute level. Annotation approaches as well as hooking rather support the implementation of additional classes and do not focus on specialisations. Multilevel Modelling provides specialisations implicitly, but has a more sophisticated focus. We therefore proclaim the application of a simple generalisation/specialisation technique in order to specify generic meta classes on the same level of classification. Several BPMN extensions already make use of this fundamental specification technique (e.g. [33]).

8.5.2 Architecture and Application

The general implementation is rather simple: An extension class specializes an original meta class in accordance with an elaboration of criteria from Sect. 4.3.1. Consequently, the properties of the superclass are reused in the subclass and extended by additional properties for extension. Three types of extensions are divided:

Punctual Specialisations (Type 1): This type refers to the introduction of perhaps multiple subclasses of original meta classes. Respective subclasses own additional properties but no references to other types. Explicitly, multiple subclasses, i.e. subsubclasses, are feasible.

Punctual-Connected Specialisations (Type 2): This type corresponds to the first type, but the introduced subclasses may own relations to other extension subclasses. Consequently, the level of dependency and complexity is higher.

Hybrid of Specialisation and Annotation (Type 3): This type corresponds to the second type, but the stated relations may also refer to original meta classes implicating partial annotations. Due to the proclaimed need of non-invasiveness, these references may not be mandatory (cf. Sect. 8.1).

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.8).

8.5.3 Pragmatics

This basal extension type can be primarily used for enhancement in the sense of refining generic concepts. It can be also applied in order to realise specialisations in the domain-specificity extension. In both cases, generic meta classes are specified to a certain degree.

8.5.4 Restrictions and Limitations

The application of this mechanism is generally unproblematic due to its simple architecture. Punctual specialisations as well as punctual-connected specialisations can be separated very well. Only the third type may implicate the issue of rather complex dependencies (cf. Sect. 8.1).

8.6 Semantic Extension Techniques

8.6.1 Motivation

The presented techniques primarily address the syntactical level of EMLs and implement extensions by the introduction of additional meta model classes. This causes an extension of the EML vocabulary and indicates some modifications of respective modelling tools (at least recompilations of the meta model). In regard to reaching a higher level of syntactical non-invasiveness, it might be promising to keep the EML syntax unaffected and solely focus on its intended interpretation, namely semantics (cf. Sect. 4.3.4). This could reduce particular implementation efforts and may enhance the level of specification in regard to domains or within a certain group of stakeholders elaborating consensus on the meaning of particular EML concepts in specific situations.

8.6.2 Architecture and Application

Semantic extensions cover adding and specifying operations. The limitation to these operations is required in order to avoid language defacements and semantic modifications of the original EML [80, 73]. In particular, additional semantic mappings as well as additional semantic constructs have to be annotated to the original semantic structure of an EML [73]. The representation of this structure should not be addressed in detail at this point, as semantics in EMLs – both at model level and meta model level – is still an under-investigated issue that requires extensive fundamental research [73, 9, 57]. A holistic analysis von EML semantics would go beyond the scope of this work significantly and should be considered in further research projects (cf. Sect. 21). Despite this limitation, ontologies seem to be promising means for the representation of semantic concepts and mappings (cf. [100, 183]). For instance, the Unified Enterprise Modelling Language (UEML) could serve as a powerful instrument for the fundamental description of enterprise-related concepts.

In particular, **adding semantics** means that additional semantic constructs and hence semantic mappings are introduced for a syntactical construct. Of course, these additional constructs must not contradict the original semantics of a meta class.

Specifying semantics refers to the introduction of additional semantic constructs, which are specialisations of the originally referred to semantic constructs (e.g. in case of under-specification). These extensions should be encapsulated within semantic extension profiles [156]. Semantic profiles should contain references to respectively extended syntactical constructs as well as the set of additional mappings and semantic constructs. In case of specialisation, also original semantic constructs have to be integrated as outlined in Sect. 4.3.4.

Further specification of the mechanism is provided in the Appendix (Sect. 1.1.9).

8.6.3 Pragmatics

Generally, extension by semantics can be used for specifying interpretation guidelines of EMLs, while the original meta model remains unaltered. It is therefore useful for sharpening the understanding of particular concepts in the context of particular domains. Semantic extensions are therefore useful for the creation of accents, in regard to formalisation as well as domain-specifity (cf. Sect. 5.4.3). In the first case, semantic extensions are used for reaching a higher level of formal invariance [14, 67]. Nevertheless, additional syntactical extensions should be examined in regard to model constrains and validity rules. In the second case, semantic extensions are used to limit particular interpretations within a group of language users (e.g. domain accents).

8.6.4 Restrictions and Limitations

As outlined in different papers (cf. discussion in [57]), specification of semantics within EML meta models has been poorly implemented and a lot of specific issues stemming from the semi-formality of EMLs has not been sufficiently considered in literature so far [9]. Consequently, also the general semantic specification of extensions (cf. [73]) and respective semantic extensions is difficult due to the lack of commonly accepted semantic description techniques. Semantic specifications could

further cause massive specification efforts due to the number of related dimensions (e.g. epistemological or lexical aspects [9]). Semantic extensions further suffer from a certain lack of explicitness due to need for providing "interpretation help" for an EML that can differ according to the group of readers. It is hence important to specify the respective group of recipients.

9

Repository

9.1 Overview

The discussed extension mechanisms can generally be divided into primarily syntactical and primarily semantic mechanisms, whereby only one mechanism exclusively addresses semantic extensions. Regarding the syntactical mechanisms, three types can be differentiated: Annotations, Specialisations, and Placeholders.

Annotations represent mechanisms which extend a meta model at specific spots in a modular way with different degrees of complexity and dependency. Thereby, the classification level and the level of abstraction remain stable, which can be referred to as horizontal extensions, aiming to realise a particular detailing. Different nuances between mechanisms are emphasised in Table 9.1.

Specialisations cover rather vertical refinements and specifications of existing meta models; either by Hooking placeholders or by multiply specialised meta classes with simple Generalisation/Specialisation. Both approaches foster a particular refinement of original language parts. In contrast to these mechanisms, Multilevel Modelling has a special role, as it covers the creation of multiple additional classification levels.

Placeholders take a special position due to the comprehensive preparation need within the meta model, i.e. the EML designer actually needs to create the particular extension areas ex ante. More precisely, meta model reference models for EM are needed (referring to [174]) and the underlying EML needs to be pre-designed in a special way.

Furthermore, *hybrid* implementations combine characteristics from multiple classification types. Indeed, Profiling provides a de facto specialisation, but it maintains also characteristics of simple annotations [45, 42]. Profiling rather enables the (probably mandatory) annotation of property instances to extended meta classes. Further, simple Generalisations/Specialisations may be combined with annotation mechanisms.

Syntax			Semantics
Annotations	**Specialisations**	**Placeholders**	
Decorator: • Annotating roles • Flexible hierarchies • Optional extensions **Plugin:** • Interfaces have multiple meta classes • Domain concepts • Complex structure, modular – **Add-On:** • Optional • Punctual-conceptual extension • Not complex, not modular – **Aspect:** • 1 meta class • Varying complexity • Cross-cutting concepts • Modular	**Hooking by Specialisation:** • Subclasses of marked spots of a meta model (predesign efforts) • Same classification level **Simple Generalisation/Specialisation:** • Subclasses of not explicitly marked spots • Same classification level **Multilevel Modelling:** • Additional classification levels • Annotation of Intrinsic Features for level-spanning classification • Duality of specialisation and instantiation	**Hooking by Placeholder:** • Filling of left open meta model parts (classes, properties, range values etc.) • Predesign efforts	**Add Semantics:** • Additional semantic mappings **Specify Semantics:** • Additional semantic mappings and specifying constructs
Hybrids			
Profiling: • Features of both types, no real specialisation **Simple Specialisation/Generalisation + Plugins or Add-Ons** • Domain-specific extension by specification and annotation			

Table 9.1. Summary of mechanisms

9.2 Comparison

The analysis and consideration of respective extension types due to the intended extension need then leads to an assessment of particular implementation consequences. A detailed analysis of mechanisms is therefore necessary. Table 9.2 summarises particular characterisations in order to provide an appropriate comparison. Single criteria are briefly discussed hereinafter.

	Decorator	**Plugin**	**Add-On**	**Aspect**	**Hooking**	**Profiling**	**Multilevel**	**Gen./Spec.**	**Semantics**
Invasiveness and Interface Structure	– –	o	+	+	o / – –	+	– –	+	++
Application Complexity	–	+	+	+	o	++	–	++	+
Multiple Applications	++	++	++	++	– –	++	++	++	++
Extended Meta Types	Classes	Classes	Classes	Classes	All	Classes, (All)	Classes, Properties, (Language)	Classes, (All)	All
Modularity, Level of Independence	– –	++	– –	++	– –	– –	o	– –	– –
Representation of complexity	o	++	–	o	– / +	o	++	o	o

Table 9.2. Comparison of mechanisms; legend: (++) very appropriate (+) appropriate (o) indifferent (–) inappropriate (– –) very inappropriate

With regard to *meta model invasiveness* and required interface structures, Add-Ons, Aspects, Profiling, Generalisation/Specialisation as well as Semantic Extension are perceived as the most appropriate techniques, as they can simply be added to existing meta models. In other words, only very limited preparations and pre-designs are necessary in the original meta model. In contrast, Plugins and Hooking by Specialisation require the explicit specification of extension interfaces that can be shaped by extensions, while Add-Ons and Aspects are rather optional. Decorators indicate complex binding routines; Multilevel Modelling requires both a paradigmatic revision of the original meta model as well as revision in terms of annotating Intrinsic Features, for instance. Hooking by Specialisation requires multiple explicitly underspecified parts in the meta model. This creates the situation that the meta model cannot be instantiated without filling in particular meta elements.

With respect to the *application complexity*, the introduction of Specialisations (simple subclass building) as well as Profiling (separate Stereotype definition and annotation) are most straightforward. Further, annotations (Plugins, Add-Ons, Aspects) serve as rather simply applicable instruments, but require the identification of respective integrator classes (cf. [45]) and probably merging procedures. Semantic Extensions may suffer from the stated representation issues. The Hooking technique requires rather extensive meta modelling works within the set boundaries and both the Decorator and Multilevel Modelling approach indicate multiple steps in application, which hamper a straightforward definition and application.

Nearly each of the elaborated mechanism supports *multiple extension* application. Hooking presents the only exception due to the implicit one-way instantiation of the hooking points.

The discussion on *extended meta class* types should be limited to the abstract syntax, as current literature strongly focusses on this part of EML definitions, while a concise specification of the concrete syntax is omitted (cf. the discussion in [42]). All annotation types cover extensions with additional meta classes, but reveal fine differences. For instance, Plugins and Aspects build a strong inner coherence, while Add-Ons are rather appropriate for punctual class-wise extensions and Decorators annotate some kind of roles to meta classes. Multilevel Modelling can be used for extending classes and properties (e.g. making them intrinsic) and for extending the entire language specification by instantiating parts of it. Both Hooking and Semantic Extension are able to specify or extend each meta model type. Profiling is primarily used for class annotations, but also generic applications to all meta model types have been discussed recently [156]. Simple Generalisation/Specialisation is naturally applied to classes, but – due to the basal character of the mechanism – application to other meta types seems to be reasonable.

Modularity plays an important role in regard to extension re-use and potential adaptation to other EML meta models. This implies high cohesion, low coupling to the original meta model, and a certain complexity within the application. Only Plugins and Aspects satisfy these requirements. Multilevel Modelling is tightly coupled to the original meta model due to the instantiation relation, but may provide independent elements on introduced classification levels. All other mechanisms reveal a high coupling to the original language.

Consequently, the achievable *level of complexity* differs among the mechanisms. Plugins and Multilevel Modelling are most prominent in terms of specifying complex extensions, since Plugins are understood as separate meta model components (within the same classification level) and Multilevel Modelling enables flexible architectures with multiple classification levels. Hooking by Placeholders also supports complex extension designs, but is limited by the set borders of the hook. Decorators, Aspects, Profiles, Generalisations, and Semantics serve limited complexity, as they annotate original meta model elements, but have only limited inner complexity. Add-Ons and Hooking by Specialisation have an even lower degree of such relations or miss them totally. The impact of their particularly produced extensions is, hence, limited locally.

9.3 Combination of Mechanisms

Combinations of mechanisms are generally possible if extension need causes the selection of multiple extension mechanisms. Extension mechanisms can be combined, but the following conditions have to be respected:

- Annotation techniques can be combined among each other without any limitation.
- If the Hooking mechanism is applied, other approaches can only be applied to those original meta model elements that are not marked as hooking points (cf. [45]). Consequently, combinations with Hooks are implicitly limited.

- If the Multilevel Modelling approach is selected, then all other mechanisms can only be applied to the original classification level indicating a sequential processing of extensions. At first, extensions (e.g. annotations) on the original level of classification are applied in order to extend the meta model. Then, respective multilevel extension can be applied to this extended version of the meta model in order to assign respective features, for instance.
- Semantic extensions can be combined with any kind of syntactical extensions. Self-evidently, particular semantic mappings have to be assigned to the extended meta element first, i.e. syntactical extensions have to be applied first in any case.

The combination of single mechanisms causes the creation of an extension bundle that consists of single extensions. Generally, these bundles serve as loose collection of extensions, which may be associated indirectly by the meta model elements they extend. Additionally, it might be feasible to integrated dependencies between single extensions (e.g. combining Add-Ons and Generalisations as outlined in Table 9.1). These aspects cause two implementation consequences.

First, it is necessary to generally explicate dependencies between extensions if one extension refers to or uses concepts from another extension. Second, it might be necessary to define a specific sequence of applying extensions in order to specify respective logical dependencies in detail. For instance, the combination of Plugins and Generalisations could require such a sequence if the Generalisation extension specialises introduced Plugin classes.

Part IV

Semantics-Driven Justification of Extension Need

10

Relevant Publications

10.1 Publication DESRIST-2015

General Information:

Title:	Proposal for Requirements Driven Design Science Research
Authors:	Richard Braun, Martin Benedict, Hannes Wendler, Werner Esswein
Year:	2015
Medium:	Donnellan, B., Helfert, M., Kenneally, J., VanderMeer, D., Rothenberger, M., Winter, R. (Eds.): New Horizons in Design Science: Broadening the Research Agenda, 10th International Conference on Design Science Research in Information Systems (DESRIST)
Series:	Lecture Notes in Computer Science, Vol. 9073
Pages:	135-151
Reference:	[79]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions:

Richard Braun:	60% (introduction and motivation, requirements engineering ontology, requirements-driven DSR framework, reasons for critique, conclusion and further research)
Martin Benedict:	20% (requirements engineering)
Hannes Wendler:	10% (related work)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	Design Science Research
Motivation:	Shortcomings in design-oriented research: issues regarding operationalisation (missing concrete guidance), procedural transparency, demarcation from professional design
Input and Method:	Adaptation of Requirements Engineering to Design Science Research
Contributions:	Requirements-driven DSR framework and DSR requirements types

Abstract:

Design Science Research (DSR) still reveals several methodical shortcomings, which need to be remedied in order to enhance the maturity of DSR and its derived artifacts. For instance, there is a remarkable lack in methodical support for problem formulation. Also, DSR does not provide detailed procedure models, which can be operationalized appropriately. This compromises rigor within the design process and hampers demarcation from professional practice. In order to tackle these issues, we propose the adaptation of Requirements Engineering for structuring the problem space and deriving design decisions systematically. Requirements are also intended to work as glue between single design stages in order to keep the design process comprehensible and transparent. We therefore justify an ontology-based analogy between requirements analysis and DSR parts and provide a requirements-driven DSR framework based on a four-part ontology that especially focuses problem analysis and design preparation. Moreover, a detailed state of the art is presented and our approach is discussed within a critical appraisal.

10.2 Publication REBPM-2014

General Information:

Title:	Requirements-Based Development of BPMN Extensions: The Case of Clinical Pathways
Authors:	Richard Braun, Hannes Schlieter
Year:	2014
Medium:	Heinrich, R., Kirchner, K., Weißbach, R. (Eds.): IEEE 1st International Workshop on the Interrelations between Requirements Engineering & Business Process Management (RE-BPM)
Pages:	39-44
Reference:	[55]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	70% (introduction and motivation, BPMN extensibility, method for extension development)
Hannes Schlieter:	30% (demonstration, conclusion, further research)

Summary of Contents:

Context:	BPMN
Motivation:	Extension justification and methodical guidance during BPMN extension design
Input and Method:	Requirements-driven design, domain analysis based on ontologies
Contributions:	Enhancing the BPMN extension method of [48] with regard to domain analysis

Abstract:

In recent years, the Business Process Model and Notation (BPMN) has evolved to one of the most applied modelling General Purpose Languages in the Business Process Management discipline. Due to its application in different domains, it becomes frequently necessary to extend the BPMN by domain-specific concepts. Extending the BPMN fosters an adequate communication between system engineers and domain experts and enhances the semantically correct representation of the domain. Therefore, the BPMN meta-model provides an extension mechanism. However, there is a remarkable lack in procedure models for the design of such extensions. This re-

search article outlines an extension method that focuses on domain analysis, extension requirements and the derivation of domain-specific extension concepts within BPMN. Therefore, the method of Stroppi et al. (2011) is extended in regard to domain analysis. The approach is motivated and demonstrated by the case of Clinical Pathways (CPs).

10.3 Publication BIBM-2015

General Information:

Title:	Clinical Processes from Various Angles – Amplifying BPMN for Integrated Hospital Management
Authors:	Richard Braun, Martin Burwitz, Hannes Schlieter, Martin Benedict
Year:	2015
Medium:	Huan, J., Miyano, S., Shehu, A., Hu, X.T., Ma, B., Rajasekaran, S., Gombar, V.K., Schapranow, M., Yoo, I., Zhou, J., Chen, B., Pai, V., Pierce, B.G. (Eds.): 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)
Pages:	837-845
Reference:	[77]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	50% (adapting and extending BPMN for hospital modelling, BPMN extension definition)
Martin Burwitz:	30% (integrated hospital modelling)
Hannes Schlieter:	10% (conclusion and further research)
Martin Benedict:	10% (demonstration)

Summary of Contents:

Context:	BPMN, EMLs
Motivation:	Need for flexible perspective architecture in BPMN
Input and Method:	Extended BPMN extension method, including the BPMN extension method of [48]
Contributions:	BPMN extension: • BPMN4CP v2.1 • In-detail specification of all components and their integration spots

Proposal for an integrated specification of the syntax of EMLs within the OMG environment:

- MOF, MOF+
- DG, DG+
- QVT for integration

Abstract:

The design and management of information systems is driven by model-oriented approaches on different levels of abstraction. For instance, enterprise models link the organizational action system and corresponding information systems. Enterprise models are generally perceived as measures to close the gap between business and IT, as it is in the healthcare domain. Clinical Pathways (CPs) represent value-added processes of hospitals and are typically described by respective process modeling languages like BPMN. However, a solitary focus on processes is insufficient for utilizing the potential of the model. Instead, it is rather advisable to consider various perspectives in order to completely represent the organizational action system of a hospital. We therefore propose to extend clinical process models with accordingly required perspectives for the representation of satellite objects, e.g. medical resources. Based on previous work, this paper motivates the approach of (process-based) integrated hospital modeling and presents the architecture and design of a revised BPMN extension for multi-perspective modeling. The applicability of the proposed BPMN extension is demonstrated by modeling a CP part for the treatment of stroke patients, which explicitly integrates the process and resource perspective.

10.4 Publication EEWC-2015

General Information:

Title:	Towards Multi-Perspective Modeling with BPMN
Authors:	Richard Braun, Werner Esswein
Year:	2015
Medium:	Aveiro, D., Pergl, R., Valenta, M. (Eds.): Advances in Enterprise Engineering IX, Proceedings of the 5th Enterprise Engineering Working Conference
Series:	Lecture Notes in Business Information Processing, Vol. 211
Pages:	67-81
Reference:	[44]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions:

Richard Braun:	90% (introduction and motivation, fundamentals, BPMN extensibility in general, extension of the BPMN meta model, methodical support and demonstration, conclusion)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Defining perspectives in BPMN as well as BPMN extensions
Input and Method:	Extended BPMN extension method, including the BPMN extension method of [48]
Contributions:	BPMN meta model extension: • Perspective definitions • Diagram definitions • Integration with DD

Enhancing the BPMN extension method of [48] in terms of defining perspectives and diagrams:

- Extended CDME
- BPMN+X CS
- BPMN DG
- BPMN DI

Outlining an extension on the M3 level:

- Integrated syntax specification by associating MOF and DD

Abstract:

BPMN is the prevalent process modeling language and a lot of domain-specific BPMN extensions have evolved during the last couple of years. Due to the plenty of extensions and elements within BPMN, it is promising to consider complexity reduction mechanisms in order to provide appropriate, purpose-specific views on BPMN models. We therefore analyze capabilities of BPMN in regard of the definition of additional perspectives and diagrams in order to provide dedicated views on aspects of business processes (e.g., separate resource diagrams). As both BPMN and BPMN-defining MOF reveal shortcomings regarding to the definition of perspectives, we introduce a BPMN meta model extension in order to allow an integrated definition of new perspectives and their respective graphical elements. We further provide methodical guidance by conducting and customizing the BPMN extension method of Stroppi et al. (2011).

10.5 Publication IDS-2015

General Information:

Title:	Epistemological Foundations for the Integrated and Multifaceted Description of Stipulated Semantics in Enterprise Modeling Languages
Authors:	Richard Braun
Year:	2015
Medium:	19. Interuniversitäres Doktorandenseminar Wirtschaftsinformatik
Reference:	[41]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	EMLs
Motivation:	Lack of integrated analysis of EML semantics on the meta model level
Input and Method:	Consideration of epistemology and its consequences to semantics
Contributions:	Epistemologically founded semantics framework (dimensions and aspects, which influence the understanding of meta models): • Ontological aspects • Subjectivist aspects • Epistemological aspects • Linguistic aspects Further implications: • Fundamental consideration of semantics in EML meta models, research challenges • Points for misunderstanding • Concepts of implicit stipulated semantics and explicit stipulated semantics

Abstract:

Enterprise modeling languages (EML) are often subject of language extensions in order to satisfy particular stakeholder needs. While there are some publications on syntactical aspects and techniques of EML extensions, there is a notable lack regarding to semantic topics, which is tightly related to the generally under-investigated field of EML semantics. This paper hence intends to propose an integrated and multifaceted framework in order to derive a comprehensive view on EML semantics, their parts and particular consequences. The framework covers the meta model layer, the meta meta model layer and respective areas of discourse. Those parts are elaborated in terms of ontological aspects, subjectivity, linguistic aspects and finally epistemological positions, which can be taken when discussing EML semantics. This paper hence aims to consider EML semantics from an epistemological point of view in order to establish a stable foundation for the creation of an integrated semantic description of EMLs in further research.

10.6 Publication HICSS-2016

General Information:

Title:	BPMN4CP Revised – Extending BPMN for Multi-perspective Modeling of Clinical Pathways
Authors:	Richard Braun, Hannes Schlieter, Martin Burwitz, Werner Esswein
Year:	2016
Medium:	Bui, T.X., Jr., R.H.S. (Eds.): 49th IEEE Hawaii International Conference on System Sciences (HICSS)
Pages:	3249-3258
Reference:	[76]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions:

Richard Braun:	60% (research approach, consequences for extension evolution, extension design, tool implementation)
Hannes Schlieter:	20% (introduction and motivation, conclusion)
Martin Burwitz:	10% (demonstration)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	BPMN
Motivation:	Multi-perspective modelling of CPs with BPMN
Input and Method:	Enhanced BPMN extension method and requirements-driven Design Science Research
Contributions:	BPMN extension: • BPMN4CP v2.0 • Additional perspectives in BPMN Further enhancement of the BPMN extension method: • Domain analysis (domain ontology) • Extension preparation (CDME) • Extension meta model (BPMN+X, BPMN+X CS, BPMN DG, BPMN DI)

Abstract:

Clinical Pathways (CPs) can be seen as business processes of hospitals or clinical institutions. Modeling these pathways is an emerging field of research, as it provides promising benefits for systems integration, quality management and documentation. The Business Process Model and Notation (BPMN) provides a range of process-related concepts but naturally lacks in representing specific aspects from the CP domain. Therefore, the BPMN extension BPMN4CP was designed in a previous research project. In accordance with research guidelines from Design Science, the extension ran through an iteration based on its practical application within a telemedical project. Based on several new requirements, the extension was revised regarding to the integration of resources, documents, objectives and quality indicators. These concepts were assigned to particular perspectives and diagrams in order to support model complexity management and provide appropriate diagrams for respective stakeholders. In order to provide a commonly usable extension, these enhancements were implemented as BPMN meta model extension.

10.7 Publication ZEUS-2016

General Information:

Title:	SemFrameX – Towards a Framework for the Semantic Justification of BPMN Adaptations
Authors:	Richard Braun
Year:	2016
Medium:	Hochreiner, C., Schulte, S. (Eds.): Proceedings of the 8th ZEUS Workshop
Series:	CEUR Workshop Proceedings, Vol. 1562
Pages:	13-20
Reference:	[9]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	BPMN
Motivation:	Semantic justification of BPMN extensions
Input and Method:	Application of the generic semantics framework from AQEMO-2016
Contributions:	SemFrameX framework for BPMN

Abstract:

In recent years numerous extensions and adaptations of the BPMN evolved, since model users aim to both exploit the benefits of the modeling standard and adapt BPMN to particular domain peculiarities or project requirements. Methodical support for conducting such adaptations is generally rare and very focused on the abstract syntax, which is actually insufficient, since particular semantics are more relevant. Consequently, it seems to be reasonable to explicitly conduct semantical analysis and comparison checks before extending or adapting BPMN. However, appropriate semantic specifications of BPMN are missing. After introducing and motivating the entire issue, we therefore outline the *SemFrameX* framework that aims to specify the BPMN meta model semantics with a special consideration of ontic, epistemological, conceptual, linguistic and pragmatics aspects.

10.8 Publication AQEMO-2016

General Information:

Title:	Towards a Multi-Faceted Framework for Semantics in Enterprise Modeling Languages
Authors:	Richard Braun
Year:	2016
Medium:	Betz, S., Reimer, U. (Eds.): Modellierung 2016 - Workshopband
Series:	Lecture Notes in Informatics, Vol. 255
Pages:	33-44
Reference:	[57]

Rankings:

VHB-JOURQUAL-3:	C
WKWI-2008:	B

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	EMLs
Motivation:	Multi-faceted framework for EML semantics
Input and Method:	Consolidation of different research streams affecting EML semantics, esp. epistemological foundations
Contributions:	SemFrame framework and types of pragmatics

Abstract:

The semantic specification of Enterprise Modeling Languages (EMLs) is a challenging task that is primarily caused by the immanent subjectivity in the context of enterprise modeling. This covers the interpretation of respective meta model constructs due to their references to the reality. In contrast, EMLs may also contain formal semantics in regard of automating specific parts. Despite the generally accepted relevance of semantics for the application of EMLs, current research lacks in the provision of appropriate description means and largely omits semantic investigations. This paper therefore proposes a multi-faceted framework for the analysis and description of EML semantics in order to increase the awareness of relevant influences. The framework consists of an ontic and epistemological dimension in its core, as material semantics finally address such fundamental aspects. On this basis, several wrapping dimensions are outlined: Conceptualization dimension, pragmatic dimension, representation dimension and the final consensus dimension.

10.9 Publication MODELSWARD-2016

General Information:

Title:	Towards Hybrid Semantics of Enterprise Modeling Languages
Authors:	Richard Braun, Werner Esswein
Year:	2016
Medium:	Hammoudi, S., Pires, L.F., Selic, B., Desfray, P. (Eds.): Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development
Pages:	412-420
Reference:	[67]

Rankings:

VHB-JOURQUAL-3:	not ranked
WKWI-2008:	not ranked

Contributions:

Richard Braun:	90% (introduction and motivation, formal semantics, material semantics, towards hybrid semantics, conclusions and further research)
Werner Esswein:	10% (research conception)

Summary of Contents:

Context:	EMLs
Motivation:	Analysing and elaborating a possible integration between formal and material semantics
Input and Method:	Consolidation of different research streams in regard to material and formal semantics
Contributions:	Framework for hybrid semantics: • Clarification of different types of semantics and their possible integration • Concept of hybrid semantics • Relevant types of semantics: Behavioural formal semantics, static formal semantics, material semantics

Abstract:

Enterprise Modeling Languages (EMLs) are generally perceived as conceptual modeling languages having a formal syntax and informal semantics. The non-formality of semantics is mainly caused by the materiality of the addressed domain (enterprises and its related aspects) and the resulting personal interpretation of syntactical con-

structs. However, EMLs may also explicitly define invariant interpretations in the sense of possible model executions or the definition of domain-specific restrictions. It is therefore promising to address a possible amalgamation of material semantics and formal semantics in order to provide an integrated and comprehensive semantic specification of EMLs. This position paper introduces and motivates the topic by systematizing and consolidating approaches from both fields and introduces a framework for so-called hybrid semantics on the meta model layer. Further, the general relevance of semantics and semantic specifications in EMLs is emphasized and prospective research challenges are proposed.

11

Relevant Unpublished Papers

11.1 Paper UNPUB-MOF4EM-2016

General Information:

Title:	Syntactical Enrichment of the Meta Object Facility for the Definition of Enterprise Modeling Languages
Authors:	Richard Braun
Year:	2016
Medium:	Unpublished (submitted to the International Journal of Intelligent Information and Database Systems)
Reference:	[78]

Contributions:

Richard Braun:	100%

Summary of Contents:

Context:	EMLs
Motivation:	Enhancing the syntax of MOF for EML definition
Input and Method:	Re-engineering of the current MOF version based on peculiarities of the EM domain; demonstration by application to BPMN
Contributions:	MOF4EM: meta meta model extension for EML specification, MOF+ package: • Views and perspectives • Diagrams and symbols • Architectural adaptation of E3 for integrated definition

Abstract:

Recent research reveals a need for flexible Enterprise Modeling Languages (EMLs) in order to integrate problem-specific or domain-specific concepts on the meta model layer. The provision of generic adaptation mechanisms requires a well-defined and integrated meta modeling language. Several approaches exist for that, differing in terms of prevalence, vendor dependency and conceptual soundness. With regard to the standardization efforts in the OMG environment, it is reasonable to contemplate the OMG standards MOF and DD for the definition of EMLs, as they feature meta modeling capabilities. However, both approaches lack in the provision of enterprise-specific concepts and reveal some architectural issues (especially in regard of abstraction layers). Therefore, an elevation of debatable packages to the meta meta model layer is proposed based on a rational discourse. The revised abstraction architecture constitutes the fundament for the elaborated MOF4EM extension. MOF4EM introduces views and perspectives to the meta meta model layer and provides an

integration between abstract and concrete syntax. The proposed framework serves as conceptual base for the later supplement of appropriate adaptation mechanism.

12

Motivation and Introduction

This Section aims to provide means for the specification of extension demand within a potential extension situation. Therefore, a strict focus on the needs and requirements of EML users is proclaimed in order to provide the most adequate extensions. Consequently, adequacy is perceived as the most relevant criteria for EML assessment and, hence, the derivation of appropriate extensions based on pragmatics and semantics. Further, the relevance of pragmatics and semantics is emphasised, as the focus on EML users requires a deep understanding of these EML dimensions. Then, a generic EML extension procedure is outlined and its single components are presented in detail.

EMLs are usually defined by meta models that formulate the grammar and the type terminology of a language [175, 13]. The grammar specifies respective rules and constraints and determines valid expressions within an EML. The grammar is concretised by the terminology and explicated conceptualisations of a particular area of discourse [13]. Consequently, an EML meta model specifies the set of valid statements (models) about a specific domain and it pre-defines and limits the possible expressiveness about an area of discourse. In the context of EML extensions, the question arises of how the perceived quality of a meta model can be assessed in order to justify potential language extensions. While explicit meta model quality frameworks are missing in literature, at least the consideration of general model quality frameworks seems to be reasonable (e.g. [184, 185, 186, 187]).

