The Ingredients of EMF
Analysis of Documentation and Implementation Topics

Johannes Härtel and Lukas Härtel and Ralf Lämmel

Software Languages Team, http://www.softlang.org/
University of Koblenz-Landau, Universitätsstraße 1, 56072 Koblenz, Germany

Abstract. The paper recovers relationships between the implementation of the Eclipse Modeling Framework (EMF) and a major documentation source on the framework, i.e., Steinberg et al.'s textbook on EMF. The underlying method recovers relationships on the grounds of semantic information that can be retrieved from implementation and documentation. To this end, LDA is applied to both implementation and documentation and the resulting topics are compared in a systematic metrics-driven manner, thereby revealing cases of alignment and also cases of misalignment, both of which are to be explained with the help of domain knowledge. Paragraph granularity is used for documentation and method granularity is used for the implementation. Thereby, we abstract from the modular structure of the information sources.

1 Introduction

Getting a better understanding of the relationships between source code and documentation can facilitate the comprehension of code, documentation, and the underlying domain(s) also in the view of use-cases such as quality assurance or traceability recovery. In this paper, we are interested in studying the relationships between implementation (i.e., source code) and documentation (e.g., a textbook) of (object-oriented) frameworks. More specifically, we are interested in recovering relationships between the implementation of the Eclipse Modeling Framework (EMF) and a major documentation source on the framework, i.e., Steinberg et al.'s textbook on EMF.

Information retrieval methods may be used to recover relationships between source code and documentation. Contrary to other approaches that establish a relationship on explicitly defined parts of the data usually facilitated by topic modes [1–10], this work examines the relationships between different