12.1 Related Work

Lindland et al. [184] propose a framework spanning the criteria semantic quality, syntactic quality, pragmatic quality, as well as appropriateness. Syntactic quality is the degree of correspondence between a particular model and the language. It represents the set of syntactic errors, i.e. if a particular model is not a valid instance of the meta model. Semantic quality reflects the degree of correspondence between model and domain by formulating the criteria feasible validity and feasible completeness. Pragmatic quality represents the degree of correspondence between model and audience interpretation. Pragmatic quality therefore indicates how a particular model is understood by a model reader [184, pp. 44ff].

Krogstie et al. [185] extend the framework of Lindland et al. [184] and introduce additional quality indicators, namely physical quality, knowledge quality, language quality, social quality, and perceived semantic quality [185]. Thereby,

perceived semantic quality is defined as the correspondence between the actor interpretation of a model and his current knowledge of the domain. The criterion relaxes the strict objective position within the framework of Lindland et al. [184]. Pragmatic quality is quantified by the level of comprehension. Further, the social quality indicator covers different types of agreement and implicates a certain consensus process, or at least a discussion process.

Schütte & Rotthowe [186] introduce several guidelines for modelling. The principle of construction adequacy reflects the degree of consensus according to the problem domain. Language adequacy covers syntactic correctness between models and their defining language. The principle of economic efficiency covers efficient model design. The principles of clarity, systematic design, and comparability focus on the comprehensive, coherent, and well-structured design of models [186].

Overhage et al. [187] present the 3QM framework for the evaluation of process models. The framework is composed of a set of quality indicators and respective quality metrics for each indicator. Single indicators are refined into more fine-grained and measurable indicators. For instance, the quality indicator "semantics" is determined by the indicators relevance, correctness, completeness, and flexibility [187, p. 222].

12.2 Pragmatics and Semantics First

The stated approaches provide general conceptual guidance regarding the creation of good models, but omit some aspects that are highly relevant for the meta modelling level as well as for operationalisation in the context of meta model extensions.

Epistemological Consequences for Semantics

Construction-oriented modelling generally requires a fundamental understanding of the addressed area of discourse as well as a consideration of respectively intended business actions based on a language [139, p. 59]. Moreover, the design of EMLs requires the identification of prospective language users and their intended modelling purposes [188, p. 31]. EM is therefore tightly coupled with epistemological and ontological considerations [41]. Despite their elementary relevance (cf. [110, 189, 85, 41]), the positions remain unclear or tacit within the stated model quality frameworks, which make it prone for several issues in regard to meta model semantics, e.g. misunderstandings, mismatched understandings or conceptual disagreements [190, 191, 101, 41, 57]. This lack of consideration is critical, as EMLs are closely related to the real-world or what is perceived as reality. Different epistemological positions and foundations could make conceptualisations about considered things incommensurable, hampering effective communication [57, p. 37].

The reflection of three main types of EM-related topics should underpin this issue. First, there is the group of concepts that are generally perceived as fundamental in EM, e.g. Processes, Actors or Resources. Despite the tacit assumption of common sense on the interpretation of these concepts, van der Linden [101] investigates the interpretative diversity regarding to the set of these basal EM concepts, which creates several sources for communication errors. Second, there might

be variance according to more specific concepts of problem spaces, especially in interdisciplinary areas [57]. Third, normative or solution-oriented concepts within EMLs require special consideration. These concepts represent targeted and intended concepts for business solutions within the mind of single model stakeholders and their common interpretation is therefore contingent.

Considerations on semantic constructs and semantic mappings in EM are hence inevitably coupled with epistemological issues within the area of tension between Methodic Constructivism and Radical Constructivism (with regarded consequences to related positions of truths [41, 57]). This implicates a reconsideration of the appropriateness of frequently used means for ontological analysis of EML, e.g. the Bunge-Wand-Weber (BWW) ontology [192, 193]. The BWW ontology proclaims a pre-existence of true and thus somehow fixed or commonly accepted concepts for the representation of IS. A complete reliance on the existence of these concepts is at least questionable (cf. the critique in [194]). It seems to be more reasonable to suppose a reality (or area of discourse) that is composed of some things that are not generally perceivable or different according to their degree of semantic variance (personal semantics as elaborated in [101]).

It is therefore useful to conduct the frequently proposed, but rarely concretised position of consensus theory of truth that is suitable for constructivist positions [110, 41, 57]. The perceived semantic quality of an EML is hence determined by a common understanding of related items. Possibly precise and comprehensive *externalisation and presentation* of the intended semantics are hence required (i.e. the semantics of the EML designers) in order to facilitate respective agreement and consensus processes [41, 57] and identify semantic misinterpretations, mismatches, and misunderstandings [101].

The *representation* and following homogenisation of semantics should enable communication about a particular domain and should facilitate the understanding of an EML within a stakeholder group that is finally explicated in the right application of an EML. Consequently, semantic considerations only constitute an intermediate step within the process of applying meta model statements adequately. Pragmatically, this represents a certain (user-specific) translation process from perceiving and understanding models towards the realisation of some business-related follow-up actions in real business life that are intended to create some added value, e.g. optimising a business process or gaining better comprehensibility of an IS in general.

Adequacy as Indicator for Pragmatic Quality

The actual quality of an EML therefore comes from the particular appropriateness as means for realising a corresponding purpose or objective. Consequently, quality criteria like correctness [187] and completeness [185] are not only debatable from an epistemological point of view (as elaborated above), but also in regard to meta model quality in general. Instead of finding objectively correct and complete meta models, the assessment of an EML should consequently be driven by the *adequacy* of an EML in regard to a particular situation and a prospective user group. Instead of proclaiming an absolute degree of correctness or completeness (as implicitly given in parts of BPMN), it is rather useful to examine models and modelling languages with a focus on the actual purpose and underlying means-end relations. This im-

plicates an analysis of considered users, contexts and objectives [40, pp. 433-437]. Karagiannis [68] proposes a shift from an inwards quality perspective of modelling languages to a rather outwards perspective. This implicates a shift from criteria like consistency, integrity or performance to user-oriented indicators, i.e. usability and adequate language applicability. The author further pointedly formulates that "all models are wrong, but some are useful" [68, p. 5]. The author elucidates that the truth of a model is not relevant, but rather its *usefulness* for businesses. It is hence necessary to focus on both the pragmatics dimension of EMLs and the semantics dimension of EMLs in detail.

Consequently, the pragmatic value of an EML is the actual driver for application, not primarily the semantics and the syntax (referring to [20]). Thereby, semantics and pragmatics should rather correspond to each other, as thinking in solutions (pragmatics) refers to respective conceptualisations (semantic constructs [57]). The investigation of *semantics and pragmatics* is therefore very promising for extension justification. It is hence advisable to focus on these rather user-related perspectives instead of solely examining syntactical tasks [64, 65, 40]. The assessment of the syntactical quality of a meta model then seems to be trivial, as it can be determined by checking the conformance of the model instance in regard to the meta model [184, 185, 186, 41].

Benefits for the Design of EML Extensions

In the context of EML extensions, it is therefore necessary to understand what is intended by prospective or current EML users (*intended pragmatics*) and which specific domain concepts are covered (*intended semantics*, referring to [9]).

Focusing on the intended semantics of users is further promising in regard to independence of any syntactical consequence or implementations in order to separate the problem space (semantic constructs, pragmatics) and the solution space (primarily syntax, semantic mappings). Specifying the intended semantics is therefore a feasible starting point both for language extension (e.g. [55]) and the composition of language modules (e.g. [168, 35, 24]). Extensions that are designed and planned once on a pragmatic and semantic level may be easily transferable to other modelling languages in order to re-apply them or examine correspondence (referring to [26, 42]).

12.2.1 Methodical Consideration of Pragmatics and Semantics in EML Extensions

Most considerations of EMLs and EML extensions are limited to syntactical investigations [68, p. 5], [40, p. 432], [33, p. 52]. Bjeković et al. [40] state that "language engineering efforts typically overemphasise the challenges of mechanical manipulation of models and neglect the variety of contexts, users and purposes for which models need to be created" [40, p. 432]. This overemphasis of the syntax corresponds with weak methodical support for justifying and designing EML extensions, e.g. in the case of BPMN [59]. Insufficient procedural transparency during extension design [33, p. 50] as well as missing semantic extension specification can be observed [33, 9, 73].

Current literature provides very few papers on methodical support for pragmatics and semantics within extension design [9]. While a few papers address syntactical issues (e.g. [48, 42]), semantics are rarely considered and pragmatics are completely omitted (cf. [9]). This leads to an imprecise understanding of the actual extension need and the negligence of the user perspective [40, p. 432], which causes missing procedural transparency and lack of externalisation [57].

12.2.2 Consequences for Extension Design

It is therefore essentially important to focus on pragmatics and semantics first instead of solely stressing syntactical structures and technical implementations. Particular grammatical rules and specifications rather come from the semantics, i.e. the intended user scope, but not from the language itself, as EMLs do not exist for reasons of self-purpose. As mentioned above, it is important to envision that EMLs are artificial means for the solution of different, usually stakeholder-specific business problems, which requires the detailed consideration, analysis, and representation of pragmatic use cases or usage scenarios, the users themselves as well as the particular context or situation. More precisely, it is important to understand (a) native concepts of a problem scope and (b) rather solution-oriented concepts.

This indicates a certain *translation of pragmatics into intended semantics*, which shapes expectations of prospective EML users according to their needs and requirements. Only such a detailed domain analysis and understanding may guide the selection of appropriate modelling languages or the extension of an EML (according to [139, p. 41], [195, p. 212]). Semantics are important for the explication of intended interpretations of syntactical constructs (cf. [101]) and for the justification of particular elements of the language [194, 65]. The specification of semantics is hence essential for inter-subjective communication and appropriate language usage.

Consequently, it is necessary to outline a method, which supports extension decisions based on intended pragmatics and semantics. This covers the specification and representation of pragmatics and semantics at first. Based on a set of specified semantic constructs (derived extension need), final syntactical extension mechanisms can be derived.

12.2.3 Assumptions and Limitations

With respect to the multi-faceted complexity of semantics and the level of underinvestigation regarding pragmatics, different limitations and assumptions should be made with regard to the topical scope of this work.

The focus of the following considerations lies on the general *explication* and *specification* of the intended pragmatics and intended semantics as well as their consequences for particular extension types. This work therefore focusses on the representation of pragmatics and semantic structures and omits concrete consensus efforts as well as different conceptual, epistemological, and linguistic questions at the current stage due to their notable level of complexity [41, 9]. In particular, deeper aspects in the context of semantics like lexical issues [196] or differing personal conceptualisations (e.g. [197, 198]) have not been discussed yet. This includes rather philosophical discussions and consensus-finding within sociological processes [62, p. 68], as each modelling language is inherently bound to the community using it [199].

For practical reasons, it is further assumed that the syntactical specification of addressed EMLs is complete and consistent (although there are some syntactical issues in BPMN, for instance [112]).

The following discussions base on the position of Methodic Constructivism, which combines Ontic Realism and Epistemological Idealism (cf. [200]). Consequently, semantics can be understood as the individual, subjective re-construction of an ontically perceptible area of discourse [57, p. 36].

13

Structure for Extension Procedure

Motivated by the above considerations on the adequacy of EMLs, it is necessary to construct a method for the systematic and transparent elaboration of EML extensions based on a profound analysis of the intended pragmatics and intended semantics of prospective EML users. Further, the well-guided derivation of potential EML extensions need has to be addressed. Therefore, the technique of *Method Engineering* is conducted.

Method Engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of IS [201]. A method can be defined as systematic derivation of specified objectives based on a dedicated set of single tasks, decisions, and guidelines (according to [47]). *Situational Method Engineering* describes the application of Method Engineering to particular situations that are specified by project characteristics [202]. This covers customisation, tailoring, and reconfiguration of existing methods and their elements, e.g. method chunks [203, 204, 205]. Consequently, Situational Method Engineering also comprises method extensions and the technique should therefore be conductible to the extension of EMLs as parts of EM methods [46]. However, a straightforward adaptation is difficult, as Method Engineering rather investigates generic, domain-independent and language-independent methods [201, 206]. Dedicated applications to EMLs or BPMN are missing so far (cf. the discussions in [26, 22, 39]).

In the field of domain-specific modelling, only Frank [25] discusses the methodically guided design of DSMLs in the EM context. The author presents a multistage method for the design of DSMLs, while alternative approaches rather focus on implementation-oriented DSLs (e.g. [207, 208, 209]). They do not match the characteristics of EM-related modelling languages as outlined in Sect. 4.1. De Kinderen & Ma [56] propose a step-wise process that is composed of the identification of stakeholders, the analysis and description of their concerns, the derivation of required information, and a final language fit in order to justify respective design decisions [56, p. 31]. Considering the stated lack of operationalisation, it is necessary to provide a pragmatic, fine-grained and detailed method for justifying and designing BPMN extensions (according to [3, p. 23]).

Table 13.1 presents the proposal for the method. The outlined method bases on previous works in the context of BPMN extensions, namely the BPMN4CP extension [55, 76]. These publications initially propose and revise an extension method for BPMN that is composed of three main steps: Domain analysis, extension prepa-

ration, and the design of the extension meta model [76]. Thereby, the last step conducts and extends the BPMN syntax extension method of Stroppi et al. [48].

This macrostructure is generally reused, but refined below. In particular, the stepwise derivation of relevant semantic constructs based on an analysis of pragmatic use cases is revised and extended in order to justify extension need on a semantic, namely, ontology-driven level (according to [98]) instead of solely addressing the syntactical dimension [48].

In order to create concise, detailed, and transparent guidance, each step of the extension method is specified by several properties: A certain *main stage* describes a comprehensive container for potential *sub-stages* that are understood as refinements in order to provide precise interfaces for input and output artefacts. A sub-stage is characterised by expected *inputs*, relevant *methods* and techniques, as well as a generated *output*. Output is understood as respective artefacts that are reused in subsequent stages as input that is somehow treated or transformed within particular methods. Thereby, results of previous work and above outlined artefacts are integrated (e.g. extension types). The attribute *focus group* describes roles which are the most relevant in single stages. Therefore, prospective EML users and EML engineers (method engineers) are differentiated.

Below, the single main stages are presented and discussed in detail. Thereby, especially the context analysis phase is investigated, i.e. Use Case Analysis (Sect. 14), Requirements Analysis (Sect. 15), and Concept Analysis (Sect. 16). Syntactical realisations are omitted due to space limitations. Respective preliminary studies both on implementations within the current BPMN meta model (e.g. [74, 55]) as well as implementations within a potentially revised BPMN meta model (e.g. [59, 76]) are directly discussed in literature.

Main Stage	Sub-Stage	Focus Group	Input	Methods	Output
Context Analysis	Use Case Analysis	Users	Use Cases, Purpose Types	Domain Analysis (Literature Analysis, Interviews, Observations etc.)	*Intended Pragmatics*
	Requirements Analysis	Users	Use Cases, Purpose Types	Requirements Engineering	*Requirements Specification* • Conceptual Req. • Capability Req. • User-related Req.
	Concept Analysis	Users, Engineer	Requirements Specification	Semantic Analysis	*Semantic Specification* • Material Constructs • Formal Constructs • Perspectives
Correspondence Analysis	Semantic Comparison	Engineer, (Users)	Semantic Specification, EML Meta Model	Ontological Comparison	*Extension Profile*
Extension Preparation	Mechanism Selection	Engineer	Mechanism Repository	–	*Proposed Mechanisms*
Extension Definition	Abstract Syntax	Engineer	Selected Mechanisms, EML Meta Model (MOF instance)	Mechanism Application	*Abstract Syntax of Extended EML*
	Concrete Syntax	Engineer	DD Instance	–	*Concrete Syntax of Extended EML*
	Semantics Specification	Engineer	Extended Meta Model	Meta Model Annotations	*Semantics of Extended EML*
Extension Labelling	Pragmatics Specification, Master Data	Engineer, (Users)	Use Cases	–	*Exemplarily Application Cases*

Table 13.1. Semantics-driven extension procedure

14

Use Case Analysis

14.1 Motivation and Fundamentals

The analysis of the context of use, the early identification of addressed language purposes, relevant user groups, and the expected utility of an EML are essential for an adequate language design [20, 210, 211, 40]. Moreover, the formulation of modelling use cases enables the assessment and selection of modelling languages in general. This stage aims to identify the purposes which should be fulfilled by a modelling language. It has to be clarified what should be done with the language. Although it is crucial to investigate, define, and explicate the expected utility and underlying purposes of EMLs [212, 25], there is little research on this issue [56], and a general lack of appropriate description means for pragmatics has to be criticised [40]. So far, only isolated approaches from different directions within EM-related research exist. These approaches are briefly considered below in order to represent the state of the art and identify applicable means for the specification of pragmatic use cases in the context of EMLs.

14.2 Related Work

Patig [70] addresses the evolution of conceptual modelling languages, especially ERMs and Petri Nets, and describes different reasons for language metamorphosis [70, pp. 38ff, 58-62]. However, the author omits any enterprise-related aspects and all investigations are limited to generic modelling approaches.

Frank [25] emphasises the relevance of understanding the domain during the development process of DSMLs. The process should start with a clarification of the covered domain scope as well as the intended purposes [25, p. 140].

Bjeković et al. [22, 40] investigate reasons for EML application in order to better understand the occurrence of EML extensions and variants. The authors emphasise the relevance of pragmatics for EM by focusing on the individual purposes of EML stakeholders as the actual drivers behind language adaptation. Explicating the modelling purpose is therefore essential important [22, 40]. However, dedicated pragmatic types are not presented.

Karagiannis [68] introduces AMME, the Agile Modelling Method Engineering approach. AMME adapts the agile paradigm and aims to timely respond to changing modelling requirements, which implicates an iterative evolution of modelling

languages. Language-changing requirements should be derived from use cases [68, p. 7], but it remains unclear how these use cases can be specified and elaborated. Also, final consequences in regard to the meta model level are not documented.

Within the AMME project scope, Visić et al. [69] introduce a domain-specific language, i.e. a method for agile method engineering. The extensibility of the DSL itself is considered as a key aspect within method development. Base for DSL design is a creation phase which aims to elaborate requirements that are obtained from questions, task identification, and respective decisions [69, p. 5]. Therefore, methods like content analysis or structured interviews are recommended in order to understand particular domain-specific knowledge. The AMME approach is generally similar to the idea of extensible EMLs. However, AMME rather focusses the compositional design of DSMLs and it is rather tool-centred [213]. Modelling standards like BPMN are not considered. The authors further propose text-based modelling based on ENBFs, while our approach strictly bases on diagrammatic meta modelling.

De Kinderen & Ma [56] proclaim value-oriented language engineering that focusses on the actual user value, which is generated or intended by a certain EML. Therefore, Goal-Oriented Requirements Engineering (GORE) is adapted, which encompasses the initial identification of stakeholders, the clarification of their intentions, and the subsequent identification of their intentions [56, p. 2].

Despite these rather generic approaches, methodical support for early stages of language design and extension design can be found in Burwitz et al. [43], Braun & Schlieter [55], as well as Braun et al. [76]. These works demonstrate the specification of scenarios as base for the identification of language requirements and concepts [43, 55, 76].

To the best of our knowledge, only one paper provides types for EM pragmatics on a generic level: Motivated by considerations of different types of semantics, Braun [57] proposes four different types of usual modelling purposes: *Documentation*, *Documentation and Automation*, *Automation*, and *Model Analysis* [57, p. 39]. These usage types are applied below.

14.3 Input, Method, Output

At first, it is important to identify current, potential, or prospective *stakeholders*, i.e. the perhaps abstract mass of potential users dealing with the same class of problems (according to [68, p. 5]). These people intend to derive some utility or expected value by creating enterprise-related models with an EML (cf. [35, 210]). However, further separation into *human stakeholders* and *technical stakeholders* might be helpful in order to assess automatisation capabilities of single use cases early.

The creation of models is driven by underlying objectives and intends the final realisation of respective *purposes* (according to [186]). However, such purposes and objectives are often difficult to explicate and specify, as users are usually uncertain about the language capabilities, on the one side, and language engineers are not familiar with the considered area of discourse, on the other side (according to [25, p. 134]). It is therefore promising to elaborate and describe expected *use cases* or application scenarios at first in order to make the user intention more concrete and explicit. Modelling use cases should describe what model users actually aim to do

with models created by a modelling language and what tasks should be conducted for accomplishing a certain goal.

Use cases can be derived twofold. First, by conducting empirical research, e.g. user interviews, questionnaires, or observations of daily business actions (e.g. [214]). It might be also useful to explicate challenges or to design prototypical models [25, pp. 143ff]. Thereby, it is important to especially examine relevant tasks, decisions, and underlying business objectives (referring to [69]). Second, theoretical and conceptual analyses can serve as an information source, e.g. content analysis or literature reviews. Use cases can be specified textually or by UML use case diagrams [107]. In the next step, the gained set of *use cases should be grouped* and perhaps integrated in order to identify similar ones. This facilitates the identification or homogenisation of stakeholder groups with differing perspectives, which is important for later semantic differentiation [211, 126, 56].

Then, the *purpose types* of each consolidated use case has to be determined in order to prepare the subsequent derivation of requirements and semantic constructs. One of the following purposes should therefore be selected [57, pp. 38ff]:

- **Documentation**: A particular modelling use case is motivated by the idea of documentation and describing business-related aspects for inter-subjective, primarily non-technical communication in order to manually derive respective follow-up actions based on designed models. Models therefore do not serve as a direct part of an action system, but rather provide documentation foundation. Documentation is suitable for *organisational engineering* (i.e. IS design [3]), *systems engineering* (i.e. application systems design [215]), and for *communication between stakeholders* (according to [4]). There is consequently no need for invariant model interpretation, semantics are hence material and prone to the above mentioned issues of subjective interpretations. This purpose type is therefore purely *descriptive*.
- **Automation**: A modelling use case with an underlying automation purpose determines an invariant and hence formal interpretation of models [25, pp. 144ff], implicating formal semantics with precise interpretation rules. This covers the application of models in transformation chains within *Model-Driven Engineering* (MDE [216]), or as dedicated parts within application systems (*models at runtime* [217]). Models are thereby part of artificially created action systems and effect the change of object states within these systems. Models are therefore not used manually and this purpose type is therefore *prescriptive*.
- **Documentation and Automation**: Some modelling use cases may affect both pure documentation aspects as well as potentially automatable aspects [67]. For instance, a process model needs to be interpreted manually (e.g. within clinical process modelling [77]), but some aspects of the entire model might be processed automatically, e.g. vertical resource flows or material flows. Consequently, hybrid semantics as a combination of material semantics and formal semantics appear as a promising instrument. However, a concise decomposition into documentation or automation should be examined for reasons of clarification and the current lack of appropriate description means for hybrid semantics [67].
- **Model Analysis**: This purpose type reflects the analysis or assessment of respectively designed models. There is consequently no direct or indirect relation to the area of discourse, as the models themselves are perceived as objects under

analysis. For instance, a use case for the calculation of model-related key figures can determine this purpose type [33, p. 45].

- **Model Usability**: Similar to the last purpose type, model usability covers use cases which refer to the manual application of models and their somehow perceived usability. This type represents a special case, as it does not create any concrete value, but rather supports the way of model creation by enhancing the perceived manner of model usage [28, p. 400].

This stage finally elaborates the *intended pragmatics* by formulating intended use cases and respective modelling purposes informally or by applying use case diagrams. Further, different stakeholder groups might be ascertained. Consequences for subsequent stages are twofold: First, the identified purpose types determined the required semantic types and may indicate specific requirements. Second, the use cases may provide first conceptual requirements, i.e. semantic constructs.

15

Requirements Analysis

15.1 Motivation and Fundamentals

The extension method is basically motivated by the requirements-driven Design Science Research (DSR) approach that postulates the investigation, classification, and description of requirements as central instruments for the rigorous analysis and conception of innovative and relevant artefacts. This should remedy existing DSR shortcomings like missing procedural transparency, insufficient comprehensibility, and unclear demarcation from professional practice [79].

The elaboration of extension need is driven by pragmatics and semantics as illustrated in Sect. 12.2. Requirements should therefore explicate the needs of prospective EML users in order to specify the required language capabilities. Requirements are understood as *transmitters between pragmatics* (explicated by rather informally specified use cases) *and semantics* (finally explicated as detailed semantic constructs). Previously introduced use cases and corresponding purposes are therefore decomposed into more specific requirements, which are the base for the later derivation of detailed semantic constructs.

In general, a requirement stands for a desired capability of a system or of an artefact that is needed for solving a specific problem or reaching an objective [218]. In respect to EMLs and especially with regard to the introduced *pragmatics and semantics first* approach, requirements represent two things:

- The intended conceptual *expressiveness* of an EML, which is finally realised by a specification of semantic constructs.
- The expected *capabilities and functionalities* of an EML in order to satisfy user-specific tasks, purposes or use cases.

Requirements should therefore support the elaboration of affected conceptualisations about things of an area of discourse, which are the base for semantic constructs (what is the language about). This refers to original constructs of the domain, explicated by EML users. This type can therefore be seen as a set of rather *problem-oriented requirements* (according to [219]).

Requirements should serve as the base for the later derivation of respectively needed semantic constructs in order to realise particular functionalities using these constructs. This stands for the way of solving expected issues and consequently refers to constructs that have to be designed and specified by EML engineers, not primarily

by EML users. This type therefore stands for rather *solution-oriented requirements* (according to [219]).

Explicitly, the stated requirement types only address the semantic dimension of a language. Specific syntactical obligations are not considered at this point, but rather intended as the final result of a continuous translation process from pragmatics to semantics and, finally, to the syntax.

15.2 Related Work

Analysing, defining, and explicating requirements for different kinds of modelling languages are perceived as essential topics within method engineering, but current research largely omits this topic so far and lacks the provision of appropriate and commonly accepted expression instruments [212, 25, 56, 68]. It has to be bemoaned that very little research is conducted on Requirements Engineering in the context of EMLs and EML extensions [55, 56], which corroborates the considered over-emphasise of syntactical and rather technical issues (cf. [40]) that might amplify the limited dissemination of EMLs in general (according to [220, 3]).

To the best of our knowledge, only the below stated papers consider the role of requirements within the process of developing modelling languages. The papers are briefly summarised in order to represent the state of affairs and enable the adaptation of single aspects.

Frank [25] presents a method for the systematically guided design of DSMLs and emphasises the significance of generic and (domain-) specific requirements [25, p. 142]. Generic requirements primarily refer to syntactical and technical aspects within the design stage, while specific requirements cover semantic constructs within the targeted domain. They can be elaborated by the analysis of usage scenarios or by prototypically designed models, for instance [25, pp. 143ff]. The analysis of usage scenarios should correspond to the identification of respective key terms of the domain that are captured in glossaries [25, p. 146]. Additionally, the level of automatisation has to be determined [25, p. 143].

De Kinderen & Ma [56] propose the definition of goal-oriented language profiles for the composition or design of EMLs in order to satisfy particular user purposes and create expected user values. Language profiles describe inherent capabilities, extra capabilities, and a list of limitations. Inherent capabilities represent the intended semantics and can be derived by analysing documents, books, and papers, for instance. Extra capabilities refer to non-functional expectations, e.g. it terms of language usability [56, p. 14].

Karagiannis [68] addresses the importance of general, domain-specific, and run-time-related requirements within the AMME framework. The author therefore proposes syntactical, semantic, functional, design-time-related, and run-time-related requirements as foundations for the design of modelling languages [68, p. 5]. Although the framework mainly focusses on technical implementations, the adaptation of semantic and functional requirements seems to be promising in order to differentiate between required conceptualisations within the domain and expected functionalities based on these concepts.

Visić et al. [69] emphasise the direct relation between requirements and semantic specifications. Similar to Karagiannis [68], the authors implicitly differentiate two kinds of semantic constructs: The first type of semantic constructs comes directly from modelling stakeholders [69, p. 1], while the second type is derived "indirectly from required functionality (e.g. model queries, run time applications)" [69, pp. 1ff]. Thereby, the proposed modelling language requirements represent semantic domain constructs. A differentiation between both types in the sense of translating less formalised requirements into detailed semantic constructs with a particular shape has been neglected so far.

15.3 Input, Method and Output

15.3.1 Requirements Classification

The initial motivation and the consideration of the state of the art as well as the consideration of use case types in Sect. 14 indicate a differentiation of requirements according to rather vocabulary-oriented requirements and rather functionality-oriented requirements. It is therefore reasonable to classify EML requirements in order to prepare the structured derivation of semantic constructs in subsequent stages. For reasons of systematisation, the generic artefact requirements classification of Braun et al. [79] is adapted and refined in regard to the elaborated peculiarities in Table 15.1 (cf. [79, p. 144]).

The adaptation of the following requirement types from Braun et al. [79] was omitted: *Feature-related requirements* are ignored, as they represent rather general requirements which are already specified through use cases in our extension method. *Non-functional requirements* are ignored, too. Non-functional requirements represent user-related requirements in terms of the nature and quality of an artefact [79, p. 144]. With regard to the intended elaboration of semantics in the extension method, this would implicate a qualitative evaluation of the final semantics in regard to the personal semantics of EML users [101]. This could require a special consideration of reducing conceptual and lexical ambiguity issues, for instance [191, 221]. Due to the complexity of these issues and missing research in this field [57], this dimension is hidden for now. Nevertheless, some non-functional requirements (in the sense of usability) should be represented by user-related requirements.

The specification of all requirement types can be realised by conducting the KAOS method [222] or the i* method [130]. Thereby, it is important to specify the respective types of requirements. Generally, additional research regarding a modelling language for EML design seems to be reasonable (cf. the AMME DSL approach [69]).

15.3.2 Derivation of Requirements from Use Cases

Below, the derivation of requirements from respective use case types is elaborated. Each derivation is briefly described in terms of analytical tasks that have to be conducted.

Documentation → Conceptual Req., Constraints: Each use case should be analysed regarding its key terms and potential relations between them in order to identify relevant conceptualisations about things and phenomena of the area of

Requirement Type	Semantics	Adaptation for EMLs
Functional	Specification of intended capabilities or features of an artefact.	Functionality in the scope of the extension method is understood as (a) conceptual expressiveness and (b) set of functional capabilities of a language. This requirement type is therefore divided into the two sub types (conceptual and capability-related requirements).
– **Conceptual**	(not specified in [79])	This type represents required *domain constructs*, i.e. conceptualisations within a particular area of discourse. It covers all things about which communication should be possible. During the extension method, these requirements are refined or translated into semantic constructs in order to represent the original conceptual scope. Conceptual requirements are rather inconcise, less detailed or specified. Hence, they represent purely problem-oriented requirements (cf. [219]). Conceptual requirements can be derived by the analysis of key terms within use cases, for instance (according to [25]).
– **Capability-related**	(not specified in [79])	This type rather represents expectations in terms of *realising specific tasks* or *operations*. Requirements of this type focus on the solution space in the sense of artificially designed constructs that are not in the mind of EML users. They hence represent rather solution-oriented requirements (cf. [219]). For instance, these requirements may cause the later specification of specific, perhaps formal semantic constructs by the language engineer in order to realise intended purposes.
Constraints	Laws, regularities, standards (referred as contextual requirements [79, p. 144])	Specific circumstances and contextual borders, which have to be respected (in addition to initially formulated use cases). Contextual requirements can determine additional conceptual requirements or may indicate capability-related requirements.
User-related	User preferences	Expectations according to usability of an EML. This type primarily affects rather soft goals and user expectations (cf. [130]). It may implicate the later derivation of capability requirements. User-related requirements typically address final language reductions, the modification of the concrete syntax, or some other annotations for general better understanding, i.e. some kind of supporting constructs.

Table 15.1. Adapted requirements types

discourse, which have to be captured in the expressive scope of the EML. This determines conceptual requirements. Additionally, constraints have to be identified and described. Constrains implicate additional conceptual requirements and the definition of specific rules on these concepts, which may further cause capability requirements. Such restrictions can be referred to as static semantics [67] or syntactic semantics, too [40].

Automation → Capability-related Req., Constraints: Automation use cases determine capability-related requirements in order to realise capabilities for formal interpretation within a language, which may require some kind of working on the above mentioned concepts of a language. This concern encompasses the derivation of requirements for a later specification of rules as well as the specifica-

tion of additional constructs for particular algorithms (according to [14, 8]). These requirements are therefore indirectly elaborated by method engineers.

Documentation and Automation: As mentioned above, a decomposition of these use cases should be attempted in order to derive respective requirement types.

Model Operations → Capability-related Req.: Model operations also implicate capability requirements and are hence treated as outlined above. However, they are not necessarily automatable, i.e. the derivation of formal limitations and restrictions is not mandatory, but possible.

Model Usage → User-related Req.: Explicated use cases for model usage can be directly mapped to user-related requirements.

15.3.3 Output

This stage yields a set of requirements which have a higher level of detail in comparison to the initial use cases. In particular, three main groups of requirements can be differentiated:

1. Conceptual requirements
2. Capability-related requirements
3. User-related requirements

These requirement classes act as input for the subsequent derivation of semantic constructs that can be used for semantic correspondence analysis.

16

Concept Analysis

This stage aims to evolve detailed semantic constructs from requirements in order to enable a reasonable comparison with the original EML. It is therefore necessary to extract and derive *relevant constructs with their characteristics* first. It is thereby important to differentiate material semantic constructs and formal semantic constructs due to their different representation techniques. Particular *description instruments* have to be selected accordingly. With respect to the limited scope of this paper, the following considerations will primarily focus on material semantic constructs. The deeper analysis and integration of formal semantics constructs in the context of EMLs is only outlined and shifted to further research.

16.1 Motivation and Fundamentals

16.1.1 Relevance of Semantics and Current Issues

Despite obviously high relevance of semantics for EML application and dissemination (cf. [195, p. 212], [22, p. 440]), semantics are still under-investigated and research on EML semantics on the meta model level is largely omitted [73]. Several authors explicitly criticise the imprecise semantic justification and specification of conceptual modelling languages, EMLs, and their extensions [62, pp. 67-69], [63, p. 19], [64, p. 108], [65, pp. 690, 706], [66, p. 485], [33, p. 52].

Poor semantic specifications of EMLs hamper language comprehensibility and may cause severe issues in terms of inter-subjective communication due to differing interpretations and conceptualisations (e.g. [223, 40, 101]). Potential problems may occur if meta model elements are differently understood by language users, but respective mismatched understandings remain undetected [190]. It is further possible that superficially similar meta concepts are erroneously treated as being the same, ignoring fine-grained but relevant conceptual differences [101]. Rudimentary semantic specifications also hinder the emergence of commonly agreed interpretations of meta concepts [101, pp. 3, 66, 104] and complicate the general discussion about them [9]. The stated issues for EMLs also apply for EML extensions, which are usually designed in an ad-hoc manner, omitting any rational and transparent justification process (cf. [64, p. 108]). This has negative effects on the entire extension design process [40, p. 432].

Reasons for insufficient semantic considerations are manifold [41, 57]. One reason might be the rather formal original of some EMLs (e.g. BPMN), which often impli-

cates a naive-realistic understanding of semantics. It means that a particular meta model element is supposed to be interpreted in exactly one way, which is generally questionable in semi-formal modelling languages and their reliance on material semantics [98]. Recipients of EMLs are mostly human beings, which in turn causes the immanent issue of subjectivity in regard to understanding a meta model differently [224, p. 6], [14, p. 112], [101, p. 5]. Technically spoken, the semantic mapping of a syntactical construct to a particular semantic domain construct differs and leads to the above mentioned issues [46, 198].

Although objectification and particular semantic harmonisation within a group of EML users are emphasised as worthwhile objectives [225, p. 25], [25, pp. 13-14], current research lacks in the provision of techniques for describing EML semantics in an appropriate manner. Also methodical guidance for detecting and eliminating misunderstandings is omitted and investigations on the integration of material and formal semantics are rather immature [67]. On the other hand, several semantic description approaches have evolved prototypically over the last years, e.g. UEML [226]. But they are neither well-disseminated nor notably applied in meta models. Instead, semantic specifications are still limited to rather simple and ambiguous textual statements (cf. [73]). Research on semantics in the context of EMLs can hence be assessed as rather immature, isolated, and less integrated (referring to [62, 210, 9]).

16.1.2 Consequences for Extension Method

Considering the state of affairs reveals two major implications for the extension method: At first, it is necessary to derive and specify semantic constructs as detailed as possible in order to avoid misunderstandings and facilitate transparent communication on EML semantics. This objective covers the initial extraction of key terms from the elaborated set of requirements and their incremental refinement subsequently. Due to the lack of appropriate and integrated methods, an integration of parts of the DSML design method of Frank [25] and the EM-related ontology language of Opdahl et al. [227] is discussed below. Therefore, ontologies are employed as central instruments for the representation of semantics. Their basal features are briefly outlined below.

16.1.3 Ontologies for Semantics Representation

Ontologies are well-established means for explicating domain knowledge and enabling semantic annotation of meta models in order to express stipulated semantics of modelling languages [100, 223]. Ontologies are also applied within early constructive stages of a language design process in order to guide the integration of domain concepts in meta models [98, 55]. Despite its high relevance for several fields of application within EM, the understanding of ontology characteristics differs significantly [183].

Ontologies therefore need to provide at least the following basal types [228]: Concepts (characterised by mutual properties), relationships (between concepts and properties), axioms (constraints and rules for concepts), and instances as exemplarily realisations.

16.1.4 From Requirements to Semantics Specification

The elaboration of required semantic concepts should be driven and structured by decisive meta types in language engineering, namely concepts, perspectives, and notations. Concepts refer to the contextual and functional vocabulary as discussed in Sect. 15. Perspectives refer to stakeholder-specific or purpose-specific aggregations of concepts or simple groupings of common understandings. Concepts and perspectives will finally provide the base for the derivation of the abstract syntax. Notations refer to any kind of user-related requirements in regard to the shape and appearance of modelling elements. Below, the determination of single types is discussed based on the above introduced requirement types.

16.2 Ontological Constructs from Conceptual Requirements

Conceptual requirements are the foundation for material semantic constructs, which is generally considered as EML semantics [57]. The specification of these constructs should be realised by combining the DSML design method of Frank [25] and the UEML approach [229].

The DSML design method is primarily adapted in regard to the initial identification of key terms as a reference for required domain concepts, as it is important to decide whether a term is suited to be incorporated in a language [25]. Therefore, Frank [25] proposes four essential criteria: (1) invariant semantics, (2) relevance, (3) variance of type semantics, and (4) instance of type intuitive [25, pp. 148ff]. The assessment of these criteria then enables a decision for or against an inclusion as meta type [138, p. 100]. The stated criteria are adapted as a set of guidelines through which a certain term has to go through.

While the externalisation of semantic constructs remains rather vague within the DSML method of Frank [25], we propose the application of a more detailed approach for their specification. Semantic constructs are thereby specified with UEML in order to enable a multi-faceted and possibly complete externalisation of conceptualisations about phenomena, which should support inter-subjective communication and transparent comparison with original semantics from EMLs [226, 229, 66].

16.3 UEML for the Representation of Material Semantics

16.3.1 Introduction

Ontologies for representing material semantics should provide concept types for EM-related aspects instead of purely generic concepts in order to facilitate the integration of specific concepts into the EM domain. Ontologies should further not be completely determined and fixed, but extensible. Also integration with syntactical constructs is preferable [100]. As mentioned above, a dedicated and commonly applied ontology for material semantics is missing. The Bunge-Wand-Weber (BWW) ontology is frequently conducted for the ex-post analysis and evaluation of modelling languages, but reveals several shortcomings as elaborated in literature [194, 230, 231]. Other

authors proclaim the application of generic ontologies like the Web Ontology Language (OWL) [232, 100, 55], but their specific application and appropriateness for EM remain vague.

Another approach is the Unified Enterprise Modelling Language (UEML). UEML serves as a well-defined modelling approach for the specification of multi-faceted, EM-related ontologies and their annotation on the meta model level [226, 229, 66]. UEML reuses some basal concepts from the BWW ontology [192, 233] in its core and extends them with structural and functional concepts from the IS and EM domain in order to provide a reference ontology that facilitates the integration, exchange, and migration of EMLs [226, p. 103], [229], [66, p. 488]. Therefore, UEML provides a reference ontology, called Unified Enterprise Modelling Ontology (UEMO), which specifies more than 200 concepts for the domain-independent description of EM-related syntactical constructs as languages are.

UEML aims to reduce semantic gaps in EM in order to foster the emergence and management of model-driven organisations by a standardised, integrative, and evolvable approach [226]. In this context, the authors of UEML stress the inherent significance of semantic constructs, i.e. ontological representations (according to [226, p. 101]) and emphasise that the meaning of a modelling language is primarily driven by its decomposition into atomic semantic constructs (cf. [66], referring to Frege's Compositionality Principle [234]). The underlying paradigm hence complies with the introduced *semantics first* principle.

Consequently, the well-structured and expressive UEML is proposed as a convenient technique for the representation and specification of intended semantics in the context of the extension method under consideration. The architecture and main components of UEML are briefly introduced below.

16.3.2 Architecture

UEML is framed by three main components as presented in Figure 16.1 [66, 235]: The abstract syntax (*language layer*), ontological scenes (*construction description layer*), and the underlying UEMO ontology (*ontology layer*).

Language Layer: The language layer covers all grammatical constructs of the EML that should be analysed and semantically annotated. This indicates that each UEML analysis bases on already existing syntactical constructs. These constructs are semantically enriched in two subsequent steps as stated below.

Construction Description Layer: The idea behind UEML is to describe each syntactical construct not only as one-to-one mapping to a certain ontological concept, but rather as representation of a certain *phenomenon* that is composed of several roles (cf. [236]). A syntactical construct is described by a *state of affairs* (scene) that is composed of one or more represented phenomena, i.e. classes, properties, states, and transformations [66, p. 487]. It is therefore possible to specify the meaning of a language construct through decomposition into its atomic parts of phenomenon representation [237, p. 9], [66, p. 490], [235, p. 56]. The modelling of a scene therefore externalises the underlying understanding of language concepts and serves as first reference to the perceived real-world (referring to [226, p. 104], [229, p. 165]). The derivation and specification of represented classes, represented properties, represented states, and represented transformations is guided by answering four essential

Dimension	Question	Outcome
Static	Which **class(es) of things** is the construct intended to represent?	• Classes of things
Static	Which **properties** is the construct intended to represent?	• Intrinsic properties • Relations • Complex (subproperties) • Laws (structural and behavioural restrictions, i.e. states and transformations)
Behavioural	Which **state** is the construct intended to represent?	• States of classes • State laws (restriction of class properties in state)
Behavioural	Which **transformation** is the construct intended to represent?	• Transformations • Transformation laws (from state to another)
Usage	Which **instantiation level** is the construct intended to represent?	• Type • Instance • Both
Usage	Which **modality** is the construct intended to represent?	• Descriptive (objectively perceived facts) • Normative (rules, prohibitions etc.) • Intentional (goals, intentions etc.)

Table 16.1. Description of a state of affairs by answering six essential questions

questions in order to evolve the manifestation of a language construct [226, 229, 66]. The relationship types between the concepts have to be elaborated in addition, e.g. generalisations, sequences, or some other kind of impact [66, p. 490]. Also the kind of things as well as the intended kind of usage can be specified (cf. [235]). Table 16.1 summarises the stated questions and outlines the represented phenomena which shape a particular scene. The evolved scene represents the actual *interpretation and meaning* of a referred syntactical construct, indicating the semantic mapping between syntax and semantics.

Ontology Layer: The above specified representation types are then mapped into a generally pre-defined set of basal ontological concepts from the EM and IS domain, the UEMO ontology. The representation types are mapped to ontological constructs and hence operationalised according to their type [66, p. 490]. UEMO therefore provides the following four taxonomies (e.g. [226, 235]):

- Taxonomy of Classes (e.g. *AnyThing*, *UnchangingThing*, *CoupledThing*)
- Taxonomy of Properties (e.g. *AnyProperty*, *IntrinsicProperty*, *ComplexProperty*)
- Taxonomy of States (e.g. *AnyState*, *StableState*, *UnstableState*)
- Taxonomy of Transformations (e.g. *AnyTransformation*, *Event*, *Execution*)

Each taxonomy is hierarchically structured and allows the specification of taxonomical, i.e. ordered, relationships (e.g. specialised, precedes) and non-taxonomical, i.e. unordered, relationships (e.g. define, restric, effect, prestate of, poststate of; according to [66]). This enables generalisation hierarchies and precedence hierarchies within the ontology (cf. [226]). Single taxonomies are further interrelated in order to represent the interdependencies between the underlying represented concepts.

Each ontological analysis finally leads to the specification of ontological concepts from UEMO, which enables language comparison and language across migration via this standardised set of ontological constructs. Potential points of subjective variance can hence be inferred as follows: First, the scene representation of a certain

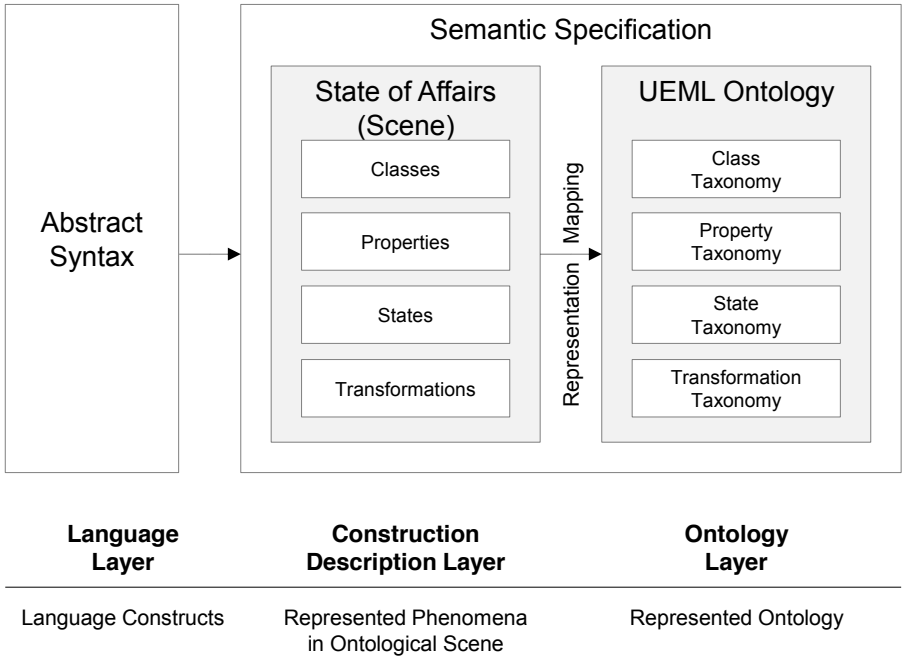

Fig. 16.1. UEML architecture

syntactical construct may vary. Second, identical scenes may be differently mapped to ontological constructs from UEMO. The resulting differences are relevant for the later consideration of semantic comparisons.

16.3.3 Application and Adaptation

The UEML is intended to be used within the analysis stage in order to express what a particular modelling language should represent. Current literature remains rather vague on this aspect and considers only classes and their attributes on a very generic level, omitting any concrete representation instruments (e.g. [25, 55]). UEML hence provides a conceptual and methodical frame for structuring the rather imprecise area of discourse and fostering its model-based externalisation with a specific modelling notation as well as with the ontology representation standard OWL, for instance (cf. [229]).

In any case, applying UEML for the extension method reveals some peculiarities. At first, UEML strongly bases on the existence of a well-specified EML, indicating already existing syntactical constructs that can be described (e.g. [66, pp. 492ff]). However, the syntax is naturally missing within the proposed extension approach, which hampers the straightforward conduction of UEML construction guidelines.

Specification of Scenes as Externalisation of Intended Semantics

Instead of considering the language syntax here, we rather proclaim scenes as externalisation instruments for describing the required language capabilities. Therefore, the UEML scene modelling notation can be applied (referring to [227, p. 4]). It is consequently necessary to translate the ascertained requirements into respective scenes. Therefore, the construction guidelines of Harzallah et al. [66] may be

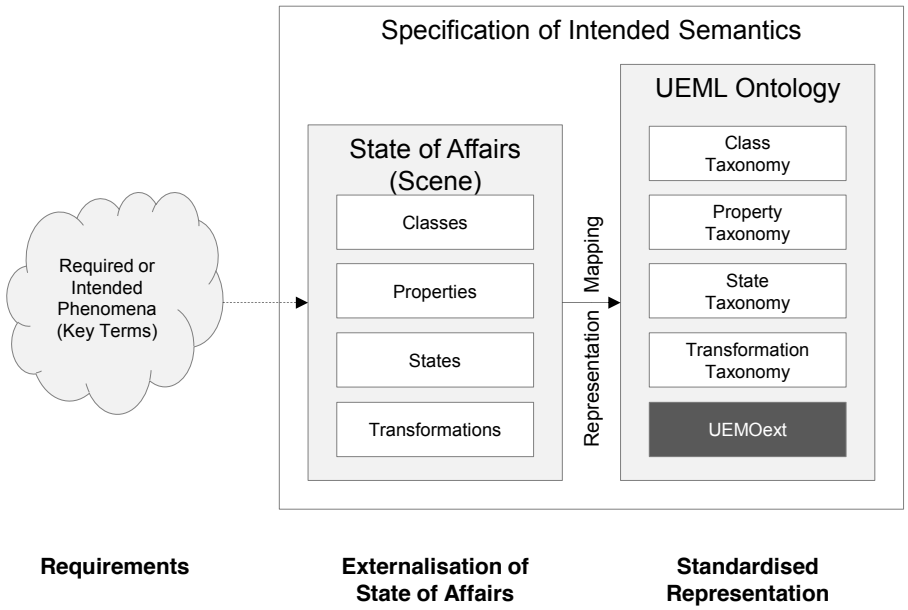

Fig. 16.2. Adaptation of the UEML approach for specifying the intended semantics

conducted. For instance, if further information on scenes is missing, then the central phenomenon for representation should be a class and respective representation types should be annotated to this central phenomenon (cf. [66, p. 494ff]). The identification of key terms from conceptual requirements seems to be promising for this step (according to [25]). The specification of represented phenomena hence externalises the intended material semantics, which are covered by requirements. The scene specification is depicted in the middle of Fig. 16.2. Figure 16.3 represents the scene modelling notation that is inferred from [227].

Standardising via UEMO Ontology and its Extension

After scene modelling, each representation phenomenon has to be mapped to the ontology layer in order to translate and hence operationalise it for language across comparison and analysis. It can hence be understood as a certain translation task that has to be fulfilled in order to provide communication and concept comparison.

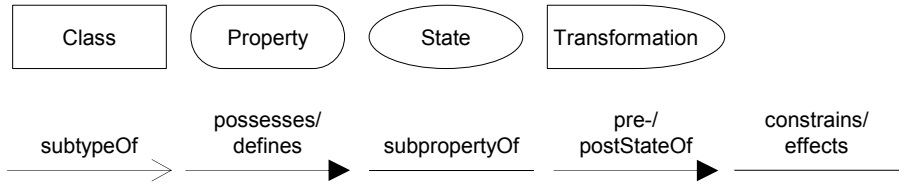

Fig. 16.3. Notation for modelling scenes [227]

The current UEMO version bases on a reverse engineering of GEMLs and PSMLs, namely BPMN 1.x, ARIS, UML, KAOS or IDEF (cf. [226, p. 108], [229, p. 173], [235, p. 50]). The UEML authors themselves note that the initial reuse of some major BWW concepts as well as the concentration on process modelling languages cause

an over-emphasise of active and interactive classes of things, while organisational aspects and social actors are rather under-represented and hence difficult to express [235]. With regard to the peculiarities of the EM domain, also other topics like material flows, resource structures, or business goals may become relevant for ontological considerations. There is consequently an inherent necessity of adding further ontological concepts to UEMO (according to [226, p. 108]).

Opdahl et al. [235] suggest an adaptation of potentially useful approaches like value modelling or language-action perspectives. In contrast, we proclaim the situational extension of UEMO with potentially existing domain ontologies from the area of discourse. This circumstance is represented by the introduced *UEMOext* ontology part in the right side of Fig. 16.2. Although UEMO is explicitly designed for enriching itself (cf. [226, p. 63], [66, p. 485]), its technical realisation has mostly been left open so far [235]. The authors propose the initial selection of a taxonomy and the subsequent integration of an additional ontological concept by applying respective taxonomical mechanisms. Consequently, there is a need of decomposing intended semantic constructs into classes, properties, states, and transformation, as well as an extension of the UEMO itself.

16.4 Semantic Comparison with UEML

Due to the intended representation of material semantic constructs with UEML, already the initial phase for semantic specification raises the issue of conducting semantic comparisons in order to clarify potential conceptual mismatches.

16.4.1 Related Work

The evaluative comparison of meta model semantics is only sparsely investigated in literature. Considerations on semantics are limited to the comparison of syntax and semantics in the light of assessing their correspondence to a given ontology (ontological analysis [193]).

Wand & Weber [192] proclaim four essential comparison types: *Construct deficit*, *construct redundancy*, *construct overload*, and *construct excess*. Construct deficit represents missing representation of an ontological construct in the modelling grammar. Redundancy stands for the mapping of multiple constructs to one ontological construct. Overload represents a mapping from a grammatical construct to multiple ontological constructs. Excess represents those grammar constructs, which have no ontological mappings [192].

Fickinger & Recker [238] expand this framework by the concepts *distinct construct redundancy* and *specialised construct redundancy*. The first describes that one ontological construct maps two different and unrelated grammatical constructs. A language construct has therefore different interpretations, which might cause misunderstandings. Specialised construct redundancy describes the mapping of one ontological construct to different grammatical constructs, while these constructs constitute parts of a generalisation. This type is less problematic, as it keeps most ontological features [238].

According to the semantic annotation of meta models, Guizzardi [223] introduces the criteria *soundness*, *completeness*, *lucidity*, and *laconicity*. Soundness claims

for the general existence of an ontological construct. Completeness describes that each ontological construct is represented in the syntax. Lucidity stands for the unambitious representation of an ontological construct and laconicity demands that a particular ontological construct is referenced only once [223].

The presented approaches seem to be generally applicable for a rapid appraisal of the appropriateness of an EML. However, the approaches involve the risk of ignoring particular knowledge that is encapsulated or referred by the intended semantics, as it directly maps to the language syntax. These mappings remain implicit, which is detrimental for procedural transparency and comprehensibility. It is more reasonable to conduct the initially proposed *semantics first* approach and focus on semantic comparisons.

Only a few research papers explicitly address the implementation of semantic comparisons. Gehlert & Esswein [231, 239] elaborate a formalised ontological analysis, leading to the specification of equivalence, similarity, and differences. Pfeiffer [240] discusses several model conflict types, e.g. homonym conflicts and synonym conflicts. Opdahl & Henderson-Sellers [241] introduce the *element of* comparison result type, which implicates that a grammar construct constitutes an element of an ontological construct. *Subtyping* represents generalisations between ontological constructs [241]. Anaya et al. [226] introduce several correspondence types, which are discussed below.

Based on these preliminary studies, the following fine-grained correspondence types are consolidated with a special focus on subsequent constructive actions. Thereby, only material semantic constructs are covered.

16.4.2 Correspondence Types in UEML

Semantic comparison in the context of UEML is driven by comparing modelling languages in order to estimate migration strategies. Language comparison is therefore elevated to a comparison of ontological scenes [226, p. 100]. The analysis is thus also driven by finding respective correspondence between language constructs. This leads to the following correspondence types which cover the comparison of ontological scenes [226, p. 105]:

1. **Equivalence**: Both scenes are identical. This means that the represented phenomena as well as their relations are congruent, indicating a replacement of respectively related syntactical constructs without information loss.
2. **Containment**: One scene is completely covered by another scene. This means that the covering scene specifies additional represented phenomena, while the covered scene serves as a valid excerpt or subset of the covering scene. If one scene covers multiple scenes, it could be described as a merge of these scenes. And a decomposition of this scene into the covered scenes is then possible. This enables a replacement during model-to-model translation procedures, for instance.
3. **Generalisation**: One scene (sub-scene) can be generalised by another scene (super scene). More precisely, respective sub/super relations between represented phenomena exist [66, p. 490]. Generalised constructs may enable model transformations with particular information loss [226, p. 105].
4. **Complex**: This type stands for the combination of containment and generalisation. Thus, it represents a hybrid type indicating various similarities.

5. **Overlapping**: This type represents partial equivalence. More precisely, some represented phenomena are mutual, while others are exclusively owned by one scene.

Further specification of conducting and elaborating these correspondence types in detail is so far missing. A certain procedure can be inferred inductively from the presented example in Anaya et al. [226]. The goal concepts from KAOS and GRL are therefore described within the construction description level in order to find represented classes and properties. These representation constructs are then mapped to the UEMO ontology, enabling a construct-wise correspondence check. However, the authors state that there is need for further research in terms of scene comparison and the establishment of a correspondence typology [226, p. 105].

16.4.3 Proposal for Correspondence Typology

We therefore outline a more fine-grained and detailed typology that aims to generically specify comparisons within the construction description layer. The typology bases on the assumption that both the scenes as well as their ontological representations serve as atomic units of analysis enabling the best possible externalisation of material semantics.

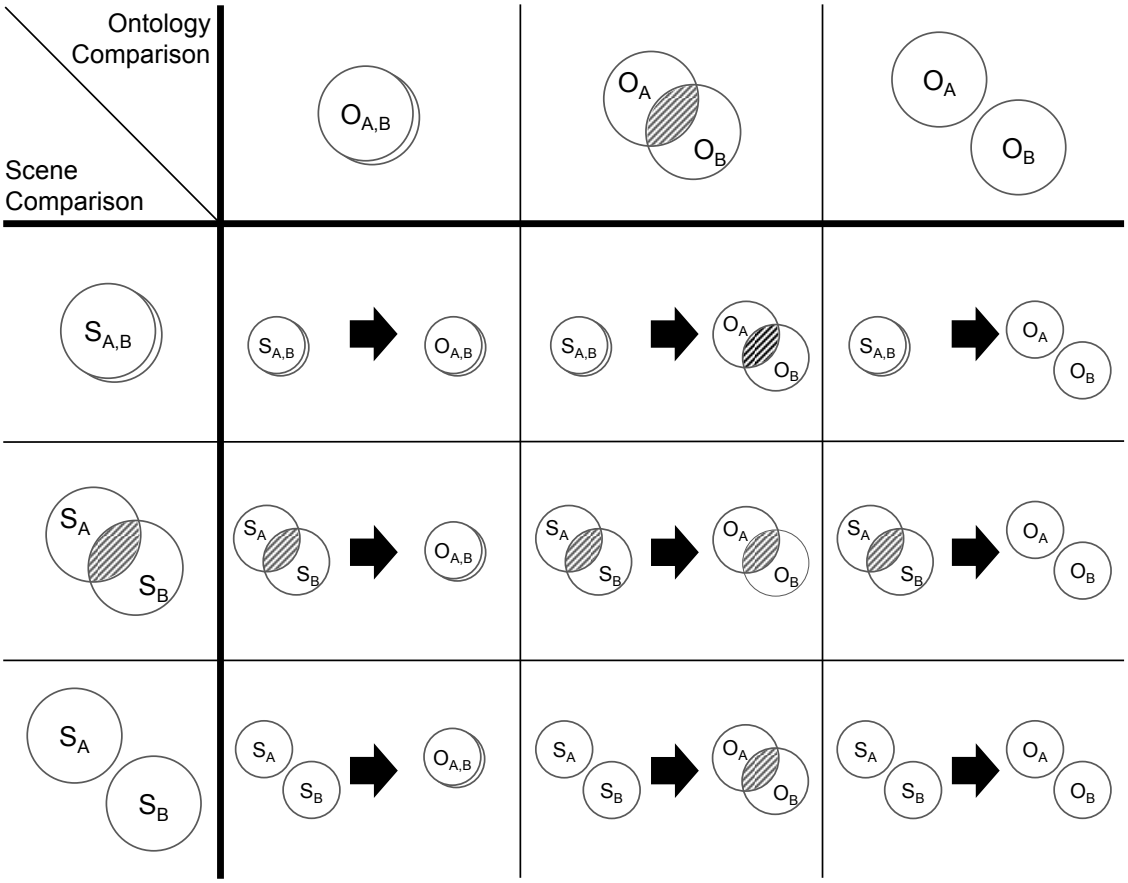

Fig. 16.4. Correspondence typology representing scene comparison and ontology comparison

Evaluating the difference between scenes – independent from any syntactical construct – in fact determines an investigation of the underlying phenomena (what is) and an investigation of terms referring to it (how is it labelled).

The first issue can be tackled by comparing the respectively defined and mapped ontological constructs. The second issue rather focusses on the scene itself in order to find homonyms that may cause semantic misunderstandings, for instance. Consequently, the model structures of two scenes as well as the model structures of the ontological representation are compared, leading to an assessment of *Equivalence*, *Similarity*, or *Difference* (according to [239]).

We therefore propose a two-dimensional framework in order to establish a correspondence typology. The framework is composed of a dimension describing the comparison of respective scenes and a dimension describing the accordingly determined comparison of respective ontologies. We thereby assume that the result of the ontology comparison also covers the respective mappings between scenes and ontologies (despite the fact that mappings between identical scenes and identical ontologies may differ, for instance).

- **Equivalence** stands for congruency between two artefacts, in regard to their elements, their relations and their particular labels.
- **Similarity** stands for missing equivalence, but congruence in regard to some elements. Naturally, the extent of congruence may differ, often leading to a situational assessment of the degree of similarity. Different degrees of similarity are especially relevant for the derivation of syntactical consequences (cf. Sect. 18.2).
- **Difference** stands for the lack of any commonalities between two artefacts.

Ontology Comparison **Scene Comparison**	**O1: Equivalence**	**O2: Similarity**	**O3: Difference**
S1: Equivalence	S1O1: Congruency	S1O2: Differing Degree of Understanding	S1O3: Homonym Conflict
S2: Similarity	S2O1: Potential Synonym	S2O2: Indifference	S2O3: Potential Difference
S3: Difference	S3O1: Synonym	S3O2: Indifference	S3O3: Difference

Table 16.2. Basic correspondence typology

Table 16.2 presents the fundamental structure of the framework. Single framework elements are discussed within the analysis scenarios below. The elaborated framework should support the comparison of initially derived scenes during requirements engineering. And it should facilitate the later comparison with already existing concepts of an EML. The in-detail consideration of each framework element is organised in accordance with the scene comparison.

Analysis Scenario 1: Equivalent Scenes

This framework layer aims to clarify whether equivalently described phenomena (things of the area of discourse) are in fact understood equivalently. From a pragmatic point of view, this is essential for becoming aware of semantic conflicts (homonymous understandings) or perhaps user-specific differences in the level of detail. In accordance with Table 16.2, the following types can be divided:

- S1O1: **Congruency**. The ontological representation is equivalent, indicating an identical understanding of one and the same state of affairs.
- S1O2: **Differing Degree of Understanding**. The understanding of one and the same state of affairs differs, either horizontally (differently annotated characteristics), vertically (different generalisation levels), or both. Particular difference types are introduced in Table 17.2. This type may indicate the derivation of user-specific perspectives, for instance.
- S1O3: **Homonym Conflict**. One and the same state of affair is understood differently, leading to homonymous understandings that represent user-specific misunderstandings. The lack of commonalities is therefore critical and has to be resolved during extension design.

Different fine-grained considerations and **Types of Similarity (S1O2)** are introduced below. The reflected types are reconsidered for correspondence analysis in Sect. 17. We therefore refer to a *need scene* (need in the sense of required demand) and an *original scene* as well as *need ontology* and *original ontology* in order express correspondence analysis between the needed capabilities and the capabilities of the original language.

- *Containment*: The same thing (or state of affairs) is meant, but in the original it is semantically specified in a more detailed way. Consequently, the need ontology is always covered by the original ontology (Fig. 16.5a).
- *Coverage*: The same thing (or state of affairs) is meant, but it is understood in a more detailed way in the need ontology. Consequently, the original language lacks in the provision of some semantic constructs (Fig. 16.5b).
- *Specialisation*: Within the need, something more specialised is meant. This indicates that constructs of the need ontology are either equivalent to constructs of the original ontology or they are specialisations of them (Fig. 16.5c).
- *Generalisation*: Within the need, something more general is meant. This indicates that constructs of the need ontology are either equivalent to constructs of the original ontology or they are generalisations of them (Fig. 16.5d).
- *Complex (with Specialisations)*: Something more specific is meant, whereby some parts of the referred state of affairs are understood in a more specialised way, while the other parts are covered by the original ontology (i.e. containment and specialisation, Fig. 16.5e).
- *Complex (with Generalisation)*: Something more general is meant, whereby some parts of the referred state of affairs are understood in a more general way, while the other parts are covered by the original ontology (i.e. containment and generalisation, Fig. 16.5f).
- *Overlapping (Complete)*: Something is understood in the same manner, but additional characteristics are needed. A few, but not all ontological constructs of the need ontology match the constructs of the original ontology. These constructs thereby represent the complete set of constructs from the original ontology (i.e. complete coverage, Fig. 16.5g).
- *Overlapping (Partial)*: Something is only understood in the same manner to a certain degree. A few, but not all ontological constructs of the need ontology match the constructs of the original ontology. These constructs thereby do not

represent the entirety of constructs from the original ontology (i.e. partial coverage, Fig. 16.5h).

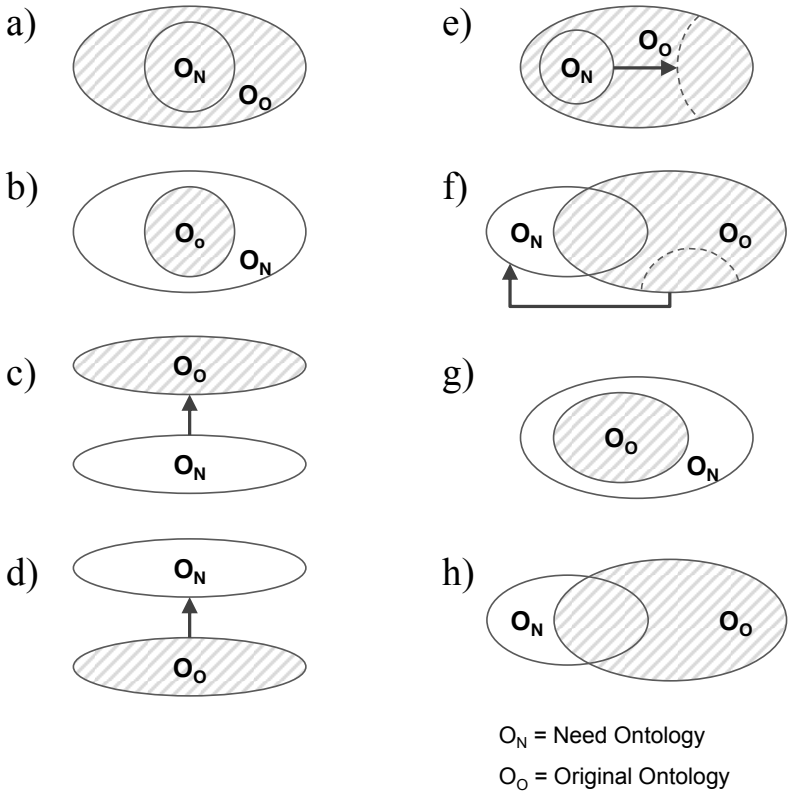

Fig. 16.5. Types of ontological similarity

Analysis Scenario 2: Similar Scenes

This framework layer represents the scenes with similarities, e.g. all cases of containment, generalisation and partial overlapping (cf. [226]). Despite the inherent challenge of assessing similarities, the analysis should foster the identification of potentially identical sets in order to become aware of actual or "de facto" matches of understanding. We therefore propose the following types:

- S2O1: **Potential Synonym**. Similar scenes are understood in the same way. It is consequently reasonable to assess the degree of similarity in order investigate whether it is (a) rather an equivalent or (b) rather a synonym. This step, of course, represents a situational decision in accordance with the respective elements.
- S2O2: **Indifference**. Similar scenes are understood in a differing detailed kind, but have some overlapping. Further consideration in the sense of changing the type to potential synonym (S2O1) is only reasonable if the similarity is perceived as large.
- S2O3: **Potential Difference**. Partly different scenes are understood in a completely different way, without any overlapping. Therefore, it is necessary to investigate the degree of scene similarity in detail. If the similarity is rather large, then

the similarity has to be treated as a homonym conflict (S1O3). If the similarity of the scene is rather low, then it should be treated as completely different for reasons of clarity (S3O3).

As stated above, the case of similar ontologies may require further discussion to identify actual strong similarities in order to better understand the kind of similarity. The comparison of the ontological similarity should therefore indicate and justify a probably derived assumption that the similar scenes under consideration actually stand for identical phenomena in their core. We therefore introduce the following types of similarity for differing scenes:

- *Containment*: Probably the same phenomenon is meant, but it is referred to differently and is understood in more detail in the original ontology. All constructs from the need ontology are therefore contained in the original ontology.
- *Coverage*: Probably the same phenomenon is meant, but it is referred to differently and is understood in more detail in the need ontology. All constructs from the original ontology are therefore contained in the need ontology.
- *Specialisation*: Probably something more specialised is meant and referred to differently, as each construct from the need ontology serves as a specialisation of the original ontology.
- *Generalisation*: Probably something more generalised is meant and referred to differently, as each construct from the need ontology serves as a generalisation of the original ontology.
- *Complex (with Specialisation)*: Probably something more specific is meant and referred to. Thereby, something more specific is understood both as a specialisation of the original ontology as well as annotating additional characteristics. Exactly these characteristics have to be examined in detail. If they extend the original ontology significantly, then it is rather unlikely that the same thing is meant. In case of a slight extension, it is probably the same.
- *Complex (with Generalisation)*: This case is similar to the *Complex (with Specialisation)* type, but the need ontology is more general in some parts. The same considerations have to be conducted in order to assess real difference.
- *Overlapping (Complete)*: This type basically corresponds to the *Coverage* type, but defines additional characteristics within the need ontology, which have to be considered as in the *Complex* type, for instance.
- *Overlapping (Partial)*: Only some ontological constructs are shared between the need ontology and the original ontology. In this case, in-depth analysis of the excluded constructs is necessary. If the non-shared constructs are non-contradictory and if they are relatively small in comparison to the overlapping, then both scenes probably mean the same. If the non-shared constructs are contradictory or have a rather low proportion in relation to the entire ontology, then they should rather be treated as differences.

It is important to note that the fine-grained analysis of similarities is limited to the ontological level, although it might be reasonable to conduct it on the construct description level, too. However, this conduction is omitted, as the construct description layer provides rather little structural information and, even more impor-

tantly, the comparison of scenes themselves is barely realisable without analysing the ontological representation.

It is further important to mention that each comparison of similarities and differences serves as a manual decision, especially in terms of qualitative assessment (strengths of overlapping, e.g.) as well as quantitative assessment (ration between shared and unshared ontological constructs, e.g.). As already stated in Anaya et al. [226], further research on operationalisation of such similarity degrees is absolutely necessary.

Analysis Scenario 3: Different Scenes

This framework layer covers the analysis of differently represented phenomena. This layer is important for the identification of synonyms and for the justification of clear differences. The followings types are differentiated:

- S3O1: **Synonym**. Differently referred state of affairs are understood the same. It is therefore reasonable to treat them as being the same.
- S3O2: **Indifference**. Differently represented state of affairs are similarity described in the ontologies. A deeper analysis of the degree of difference could enable a classification as a de facto synonym, if reasonable. Otherwise, such similarities should rather be assessed as clear differences.
- S3O3: **Difference**. There are neither commonalities in the scene nor in the ontology. Consequently, totally different things are covered.

Similar to the above mentioned analysis scenarios, it is reasonable to analyse the differences between ontological scenes in order to identify potential synonyms, i.e. largely similar ontologies that justify de-facto equivalence of them. However, we propose to only consider those, which allow a precise commitment and omit any further manual decision. Therefore, the following types from Sect. 16.4.2 can be reused: *Containment*, *Coverage*, *Specialisation*, and *Generalisation*.

16.5 Justifying and Modelling Intended Semantic Constructs

After elaborating an appropriate description format in Sect. 16.4, the determination of respective scenes and their covered phenomena remains open. Therefore, the DSML design method of Frank [25] is adapted and partially integrated with UEML in order to (a) reuse the elaborated procedure of Frank [25] and (b) leverage the manifold means of expression from UEML. Specifically, the intended semantic constructs are not only externalised by terms of respective "concepts", but as scenes with different kinds of represented phenomena. Although syntactical constructs are missing, the specification guidelines of Harzallah et al. [66] as well as the modelling notation of Opdahl et al. [227] can therefore be leveraged.

Consequently, the key terms from the conceptual requirements should be extracted and refined accordingly in order to elaborate the scenes first. In accordance with Frank [25] and Harzallah et al. [66], we suggest using classes of things as the starting point for analysis. This analysis can be conducted by introspective,

for instance [25]. Then, the phenomena of scenes have to be mapped to the UEMO ontology or its extended version in order to operationalise them.

According to Frank [25], the identified scenes need to satisfy two requirements. They should have the same meaning across all application areas and it should be possible to define the essential meaning of each scene respectively. These requirements are necessary in order to avoid language overload and enable ontologically clear extensions. The particular meaning is thereby understood as externalisation of scenes by their ontological constructs in UEMO. The term "essential" stands for the ability to distinguish one scene from another scene in a significant way. The higher the levels of semantics, the more interpretations are excluded and the more information is required about potential instances. It is therefore necessary to identify potential mismatches or misunderstandings based on the ontological analysis of scenes. Frank [25] therefore proclaims four analysis steps: invariant semantics, variance of type semantics, relevance, and types intuitively. These are conducted and adapted in the light of UEML below.

16.5.1 Invariant Semantics

The criterion of invariant semantics claims an invariant interpretation of terms across different user groups and fields of application [25]. With respect to the elaborated adaptation of the UEML approach, it is therefore necessary to examine the ontological representation of equivalent (S1) and very similar (S2) scenes in order to identify potential misunderstandings in the sense of homonyms and largely different understandings. At first, equivalent scenes have to be analysed and the following results are considerable:

- *Congruency (S1O1)*: The ontological representations of the scenes are identical. They are hence treated as one scene. The semantics are consequently treated as invariant.
- *Differing Degree of Understanding (S1O2)*: The ontological representations of the scenes are not identical, but similar. According to the stated similarity types, the following actions are recommended:
  - *Containment and Coverage*: In this case, identical scenes are understood with a different level of detail. It is recommended to use the larger ontological representation if there are no contradictions. Then, the creation of a user-specific perspective in the sense of a reduced view on the entirety of the referred ontological concepts should be contemplated.
  - *Specialisation and Generalisation*: Identical scenes are understood with a different level of abstraction. It is recommended to use the more general ontology and rename the more specific one in order to express its role. This more specific ontology may serve as a potential sub-type within later syntactical considerations.
  - *Complex and Overlapping*: As mentioned above, these types require special attention. For both types, it is recommended to examine a merger of ontologies in order to create a common understanding. Then, user-specific perspectives can be created later. In case of any contradictions or a low degree of similarity, both ontologies should not be merged and rather treated as being different (see below).

- *Homonym Conflict (S1O3)*: The ontological representation of one scene does not have any overlap with the other scene and there are consequently no semantic similarities. Despite the same terms, the scenes are interpreted in a totally different way and hence prone for misunderstanding (cf. [101]). It is therefore necessary to rename one scene and propagate this renaming within the addressed user group in order to separate them.

In a second step it is necessary to examine scenes with a high degree of similarity, i.e. these two scenes only differ marginally. A discursive analysis (with user groups) should then elaborate if the regarded scenes can be seen as one. Then, the above introduced rules for the case of equivalence can be conducted.

It becomes obvious that even the early analysis of same or similar scenes reveals some syntactical hints, especially regarding sub-types and required perspectives. These aspects should be recorded in order to apply them during extension design.

16.5.2 Variance of Type Semantics

Frank [25] further claims for variance of type semantics. Variance of type semantics, i.e. the semantics of intended language constructs, implicates a noteworthy semantic distance between particularly identified scenes in order to avoid redundant language types. However, concrete operationalisation remains unclear (also in the context of UEML). We therefore recommend comparing the ontological representation of each scene with all other ontological representations. A particular distance is understood as a comparison of these ontologies. It is intended to identify synonyms (S3O1) and potential synonyms (S2O1), as these types indicate low semantic distance. Two result types are consequently possible:

- **Same meaning**: Two different scenes are characterised by the same set of ontological constructs, causing a synonymous understanding. It is then necessary to record these synonym structures in order to keep track of them during extension design. Synonyms should be covered by semantic specifications of potential extension constructs.
- **Slightly differing meaning**: The analysis may also reveal very similar sets of ontological concepts that would negate the criterion of significant distance (cf. [25, p. 147]). They are rather seen as potential synonyms. In this case, reconsideration is necessary in order to find out whether different ontological phenomena are indeed referred to or whether one phenomenon is referred with slightly different characteristics. In the first case, we recommend keeping both scenes differently, but treating them in different perspectives later in order to emphasise the different meaning. The latter case should be treated as one scene having differing attributes in single perspectives. Similar to the above mentioned synonyms, the (de facto) synonym should be recorded and the entire concept should be treated as a candidate for perspectives.

As mentioned earlier, it is necessary to elaborate powerful mechanisms for ontological comparison on the meta model layer in order to provide better means for assessing the degree of similarity, for instance [226].

16.5.3 Relevance

Frank [25] proclaims the integration of only those concepts into a language which are perceived as relevant. Also other authors emphasise that only necessary domain concepts should be considered in order to avoid over-complicated languages [211]. In contrast to the stated works, we rather intend to relax this criterion, as the actual relevance of a determined requirement is impossible to satisfactorily assess from a language engineer point of view. This position might be seen as naive, but we take the expressed conceptual requirements and constraints as granted and solely recommend reconsidering single concepts of the above mentioned list by EML users. Potential pre-exclusion of concepts should be avoided in order to enable the user-centric creation of value.

16.5.4 Instances as Types Intuitively

This criterion aims to investigate, whether a particularly identified ontological scene may act as a later language type for instances, which are intuitively perceived as types themselves [25]. Consequently, the level of classification has to be evaluated as outlined in Sect. 8.4.5.

Frank [25] suggests discursive evaluations for this purpose, as it depends chiefly on reflection and introspection to find out whether a term that does not allow for further instantiation can still be regarded as an abstraction [25]. The usage dimension of each ontological scene can be consulted (cf. Table 16.1). Scenes with a classification as "instance" should be excluded at this point. Scenes with a classification as "type" can be retained. Scenes with a classification of "both" should also be retained and need to be considered as potential Clabjects within language specification (cf. Sect. 8.4).

16.5.5 Output

The elaboration of material semantic constructs brings out a consolidated set of ontological scenes, which are respectively decomposed into UEMO ontologies. UEML scenes and ontologies can be modelled by a lightweight domain-specific approach and with the help of the ontology modelling language OWL [229, 227]. Further, some scenes should be annotated with advice for required perspectives or advice for respective synonym relations. It is further necessary to keep track of respectively referred ontological scenes by referencing them within the scene specification.

16.6 Ontological Constructs from Capability-Related Requirements

The above introduced concepts focus on directly expressed terms of stakeholders that shape the addressed, somehow natural area of discourse. In contrast, capability-related requirements cause the early derivation of concepts for realising specifically intended capabilities within an EML. In contrast to conceptual requirements, language engineers themselves serve as originators of such requirements (similar to indirect requirements in [69]). Consequently, the set of already specified semantic

concepts can be extended arbitrarily. Elaborated concepts can thereby refer to previously introduced concepts in order to represent dependencies, as capability-related requirements often cause a certain analytical layer (also referred as solution space [139]) that enables some kind of invariant working "on" domain concepts (referring to [67]).

This aspect especially encompasses the consideration of formal semantics and approaches for their representation [14, 8, 67]. More specifically, formal semantics requires the specification of determined and invariantly interpretable syntax transformations. In addition to these formal semantic constructs, it might also be necessary to integrate further material semantic constructs for design reasons.

The analysis of formal semantic constructs differs remarkably from material semantics. The initial analysis of formal semantic constructs like material semantic constructs is therefore contingent for at least two major reasons: First, the language engineer himself designs them and additional language engineers are required for a critical analysis, which serves as a requirement that is indeed rational but unrealistic in engineering projects. Second, especially the above discussed variance and instance criteria are contingent, as formal semantics invariantly map syntax to other completely known sign systems, omitting any interpretative variance [14, 8, 67]. Their final relation to real-world phenomena, however, depends on at least one intended mapping to material concepts (cf. [67]). Such a mapping generally serves as an integration point between material and formal semantics, representing a particular interpretation.

There are consequently two challenges that must be tackled. It is first necessary to find a mechanism for the specification of formal semantics, as current literature is rather split over this topic. For instance, there is no commonly accepted specification for workflows in BPMN [15, pp. 435ff]. Second, an appropriate integration mechanism with material semantic constructs has to be designed in terms of representing their dependencies. Additionally, the specification of a required perspective for these analytical or strictly functional tasks should be considered if not already done within the requirements stage (i.e. as perspective for non-human actors). This is important for the avoidance of potential amalgamation between conceptual and rather technical semantics, which hampers separation of concern [33].

16.6.1 DMM for the Representation of Formal Semantics

Formal semantics requires the specification of two formal syntax definitions and a concise mapping specification between them [242, pp. 34ff], [139, p. 37]. Despite the clarity of the challenge and frequent implementations, generic approaches are largely missing, as demonstrated in Hausmann [63].

To the best of our knowledge, only one approach enables the generic specification of formal semantics and mappings, the Dynamic Meta Modelling (DMM) approach of Engels et al. [243]. DMM was originally designed for the specification of behavioural semantics of state charts and activity diagrams, but its theoretical foundation is promising for adaptation. The application of DMM is constituted of three major steps: In the first step, the addressed meta model has to be slightly extended [243]. Then, the behavioural semantics are formally specified and transformed into a typed graph structure. Particular syntactical expressions are therefore coupled with

pre-conditions (i.e. events), transformation rules (i.e. actions), and postconditions [244]. Consequently, a particular model state is proved according to preconditions. If these conditions are met, then the action can apply until the model satisfies the postconditions, leading to a new valid model state [243, 245].

16.6.2 Outline and Discussion of a Possible Implementation

The idea of the DMM approach should be generally applied for the extension method, but the concrete application causes several issues. First and foremost, there is the problem of missing syntactical constructs that are yet undefined within the analysis stage. Further, the introduced ontological constructs are contingent for DMM, as formal semantics require a mapping from one formal syntax to another formal syntax.

This issue could be tackled if an ontological scene is interpreted as a precursor of a later specified syntactical construct, assuming a later one-to-one mapping. The scene is therefore operationalised from the perspective of formal semantics, ignoring the underlying ontological references as outlined on the right side of Fig. 16.6. Technically speaking, an additional interpretation of the scene is added in order to correspond to the specific invariant interpretation a technical actor grasps as semantics. This could represent integration between formal and material semantics (according to [67]). Therefore, the original semantics has to be considered in detail.

In case of *Model Analysis* (i.e. the intended on-top analysis of models that represent material semantics), an additional analytical layer with formal specifications should be introduced. The particular types of the model elements are hence extended with an additional role in order to grasp them formally as input data for some computations. In this case, the target syntax has to be specified separately, since the state of a particular model will not be changed.

In the case of *Automation*, only those scenes which are explicitly considered as automatable, can act as input for formal operations. Regarded scenes can be labelled as automatable [25]. Also a deeper analysis of *StateLaws* and *TransformationLaws* (cf. [235]) may allow further inferences. In this case, the "pre state" syntax corresponds with the target "post state" syntax, i.e. particular transformations affect the same state space.

However, the concrete realisation of the outlined architecture remains tricky and is not trivial. Fig. 16.6 presents a potential implementation. Formal representation constructs operationalise particular ontological scenes in order to define a specific formal state. Additionally, respective preconditions, postconditions and transitions have to be defined in a generic manner, as a specified modelling grammar is undefined at this stage. It seems to be reasonable to introduce an additional taxonomy into UEMO in order to provide formal operators and respective concepts. These elements need to be transferred to the final syntax later.

In its current version, we suppose to specify formal semantics with pseudo code that refers to elements of the scenes. The pseudo code needs to explicitly express particular preconditions, postconditions, and state transitions. Also the components of formal states need to be described. The entirety of these elements is understood as *formal semantic construct*.

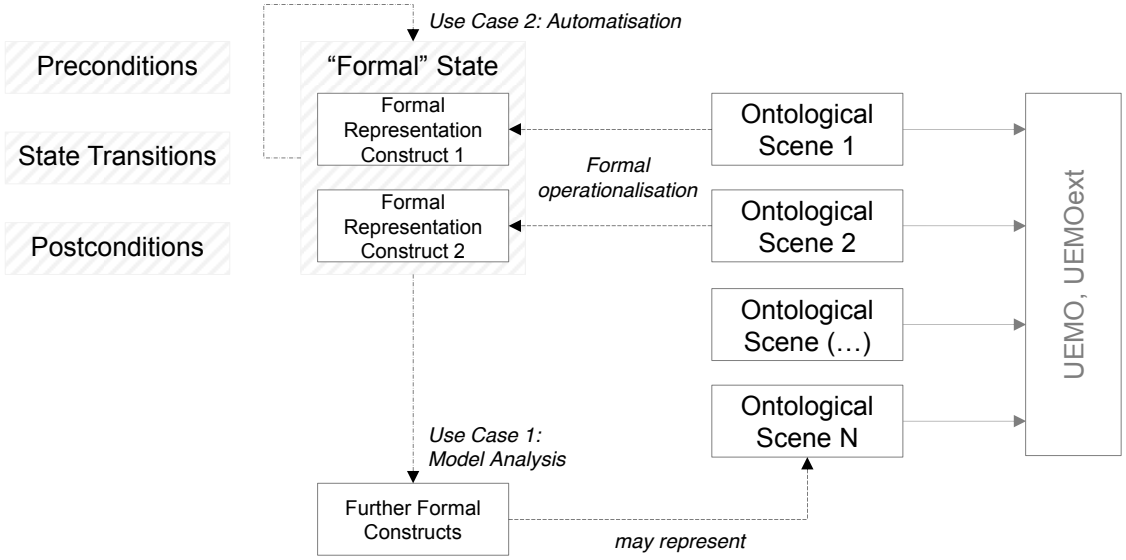

Fig. 16.6. Outlining a potential integration between material and formal semantics by mapping DMM and UEML

Nevertheless, it has to be stated that the early specification of formal semantics and especially its integration with material semantics is a complicated task that calls for massive further research (cf. [67]), which cannot be conducted within this paper due to space limitations. However, the introduced framework should provide guidance and orientation for respective work. Further consideration in this work will hence focus on material semantics.

16.7 Perspectives and User-Related Requirements

16.7.1 Perspectives

Perspectives are central means within EM, as they are feasible for representing user-specific or purpose-specific direction of views or foci on the entirety of all language concepts [5]. Perspectives may correspond to the way of thinking about particular concerns from different angles (according to [96]) and they may also be simply used for reducing complexity of the entire vocabulary. They can hence be understood as purpose-specific, pre-defined filter mechanisms (according to [47, 44]).

Perspectives should already be pre-specified at this stage in order to facilitate the management of different semantic concepts or correspond to the initially identified stakeholder groups. The derivation of perspectives can be consolidated as follows:

- **Perspectives from different foci (*conceptually* driven)**: As stated above, *interpretations of concepts can slightly differ* in regard to some facets. These differences may be reflected in perspectives, if respective concepts are integrated with other concepts that are represented within the final language.
- **Perspectives from different stakeholders (*user*-driven)**: According to the specified use cases, a *grouping* of respectively derived concepts could support the realisation of intended purposes by focusing only on relevant ones and en-

abling complexity reduction. Of course, this type can correspond with conceptually driven perspectives.
- **Perspectives from differentiation between material and formal semantics (*formally* driven)**: *Formal semantics* should require the definition of separate perspectives for reasons of separation of concern.

Derived perspectives should be *stored in a separate list of perspective candidates*. Each perspective should be labelled appropriately and all related material semantic constructs (Sect. 16.5) and formal semantic constructs (Sect. 16.6) need to be listed.

16.7.2 User-Related Requirements

User-related requirements may cause the consideration of specific notational elements in the modelling language, as these requirements cover soft goals like specific icons or colours that are not closely related to the semantic scope of a language. Therefore, it is advisable to specify these requirements in terms of the following aspects:

- Which concept should be represented in which perspective?
- What symbols, icons, colours and fonts should be utilised?
- How should the arrangement of these elements be realised and what is the relation to other graphical elements?

Sketches and prototypical diagrams can facilitate the derivation of respectively required graphical elements for the graphical representation (cf. [102, 246, 25]). Graphical user expectations are subsequently considered for potential extensions of the concrete syntax, while the abstract syntax remains unaffected. This aspect is therefore not relevant for the discourse on semantic representations.

16.8 Output

This sub-stage should generate a consolidated set of ontological scenes, which are represented by a particular set of ontological concepts within UEMO or annotated ontologies (UEMOext). Ontological scenes should be specified regarding to the required (or derived) perspective, existing synonym relations, and the respectively related scene, if necessary. Further, formal semantic constructs can be specified by adapting DMM. Potential mappings between formal and material semantics should be specified textually. In addition, a set of different perspectives and respectively contained or referred concepts should be explicated. The summarised output of this stage is summarised in Fig. 16.7.

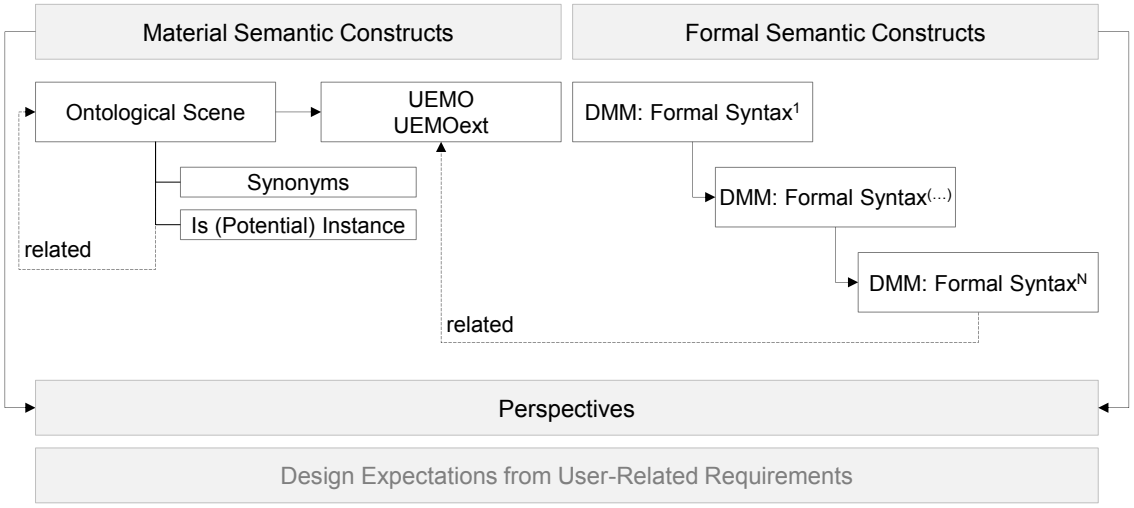

Fig. 16.7. Summarised output of the semantic specification

17

Correspondence Analysis

In the previous Section, requirements were decomposed into their smallest semantic entities by using ontologies. This procedure enables the comparison of semantic constructs and hence serves as a convenient instrument for justifying extension need by comparing the elaborated ontological scenes with the respective counterparts of an addressed EML on the meta model level.

The consideration of a pair-wise scene comparison causes a particular correspondence analysis between needed constructs and original constructs. The semantic distance between them as well as the type of difference has to be identified for this reason. This means that particular distance types have to be specified for each needed scene and potential differences are condensed within an integrated extension profile that facilitates the guided selection of appropriate extension mechanisms. This fine-grained procedure should enable a well-justified extension construction.

For reasons of applicability, the following assumptions and limitations are stated: All considerations assume the existence of a semantic specification of an EML, although current research lacks in the provision of such specifications as outlined in Sect. 16.1.1. It is further assumed that each ontological representation in the original EML is mapped by exactly one syntactical construct (cf. [247]). Finally, we focus on the correspondence analysis of material semantic constructs and omit any deeper analysis of formal semantic constructs due to the limited space of this work. Respective research – also in regard to hybrid semantics – has to be conducted in further research.

17.1 Material Semantic Constructs

Basically, semantic comparison is conducted by comparing the scene and its ontological representation of the identified need (*need scene*, *need ontology*) and the original EML under consideration (*original scene*, *original ontology*). Consequently, the introduced comparison types from Sect. 16.4.2 are taken up again in order to derive respective design consequences. Each comparison result type can determine one of the following design consequences:

1. **Apply Syntax and Semantics**: This type represents the actual usage and application of original language constructs, without any extension.
2. **Apply Syntax**: This type stands for solely application of the syntax, while the semantics are extended (see below).

3. **Extend Syntax**: This type stands for an extension of the original syntax of a language construct, usually causing the definition of additional semantics for these introduced constructs.
4. **Extend Semantics**: This type stands for an extension of the original semantics of a language construct, while its syntax remains unaffected. This has to be conducted in order to refine or specify the intended interpretation of a construct.
5. **Reject**: Rejection means that an examined scene should neither be implemented as an application nor extension due to semantic conflicts with the original language and the outlined extension principles from Sect. 4.3.
6. **Perspective**: Additionally, it might be reasonable to specify perspectives on language constructs in order to correspond to explicit user needs and emphasise particular foci on constructs.

The mentioned comparison consequences correspond with the syntactical constructs that are referred to by particular scenes and their ontological representation. The following analysis is aligned with the general reflections on semantic comparisons from Sect. 16.4.3. We therefore start with equivalent scenes, subsequently consider similar scenes, and finally consider completely different scenes.

17.1.1 Equivalent Scenes

Type	**Sub-Type**	**Design Consequence**
Congruency		• Apply Syntax and Semantics
Differing Degree of Understanding	Containment	• Apply Syntax and Semantics • (Perspective for covering the focussing on the contained attributes)
	Coverage	Option 1: • Apply Syntax • Extend Semantics Option 2: • Extend Syntax (explication and annotation of missing characteristics to one construct) Option 3: • Reject (in case of overlap with ontological representations of other constructs)
	Specialisation	Option 1: • Apply Syntax • Extend Semantics (refinement in the sense of specialising interpretations) Option 2: • Extend Syntax (explication as subtype, refining all super properties with new terms, e.g.)
	Generalisation	• Reject
	Complex (with Specialisations)	Option 1: • Apply Syntax • Extend Semantics (refinement in the sense of specialising interpretations) Option 2: • Extend Syntax (explication as subtype, refining regarded super properties with new terms, e.g.)
	Complex (with Generalisation)	• Reject
	Overlapping (Complete)	• cf. *Coverage*

	Overlapping (Partial)	Case 1: Non-matching ontological constructs do not contradict each other (no misunderstanding) • Option 1: – Apply Syntax – Extend Semantics • Option 2: – Extend Syntax – Perspective Case 2: Non-matching ontological constructs contradict each other (misunderstanding) • Reject
Homonym Conflict		• Reject

Table 17.1: Design consequences for equivalent scenes

Table 17.1 describes design consequences resulting from the comparison of equivalent scenes. In case of *Congruency*, the extension need corresponds to already specified language capabilities. Syntax and semantics of a referred to original construct can be reused. If both ontological scenes have a different degree of understanding, then an in-detail analysis is necessary.

In case of *Containment*, syntax and semantics of the original construct can be applied. However, the introduction of a perspective or a construct-specific view should be examined in order to provide an appropriate, complexity-reducing view on this construct.

In case of *Coverage*, some kind of extension has to be conducted, as not all semantic features are provided by the original language. Three design alternatives therefore exist. Option 1 covers the application of the underlying language construct and an extension of the semantics in order to explain additional semantics. Option 2 represents a more explicit form by extending the regarded syntactical element and annotates missing features by properties, for instance. If the coverage of the need ontology affects ontological representations of other language constructs, then the scene has to be rejected, as it would redefine the original language.

In case of *Specialisation*, all needed ontological constructs serve as specialisations of the original ontology. Consequently, two design options exist. Option 1 represents the application of the original syntax and the extension of its semantics by refining it for special interpretations, i.e. the understanding is sharpened. Option 2 enables the explication of this refined understanding by defining additional syntactical constructs, i.e. a sub-type with refining properties for all super properties. While this manifestation may cause language overload (cf. [193]), the first option just causes tacit language interpretation.

In case of *Generalisation*, a particular scene is understood in a more general way than in the original language. Is has to be rejected, since any extensions for the reason of generalisation should be avoided as mentioned in Sect. 4.3, both syntactically and semantically.

In the cases of *Complex Similarities*, it is assumed that the refinement differences (generalisation and specialisation) dominate the detailing differences, as the overlapping actually represents the assumed equivalence, leading to a single consideration of the hierarchy. Consequently, the same rules as in the case of specialisation and generalisation have to be applied.

The case of *Complete Overlapping* can be managed as coverage, as the original ontology is completely covered by the need ontology.

In contrast, *Partial Overlapping* provides three design options, depending on an evaluation of the relation between the non-shared ontological constructs. If they do not contradict each other (case 1), then two options are possible. Either the syntax is applied and semantics are extended, or the syntax is extended and a perspective or view is created in order to emphasise the need attributes. If the ontological constructs contradict each other (case 2), then the needed scene should be rejected, as it does not comply with the original semantics of a language.

This is also required for any *Homonym Conflict*, as they regard the respective misunderstandings of things from the area of discourse. However, this would implicate misunderstandings with users of the original language.

Rejected scenes should generally not be omitted and ignored after their rejection. Instead, a reconsideration of their terms might be useful in order to specify their particular meaning. Such a revision could lead to a re-evaluation of scenes as differences.

17.1.2 Similar Scenes

Table 17.3 summarises the design consequences resulting from the analysis of similar scenes, which refers to the second horizontal line of the proposed correspondence typology in Table 16.2.

In case of a *Potential Synonym*, the degree of similarity has to be assessed. If it is evaluated as de facto equivalence, then the scenes can be treated as in the case of congruency (option 1). If such an inference cannot be justified, then both scenes are treated as synonymous scenes. This determines an application of the syntax and an extension of semantics in regard to the postulation of alternative terms and denotations.

As stated in Sect. 16.4, the consideration of similar scenes always requires situational analyses of the degree and extent of similarity. Consequently, the indifference sub-types usually provide more than one design option. In particular, the *Containment* type requires the analysis of two dimensions, which is outlined in Table 17.2. Thereby, the degree of similarity is only assessed as "high" or "low" for pragmatic reasons. This enables relocation within the correspondence typology, enabling a reassessment of following design consequences, which are adapted from other types. It is recommended to evaluate an aspect only then as "high" if the degree of similarity is really significant (e.g. only one or two represented phenomena are different). Otherwise, it is rather useful to treat them as being different and taking potential redundancy as a consequence.

In case of *Coverage*, the application of comparison is similar. The only difference is the derived *Coverage* type in case of evaluating the difference as a *Differing Degree of Understanding*.

In case of *Specialisation*, it can be stated that the ontological similarity is already high due to the tight binding between sub and super types. Therefore, only the assessment of the similarity degree of the scene is required and respective options can be derived. In case of *Generalisation*, the scene under consideration has to be rejected

Ontology Similarity **Scene Similarity**	**High**	**Low**
High	De Facto Congruency (S1O1)	Differing Degree of Understanding (S1O2) Homonym Conflict (S1O3)
Low	De Facto Synonym (S3O1)	Difference (S3O3)

Table 17.2. Detailed assessment of the degree of similarity for scenes and ontologies as well as respective design consequences

for reasons of possible language defacement and the avoidance of generalisation extensions in this work.

The complex types have to be assessed in accordance with the particularly referred to types of abstraction. The case of *Partial Overlapping* is treated in a similar manner as the already discussed type of coverage. The case of *Complete Overlapping* requires a consideration of these ontological features that are not shared. If they contradict each other, then the scene has to be treated as different. If they do not contradict each other, then the ontological similarity can be specified as rather low in accordance with Table 17.2, since the binding of respective ontological constructs is not as strong as in the case of containment, coverage, or specialisation. Accordingly, an assessment of the scene similarity determines further design consequences.

Type	**Sub-Type**	**Design Consequence**
Potential Synonym		Option 1: Treated as equivalent • cf. *Congruency* Option 2: Treated as synonym • Apply Syntax • Extend Semantics (especially in regard to alternative terms)
Indifference	Containment	Option 1: Treated as *Congruency* Option 2a: Treated as *Different Degree of Understanding*, sub-type *Containment* Option 2b: Treated as *Homonym Conflict* Option 3: Treated as *Synonym* Option 4: Treated as *Difference*
	Coverage	Option 1: Treated as *Congruency* Option 2a: Treated as *Different Degree of Understanding*, sub-type *Coverage* Option 2b: Treated as *Homonym Conflict* Option 3: Treated as *Synonym* Option 4: Treated as *Difference*
	Specialisation	Option 1: Treated as *Differing Degree of Understanding*, sub-type *Specialisation* Option 2: Treated as *Difference*, sub type *Specialisation*
	Generalisation	• Reject
	Complex (with Specialisation)	Option 1: Treated as *Differing Degree of Understanding*, sub-type *Specialisation* Option 2: Treated as *Difference*, sub-type *Specialisation*
	Complex (with Generalisation)	• Reject
	Overlapping (Complete)	cf. *Coverage*

Overlapping (Partial)	Case 1: Non-matching ontological constructs do not contradict each other • Option 1: Treated as *Differing Degree of Understanding*, sub type *Overlapping* • Option 2: Treated as *Homonym Conflict* Case 2: Non-matching ontological constructs contradict each other • Treated as *Difference*
Potential difference	Option 1: Treated as *Homonym Conflict* Option 2: Treated as *Difference*

Table 17.3: Design consequences for similar scenes

In case of a *Potential Difference*, dedicated investigations should be conducted on the difference of scenes in order to elaborate potential de facto homonyms with large scene similarities. Otherwise, it should be generally treated as different, indicating the final introduction of new syntactical constructs. It becomes obvious that the similarity type always requires a shift within the typology framework. Concrete design actions can therefore be found in Tables 17.1 and 17.4.

17.1.3 Different Scenes

The consideration of different scenes assumes that different things are meant. However, respective ontological analysis has to be conducted in order to find out whether similar or even the same things are actually being referred to. The stated correspondence types from Sect. 16.4.3 can therefore be reused and Table 17.4 presents particular design consequences for single types.

In case of *Synonyms*, the original syntax can be applied and the synonym term of the presented scene should be captured within slightly extended semantics.

In case of *Containment*, the non-matching attributes have to be considered. If they reveal no contradictions, then they can be treated as synonyms and probably an additional perspective should be introduced in order to focus the subset of characteristics.

In case of *Coverage*, missing ontological constructs should be represented by extending the original syntactical construct by a specialisation with additional attributes. If the need ontology overlays multiple ontological representations of original elements, then the introduction of compositions or aggregations should be examined if respective "part-of" relations can be identified.

The case of *Specialisation* represents the situation where differently perceived scenes are completely related based on an ontological inheritance. Two options are possible. In the first option, the syntax is extended by a sub-type and properties which serve as one-to-one refinements of properties from the super type. In contrast to this externalised implementation, the second option only conducts semantic extensions in order to state how the original syntactical construct has to be understood and how it is alternatively referred to.

In case of *Generalisation*, an additional syntactical construct has to be introduced. This construct "rebuilds" some ontological characteristics from an original construct, but this type of redundancy is necessary in order to avoid generalisation-based modification.

In case of *Complex Similarities (with Specialisations)*, some ontological constructs are reused and other ontological constructs are refined by the need ontology. Two options are possible. At first, a specialisation might be introduced in order to provide a particularly labelled syntactical construct reusing super properties. Besides, additional properties have to be defined, which specify the refined properties either syntactically or semantically. Instead of this rather complex implementations, the definition of an independent syntactical construct is also possible.

In case of *Complex Similarities (with Generalisations)*, an additional syntactical construct should be specified due to the prohibition of generalisations on original constructs.

Finally, the case of *Overlapping (Partial)* requires special consideration. At first, it has to be evaluated if the overlapping is perceived as large. If not, then the syntax should be extended with a new construct. If the overlapping is large and there are no contradictions within the non-matched ontological constructs, then two design options remain. In the first option, a specialisation is introduced which both reuses shared as well as non-shared attributes from the original construct. In the second option, the original construct is extended with additional properties, indicating a particular overloading. Further, the semantics have to be extended in order to record the needed scene terms.

Type	**Sub-Type**	**Design Consequence**
Synonym		• Apply Syntax • Extend Semantics (explication of synonym meaning)
Indifference	Containment	Case 1: Non-matching attributes have no contradictions • Treated like *Synonym* • Perspective Case 2: Non-matching attributes have contradictions • Extend Syntax (new construct)
	Coverage	Case 1: Covering one ontological construct • Extend Syntax (specialisation and additional attributes) Case 2: Covering multiple ontological constructs • Extend Syntax (e.g. composition, if "part-of" indications)
	Specialisation	Option 1: • Extend Syntax (specialisation with additional attributes that are refinements of inherited attributes) Option 2: • Apply Syntax • Extend Semantics (especially alternative terms)
	Generalisation	• Extend Syntax (dedicated language construct)
	Complex (with Specialisation)	Option 1: • Extend Syntax (specialisation) • Extend Semantics (refined properties on sub level) Option 2: • Extend Syntax (dedicated language construct)
	Complex (with Generalisation)	• Extend Syntax (dedicated language construct)
	Overlapping (Complete)	cf. *Coverage*

	Overlapping (Partial)	Case 1: Non-matching attributes have no contradictions
		• Option 1:
		  • Extend Syntax (specialisation)
		• Option 2:
		  • Extend Syntax (additional properties)
		  • Extend Semantics (alternative terms)
		Case 2: Non-matching attributes have contradictions
		• Extend Syntax (new construct)

Difference		• Extend Syntax

Table 17.4: Design consequences for different scenes

17.1.4 Instances

The elaborated proposal for semantic comparison on the meta model level implicitly focusses on the identical abstraction level and, hence, supposes the same classification level for potential extension techniques. However, the requirements analysis and semantic specification stage may also yield type-instance relationships between intended constructs and original semantic constructs (cf. Sect. 16.3.2). Due to the lack of comparison instruments, instantiation relations cannot be represented in UEML appropriately in the current version and it is hence difficult to systematically identify them during correspondence analysis. Thus, it is most likely that instances are evaluated as differences or specialisations.

In regard to a potential application of the Multilevel Modelling approach, we recommend dedicatedly examining all properties of the underlying constructs according to the presented guidelines on differentiation between generalisation and classification from Sect. 8.4.3, which may provide reasons for type instance relations. In case of arguments for instantiations, a respective material semantic construct has to be recorded as an instance or potential instance in order to generally exclude it from traditional extension design, but prove potential multilevel capabilities later, too.

17.1.5 Consolidation, Application, and Remarks

The introduced comparison types should be applied in the following regime. In the first step, each similar scene is examined and shifted within the typology, if necessary. In the second step, equivalent scenes are analysed, as they implicate a specific and hence efficiently conductible point for comparison. In the last step, all remaining scenes have to be examined by comparing them to each original ontological representation. The search for the most appropriate ontological representation should be applied in a descending manner, whereby equivalence serves as the top and difference as the bottom of comparison results. Consequently, this serves as the most time-consuming task within the correspondence analysis.

Finally, the mentioned scene-based comparisons lead to the identification of different syntactical design recommendations, i.e. introducing additional properties, additional specialisations, as well as dedicated language constructs. Furthermore, element-specific views or perspectives are derived and also semantic specifications are stated.

After specifying particular extension constructs, it is extremely important to focus on their relations to other constructs. Therefore, the ontological representation can be reconsidered in order to identify "part-of" relations (for compositions or aggregations, e.g.) as well as directed or undirected relations (according to [235]). These considerations should then be reused within the definition of the extension syntax.

17.2 Formal Semantic Constructs and Perspectives

17.2.1 Formal Semantic Constructs

The consideration of correspondence checks in the area of formal semantic constructs should be omitted in this paper and only the following guidelines are postulated in accordance with the pragmatic usage types.

In case of *Automation*, defined formal algorithms and rules must not override or contradict any formal statement, i.e. model transformation rules of the original language (according to [80]). While this is difficult to describe formally in a generic manner, it is important to note that each originally specified transformation of models instantiated from the original language has to be valid after extension.

In case of *Model Analysis*, the in-depth analysis of potential modifications is less critical, as these operations are understood as "on-top" analyses of the probably extended model base.

17.2.2 Perspectives

Section 16.7 already specifies the first list of required perspectives, whereby each perspective associates contained semantic constructs, either motivated by user-specific views of constructs, particular grouping of constructs, or a differentiation of material and formal semantics. The conduction of the correspondence analysis leads to a revision of this list as follows.

A different degree of detailed understanding of a construct may require a particular sub-view of a construct that might be captured within a perspective. The particular ontological representation of a scene (probably of an original construct) should therefore be added to an already specified perspective candidate or it should determine the creation of one.

Scenes that determine additional language constructs (no specialisations) should also be considered in regard to the perspectives they are integrated in, if it cannot be referred from the already defined list of perspectives.

Subsequently, the specified list of perspectives should be revised. More precisely, a homogenisation of conceptually-driven perspectives and user-driven perspectives should be examined. These perspectives should then be compared to possibly defined perspectives within the original language, which could result in the definition of additional perspectives or sub perspectives (cf. [80]). Due to the rather constructivist character of the perspective concept, it should be realised by the language engineer, in close coordination with the prospective EML users.

17.3 Output

This stage finally provides a semantically justified extension profile that is composed of material semantic constructs and their required type of implementation, respectively derived perspectives, as well as their covered semantic constructs and formal semantic constructs that are reused from the concept analysis stage. The extension profile is summarised in Table 17.5.

Aspect	Representation	Specified Characteristics
Material Semantic Constructs	Scenes (represented phenomena), Ontologies (represented scenes)	• Identifier (according to central phenomenon in scene) • Realisation type • Referred original construct (ontology and syntactical construct) • Relation to other constructs
– **Instances**		• Specified as *Instance* or *Potential Instance* • Responsive type for type instance relation

Formal Semantic Constructs	DMM model, computation rules and algorithms (integration model with scene)	• Realisation types:
		  • "Extend Syntax and Semantics"
		  • "Extend Semantics"
		  • Perspective
		• Relation to material semantic constructs

Perspective	List	• Identifier (name) • List of semantic constructs • Relation to other perspectives • Type of represented constructs (material, formal)

Table 17.5. Structure of an extension profile

The consolidated extension profile should work as base for the following selection of appropriate extension mechanisms from the mechanism repository, which inherently serves as engineering-oriented task.

18

Extension Preparation and Subsequent Stages

The above discussed Sections evolve a set of required language constructs by the definition of a consolidated extension profile. It is now necessary to select an appropriate extension mechanism, prepare the respective application, and finally implement the extension as an applicable meta model extension. It is hence necessary to translate the above formulated design use cases as well as the extension profile to respective syntactical structures. Due to the characteristics of semi-formal modelling languages, a straightforward or even automatable translation procedure is impossible. Nonetheless, it is intended to provide methodical support for this translation by introducing some guidelines for the efficient preparation, selection, and application of appropriate extension mechanisms.

18.1 Pragmatics-Driven Pre-Selection

The actual selection of one mechanism should not be conducted in an ad-hoc manner, but rather guided by the intended extension type as introduced in Sect. 5. It is therefore reasonable, to select one or more extension types first and respective extension mechanisms afterwards. It is hence important to gain a black-box-view on a particular mechanism and reconsider the actual pragmatic extension purpose behind it.

In case of *Documentation*, usually the view and domain-specificity dimensions are of interest for conducting some extensions. If the business level and the application level are addressed, then the formalisation dimension is considered, too. In case of *Automation*, the formalisation dimension is relevant, while *Documentation and Automation* refers to all dimensions if particular use cases are not decomposed. In case of *Model Analysis*, the formalisation dimension (in particular model operations) is relevant. The applicable extension mechanisms for each dimension are represented in Table 18.1 and briefly summarised below.

18.1.1 Formalisation Dimension

Within the formalisation dimension, extensions affect the level of interpretation invariance in regard to the tension between informal and formal model interpretation (cf. [14, 8, 73, 67]). Technically, invariance creation requires the introduction of syntactical concepts, which enables the mapping of a particular syntactical structure

Formalisation Dimension	
Behavioural-formal Semantics / Static-formal Semantics	• Annotation (Plugin) • Hooking by Specialisation • Profiling (mandatory application) • (Simple Generalisation/Specialisation) • (Semantic Extensions)
Model Operations	• Annotations (Decorator, Plugin, Aspects)

View Dimension	
Enhancement	Additional features of the area of discourse: • Annotation (Plugins, Add-Ons, Decorators) Refinements within the area of discourse: • Hooking by Placeholders • Profiling • Simple Generalisation/Specialisation • Multilevel Modelling
Augmentation	• Annotation (Plugins, Aspects) • Hooking by Placeholders

Domain-Specificity Dimension	

Domain-Specialisation	Annotating:
	• Additional features:
	  • Annotation (Add-Ons)
	  • Profiling
	  • Hooking by Specialisation
	• Additional interpretation:
	  • Semantic Extension
	Specialising:
	• Simple Generalisation/Specialisation
	• Hooking by Specialisation
	Additional classification levels:
	• Multilevel Modelling

Table 18.1. Applicable extension mechanisms within the EML framework

to another syntactical system having determined semantics (cf. [139, 67]). This usually causes non-trivial, complex syntactical transformations, i.e. interpretation rules, which have to be represented by particular extension mechanisms. These mechanisms need to realise this level of complexity (in the sense of modularity) and may also require mandatory applications of extensions in order to implement those rules. The following mechanisms have to be considered therefore in regard to behavioural semantics and static semantics.

- *Plugins* provide complex, perhaps domain-independent modules that can be applied mandatorily if respective interfaces are pre-defined.
- *Hooking by Specialisation* enables element-wise instantiation of under-specified meta elements, definition of their interpretation, and the annotation of respective conditions (e.g. by OCL statements).
- The issue of possible non-applicability of additional properties that affect semantics due to missing mandatory instantiation after extension application can be solved by using *Profiling*, as Stereotypes can be applied mandatorily. The above mentioned issue of missing generalisation turns around into an advantage in this case.
- Also simple *Generalisation/Specialisation* can be used for defining stricter interpretation means in the sense of creating subclasses with specific features. How-

ever, the original meta classes remain unaffected and the degree of invasiveness is consequently lower.
- Finally, it is naturally possible to specify additional *semantic mappings* (cf. [67]). However, the precise syntactical appearance and structure of the target model (cf. [245, 63]) remains unclear at this point and needs to be specified separately.

18.1.2 View Dimension

Within the view dimension, enhancement and augmentation need to be supported by respective mechanisms. In case of *enhancement*, two implementation types can be differentiated:

- First, adding features to original concepts, without any refinement of them. This means that additional features are introduced, independent of existing ones. Therefore, annotations can be used in general, i.e. *Decorators*, *Plugins*, and *Add-Ons*.
- In the second case, existing concepts are refined, which means that an extension concisely refers to original features and extends them within a particular area of discourse by adding invariants, i.e. conceptual conditions, range values, or cardinalities (cf. [80]). Therefore, *Profiling* as well as *Simple Generalisation/Specialisation* can be used. Also the *Hooking by Specialisation* technique can be applied with respective architectural pre-design work.

In case of augmentation, concepts and features of additional areas of discourse are annotated and the EML vocabulary is thematically extended. Extension mechanisms therefore need to provide the module-like definition of additional concepts that indicates high cohesion and independence of the original model. Consequently, annotations (*Plugins* and *Aspects*) as well as *Hooking by Placeholders* are most appropriate.

18.1.3 Domain-Specificity Dimension

Within this dimension, three implementation types can be identified: annotating, specialising, and introducing additional classification levels:

- **Annotating**: Similar to the above mentioned case of refinement; additional features are added in order to specify a particular concept for a problem scope. Original concepts are thus extended directly. *Add-Ons*, *Profiling*, as well as *Hooking by Specialisation* can be applied for that purpose, as they extend referred concepts incrementally. Slight syntactical annotations are therefore referred to as *Additional Features*. The type of *Additional Interpretation* covers the introduction of additional semantic expressions in order to specify the intended interpretation of concepts and reduce informal variance.
- **Specialising**: In the second case, additional features of a domain are introduced by the generation of subclasses that encapsulate them coherently. The originally intended interpretation of original meta classes remains unaffected, which increases non-invasiveness. *Simple Generalisation/Specialisation* as well as *Hooking by Specialisation* can be used here.

- **Classification Levels**: The third case represents the introduction of additional classification levels by the *Multilevel Modelling* technique. In contrast to the above stated types, additional classification levels cause massive re-engineering due to the paradigmatic shift.

18.2 Mechanism Selection

The pragmatics-driven selection of extension mechanisms reveals that multiple mechanisms can be conducted for particular pragmatic types. This pre-selection serves as helpful limitation of suitable mechanisms according to the intended pragmatics. It is now necessary to finally select the most suitable mechanism from the pre-selected set in accordance to the rather fine-grained structure of the actual extension concepts that are elaborated within the correspondence analysis phase.

Pre-Version of Extension

In addition to the justified construct-wise realisation types in this phase, it is hence necessary to particularly become aware of the relationships between potential extension constructs and relationships between extension constructs and original constructs; probably resulting in larger, coherent modules.

It is therefore helpful to translate the extension need into a *pre-version* of the intended meta model (e.g. as simple meta class diagram) in order to specify relationships between single elements. The name of model elements can be adapted from the extension profile as well as from the underlying ontological representation. Further, generalisations or specialisations can be inferred from respective realisation types, whereas associations, aggregations, and compositions have to be modelled and attributed in accordance with the underlying scene. This especially covers cardinality values (e.g. min, max). It is important to keep in mind that no mandatory extension must be defined. Compositions and aggregations can be used for the definition of constructs that aim to represent differences with coverages, for instance.

Such a pre-version of the actual extension model facilitates the analysis of interdependencies between constructs and externalises some aspects that are relevant for method selection, e.g. the kind of dependency from original constructs and the type of complexity within extension constructs. The mentioned structuring is also referred to in existing extension methods (cf. [48]), but specific references to semantic analyses are missing and the domain models therefore come more or less out of the blue.

With respect to the focus of this Part and the limited space of the entire work, detailed considerations on translating semantic domain analyses to domain models, i.e. extension pre-versions, are omitted here and should be tackled in further research. Nevertheless, the following guidelines are proposed for a final mechanism selection in order to conduct the required extensions.

Guidelines for "Extend Syntax"

1. For the assignment of *additional properties*, either use Add-Ons or Profiling.

2. If additional constructs (without specialisations to original constructs) have to be integrated, then several sub-types have to be considered in detail:
   - *Representation of roles and complex hierarchies*: Decorators.
   - *Multiple applications to many original classes*: Aspects.
   - *Complex inner structure and missing generalisation/specialisation relation to original meta classes*: Plugins.
   - *Complex inner structure and generalisation/specialisation relations to original meta classes*: Hybrid (Simple Generalisation/Specialisation with Plugins or Add-Ons) or Hooking by Placeholders (if under-specified).
   - *Simple structure and highly dependent on a few meta classes*: Add-Ons.
3. If an *additional construct* has to be integrated as a *composition of parts*, then use hybrid approaches or Add-Ons.
4. In case of *specialisation with additional properties*, the Simple Generalisation/Specialisation technique can be applied.
   - If *specialisation with a refinement of all super properties* has to be implemented, then Profiling can be conducted, if no explicit extension points are given. If those spots are explicitly pre-defined, then the Hooking by Specialisation mechanism has to be applied.
5. If *multiple classification levels* are needed or some *instances are also perceived as types* within the intended extension, then Multilevel Modelling needs to be used.
6. *Formally driven* syntactical extensions should be realised by annotations, namely Plugins or Aspects.
7. In the rare situation of *large under-specification of a particular meta model*, i.e. large parts are explicitly left open on purpose, then Hooking by Placeholders has to be applied, which indicates a maximum of design freedom.

Guidelines for "Extend Semantics"

1. If *alternative terms or labels (synonyms)* have to be integrated, then use a *Semantic Extension*, in particular the *Add Semantics* operation in order to add semantic mappings.
2. If *refinements* have to be conducted, then use *Semantic Extension*; in particular *Specify Semantics* in the form of integrating additional ontological constructs to entire classes or properties.
3. If semantic extensions are *formally driven*, then use *Add Semantics* in the sense of specifying additional syntactical rules for dedicated mappings to formal sign systems (cf. [244]).

18.3 Mechanism Application and Subsequent Steps

As this Section solely focusses on the semantic preparation and selection of extension mechanisms, the actual conduction of those methods as well as syntactical consequences are only briefly outlined below.

18.3.1 Extension Definition in General

This stage covers the application of respectively selected extension mechanisms in order to define the meta model extension syntactically and semantically. Extension meta classes, extension perspectives, as well as formal statements have to be defined with regarded meta modelling languages like MOF (e.g. [45, 76]).

In addition, the semantics of the extension has to be defined (cf. [73]). This covers both the annotation of "extend semantics" realisation types to original meta classes as well as the semantic specification of introduced language constructs. Already defined UEMO ontologies from the semantic specification stage can therefore be reused and revised. These semantic specifications are then annotated to the extension meta model or prepared for annotation to the EML meta model.

Further, the concrete syntax of extension elements or respectively modified original notational elements have to be specified (cf. Sect. 15.3). In the context of MOF, the DD standard serves as a powerful and expressive instrument for the generic specification of notational elements and containers (cf. [42, 78]). All the stated tasks need to be realised by the EML language engineer, since all tasks require profound constructive skills and meta modelling knowledge.

18.3.2 Extension Definition in BPMN

The specific implementation of EML extensions in the context of BPMN, for instance, can be realised in two ways. The first way represents an implementation within the current BPMN environment and the state of the art in (meta) meta modelling with MOF. In this case, several methodical supplements have to be conducted in order to realise the above stated definitions. Generally, the BPMN extension method of Stroppi et al. [48] can be applied in order to derive meta model extensions that are compliant with the BPMN core, namely the BPMN extension mechanism. Therefore, the syntactical meta model extension can be integrated within the Conceptual Domain Model of the Extension (CDME), which actually represents the set of required concepts for an extension. These domain model classes are then transformed into a BPMN-compliant meta model based on respective transformation rules [48]. Braun & Esswein [44] further demonstrate how to integrate perspectives into this extension procedure. Figure 18.1 presents the current standard-compliant version of the BPMN extension method that includes the pragmatics-driven and semantics-driven enhancements within the analysis and conception phases, which are elaborated in Part IV.

However, the entire procedure builds on the BPMN extension mechanism that reveals several shortcomings [39]. Ascertained issues are largely caused by limited capabilities of the applied meta modelling language MOF [78]. The second implementation option therefore builds on a particular re-engineering of parts of the MOF as well as BPMN [42, 76, 78]. These re-engineering works mainly address the revision of the MOF meta meta model in regard to EM capabilities as well as features for the explicit consideration of EML extensions on the meta model layer [77]. Further, the application of the DD standard on the meta meta model layer as well as its generic application to EM-related meta concepts is discussed in previous works [42, 77, 78].

The pros and cons of both implementation strategies are obvious. While the first option enables extension application within the current state of the art, it

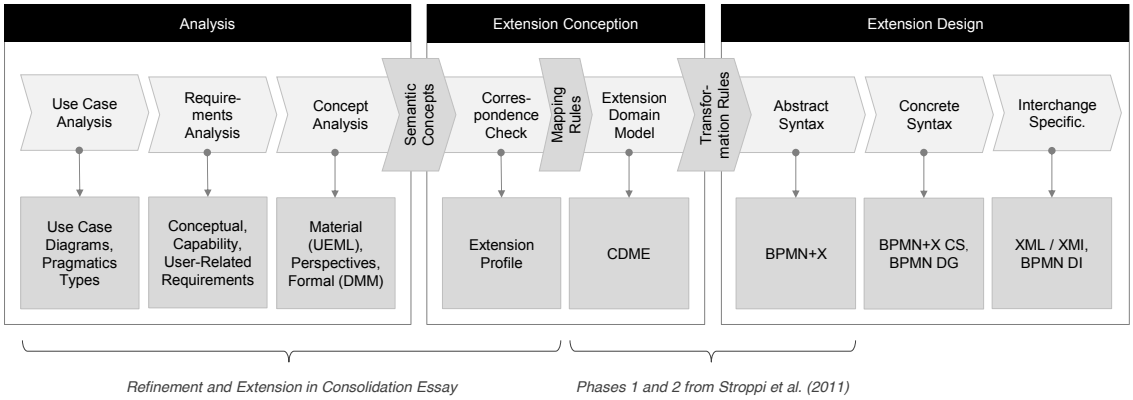

Fig. 18.1. Extended and refined version of the standard-compliant BPMN extension method from [76]

serves as a workaround for numerous reasons. The second option rather intends a meta modelling language based improvement by extending MOF in terms of issues arising from the peculiarities of EM, enabling a straightforward implementation of EML extensions on the meta model layer. However, the proposed re-engineering proposals have not yet been conducted nor disseminated in the EM community.

Finally, there remains the so far unsolved issue of integrating semantic specifications and (extension) meta models. While some preliminary studies outline general principles [100, 229], it is not specifically conducted in the context of MOF and EMLs.

18.3.3 Extension Labelling

Finally, the entire extension should be annotated with additional, rather organisational information in order to facilitate re-usage and to support an appropriate understanding of the purposes and characteristics of an extension from an outer perspective (e.g. within an extension marketplace).

Extension labelling therefore aims to provide information about the intended pragmatics by stating the initially derived pragmatic types as well as specific use cases for application. Further, the syntactical structure of the extension should be summarised by stating the main components (e.g. applied mechanisms) and respectively extended original EML meta classes.

Accordingly, the semantics of each extension class has to be provided in order to facilitate specific application decisions by other users. Finally, various types of master data should be provided, e.g. current version, related version of the extended meta model, and application information. Application information may cover specific sequences for conducting single components of extensions (e.g. according to extension bundles).

19

Conclusion

Figure 19.1 depicts the proposed EML extension method that is mainly driven by the initial analysis of the contextual scope, i.e. the intended pragmatics and semantics. It summarizes the main steps and contributions from the considered Sections aiming at providing an overall method understanding. Main results within the method are emphasized by a thick borderline and the leading relations are also represented by thicker arrows. The particular output type, i.e. the expected representation format, is stated in square brackets underneath the label of the task name.

Within the *Use Case Analysis*, expected or intended modelling cases have to be investigated. An externalization into UML use case diagrams is advisable. Additionally, respective purpose types have to be annotated to each of them in order to prepare the subsequent derivation of requirement types in the ensuing *Requirements Analysis* phase.

In case of *Documentation*, conceptual requirements are derived. In case of *Automation* or *Model Operations*, capability-related requirements can be specified. In case of *Model Usage*, user-related requirements are recorded. With respect to the focus on semantics, these requirements are not further considered. Besides, different constraints may be elaborated from the initially found use cases, implicating additional conceptual or capability-related requirements. The use cases further implicitly refer to semantic constructs by using particular terms from the area of discourse. These terms should be reused for conceptual requirements of the language, i.e. the required vocabulary.

Subsequently, the intended semantic constructs have to be specialised based on these requirements in the *Concept Analysis* phase. Material semantic constructs have to be elaborated from conceptual requirements, possibly implicating an in-depth analysis of respective constructs. These semantic constructs are externalised in UEML and mapped to the UEMO ontology or its extension (UEMOext), respectively. Capability-related requirements determine formal semantic constructs, which are specified by DMM models and respective transformation rules to additional formal systems. Also mappings to UEML models are proposed in order to integrate both types, but further research on this issue is necessary. Moreover, a list of potential perspectives has to be filled. Perspectives are either gained from stakeholder-specific groupings of use cases (user driven), from slightly differing, but non-contradictory understandings of semantic constructs (conceptually driven), or from formal semantic constructs (formally driven).

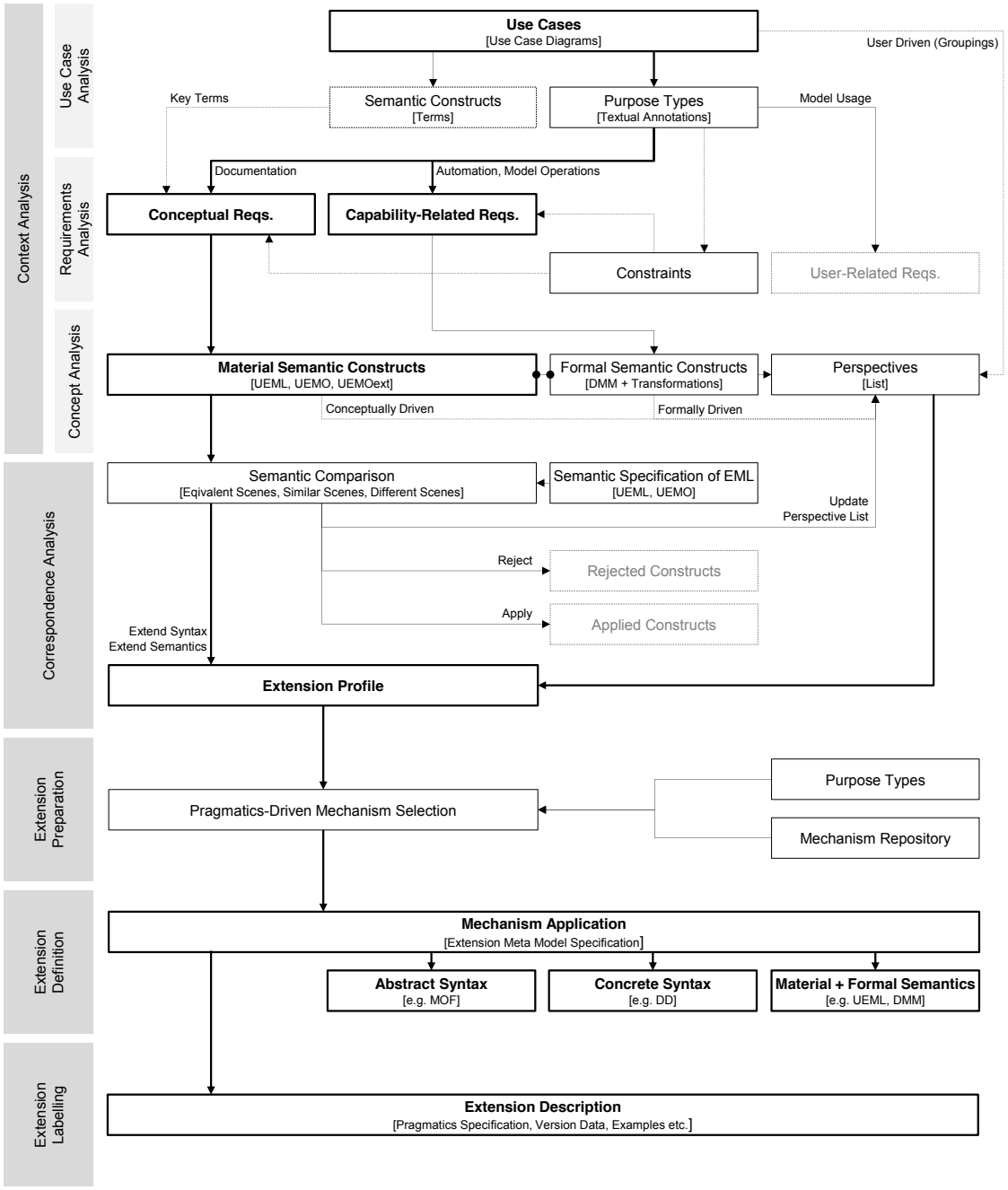

Fig. 19.1. Integrated EML extension method with a special focus on context analysis

This specification enables a *Correspondence Analysis* against the semantic specification of an EML (which is assumed to be defined). Therefore, the ontological scenes have to be compared, i.e. equivalent scenes, similar scenes, and different scenes. In their simplest form, scenes may be specified by only a few classes of things and a few attributes, indicating rather fundamental references of some state of affairs. The actual semantic comparison is realised by conducting model-to-model comparison between the UEMO ontologies. The correspondence analysis yields a design decision for each semantic construct. In case of rejection, a particular construct cannot be considered, as it contradicts the stipulated semantics of the addressed EML. In case of application, a construct semantically maps to an original construct, implicating a

simple language use. In case of determined syntax extension or semantics extension, an extension of the EML is identified. The relation between rejected constructs and applied or extended constructs guides a decision on designing an EML extension or rather designing a DSML (cf. [75]). Further, the list of perspectives can be updated due to slightly differing understandings of original constructs. Perspectives and semantic constructs establish an extension profile, which acts as integrated summary of the extension need that can be further detailed within pre-versions of the extension meta model.

In the *Extension Preparation* phase, the initially determined purpose types are reused in order to choose the relevant dimension within the EML extension framework and consequently select applicable extension mechanisms from the repository according to their pragmatic scope. It is thereby important to keep track of the concepts and the respective use cases from the beginning. Besides, the above mentioned pre-version of the extension meta model facilitates a better evaluation of the dependencies between semantic constructs. Based on that, particular extension mechanisms can finally be selected by conducting syntax-oriented and semantics-oriented guidelines.

Afterwards, the extension is specified within the *Extension Definition* phase by conducting the appropriately selected extension mechanisms and applying the derived extension model to the referred EML. Thereby, the abstract syntax, the concrete syntax, as well as material and formal semantics has to be defined. This stage primarily addresses syntactical operations and is hence only outlined due to the user-focus of the proposed method.

Finally, the extension has to be described from an outside perspective within the *Extension Labelling* phase in order to foster their assessment and dissemination within the particular EML community.

Part V

Conclusion and Further Research

20

Contributions

The referred research papers as well as the consolidation essay itself provide several contributions to the EM community. Below, the most relevant contributions are summarised within two parts. First, specific research artefacts are stated. Second, rather general insights and constructs are given. Afterwards, the realisation of the research objectives is discussed.

20.1 Research Artefacts, Studies and Consolidations

In the field of BPMN, a standard-compliant BPMN extension for resource modelling in the area of Product Engineering is proposed [36, 74]. Additionally, a revised BPMN extension for multi-perspective hospital modelling is designed [27, 75, 77, 76]. Motivated by rigorous artefact design in DSR [79], a requirements-driven method for integrated BPMN extension design is further elaborated (cf. [55, 77, 76]). The BPMN extension method builds on the BPMN standard and its methodical extension [48]. It is hence standard-compliant. Additionally, several architectural enhancements are proposed in order to overcome identified shortcomings and contradictions (cf. [42, 44]). Fig. 20.1 represents the method evolvement in the area of tension between standard compliance and further development of the BPMN environment.

The thesis further provides contributions in regard to EMLs and EML extensibility in general. For instance, a meta meta model extension for the integrated EML specification with OMG standards is outlined [45, 77, 78]. Further, a generic repository of EML extension mechanisms is presented (cf. Part III) and single mechanisms are exemplarily applied to BPMN [42, 59]. In addition, an EML classification framework and an EML extension typology is proposed in order to facilitate communication about EML extensions (cf. Part II). Finally, a method fragment for user-centric EML extension design is elaborated, which focusses on a detailed domain analysis as well as a step-by-step extension specification and justification (cf. Part IV).

The analysis of BPMN and its underlying meta modelling language MOF provides several contributions to the knowledge base in EM. An extensive analysis of BPMN extensions from different perspectives enables an overview of the current state of the art [33, 73]. The proposed classification frameworks can be adapted to other EMLs and the overview of all extensions might support the evolution of potentially official BPMN dialects or accents (cf. [33, p. 54]). In this context, different OMG standards that are relevant for EML specifications are critically analysed and discussed in order

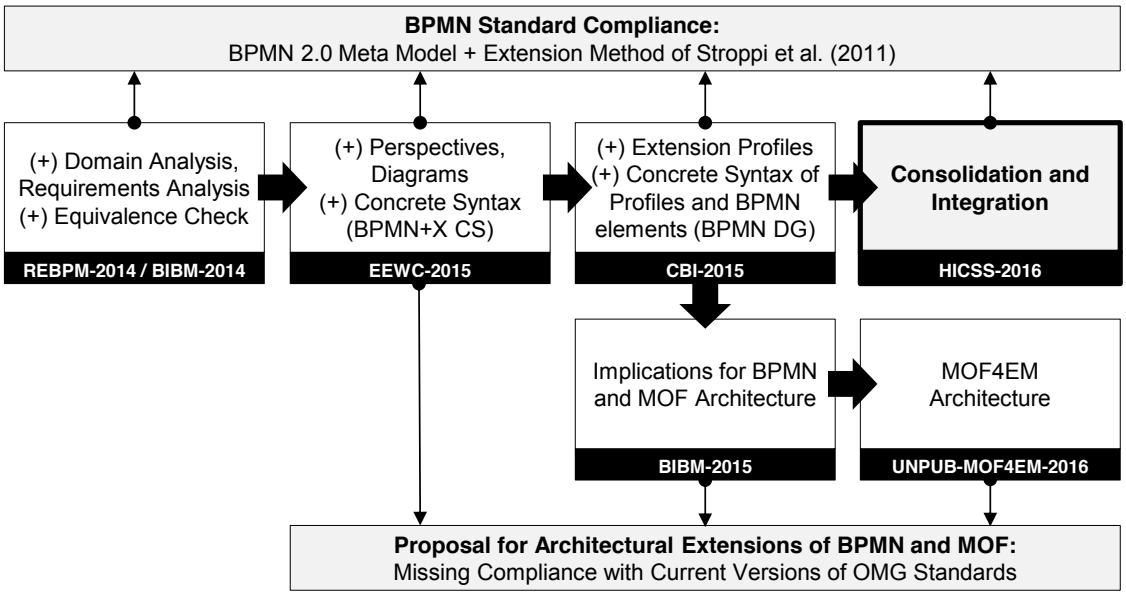

Fig. 20.1. Evolution of the BPMN extension method in the field of existing OMG standards and proposed enhancements of these standards

to elaborate current deficiencies and propose respective enhancements [39, 59, 45, 78].

Further, the state of affairs in regard to extensibility of EMLs is analysed and discussed both on the meta model level and the meta meta model level [28]. The consideration of extensibility also requires the in-depth analysis of semantic specifications on the meta model level, leading to a consolidation of different relevant dimensions [57, 9], with a special focus on epistemological foundations [41]. The concerned discussions further reveal a proposal for hybrid semantic specifications covering a potential integration between formal and material semantics [67].

20.2 Reflection of Research Objectives

With regard to Sect. 1.3, most of the research objectives can be rated as achieved. The state of the art of BPMN extensibility is elaborated from different perspectives, enabling a derivation of adequate terminology and fundamentals on EML extensibility in general (Research Objectives 1a and 2a). Also initial versions of two BPMN extensions were designed and partly revised according to altered requirements (Research Objective 1b). Moreover, an extension mechanism repository has been evolved and prototypically conducted using BPMN (Research Objective 2b).

In addition, the BPMN extension mechanism was incrementally enhanced according to domain analysis, extension justification, as well as an integrated extension specification in accordance with specifics of EM and existing standards from the OMG environment (Research Objective 1c). In particular, the enhanced BPMN extension method is compliant with the existing BPMN meta model by providing respective transformation rules as outlined in [42]. In parallel, several revision proposals for BPMN and the meta modelling language MOF are outlined and proposed in accordance to Research Objective 1d [45, 44, 78]. However, their final implemen-

tation within the language specifications remains open and is not further discussed. Research Objectives 1c and 1d are hence not completely achieved and need to be further investigated.

Finally, Research Objective 2c can be seen as achieved in respect of the scope of the thesis as it reviews, discusses and integrates different relevant topics for pragmatics-driven and semantics-driven extension design. Nevertheless, the consideration of pragmatics and semantics in EML design reveals several issues for further research that are outlined in Sect. 21.

21

Implications for Further Research

Investigating the issue of methodically guided and integrated extensibility of EMLs mainly using the example of BPMN brings up a range of resultant problems which do not cover extensibility exclusively, but rather concern underlying issues and unsolved shortcomings of EML definitions in general. The designed artefacts as well as the consolidation essay serves as an intermediate step within a larger research stream that covers several fundamental issues within EML design. Accordingly, several topics for further research have evolved, requiring additional investigations and implementation efforts in various fields. Below, those topics are outlined in regard to the main components of modelling methods (i.e. pragmatics, semantics, and syntax) as well as the entire method.

21.1 Pragmatics

Obviously, there is a lack of profound understanding of the actual application of enterprise models. Accordingly, there are only a few inferences in terms of revising an EML based on its application scope. This is generally problematic since speaking a language (i.e. modelling with an EML) determines the real added value of a language and its design should therefore be aligned with its pragmatic use cases. It is advisable to conduct further research in regard to the specification of such typical use case types and their characteristics in order to derive design consequences or to provide respectively needed EML dialects or EML accents. An ex-post analysis of already applied EMLs and EML extensions may support this purpose. Also, a stronger focus on the actual language user is needed.

21.2 Semantics

The significance and complexity of semantics has already been emphasised and recent works firmly elucidated the multifacetedness of semantics in the context of EML definitions, both in regard to actual language application as well as for justifying extension need. Accordingly, several research topics have emerged.

Consensus finding within material semantics: EML semantics are eventually determined by an integrated analysis of personal conceptualisations, their externalisation, as well as particular agreement and consensus processes [248, 41, 9].

This covers the definition of reference processes for general consensus finding regarding EM-related concepts in order to either homogenise different world-views or at least highlight particular differences (cf. [101]). Naturally, a complete and multi-layered understanding of enterprises is illusory, but a commonly accepted definition of essential constructs from the EM domain as well as their nuancing seems to be attainable. Such fundamental questions consequently require the analysis of fundamental research disciplines like Epistemology, Cognition Theory, or Linguistics (cf. [41]). The adaptation of theories and techniques from those disciplines seems to be promising in regard to semantic specifications, for instance, in order to investigate lexical ambiguity [249, 196]. Also techniques from the field of Semantic Web may be helpful for a sophisticated definition of material semantics.

Specifying material semantics of EMLs, e.g. with UEML: The significance of semantics requires the comprehensive and consistent semantic specification of EMLs like BPMN, although current research primarily focusses on the abstract syntax [100, 226, 41]. Already started research on the semantic annotation of EMLs (cf. [226]) has to be further extended, especially in regard of retrieving a semantic specification for BPMN (cf. [9]). This implicates the elaboration of semantic standards as well as their adequate fusion with the syntax specification of an EML. In case of UEML, it is necessary to design explicit mechanisms for extending UEMO and support its integration with similar ontologies from specific domains. Applying UEML for extension justification further reveals the need of additional taxonomies in regard to representing generic classification features (e.g. "instance of" relations). This aspect is closely related to the derivation of design recommendations for syntactic operations. In addition, there is need for further research concerning detailed operationalisation of correspondence types and distance measures as outlined in Anaya et al. [226].

Hybrid semantics as integration approach between material and formal semantics: The elaborated pragmatic types as well as existing EML specifications motivate the partly integrated consideration of material (non-invariant) and formal (invariant) semantics. Thereby, invariant semantics cover the intended automatisation or transformation of models, on the one side, and restrictions in regard to valid interpretations of models, on the other side [67]. The analysis and construction of appropriate representation techniques is therefore required (e.g. according to the outlined integration of UEML and DMM).

21.3 Syntax

Extensibility as inherent component of meta modelling languages: Discussed papers on the extensibility of BPMN and EMLs reveal a need for consideration and implementation of reference extension mechanisms on the meta meta model level [42]. This indicates that an EML like BPMN can only then be extended systematically and usefully if the discussed extension mechanisms are respectively defined based on instantiation from the meta meta model level instead of creating conflicting intermediate levels or pre-instantiations as in current meta model based solutions [39].

MOF for Enterprise Modelling in general: The stated need for inherent consideration of extension mechanisms on the meta meta model level causes further issues when MOF-based languages like BPMN are investigated, as MOF itself is limited in terms of defining EMLs. Only those types can be extended which are actually defined within an EML meta model (cf. the discussion in [156]). MOF lacks in regard to capabilities for perspectives and concrete syntax, as well as partly amalgamating conceptual and technical aspects [60, 78]. There are no means for integrating abstract and concrete syntax on the meta model level and semantic integration is also omitted (e.g. [42]). Potentially, some principles from other meta modelling languages, e.g. Multilevel Modelling [250, 123], E3 [47], or MM-DSL [69], can be leveraged in order to enhance MOF for EM, while keeping its benefits [78].

Inevitable revision of MOF and BPMN: The stated issues require two main aspects. First, it is necessary to specify and disseminate a MOF dialect for defining extensible EMLs, as outlined in [78]. According to the proposed *Standard+X* approach, this dialect needs to be defined as a non-invasive extension on the meta meta model level. Secondly, this dialect has to be applied for EMLs like BPMN, causing a revision of the language, which requires a request for revision within the OMG organisation. Although several preliminary studies have been completed (e.g. [42, 59, 77, 76, 78]), such an OMG revision submission constitutes a time-consuming and laborious task [111]. Proposing a standard revision hence requires massive organisational efforts and it is therefore necessary to carefully ponder such a fundamental but conceptually profound revision, or rather rely on particular workarounds and extensions methods (e.g. [48, 77]).

Definition of frozen spots and hot spots: Especially in regard to the Hooking mechanism, it could be reasonable to mark some meta model parts as frozen spots in order to explicitly limit the extensibility of meta models, while marked hot spots can be intentionally under-specified and open for specification. Such kinds of limitations may constrain massive language extensions, leading to potential "language blow ups" on the one side, but require meta model revisions on the other side.

Extending the understanding of extensions: Extensions by generalisations are omitted within this thesis in order to avoid mandatory type modifications of original meta classes. However, generalising rather domain-specific language constructs might be a reasonable effort in regard to navigating vertically within the domain-specificity dimension of the EML extension framework. Also "extension by reduction" techniques should be considered in further research (cf. Sect. 4.3.2). These extensions cover user-specific vocabulary reductions by providing particular subsets of the original meta model, which is especially relevant for PSMLs with many concepts, e.g. BPMN.

21.4 Method

This part addresses topics for further research which concern the outlined EML extension method and the underlying integration of respective parts (i.e. pragmatics, semantics, and syntax).

Method operationalisation and integration: The stated research papers as well as the achievements in this consolidation essay evolve and outline an integrated

BPMN extension method, whereby single method fragments, e.g. concept analysis or extension design, are examined with a great level of detail. However, the integration between single phases, e.g. the integration between different specification instruments, has to be designed and supported by an appropriate integration of the proposed specification instruments. So far, research on single modelling method parts within extension design is widely isolated and infrequently integrated. This concerns especially respective description formats, e.g. for the abstract syntax (MOF [38]), concrete syntax (DD [251]), material semantics (UEML [229]), formal semantics (DMM [243]), as well as intended pragmatics (e.g. use case diagrams). Integrating these parts in a reasonable manner may foster the integrated design of EMLs and their extensions; for example, in the case of BPMN.

Evaluation: Design Science Research calls for explicit evaluation of proposed artefacts. Within this work, designed artefacts are evaluated based on their demonstration, aiming to realise a particular proof-of-concept (according to [53, 83]). Accordingly, this has to be conducted for the proposed extension method, e.g. by applying it to the BPMN and assessing its usefulness.

DSML vs. extension: Further research should also provide instruments and metrics to support the decision of whether a DSML-based approach should be used or an extension-based approach is more reasonable. The elaborated requirements-based extension profiles may serve as an appropriate starting point for that.

Organisation of extensions: Besides the procedural view on an extension method, also the meaningful specification of the outside view is important in order to provide an extension for other potential users within an extension repository and foster extension combination. A reference frame for extension labelling needs to be specified and a particular marketplace or extension repository needs to be outlined and implemented (referring to the Open Models Initiative, e.g. [252]). Establishing such a marketplace further requires the definition and application of respective mechanisms for quality assurance.

List of Figures

1.1 Consolidated research problems 9
1.2 Research objectives and associated research problems 10
1.3 Legend for structuring research papers and the consolidation essay . . 12
1.4 Research design, relevant publications and considered Parts of the consolidation essay in the context of Research Objective 1 15
1.5 Research design, relevant publications and considered Parts of the consolidation essay in the context of Research Objective 2 16

4.1 Types of semantic alterations [73, p. 1124] 39

5.1 Extension framework with a localisation of the BPMN 44

16.1 UEML architecture 142
16.2 Adaptation of the UEML approach for specifying the intended semantics 143
16.3 Notation for modelling scenes [227] 143
16.4 Correspondence typology representing scene comparison and ontology comparison 146
16.5 Types of ontological similarity 149
16.6 Outlining a potential integration between material and formal semantics by mapping DMM and UEML 157
16.7 Summarised output of the semantic specification 159

18.1 Extended and refined version of the standard-compliant BPMN extension method from [76] 177

19.1 Integrated EML extension method with a special focus on context analysis 180

20.1 Evolution of the BPMN extension method in the field of existing OMG standards and proposed enhancements of these standards 186

List of Tables

4.1 Characteristics of extensions 35
4.2 Characteristics of modifications 37

5.1 Language types 45
5.2 Language extension types 50

8.1 Comparison of generalisation/specialisation and classification/instantiation 79
8.2 Property types 80
8.3 Requests for concretisation 80

9.1 Summary of mechanisms 88
9.2 Comparison of mechanisms; legend: (++) very appropriate (+) appropriate (o) indifferent (–) inappropriate (– –) very inappropriate 89

13.1 Semantics-driven extension procedure 125

15.1 Adapted requirements types 134

16.1 Description of a state of affairs by answering six essential questions . . 141
16.2 Basic correspondence typology 147

17.1 Design consequences for equivalent scenes 163
17.2 Detailed assessment of the degree of similarity for scenes and ontologies as well as respective design consequences 165
17.3 Design consequences for similar scenes 166
17.4 Design consequences for different scenes 168
17.5 Structure of an extension profile 170

18.1 Applicable extension mechanisms within the EML framework 172

1.1 Characteristics of the Decorator mechanism 209
1.2 Characteristics of the Plugin mechanism 210
1.3 Characteristics of the Aspects mechanism 211
1.4 Characteristics of the Add-On mechanism 212
1.5 Characteristics of the Hooking mechanism 213
1.6 Characteristics of the Profiling mechanism 214
1.7 Characteristics of the Multilevel Modelling mechanism 215

1.8 Characteristics of the Generalisation/Specialisation mechanism 216
1.9 Characteristics of the Semantic Extension mechanism 217

References

1. Vernadat, F.: *Enterprise Modelling and Integration*. In Kosanke, K., Jochem, R., Nell, J.G., Bas, A.O., eds.: Enterprise Inter- and Intra-Organizational Integration. Springer (2003) 25–33
2. Sandkuhl, K., Lillehagen, F.: *The Early Phases of Enterprise Knowledge Modelling: Practices and Experiences from Scaffolding and Scoping*. In Stirna, J., Persson, A., eds.: The Practice of Enterprise Modeling: First IFIP WG 8.1 Working Conference, PoEM 2008, Stockholm, Sweden, November 12-13, 2008. Springer (2008) 1–14
3. Frank, U.: *Enterprise Modelling: The Next Steps*. Enterprise Modelling and Information Systems Architectures 9(1) (2014) 22–37
4. Lankhorst, M.: *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer (2009)
5. Frank, U.: *Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages*. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences. (2002) 1258–1267
6. Vernadat, F.: *Enterprise Modeling in the Context of Enterprise Engineering: State of the Art and Outlook*. International Journal of Production Management and Engineering 2(2) (2014) 57–73
7. Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., Tang, A.: *What Industry Needs from Architectural Languages: A Survey*. IEEE Transactions on Software Engineering 39(6) (2013) 869–891
8. Bork, D., Fill, H.G.: *Formal Aspects of Enterprise Modeling Methods: A Comparison Framework*. In Sprague, R.H., ed.: Proceedings of the 47th Annual Hawaii International Conference on System Sciences, Waikoloa, Hawaii, USA, January 6-9, 2014. (2014) 3400–3409
9. Braun, R.: *SemFrameX – Towards a Framework for the Semantic Justification of BPMN Adaptations*. In Hochreiner, C., Schulte, S., eds.: Proceedings of the 8th ZEUS Workshop, Vienna, Austria, January 27-28, 2016. Volume 1562 of CEUR Workshop Proceedings. (2016) 13–20
10. Lankhorst, M.M., Proper, H.A., Jonkers, H.: *The Architecture of the ArchiMate Language*. In Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R., eds.: Enterprise, Business-Process and Information Systems Modeling: 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings. Springer (2009) 367–380
11. Sandkuhl, K., Stirna, J., Persson, A., Wissotzki, M.: *Enterprise Modeling – Tackling Business Challenges with the 4EM Method*. Springer (2014)
12. Frank, U.: *Multi-Perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges*. Software & Systems Modeling 13(3) (2014) 941–962
13. Wand, Y., Weber, R.: *Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda*. Information Systems Research 13(4) (2002) 363–376
14. Pfeiffer, D., Gehlert, A.: *A Framework for Comparing Conceptual Models*. In Desel, J., Frank, U., eds.: Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2015), Klagenfurt, Austria, October 24-25, 2005. Volume 75 of Lecture Notes in Informatics. (2005) 108–122
15. Object Management Group (OMG): *Business Process Model and Notation (BPMN), Version 2.0*. (2011)
16. Chinosi, M., Trombetta, A.: *BPMN: An Introduction to the Standard*. Computer Standards & Interfaces 34(1) (2012) 124–134
17. The Open Group: *Archimate 2.0 Specification*. Van Haren Publishing (2012)
18. Scheer, A.W., Nüttgens, M.: *ARIS Architecture and Reference Models for Business Process Management*. In van der Aalst, W., Desel, J., Oberweis, A., eds.: Business Process Management: Models, Techniques, and Empirical Studies. Springer (2000) 376–389

19. Riehle, D.M., Jannaber, S., Karhof, A., Thomas, O., Delfmann, P., Becker, J.: *On the De-facto Standard of Event-Driven Process Chains: How EPC is Defined in Literature*. In Oberweis, A., Reussner, R., eds.: Modellierung 2016, 2.-4. März 2016, Karlsruhe, Deutschland. Volume 254 of Lecture Notes in Informatics. (2016) 61–76
20. Proper, H.A., Verrijn-Stuart, A.A., Hoppenbrouwers, S.: *On Utility-Based Selection of Architecture-Modelling Concepts*. In Hartmann, S., Stumptner, M., eds.: Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM 2005), Newcastle, NSW, Australia, January/February 2005. Volume 43 of CRPIT. Australian Computer Society (2005) 25–34
21. Bjeković, M., Proper, H.A., Sottet, J.S.: *Towards a Coherent Enterprise Modelling Landscape*. In Sandkuhl, K., Seigerroth, U., Stirna, J., eds.: Short Paper Proceedings of the 5th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling, Rostock, Germany, November 7-8, 2012. Volume 933 of CEUR Workshop Proceedings. (2012) 1–12
22. Bjeković, M., Proper, H.A., Sottet, J.: *Enterprise Modelling Languages – Just Enough Standardisation*. In Shishkov, B., ed.: Business Modeling and Software Design - Third International Symposium, BMSD 2013, Noordwijkerhout, The Netherlands, July 8-10, 2013, Revised Selected Papers. Volume 173 of Lecture Notes in Business Information Processing. Springer (2014) 1–23
23. Del Fabro, M.D., Valduriez, P.: *Towards the Efficient Development of Model Transformations Using Model Weaving and Matching Transformations*. Software & Systems Modeling 8(3) (2009) 305–324
24. Živković, S., Karagiannis, D.: *Towards Metamodelling-in-the-Large: Interface-Based Composition for Modular Metamodel Development*. In Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q., eds.: Enterprise, Business-Process and Information Systems Modeling: 16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015. Springer International Publishing (2015) 413–428
25. Frank, U.: *Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines*. In Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J., eds.: Domain Engineering, Product Lines, Languages, and Conceptual Models. Springer (2013) 133–157
26. Atkinson, C., Gerbig, R., Fritzsche, M.: *Modeling Language Extension in the Enterprise Systems Domain*. In: 17th IEEE International Enterprise Distributed Object Computing Conference, Vancouver, BC, 2013. (2013) 49–58
27. Braun, R., Schlieter, H., Burwitz, M., Esswein, W.: *BPMN4CP: Design and Implementation of a BPMN Extension for Clinical Pathways*. In Zheng, H.J., Dubitzky, W., Hu, X., Hao, J., Berrar, D.P., Cho, K., Wang, Y., Gilbert, D.R., eds.: 2014 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2014, Belfast, United Kingdom, November 2-5, 2014. (2014) 9–16
28. Braun, R.: *Towards the State of the Art of Extending Enterprise Modeling Languages*. In Hammoudi, S., Pires, L.F., Desfray, P., Filipe, J., eds.: MODELSWARD 2015 - Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, ESEO, Angers, Loire Valley, France, 9-11 February, 2015. (2015) 394–402
29. Loos, P., Mettler, T., Winter, R., Goeken, M., Frank, U., Winter, A.: *Methodological Pluralism in Business and Information Systems Engineering*. Business & Information Systems Engineering 5(6) (2013) 453–460
30. Patig, S.: *Evolution of Entity Relationship Modelling*. Data & Knowledge Engineering 56(2) (2006) 122–138
31. Pardillo, J.: *A Systematic Review on the Definition of UML Profiles*. In Petriu, D.C., Rouquette, N., Haugen, Ø., eds.: Model Driven Engineering Languages and Systems: 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Proceedings, Part I. Springer (2010) 407–422
32. Azevedo, C.L., Almeida, J.P.A., van Sinderen, M., Quartel, D., Guizzardi, G.: *An Ontology-Based Semantics for the Motivation Extension to Archimate*. In: 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), IEEE (2011) 25–34
33. Braun, R., Esswein, W.: *Classification of Domain-Specific BPMN Extensions*. In Frank, U., Loucopoulos, P., Pastor, O., Petrounias, I., eds.: The Practice of Enterprise Modeling - 7th IFIP WG 8.1 Working Conference, PoEM 2014, Manchester, UK, November 12-13, 2014. Proceedings. Volume 197 of Lecture Notes in Business Information Processing. Springer (2014) 42–57
34. Kopp, O., Görlach, K., Karastoyanova, D., Leymann, F., Reiter, M., Schumm, D., Sonntag, M., Strauch, S., Unger, T., Wieland, M., Khalaf, R.: *A Classification of BPEL Extensions*. Journal of Systems Integration 2(4) (2011) 3–28
35. de Kinderen, S., Ma, Q., Proper, H.A.: *Model Bundling: Towards a Value-Based Componential Approach for Language Engineering*. In: 8th International Workshop on Value Modelling and Business Ontology (VMBO 2014), Berlin, Germany, March 3-4, 2014. (2014) 1–9
36. Braun, R., Esswein, W.: *Extending BPMN for Modeling Resource Aspects in the Domain of Machine Tools*. WIT Transactions of Engineering 87 (2014) 450–458
37. Wiemer, H., Neidhardt, L., Esswein, W., Braun, R.: *Technical and Economic Benchmarking Guideline for the Compensation and Correction of Thermally Induced Machine Tool Errors*. In Großmann, K.,

ed.: Thermo-energetic Design of Machine Tools: A Systemic Approach to Solve the Conflict between Power Efficiency, Accuracy and Productivity Demonstrated At the Example of Machining Production. Lecture Notes in Production Engineering. Springer International Publishing (2015) 233–246
38. Object Management Group (OMG): *Meta Object Facility (MOF) Core Specification, Version 2.4.2*. (2014)
39. Braun, R.: *Behind the Scenes of the BPMN Extension Mechanism – Principles, Problems and Options for Improvement*. In Hammoudi, S., Pires, L.F., Desfray, P., Filipe, J., eds.: MODELSWARD 2015 - Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, ESEO, Angers, Loire Valley, France, 9-11 February, 2015. (2015) 403–410
40. Bjeković, M., Proper, H.A., Sottet, J.S.: *Embracing Pragmatics*. In Yu, E., Dobbie, G., Jarke, M., Purao, S., eds.: Conceptual Modeling: 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings, Springer International Publishing (2014) 431–444
41. Braun, R.: *Epistemological Foundations for the Integrated and Multifaceted Description of Stipulated Semantics in Enterprise Modeling Languages*. In: 19. Interuniversitäres Doktorandenseminar Wirtschaftsinformatik. (2015)
42. Braun, R.: *BPMN Extension Profiles – Adapting the Profile Mechanism for Integrated BPMN Extensibility*. In: 17th IEEE Conference on Business Informatics, CBI 2015, Lisbon, Portugal, July 13-16, 2015 - Volume 1. (2015) 133–142
43. Burwitz, M., Schlieter, H., Esswein, W.: *Modeling Clinical Pathways – Design and Application of a Domain-Specific Modeling Language*. In Alt, R., Franczyk, B., eds.: Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013), February 27 - March 01, 2013, Leipzig, Germany. (2013)
44. Braun, R., Esswein, W.: *Towards Multi-Perspective Modeling with BPMN*. In Aveiro, D., Pergl, R., Valenta, M., eds.: Advances in Enterprise Engineering IX - 5th Enterprise Engineering Working Conference, EEWC 2015, Prague, Czech Republic, June 15-19, 2015, Proceedings. Volume 211 of Lecture Notes in Business Information Processing. Springer (2015) 67–81
45. Braun, R., Esswein, W.: *Towards an Integrated Method for the Extension of MOF-Based Modeling Languages*. In Bellatreche, L., Manolopoulos, Y., eds.: Model and Data Engineering – 5th International Conference, MEDI 2015, Rhodes, Greece, September 26-28, 2015, Proceedings. Number 9344 in Lecture Notes in Computer Science, Springer (2015) 103–115
46. Karagiannis, D., Kühn, H.: *Metamodelling Platforms*. In Bauknecht, K., Min Tjoa, A., Quirchmayer, G., eds.: Proceedings of the Third International Conference EC-Web 2002, Dexa 2002, Aix-en-Provence, France, September 2-6, 2002. Volume 2455 of Lecture Notes in Computer Science. Springer (2002) 182–195
47. Greiffenberg, S.: *Methodenentwicklung in Wirtschaft und Verwaltung*. Kovač (2004)
48. Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: *Extending BPMN 2.0: Method and Tool Support*. In Dijkman, R., Hofstetter, J., Koehler, J., eds.: Business Process Model and Notation. Volume 95 of Lecture Notes in Business Information Processing. Springer (2011) 59–73
49. Mendling, J., Neumann, G., Nüttgens, M.: *Yet Another Event-driven Process Chain*. In van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F., eds.: Business Process Management: 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005. Proceedings, Springer (2005) 428–433
50. Atkinson, C., Kühne, T.: *A Tour of Language Customization Concepts*. Advances in Computers 70 (2007) 105–161
51. Weller, J., Esswein, W.: *Consequences of Meta-Model Modifications within Model Configuration Management*. In Brockmans, S., Jung, J., Sure, Y., eds.: Proceedings of the 2nd International Workshopon Meta-Modelling, WoMM 2006, October 12-13, 2006, Karlsruhe, Germany. Volume 96 of Lecture Notes in Informatics. (2006) 125–139
52. Chmielewicz, K.: *Forschungskonzeptionen der Wirtschaftswissenschaft*. Schäffer-Poeschel (1994)
53. Winter, R.: *Design Science Research in Europe*. European Journal of Information Systems 17(5) (2008) 470–475
54. Albani, A., Raber, D., Winter, R.: *A Conceptual Framework for Analysing Enterprise Engineering Methodologies*. Enterprise Modelling and Information Systems Architectures 11(1) (2016) 1–26
55. Braun, R., Schlieter, H.: *Requirements-Based Development of BPMN Extensions: The Case of Clinical Pathways*. In Heinrich, R., Kirchner, K., Weißbach, R., eds.: 1st IEEE International Workshop on the Interrelations Between Requirements Engineering and Business Process Management, REBPM 2014, Karlskrona, Sweden, August 25, 2014. (2014) 39–44
56. De Kinderen, S., Ma, Q.: *Requirements Engineering for the Design of Conceptual Modeling Languages – A Goal-and Value-Oriented Approach*. Applied Ontology (2015) 7–24
57. Braun, R.: *Towards a Multi-Faceted Framework for Semantics in Enterprise Modeling Languages*. In Betz, S., Reimer, U., eds.: Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband. Volume 255 of Lecture Notes in Informatics. (2016) 33–44

58. de Kinderen, S., Gaaloul, K., Proper, H.A.: *Bridging Value Modelling to ArchiMate via Transaction Modelling*. Software & Systems Modeling 13(3) (2014) 1043–1057
59. Braun, R.: *Meta Model Extensibility of BPMN: Current Limitations and Proposed Improvements*. In Desfray, P., Filipe, J., Hammoudi, S., Luís, Pires, F., eds.: Model-Driven Engineering and Software Development - Third International Conference, MODELSWARD 2015, Angers, France, February 9-11, 2015, Revised Selected Papers. Volume 580 of Communications in Computer and Information Science. (2015) 230–247
60. Frank, U.: *The MEMO Meta Modelling Language (MML) and Language Architecture*. ICB Research Report 24, Universität Duisburg-Essen (2011)
61. Weisemöller, I., Schürr, A., Weisemöller, I., Schürr, A.: *A Comparison of Standard Compliant Ways to Define Domain Specific Languages*. In Giese, H., ed.: Models in Software Engineering: Workshops and Symposia at MoDELs, 2007, Nashville, TN, USA, September 30 - October 5, 2007, Reports and Revised Selected Papers. Springer (2008) 47–58
62. Harel, D., Rumpe, B.: *Meaningful Modeling: What's the Semantics of Semantics*. Computer 37(10) (2004) 64–72
63. Hausmann, J.H.: *Dynamic Meta Modeling: A Semantics Description Technique for Visual Modeling Languages*. PhD thesis, University of Paderborn (2005)
64. Siau, K.: *The Psychology of Information Modeling*. In Siau, K., ed.: Advanced Topics in Database Research. Volume 1. (2003) 106–118
65. Santos, P.S., Almeida, J.P.A., Guizzardi, G.: *An Ontology-Based Analysis and Semantics for Organizational Structure Modeling in the ARIS Method*. Information Systems 38(5) (2013) 690–708
66. Harzallah, M., Berio, G., Opdahl, A.L.: *New Perspectives in Ontological Analysis: Guidelines and Rules for Incorporating Modelling Languages into UEML*. Information Systems 37(5) (2012) 484–507
67. Braun, R., Esswein, W.: *Towards Hybrid Semantics of Enterprise Modeling Languages*. In Hammoudi, S., Pires, L.F., Selic, B., Desfray, P., eds.: Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development. (2016) 412–420
68. Karagiannis, D.: *Agile Modeling Method Engineering*. In: Proceedings of the 19th Panhellenic Conference on Informatics, ACM (2015) 5–10
69. Visić, N., Fill, H.G., Buchmann, R.A., Karagiannis, D.: *A Domain-Specific Language for Modeling Method Definition: From Requirements to Grammar*. In Rolland, C., Anagnostopoulos, D., Loucopoulos, P., Gonzalez-Perez, C., eds.: 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), Athens, 2015, IEEE (2015) 286–297
70. Patig, S.: *Die Evolution von Modellierungssprachen*. Frank & Timme GmbH (2006)
71. Hevner, A., Chatterjee, S.: Design Science Research in Information Systems. In: *Design Research in Information Systems*. Volume 22 of Integrated Series in Information Systems. Springer (2010) 9–22
72. Offermann, P., Blom, S., Schönherr, M., Bub, U.: *Artifact Types in Information Systems Design Science – A Literature Review*. In Winter, R., Zhao, J.L., Aier, S., eds.: Global Perspectives on Design Science Research: 5th International Conference, DESRIST 2010, St. Gallen, Switzerland, June 4-5, 2010. Volume 6105 of Lecture Notes in Computer Science. Springer (2010) 77–92
73. Braun, R., Esswein, W.: *Semantics in the Context of BPMN Extensions – State of Affairs and Research Challenges*. In Nissen, V., Stelzer, D., Straßburger, S., Fischer, D., eds.: Tagungsband Multikonferenz Wirtschaftsinformatik 2016, Band II. (2016) 1119–1130
74. Braun, R., Esswein, W.: *Entwicklung einer BPMN-Extension für ressourcen-intensive Prozesse im Maschinenbau*. In Kundisch, D., Suhl, L., Beckmann, L., eds.: Tagungsband Multikonferenz Wirtschaftsinformatik 2014. (2014) 1574–1586
75. Braun, R., Schlieter, H., Burwitz, M., Esswein, W.: *Extending a Business Process Modeling Language for Domain-Specific Adaptation in Healthcare*. In Thomas, O., Teuteberg, F., eds.: Smart Enterprise Engineering: 12. Internationale Tagung Wirtschaftsinformatik, WI 2015, Osnabrück, Germany, March 4-6, 2015. (2015) 468–481
76. Braun, R., Schlieter, H., Burwitz, M., Esswein, W.: *BPMN4CP Revised – Extending BPMN for Multi-perspective Modeling of Clinical Pathways*. In Bui, T.X., Jr., R.H.S., eds.: 49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016. (2016) 3249–3258
77. Braun, R., Burwitz, M., Schlieter, H., Benedict, M.: *Clinical Processes from Various Angles – Amplifying BPMN for Integrated Hospital Management*. In Huan, J., Miyano, S., Shehu, A., Hu, X.T., Ma, B., Rajasekaran, S., Gombar, V.K., Schapranow, M., Yoo, I., Zhou, J., Chen, B., Pai, V., Pierce, B.G., eds.: 2015 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2015, Washington, DC, USA, November 9-12, 2015. (2015) 837–845
78. Braun, R.: *Syntactical Enrichment of the Meta Object Facility for the Definition of Enterprise Modeling Languages*. Unpublished Paper (2016)
79. Braun, R., Benedict, M., Wendler, H., Esswein, W.: *Proposal for Requirements Driven Design Science Research*. In Donnellan, B., Helfert, M., Kenneally, J., VanderMeer, D., Rothenberger, M., Winter, R., eds.: New Horizons in Design Science: Broadening the Research Agenda - 10th International

Conference, DESRIST 2015, Dublin, Ireland, May 20-22, 2015, Proceedings. Number 9073 in Lecture Notes in Computer Science, Springer (2015) 135–151
80. Braun, R., Esswein, W.: *A Generic Framework for Modifying and Extending Enterprise Modeling Languages*. In Hammoudi, S., Maciaszek, L.A., Teniente, E., eds.: ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, Barcelona, Spain, April 27-30, 2015. (2015) 277–286
81. Hevner, A.R.: *The Three Cycle View of Design Science Research*. Scandinavian Journal of Information Systems 19(2) (2007) Article 4
82. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: *A Design Science Research Methodology for Information Systems Research*. Journal of Management Information Systems 24(3) (2007) 45–77
83. Alturki, A., Gable, G.G., Bandara, W.: *A Design Science Research Roadmap*. In Jain, H., Sinha, A.P., Vitharana, P., eds.: Service-Oriented Perspectives in Design Science Research: 6th International Conference, DESRIST 2011, Milwaukee, WI, USA, May 5-6, 2011. Proceedings. Springer (2011) 107–123
84. Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J.: *Memorandum on Design-Oriented Information Systems Research*. European Journal of Information Systems 20(1) (2011) 7–10
85. Frank, U.: *Towards a Pluralistic Conception of Research Methods in Information Systems Research*. Technical report, University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB) (2006)
86. Verband der Hochschullehrer für Betriebswirtschaft e.V. (VHB): *VHB-JOURQUAL 3 - Teilrating Wirtschaftsinformatik*. VHB (2016)
87. Die Sprecher der Wissenschaftlichen Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft (WKWI) und des Fachbereichs Wirtschaftsinformatik der Gesellschaft für Informatik (GI-FB WI): *WI-Orientierungslisten*. Wirtschaftsinformatik 50(2) (2008) 155–163
88. Wilde, T., Hess, T.: *Forschungsmethoden der Wirtschaftsinformatik*. Wirtschaftsinformatik 49(4) (2007) 280–287
89. Fox, M.S., Gruninger, M.: *Enterprise Modeling*. AI Magazine 19(3) (1998) 109–121
90. Kosanke, K., Nell, J.G.: *Enterprise Engineering and Integration: Building International Consensus: Proceedings of ICEIMT 97, International Conference on Enterprise Integration and Modeling Technology, Torino, Italy, October 28–30, 1997*. Springer Science & Business Media (1997)
91. Fowler, M.: *Domain-Specific Languages*. Pearson Education (2010)
92. Krahn, H., Rumpe, B., Völkel, S.: *MontiCore: A Framework for Compositional Development of Domain Specific Languages*. International Journal on Software Tools for Technology Transfer 12(5) (2010) 353–372
93. Breu, K., Hemingway, C.J., Strathern, M., Bridger, D.: *Workforce Agility: The New Employee Strategy for the Knowledge Economy*. Journal of Information Technology 17(1) (2002) 21–31
94. Sherehiy, B., Karwowski, W., Layer, J.K.: *A Review of Enterprise Agility: Concepts, Frameworks, and Attributes*. International Journal of Industrial Ergonomics 37(5) (2007) 445–460
95. Cruse, D.A.: *Meaning in Language: An Introduction to Semantics and Pragmatics*. Oxford University Press UK (2011)
96. Strecker, S., Heise, D., Frank, U.: *RiskM: A Multi-Perspective Modeling Method for IT Risk Assessment*. Information Systems Frontiers 13(4) (2011) 595–611
97. Becker, J.: *Interview with Reinhard Schütte on "Managing Large-Scale BPM Projects"*. Business & Information Systems Engineering 6(4) (2014) 1–3
98. Gehlert, A., Buckmann, U., Esswein, W.: *Ontology Based Method Engineering*. Proceedings of the 11th Americas Conference on Information Systems, Omaha, NE, USA, August 11-14 2005 (2005) 2824–2833
99. Pfeiffer, D., Niehaves, B.: *Evaluation of Conceptual Models – A Structuralist Approach*. In Bartmann, D., Rajola, F., Kallinikos, J., Avison, D.E., Winter, R., Ein-Dor, P., Becker, J., Bodendorf, F., Weinhardt, C., eds.: Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy, ECIS 2005, Regensburg, Germany, May 26-28, 2005. (2005) 459–470
100. Höfferer, P.: *Achieving Business Process Model Interoperability Using Metamodels and Ontologies*. In Österle, H., Schelp, J., Winter, R., eds.: Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007, St. Gallen, Switzerland, 2007. (2007) 1620–1631
101. van der Linden, D.: *Personal Semantics of Meta-Concepts in Conceptual Modeling Languages*. PhD thesis, Radboud University Nijmegen (2015)
102. Moody, D.L.: *The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*. IEEE Transactions on Software Engineering 35(6) (2009) 756–779

103. Riehle, D.M., Jannaber, S., Karhof, A., Delfmann, P., Thomas, O., Becker, J.: *Towards an EPC Standardization – A Literature Review on Exchange Formats for EPC Models*. In Nissen, V., Stelzer, D., Straßburger, S., Fischer, D., eds.: Tagungsband Multikonferenz Wirtschaftsinformatik 2016, Band II. (2016) 1167–1178
104. David, P.A., Greenstein, S.: *The Economics of Compatibility Standards: An Introduction to Recent Research*. Economics of Innovation and New Technology 1(1-2) (1990) 3–41
105. Fomin, V., Keil, T., Lyytinen, K.: *Theorizing About Standardization: Integrating Fragments of Process Theory in Light of Telecommunication Standardization Wars*. Sprouts: Working Papers on Information Environments, Systems and Organizations 3(1) (2003) 29–60
106. Object Management Group (OMG): *Unified Modeling Language, Version 2.4.1, Part 1: Infrastructure: 19505-1:2012*. OMG. (2012)
107. Object Management Group (OMG): *Unified Modeling Language, Version 2.4.1, Part 2: Superstructure: 19505-2:2012*. (2012)
108. Decker, G., Puhlmann, F.: *Extending BPMN for Modeling Complex Choreographies*. In Meersman, R., Tari, Z., eds.: On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS: OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I. Springer (2007) 24–40
109. Falkenberg, E.D., Hesse, W., Lindgreen, P., Nilsson, B.E., Oei, J.H., Rolland, C., Stamper, R.K., Van Assche, F.J., Verrijn-Stuart, A.A., Voss, K.: *A Framework of Information Systems Concepts*. Technical report, IFIP (1998)
110. Becker, J., Holten, R., Knackstedt, R., Niehaves, B.: *Epistemologische Positionierungen in der Wirtschaftsinformatik am Beispiel einer konsensorientierten Informationsmodellierung*. Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik – Theoriebildung und -bewertung, Ontologien, Wissensmanagement. Wiesbaden (2004) 335–366
111. Morales-Trujillo, M.E., Oktaba, H., Piattini, M.: *The Making of an OMG Standard*. Computer Standards & Interfaces 42 (2015) 84–94
112. Natschläger, C.: *Towards a BPMN 2.0 Ontology*. In Dijkman, R., Hofstetter, J., Koehler, J., eds.: Business Process Model and Notation: Third International Workshop, BPMN 2011, Lucerne, Switzerland, November 21-22, 2011. Proceedings. Springer (2011) 1–15
113. Parnas, D.L.: *Designing Software for Ease of Extension and Contraction*. IEEE Transactions on Software Engineering (2) (1979) 128–138
114. Krishnamurthi, S., Felleisen, M.: *Toward a Formal Theory of Extensible Software*. In: Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering. SIGSOFT 98/FSE-6 (1998) 88–98
115. Rytter, M., Jørgensen, B.N.: *Independently Extensibile Contexts*. In Babar, M.A., Gorton, I., eds.: Software Architecture: 4th European Conference, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. Proceedings. Springer (2010) 327–334
116. Esswein, W., Weller, J.: *Method Modifications in a Configuration Management Environment*. In Österle, H., Schelp, J., Winter, R., eds.: Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007, St. Gallen, Switzerland, 2007. (2007) 2002–2013
117. Fernández, H.F., Palacios-González, E., García-Díaz, V., García-Bustelo, C.P., Martínez, O.S., Lovelle, J.M.C.: *SBPMN – An Easier Business Process Modeling Notation for Business Users*. Computer Standards & Interfaces 32(1) (2010) 18–28
118. Bae, J.H., Lee, K.M., Chae, H.S.: *Modularization of the UML Metamodel Using Model Slicing*. In Latifi, S., ed.: Fifth International Conference on Information Technology: New Generations, 2008, April 7-9, 2008, Las Vegas, Nevada. (2008) 1253–1254
119. zur Muehlen, M., Recker, J.: *How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation*. In Bellahsène, Z., Léonard, M., eds.: Advanced Information Systems Engineering: 20th International Conference, CAiSE 2008 Montpellier, France, June 16-20, 2008. Proceedings, Springer (2008) 465–479
120. Fondement, F., Muller, P.A., Thiry, L., Wittmann, B., Forestier, G.: *Big Metamodels Are Evil - Package Unmerge - A Technique for Downsizing Metamodels*. In Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P., eds.: Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings. Springer (2013) 138–153
121. Indulska, M., Recker, J., Rosemann, M., Green, P.: *Business Process Modeling: Current Issues and Future Challenges*. In van Eck, P., Gordijn, J., Wieringa, R., eds.: Advanced Information Systems Engineering: 21st International Conference, CAiSE 2009, Amsterdam, The Netherlands, June 8-12, 2009. Proceedings, Springer (2009) 501–514
122. Recker, J.: *Opportunities and Constraints: The Current Struggle with BPMN*. Business Process Management Journal 16(1) (2010) 181–201
123. Frank, U.: *Multilevel Modeling*. Business & Information Systems Engineering 6(6) (2014) 319–337

124. Stachowiak, H.: *Allgemeine Modelltheorie*. Springer (1973)
125. Esswein, W., Weller, J.: *Unternehmensarchitekturen – Grundlagen, Verwendung und Frameworks*. HMD Praxis der Wirtschaftsinformatik 45(4) (2008) 6–18
126. Frank, U.: *Multi-Perspective Enterprise Modelling: Background and Terminological Foundation*. Technical report, University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB) (2011)
127. Adam, S., Esswein, W.: *Untersuchung von Architekturframeworks zur Strukturierung von Unternehmensmodellen*. Dresdner Beiträge zur Wirtschaftsinformatik 50 (2007) 1–25
128. Kulesza, U., Alves, V., Garcia, A., de Lucena, C.J.P., Borba, P.: *Improving Extensibility of Object-Oriented Frameworks with Aspect-Oriented Programming*. In Morisio, M., ed.: Reuse of Off-the-Shelf Components: 9th International Conference on Software Reuse, ICSR 2006 Turin, Italy, June 12-15, 2006. Proceedings. Springer (2006) 231–245
129. Aagesen, G., Krogstie, J.: *BPMN 2.0 for Modeling Business Processes*. In vom Brocke, J., Rosemann, M., eds.: Handbook on Business Process Management 1: Introduction, Methods, and Information Systems. Springer (2015) 219–250
130. Yu, E.S.: *Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering*. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997. IEEE (1997) 226–235
131. Heaven, W., Finkelstein, A.: *UML Profile to Support Requirements Engineering with KAOS*. IEE Proceedings - Software 151(1) (2004) 10–27
132. Pohl, K.: *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer (2010)
133. Kelly, S., Tolvanen, J.P.: *Visual Domain-Specific Modeling: Benefits and Experiences of Using Meta-CASE Tools*. In Bezivin, J., Ernst, J., eds.: Proceedings of the International Workshop on Model Engineering, ECOOP 2000. Volume 2000. (2000)
134. Kleppe, A.: *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Pearson Education (2008)
135. Frank, U.: *Outline of a Method for Designing Domain-Specific Modelling Languages*. ICB Research Report 42, University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB) (2010)
136. Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L.C., Visser, E., Wachsmuth, G.: *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. dslbook.org (2013)
137. Saleem, M., Jaafar, J., Hassan, M.: *A Domain-Specific Language for Modelling Security Objectives in a Business Process Models of SOA Applications*. Advances in Information Sciences and Service Sciences 4(1) (2012) 353–362
138. Frank, U.: *Some Guidelines for the Conception of Domain-Specific Modelling Languages*. In Nüttgens, M., Thomas, O., Weber, B., eds.: Enterprise Modelling and Information Systems Architectures: Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011, Hamburg, Germany, September 22-23, 2011. Volume 190 of Lecture Notes in Informatics. (2011) 93–106
139. Gehlert, A.: *Migration fachkonzeptueller Modelle*. Logos (2007)
140. Jung, J.S.: *Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessftsprozessmodellierung*. Logos (2007)
141. Heise, D., Strecker, S., Frank, U.: *ControlML: A Domain-Specific Modeling Language in Support of Assessing Internal Controls and the Internal Control System*. International Journal of Accounting Information Systems 15(3) (2013) 224–245
142. Murata, T.: *Petri Nets: Properties, Analysis and Applications*. Proceedings of the IEEE 77(4) (1989) 541–580
143. Overbeek, S., Frank, U., Köhling, C.: *A Language for Multi-Perspective Goal Modelling: Challenges, Requirements and Solutions*. Computer Standards & Interfaces 38 (2015) 1–16
144. Frank, U., Heise, D., Kattenstroth, H., Ferguson, D., Hadar, E., Waschke, M.: *ITML: A Domain-Specific Modeling Language for Supporting Business Driven IT Management*. In Tolvanen, J.P., Rossi, M., Gray, J., Sprinkle, J., eds.: Proceedings of the 9th Workshop on Domain-Specific Modeling (DSM) at the International Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA), Orlando, Florida, USA. (2009)
145. Hamann, L., Gogolla, M.: *Endogenous Metamodeling Semantics for Structural UML 2 Concepts*. In: Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings. (2013) 488–504
146. Kossak, F., Illibauer, C., Geist, V., Kubovy, J., Natschläger, C., Ziebermayr, T., Kopetzky, T., Freudenthaler, B., Schewe, K.D.: *A Rigorous Semantics for BPMN 2.0 Process Diagrams*. Springer International Publishing (2014)

147. Selic, B.: *A Systematic Approach to Domain-Specific Language Design Using UML*. In de Miguel, M., Kalogeraki, V., Kim, D.H., eds.: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing. (2007) 2–9
148. Mohagheghi, P., Haugen, Ø.: *Evaluating Domain-Specific Modelling Solutions*. In Trujillo, J., Dobbie, G., Kangassalo, H., Hartmann, S., Kirchberg, M., Rossi, M., Reinhartz-Berger, I., Zimányi, E., Frasincar, F., eds.: Advances in Conceptual Modeling – Applications and Challenges: ER 2010 Workshops ACM-L, CMLSA, CMS, DEAER, FP-UML, SeCoGIS, WISM, Vancouver, BC, Canada, November 1-4, 2010. Proceedings. Springer (2010) 212–221
149. Oxford University Press: *Oxford English Dictionary* (1989)
150. Lippi-Green, R.: *English with an Accent: Language, Ideology, and Discrimination in the United States*. Psychology Press (1997)
151. Petyt, K.M.: *The Study of Dialect: An Introduction to Dialectology*. Westview Press (1980)
152. Pierce, B.C.: *Types and Programming Languages*. MIT Press (2002)
153. Zor, S., Leymann, F., Schumm, D.: *A Proposal of BPMN Extensions for the Manufacturing Domain*. In Duffie, N., Westkämper, E., eds.: Proceedings of 44th CIRP International Conference on Manufacturing Systems. (2011)
154. Lara, J.A., Lizcano, D., Martínez, M.A., Pazos, J., Riera, T.: *A UML Profile for the Conceptual Modelling of Structurally Complex Data: Easing Human Effort in the KDD Process*. Information and Software Technology 56(3) (2014) 335–351
155. Langer, P., Wieland, K., Wimmer, M., Cabot, J.: *EMF Profiles: A Lightweight Extension Approach for EMF Models*. Journal of Object Technology 11(1) (2012) 1–29
156. Braun, R., Esswein, W.: *Designing Dialects of Enterprise Modeling Languages with the Profiling Technique*. In Hallé, S., Mayer, W., Ghose, A.K., Grossmann, G., eds.: 19th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2015, Adelaide, Australia, September 21-25, 2015. (2015) 60–67
157. Bravenboer, M., de Groot, R., Visser, E.: *MetaBorg in Action: Examples of Domain-Specific Language Embedding and Assimilation Using Stratego/XT*. In Lämmel, R., Saraiva, J., Visser, J., eds.: Generative and Transformational Techniques in Software Engineering: International Summer School, GTTSE 2005, Braga, Portugal, July 4-8, 2005. Revised Papers. Springer (2006) 297–311
158. Zenger, M.: *Programming Language Abstractions for Extensible Software Components*. PhD thesis, École Polytechnique Fédérale de Lausanne (2004)
159. Klatt, B., Krogmann, K.: *Software Extension Mechanisms*. TH Karlsruhe, Research Report 8 (2008) 2008
160. Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice*. Addison Wesley (1998)
161. Braun, R., Esswein, W.: *Extending the MOF for the Adaptation of Hooks, Aspects, Plug-Ins and Add-Ons*. In Bellatreche, L., Manolopoulos, Y., eds.: Model and Data Engineering – 5th International Conference, MEDI 2015, Rhodes, Greece, September 26-28, 2015, Proceedings. Volume 9344 of Lecture Notes in Computer Science. Springer (2015) 28–38
162. Atkinson, C., Kühne, T.: *The Essence of Multilevel Metamodeling*. In Gogolla, M., Kobryn, C., eds.: UML 2001 – The Unified Modeling Language. Modeling Languages, Concepts, and Tools: 4th International Conference Toronto, Canada, October 1-5, 2001. Proceedings. Springer (2001) 19–33
163. Atkinson, C., Gutheil, M., Kennel, B.: *A Flexible Infrastructure for Multilevel Language Engineering*. IEEE Transactions on Software Engineering 35(6) (2009) 742–755
164. Martin, R.C.: *Design Principles and Design Patterns*. Object Mentor 1 (2000) 1–34
165. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-oriented Software*. Pearson Education (1994)
166. Wolfinger, R.: *Plug-In Architecture and Design Guidelines for Customizable Enterprise Applications*. In Harris, G.E., ed.: Companion to the 23rd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2008, October 19-23, 2007, Nashville, TN, USA, ACM (2008) 893–894
167. Marquardt, K.: *Patterns for Plug-Ins*. In Dyson, P., Devos, M., eds.: Proceedings of the 4th European Conference on Pattern Languages of Programms (EuroPLoP 1999), Irsee, Germany, July 7-11, 1999. (1999) 203–232
168. Kühn, H., Bayer, F., Junginger, S., Karagiannis, D.: *Enterprise Model Integration*. In Bauknecht, K., Tjoa, A.M., Quirchmayr, G., eds.: E-Commerce and Web Technologies: 4th International Conference, EC-Web, Prague, Czech Republic, September 2-5, 2003. Proceedings. Springer (2003) 379–392
169. Živković, S., Kuhn, H., Karagiannis, D.: *Facilitate Modelling Using Method Integration: An Approach Using Mappings and Integration Rules*. In Österle, H., Schelp, J., Winter, R., eds.: Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007, St. Gallen, Switzerland, 2007. (2007) 2038–2049
170. Jossic, A., Del Fabro, M.D., Lerat, J.P., Bézivin, J., Jouault, F.: *Model Integration with Model Weaving: A Case Study in System Architecture*. In: 2007 International Conference on Systems Engineering and Modeling, IEEE (2007) 79–84

171. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: *Aspect-Oriented Programming*. In Akşit, M., Matsuoka, S., eds.: ECOOP 97 – Object-Oriented Programming: 11th European Conference, Jyväskylä, Finland, June 9-13, 1997 Proceedings. Number 1241 in Lecture Notes in Computer Science, Springer (1997) 220–242
172. Birsan, D.: *On Plug-ins and Extensible Architectures*. Queue – Patching and Deployment 3(2) (2005) 40–46
173. Becker, J., Delfmann, P., Knackstedt, R.: *Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models*. In Becker, J., Delfmann, P., eds.: Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models. Physica-Verlag HD (2007) 27–58
174. Becker, J., Delfmann, P.: *Reference Modeling*. Physica-Verlag HD (2007)
175. Strahringer, S.: *Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips*. In Pohl, K., Schürr, A., Vossen, G., eds.: Modellierung 1998, Proceedings des GI-Workshops in Münster, 11.-13. März 1998. Number 9 in CEUR Workshop Proceedings (1998)
176. Atkinson, C., Gerbig, R., Kennel, B.: *On-the-Fly Emendation of Multi-Level Models*. In Vallecillo, A., Tolvanen, J.P., Kindler, E., Störrle, H., Kolovos, D., eds.: Modelling Foundations and Applications: 8th European Conference, ECMFA 2012, Kgs. Lyngby, Denmark, July 2-5, 2012. Proceedings. Volume 7349 of Lecture Notes in Computer Science. Springer (2012) 194–209
177. Frank, U.: *Thoughts on Classification/Instantiation and Generalisation/Specialisation*. Technical report, University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB) (2012)
178. Atkinson, C., Gerbig, R., Kennel, B.: *Symbiotic General-Purpose and Domain-Specific Languages*. In Glinz, M., Murphy, G.C., Pezzè, M., eds.: 34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland. (2012) 1269–1272
179. Odell, J.J.: *Power Types*. Journal of Object-Oriented Programming 7(2) (1994)
180. Gonzalez-Perez, C., Henderson-Sellers, B.: *Metamodelling for Software Engineering*. Wiley Publishing (2008)
181. Clark, T., Sammut, P., Willans, J.: *Super-Languages: Developing Languages and Applications with XMF*. CoRR (2015)
182. Kühne, T.: *Matters of (Meta-) Modeling*. Software & Systems Modeling 5(4) (2006) 369–385
183. Kaczmarek, M.: *Categories of Ontologies' Applications in the Realm of Enterprise Modeling*. In: 17th IEEE Conference on Business Informatics, CBI 2015, Lisbon, Portugal, July 13-16, 2015 - Volume 1. Volume 1. (2015) 98–107
184. Lindland, O.I., Sindre, G., Solvberg, A.: *Understanding Quality in Conceptual Modeling*. IEEE Software 11(2) (1994) 42–49
185. Krogstie, J., Lindland, O.I., Sindre, G.: *Towards a Deeper Understanding of Quality in Requirements Engineering*. In Iivari, J., Lyytinen, K., Rossi, M., eds.: Advanced Information Systems Engineering: 7th International Conference, CAiSE 95, Jyväskylä, Finland, June 12–16, 1995 Proceedings. Volume 932 of Lecture Notes in Computer Science. Springer (1995) 82–95
186. Schuette, R., Rotthowe, T.: *The Guidelines of Modeling – An Approach to Enhance the Quality in Information Models*. In Ling, T.W., Ram, S., Li Lee, M., eds.: Conceptual Modeling – ER 98: 17th International Conference on Conceptual Modeling, Singapore, November 16-19, 1998. Proceedings. Volume 1507 of Lecture Notes in Computer Science. Springer (1998) 240–254
187. Overhage, P.D.S., Birkmeier, D.Q., Schlauderer, S.: *Quality Marks, Metrics, and Measurement Procedures for Business Process Models*. Business & Information Systems Engineering 4(5) (2012) 229–246
188. Weller, J.: *Modellgestützte Prozessverbesserung*. PhD thesis, TU Dresden (2009)
189. Becker, J., Niehaves, B., Knackstedt, R.: *Bezugsrahmen zur epistemologischen Positionierung der Referenzmodellierung*. In Becker, J., Delfmann, P., eds.: Referenzmodellierung: Grundlagen, Techniken und domänenbezogene Anwendung, Physica-Verlag HD (2004) 1–17
190. Kaidalova, J., Seigerroth, U., Kaczmarek, T., Shilov, N.: *Practical Challenges of Enterprise Modeling in the Light of Business and IT Alignment*. In Sandkuhl, K., Seigerroth, U., Stirna, J., eds.: The Practice of Enterprise Modeling: 5th IFIP WG 8.1 Working Conference, PoEM 2012, Rostock, Germany, November 7-8, 2012. Proceedings. Volume 134 of Lecture Notes in Business Information Processing. Springer (2012) 31–45
191. van der Linden, D., Hoppenbrouwers, S.: *Challenges of Identifying Communities with Shared Semantics in Enterprise Modeling*. In Sandkuhl, K., Seigerroth, U., Stirna, J., eds.: The Practice of Enterprise Modeling: 5th IFIP WG 8.1 Working Conference, PoEM 2012, Rostock, Germany, November 7-8, 2012. Proceedings. Volume 134 of Lecture Notes in Business Information Processing. Springer (2012) 160–171
192. Wand, Y., Weber, R.: *On the Ontological Expressiveness of Information Systems Analysis and Design Grammars*. Information Systems Journal 3(4) (1993) 217–237

193. Rosemann, M., Green, P., Indulska, M.: *A Reference Methodology for Conducting Ontological Analyses*. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.W., eds.: Conceptual Modeling – ER 2004: 23rd International Conference on Conceptual Modeling, Shanghai, China, November 8-12, 2004. Proceedings. Volume 3288 of Lecture Notes in Computer Science. Springer (2004) 110–121
194. Wyssusek, B.: *On Ontological Foundations of Conceptual Modelling*. Scandinavian Journal of Information Systems 18(1) (2006) Article 9
195. Brinkkemper, S., Saeki, M., Harmsen, F.: *Meta-Modelling Based Assembly Techniques for Situational Method Engineering*. Information Systems 24(3) (1999) 209–228
196. Pittke, F., Leopold, H., Mendling, J.: *Automatic Detection and Resolution of Lexical Ambiguity in Process Models*. IEEE Transactions on Software Engineering 41(6) (2015) 526–544
197. Ullmann, S.: *Semantics: An Introduction to the Science of Meaning*. Barnes & Noble (1979)
198. van der Linden, D., van Zee, M.: *On the Semantic Feature Structure of Modeling Concepts: An Empirical Study*. In: IEEE 16th Conference on Business Informatics, CBI 2014, Geneva, Switzerland, July 14-17, 2014 - Volume 2, IEEE (2014) 158–165
199. Perelman, C., Olbrechts-Tyteca, L.: *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press (1969)
200. Kamlah, W.: *Philosophische Anthropologie – Sprachkritische Grundlegung und Ethik*. Volume 238 of Hochschultaschenbücher. Bibliographisches Institut, Mannheim (1972)
201. Brinkkemper, S., Saeki, M., Harmsen, F.: *Assembly Techniques for Method Engineering*. In Pernici, B., Thanos, C., eds.: Advanced Information Systems Engineering: 10th International Conference, CAiSE 98, Pisa, Italy, June 8-12, 1998 Proceedings. Volume 1413 of Lecture Notes in Computer Science. Springer (1998) 381–400
202. Henderson-Sellers, B., Ralyté, J.: *Situational Method Engineering: State-of-the-Art Review*. Journal of Universal Computer Science 16(3) (2010) 424–478
203. Harmsen, F., Brinkkemper, S., Oei, J.L.H.: *Situational Method Engineering for Informational System Project Approaches*. In Verrijn-Stuart, A.A., Olle, T.W., eds.: Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, Maastricht, The Netherlands, 26-28 September, 1994. Volume 55 of IFIP Transactions. Elsevier (1994) 169–194
204. Karlsson, F., Ågerfalk, P.J.: *Method Configuration: Adapting to Situational Characteristics while Creating Reusable Assets*. Information and Software Technology 46(9) (2004) 619–633
205. Bucher, T., Klesse, M., Kurpjuweit, S., Winter, R.: *Situational Method Engineering*. In Ralyté, J., Brinkkemper, S., Henderson-Sellers, B., eds.: Situational Method Engineering: Fundamentals and Experiences: Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland. Volume 244 of IFIP – The International Federation for Information Processing (IFIP-AICT). Springer (2007) 33–48
206. Ralyté, J., Rolland, C., Deneckère, R.: *Towards a Meta-Tool for Change-Centric Method Engineering: A Typology of Generic Operators*. In Persson, A., Stirna, J., eds.: Advanced Information Systems Engineering: 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004. Proceedings. Volume 3084 of Lecture Notes in Computer Science. Springer (2004) 202–218
207. Tolvanen, J., Rossi, M.: *MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators*. In Crocker, R., Steele, G.L., eds.: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2003, October 26-30, 2003, Anaheim, CA, USA, ACM (2003) 92–93
208. Gray, J., Neema, S., Tolvanen, J., Gokhale, A.S., Kelly, S., Sprinkle, J.: *Domain-Specific Modeling*. In Fishwick, P.A., ed.: Handbook of Dynamic System Modeling. Chapman and Hall/CRC (2007)
209. Kelly, S., Tolvanen, J.: *Domain-Specific Modeling – Enabling Full Code Generation*. Wiley (2008)
210. Thalheim, B.: *Syntax, Semantics and Pragmatics of Conceptual Modelling*. In Bouma, G., Ittoo, A., Métais, E., Wortmann, H., eds.: Natural Language Processing and Information Systems: 17th International Conference on Applications of Natural Language to Information Systems, NLDB 2012, Groningen, The Netherlands, June 26-28, 2012. Proceedings. Volume 7337 of Lecture Notes in Computer Science. Springer (2012) 1–10
211. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Völkel, S.: *Design Guidelines for Domain Specific Languages*. CoRR 1409.2378 (2014)
212. Mernik, M., Heering, J., Sloane, A.M.: *When and How to Develop Domain-Specific Languages*. ACM Computing Surveys 37(4) (2005) 316–344
213. Fill, H.G., Karagiannis, D.: *On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform*. Enterprise Modelling and Information Systems Architectures 8(1) (2013)
214. Schultz, M., Radloff, M.: *Modeling Concepts for Internal Controls in Business Processes – An Empirically Grounded Extension of BPMN*. In Sadiq, S., Soffer, P., Völzer, H., eds.: Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings. Volume 8659 of Lecture Notes in Computer Science. Springer International Publishing (2014) 184–199

215. Micouin, P.: *Model Based Systems Engineering: Fundamentals and Methods*. John Wiley & Sons (2014)
216. Schmidt, D.C.: *Model-Driven Engineering*. Computer 39(2) (2006) 25–31
217. Blair, G., Bencomo, N., France, R.B.: *Models@ Run. Time*. Computer 42(10) (2009) 22–27
218. Institute of Electrical and Electronics Engineers (IEEE): *Systems and Software Engineering – Vocabulary*. (2010)
219. Wieringa, R.J.: *Requirements Engineering: Problem Analysis and Solution Specification (Extended Abstract)*. In Koch, N., Fraternali, P., Wirsing, M., eds.: Web Engineering: 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004. Proceedings. Volume 3140 of Lecture Notes in Computer Science. Springer (2004) 13–16
220. Frank, U., Strecker, S., Fettke, P., vom Brocke, J., Becker, J., Sinz, E.: *The Research Field "Modeling Business Information Systems"*. Business & Information Systems Engineering 6(1) (2014) 39–43
221. Pittke, F., Nagel, B., Engels, G., Mendling, J.: *Linguistic Consistency of Goal Models*. In Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P., eds.: Enterprise, Business-Process and Information Systems Modeling: 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings. Volume 175 of Lecture Notes in Business Information Processing. Springer (2014) 393–407
222. Nwokeji, J.C., Clark, T., Barn, B.S.: *Towards a Comprehensive Meta-Model for KAOS*. In Moreira, A., Mussbacher, G., Araújo, J., Bencomo, N., Sánchez, P., eds.: International Workshop on Model-Driven Requirements Engineering, MoDRE 2013, Rio de Janeiro, Brasil, July 15, 2013. (2013) 30–39
223. Guizzardi, G.: *On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models*. In Vasilecas, O., Eder, J., Caplinskas, A., eds.: Databases and Information Systems IV - Selected Papers from the Seventh International Baltic Conference, DB&IS 2006, July 3-6, 2006, Vilnius, Lithuania. Volume 155 of Frontiers in Artificial Intelligence and Applications. IOS Press (2006) 18–39
224. Searle, J.R.: *Minds, Brains and Science*. Harvard University Press (1984)
225. Schütte, R.: *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle*. Gabler (1998)
226. Anaya, V., Berio, G., Harzallah, M., Heymans, P., Matulevičius, R., Opdahl, A.L., Panetto, H., Verdecho, M.J.: *The Unified Enterprise Modelling Language – Overview and Further Work*. Computers in Industry 61(2) (2010) 99–111
227. Opdahl, A.L.: *Semantic Annotations for Modelling Language Interoperability*. In Akerkar, R., ed.: Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, Sogndal, Norway, May 25-27, 2011, ACM (2011) 1–9
228. Kluth, M.: *Semantisches Benchmarking von Geschäftsprozessen: Konzeption, Evaluation und Anwendungspotenziale*. Books on Demand (2013)
229. Opdahl, A.L.: *Anatomy of the Unified Enterprise Modelling Ontology*. In van Sinderen, M., Johnson, P., eds.: Enterprise Interoperability: Third International IFIP Working Conference, IWEI 2011, Stockholm, Sweden, March 23-24, 2011. Proceedings. Volume 76 of Lecture Notes in Business Information Processing. Springer (2011) 163–176
230. Guarino, N., Guizzardi, G.: *In the Defense of Ontological Foundations for Conceptual Modeling*. Scandinavian Journal of Information Systems 18(1) (2006) 1–12
231. Gehlert, A., Esswein, W.: *Toward a Formal Research Framework for Ontological Analyses*. Advanced Engineering Informatics 21(2) (2007) 119–131
232. McGuinness, D.L., Van Harmelen, F.: *OWL Web Ontology Language Overview*. W3C Recommendation 10(10) (2004)
233. Wand, Y., Weber, R.: *On the Deep Structure of Information Systems*. Information Systems Journal 5(3) (1995) 203–223
234. Pelletier, F.J.: *The Principle of Semantic Compositionality*. Topoi 13(1) (1994) 11–24
235. Opdahl, A.L., Berio, G., Harzallah, M., Matulevicius, R.: *An Ontology for Enterprise and Information Systems Modelling*. Applied Ontology 7(1) (2012) 49–92
236. Opdahl, A.L., Henderson-Sellers, B.: *A Template for Defining Enterprise Modelling Constructs*. Journal of Database Management 15(2) (2004) 39–73
237. Harzallah, M., Berio, G., Opdahl, A.L.: *Incorporating IDEF3 into the Unified Enterprise Modelling Language*. In: Workshops Proceedings of the 11th International IEEE Enterprise Distributed Object Computing Conference, ECOCW 2007, 15-16 October 2007, Annapolis, Maryland, USA. (2007) 133–140
238. Fickinger, T., Recker, J.C.: *Construct Redundancy in Process Modeling Grammars: Improving the Explanatory Power of Ontological Analysis*. In: 21st European Conference on Information Systems, ECIS 2013, Utrecht, The Netherlands, June 5-8, 2013, Association for Information Systems (2013) 1–12

239. Gehlert, A., Esswein, W.: *Toward More Rigor in Ontological Analyses*. In Ljungberg, J., Andersson, M., eds.: Proceedings of the Fourteenth European Conference on Information Systems, ECIS 2006, Göteborg, Sweden, 2006. (2006) 984–994
240. Pfeiffer, D.: *Constructing Comparable Conceptual Models with Domain Specific Languages*. In Österle, H., Schelp, J., Winter, R., eds.: Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007, St. Gallen, Switzerland, 2007. (2007) 876–888
241. Opdahl, A.L., Henderson-Sellers, B.: *Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model*. Software and System Modeling 1(1) (2002) 43–67
242. Kühn, H.: *Methodenintegration im Business Engineering*. PhD thesis, University of Vienna (2004)
243. Engels, G., Hausmann, J.H., Heckel, R., Sauer, S.: *Dynamic Meta Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML*. In Evans, A., Kent, S., Selic, B., eds.: UML 2000 - The Unified Modeling Language, Advancing the Standard, Third International Conference, York, UK, October 2-6, 2000, Proceedings. Volume 1939 of Lecture Notes in Computer Science. Springer (2000) 323–337
244. Soltenborn, C., Engels, G.: *Towards Test-Driven Semantics Specification*. In: Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings. (2009) 378–392
245. Engels, G., Küster, J.M., Heckel, R., Groenewegen, L.: *A Methodology for Specifying and Analyzing Consistency of Object-Oriented Behavioral Models*. In Tjoa, A.M., Gruhn, V., eds.: Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering 2001, Vienna, Austria, September 10-14, 2001, ACM (2001) 186–195
246. Stark, J., Esswein, W.: *Rules from Cognition for Conceptual Modelling*. In Atzeni, P., Cheung, D., Ram, S., eds.: Conceptual Modeling: 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings. Volume 7532 of Lecture Notes in Computer Science. Springer (2012) 78–87
247. Guizzardi, G., Ferreira Pires, L., van Sinderen, M.: *An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages*. In Briand, L., Williams, C., eds.: Model Driven Engineering Languages and Systems: 8th International Conference, MoDELS 2005, Montego Bay, Jamaica, October 2-7, 2005. Proceedings. Volume 3713 of Lecture Notes in Computer Science. Springer (2005) 691–705
248. Esswein, W., Lehrmann, S.: *About the Need for Semantically Enriched Reference Models*. In: 19th Americas Conference on Information Systems, AMCIS 2013, Chicago, Illinois, USA, August 15-17, 2013. (2013)
249. Leopold, H.: *Natural Language in Business Process Models*. Volume 168 of Lecture Notes in Business Information Processing. Springer (2013)
250. Clark, T., Gonzalez-Perez, C., Henderson-Sellers, B.: *A Foundation for Multi-Level Modelling*. In Atkinson, C., Grossmann, G., Kühne, T., de Lara, J., eds.: Proceedings of the Workshop on Multi-Level Modelling co-located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2014), Valencia, Spain, September 28, 2014. Volume 1286 of CEUR Workshop Proceedings. (2014) 43–52
251. Object Management Group (OMG): *Diagram Definition (DD), Version 1.0*. (2012)
252. Karagiannis, D., Grossmann, W., Höfferer, P.: *Open Model Initiative: A Feasibility Study*. University of Vienna, Department of Knowledge Engineering (2008)

1

Appendix

1.1 Overview of Extension Mechanisms

1.1.1 Decorators

User Perspective:

General Scope:	Different concepts can be multiply wrapped with additional capabilities in order to represent complex classification hierarchies or roles at runtime.
Dissemination and Occurrence:	Not applied in EM
Pragmatics:	Representation of hierarchies, combinations and variants
Appropriate Extension Type:	• Model Operations • (Enhancement)
Main Benefits:	• Representation of multiple roles
Limitations and Problems:	• Complex wrapping logic, binding and releasing • Rather useful for runtime-related annotations

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	Highly invasive, decorating logic requires complex environment
Implementation and Application:	• Decorator definition (runtime) • Binding (merging of meta elements) • Instantiation • Unbinding, releasing
Multiple Application:	Possible (explicitly intended by definition)
Host Dependency:	High

Syntactical Mirco View:	
Extension Elements:	Meta classes
Modularity:	Low (high coupling, low cohesion, separate instantiation rather useless)
Inner Complexity and Design Freedom:	Low (simple conceptual structures)
Interface Structure:	• Not explicit • M-N relation (host meta class can have multiple decorators, a decorator can be assigned to multiple host meta classes)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	General implementation of the Decorator Pattern on type level

Table 1.1: Characteristics of the Decorator mechanism

1.1.2 Plugins

User Perspective:

General Scope:	Provision of complex modules for the solution of specific business problems
Dissemination and Occurrence:	• Not explicitly defined • Similar approaches in the field of meta model composition and meta model integration [168, 169]
Pragmatics:	Annotation of additional expressiveness from other areas of discourse, both syntactically and semantically
Appropriate Extension Type:	• Augmentation, enhancement (syntax and semantics) • Formal Semantics, model operations (semantics)
Main Benefits:	• Separation of concern, modularisation
Limitations and Problems:	• Interface specification

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	High (e.g. definition of interfaces in host meta model)
Implementation and Application:	• Definition of extension interfaces (if required) • Plugin definition • Merge and application
Multiple Application:	Possible
Host Dependency:	Low

Syntactical Mirco View:	
Extension Elements:	Meta classes (single class or set of classes)
Modularity:	Maximum (high cohesion, low coupling, separate instantiation is possible)
Inner Complexity and Design Freedom:	High
Interface Structure:	• 1-M (extension interface can have multiple adaptive Plugins, a particular Plugin corresponds to exactly one interface)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Definition of interfaces

Table 1.2: Characteristics of the Plugin mechanism

1.1.3 Aspects

User Perspective:

General Scope:	Central provision of generic supporting concepts and capabilities
Dissemination and Occurrence:	Not applied in EM
Pragmatics:	Annotation of generic and reusable capabilities for model analysis
Appropriate Extension Type:	• Model operations • Augmentation (often required attributes, without generalisations)
Main Benefits:	• Cross-cutting reuse of rather generic concepts
Limitations and Problems:	• Potential amalgamation of conceptual and analytical concepts

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	Medium (low coupling, but integration and merge logic is required)
Implementation and Application:	• Aspect definition • Merge and application
Multiple Application:	Possible
Host Dependency:	Low

Syntactical Mirco View:	
Extension Elements:	Meta classes (one particular)
Modularity:	High (high cohesion, low coupling, separate instantiation is possible)
Inner Complexity and Design Freedom:	Differs
Interface Structure:	• Not explicit • M-N (host meta class can have multiple Aspects, a particular Aspect can be assigned to multiple meta classes)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Simple (merging of additional concepts)

Table 1.3: Characteristics of the Aspects mechanism

1.1.4 Add-Ons

User Perspective:

General Scope:	Provision of supplemented information with a limited complexity in order to enhance existing concepts incrementally for specific purposes
Dissemination and Occurrence:	Not *explicitly* applied in EM
Pragmatics:	Annotation of additional information within the existing area of discourse, both syntactically and semantically
Appropriate Extension Type:	• Enhancement • Domain specialisation (by detailing on the same level of abstraction)
Main Benefits:	• Incremental, marginal extension
Limitations and Problems:	• Interface specification

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	High (e.g. definition Plugin interfaces in host meta model)
Implementation and Application:	• Definition of extension interfaces • Add-On definition • Merge and application
Multiple Application:	Possible
Host Dependency:	High

Syntactical Mirco View:	
Extension Elements:	Meta classes (single class or set of classes)
Modularity:	Low (low cohesion, high coupling, separate instantiation is not possible)
Inner Complexity and Design Freedom:	Simple, supplementing
Interface Structure:	• Explicit • 1-M (extension interface can have multiple adaptive Add-Ons, a particular Add-On corresponds to exactly one interface)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Interface design

Table 1.4: Characteristics of the Add-On mechanism

1.1.5 Hooking

User Perspective:

General Scope:	Parts are left open or remain under-specified on purpose
Dissemination and Occurrence:	• Not explicitly defined • Single approaches in the context of reference modelling [173, 174]
Pragmatics:	Specialising: Implementation of intentional extensibility in some sorts of generic EMLs
Appropriate Extension Type:	• Placeholder: Enhancement, augmentation • Specialisation: Domain-specification, (formal semantics)
Main Benefits:	• Explicit specification opportunity • Design freedom
Limitations and Problems:	• Level of predesign (especially for Hooking by Specialisation) • Need for specialisation (parts of original meta model may not be applicable) • Abstraction differences in case of Hooking by Specialisation

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	• Placeholder: Low (definition of hooking hot spots that can be specialised) • Specialisation: High (explicitly prepared meta model)
Implementation and Application:	• Definition of hot spots for specialisation and placeholders • Hook definition • Merge and thus pre-instantiation of the meta model in order to establish a revised meta model version
Multiple Application:	Not possible
Host Dependency:	Very high

Syntactical Mirco View:	
Extension Elements:	• Placeholder: All meta types • Specialisation: Usually on meta class level: attribute values, ranges, constraints, particular terms and atomic values (cf. [161])
Modularity:	• Placeholder: Low (high cohesion, but high coupling) • Specialisation: Very low (low cohesion, very high coupling)
Inner Complexity and Design Freedom:	• Placeholder: Various • Specialisation: Rather simple
Interface Structure:	• Explicit • 1-1 (hooking spot or area is replaced by exactly one hooking extension)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Large [161]

Table 1.5: Characteristics of the Hooking mechanism

1.1.6 Profiling

User Perspective:	
General Scope:	Attribute-wise annotation of concepts within well-defined packages (Profiles)
Dissemination and Occurrence:	Common UML extension mechanism [31], few adaptations to engineering-oriented modelling languages [155], BPMN [42] and EMLs in general [156]
Pragmatics:	Specifying rather generic concepts for particular domains, industries or enterprises
Appropriate Extension Type:	• Domain-specialisation • Enhancement (by refinements) • Formalisation (e.g. by annotation of constraints)
Main Benefits:	• Well established mechanism, reuse of well-known meta concepts from MOF, lightweight integration opportunities • Generic property-wise extension (cf. [156]) • Combination of different Profiles • Profile concept as packaging instrument
Limitations and Problems:	• Basic approach (extending meta classes): No real type creation, no subclasses • Advanced approach (extending all meta types [156]): The creation of "top level" Stereotypes indicates huge language extensions in the sense of the creation of additional meta model classes, for instance [156, p. 64], which actually does not correspond to the localised "specification" character of Profiling.
Technical Perspective:	
Syntactical Macro View:	
Invasiveness and Architectural Predesign:	Low
Implementation and Application:	• Stereotype definition • Instantiation of the stereotyped meta class requires the optional or mandatory instantiation of the annotated Stereotype, indicating a certain post processing of the instantiated meta classes (e.g. validation checks)
Multiple Application:	Possible
Host Dependency:	Very high
Syntactical Mirco View:	
Extension Elements:	Meta classes (application to different meta types is possible [156])
Modularity:	Low (low cohesion, high coupling, separate instantiation is not possible)
Inner Complexity and Design Freedom:	• Basic approach: Simple, structure depends on host meta classes • Complex approach: The degree of design freedom depends on the meta class type that is extended [156, p. 64]. For instance, the creation of view stereotypes could lead to entirely new diagrams based on a stable conceptual set. In contrast, stereotypes of meta model classes have a rather limited expressiveness, as it must conform to the conceptual boundaries of the stereotypes class.
Interface Structure:	• Implicit • 1-M (one meta class could apply multiple Profiles, a particular Profile corresponds to a particular set of meta classes, shaping an implicit interface)
Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Lifting the profiling mechanism to the meta meta level [42]

Table 1.6: Characteristics of the Profiling mechanism

1.1.7 Multilevel Modelling

User Perspective:

General Scope:	Introduction of additional classification levels
Dissemination and Occurrence:	• Novel approach, limited to dedicated modelling environments (FMMLx) • Not integrated in common meta modelling languages like MOF
Pragmatics:	Localising EMLs for domains, enterprises and situations
Appropriate Extension Type:	• Domain-Specialisation • (Enhancement, augmentation on lower classification levels)
Main Benefits:	• Flexible derivation of localised languages • Separation of classification levels according to conceptualisations (avoidance of flattening)
Limitations and Problems:	• Expensive technical implementation due to paradigmatic novelty • Invasiveness: Annotation of attributes and classes in regard to defining Intrinsic Features and particular levels of instantiation

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	High (multilevel paradigm actually contradicts traditional meta modelling)
Implementation and Application:	• Selection and annotation of Intrinsic Features • Definition of additional classification levels • Instantiation of the multilevelled meta model
Multiple Application:	Possible
Host Dependency:	High (instantiation from original meta classes)

Syntactical Mirco View:	
Extension Elements:	• Meta classes • (Entire classification level)
Modularity:	Low (different degree of cohesion, high coupling, separate instantiation is not useful due to "instance of"-relation to original host meta classes)
Inner Complexity and Design Freedom:	• Inner complexity could range from simple specialisations to complex inner-level structures • Design freedom is principally high (within additional classification levels), but naturally restricted by the existing meta classes
Interface Structure:	• Not explicit • 1-M (meta classes can be "extended" by multiple classes on different classification levels, while an introduced class serves as instance of an original meta class)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Comprehensive redesign efforts due to the paradigmatic novelty

Table 1.7: Characteristics of the Multilevel Modelling mechanism

1.1.8 Generalisation/Specialisation

User Perspective:

General Scope:	Specialisation by additional subclasses
Dissemination and Occurrence:	Often implicitly applied (cf. [33, p. 52])
Pragmatics:	Specifying rather generic concepts for particular domains, industries or enterprises
Appropriate Extension Type:	• Domain-specialisation • Enhancement (refinement)
Main Benefits:	• Simple application and clear structure • Simple combination with other approaches (especially Plugins and Add-Ons)
Limitations and Problems:	• Precise separation in hybrid forms

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	• Type 1 and 2: Low (extension by specialisation) • Type 3: Differs according to "min" cardinality of the introduced association
Implementation and Application:	• Definition of subclasses (and associations in case of type 3) • Merge and instantiation
Multiple Application:	Possible
Host Dependency:	Very high

Syntactical Mirco View:	
Extension Elements:	Meta classes (application to other meta types, e.g. perspectives, is also possible)
Modularity:	Minimum (low cohesion, high coupling, separate instantiation is not possible due to lack of super properties)
Inner Complexity and Design Freedom:	• Type 1: limited • Type 2: differs according to number of associations • Type 3: usually high (hybrid type, Plugins and Add-Ons)
Interface Structure:	• Implicit • 1-M (meta class can be specialised by multiple extension classes, one extension class refers to or specializes one meta class)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	None

Table 1.8: Characteristics of the Generalisation/Specialisation mechanism

1.1.9 Semantic Extension

User Perspective:

General Scope:	Extending the interpretation of an EML meta model
Dissemination and Occurrence:	Not explicitly specified and applied
Pragmatics:	Specifying the interpretation of an EML according to a domain or stakeholder group
Appropriate Extension Type:	• Accent building • (domain-specialisation, formalisation)
Main Benefits:	• Syntax remains unaffected • Flexible stakeholder-dependent interpretation of one language
Limitations and Problems:	• Lack of explicitness • Specification efforts

Technical Perspective:

Syntactical Macro View:	
Invasiveness and Architectural Predesign:	No invasiveness (syntax remains unaffected)
Implementation and Application:	• Specification of additional semantic mappings and semantics constructs • Specification of the relation to original semantic constructs in case of specialisation
Multiple Application:	Possible
Host Dependency:	Very high

Syntactical Mirco View:	
Extension Elements:	Each syntactical construct (single classes as well as combinations)
Modularity:	Low (naturally high coupling of semantic relations, differing degree of cohesion, separate instantiation is contingent)
Inner Complexity and Design Freedom:	• Inner complexity: could vary, depending of the set of introduced semantic mappings • Design freedom: limited to the semantic borders of the extended constructs
Interface Structure:	• Not explicit nor required • 1-M (meta element can be extended multiply, while an additional semantic mappings refers to exactly one meta element)

Consequences for Meta Modelling and Meta Meta Modelling:	
Redesign Consequences (if required):	Semantic representation techniques

Table 1.9: Characteristics of the Semantic Extension mechanism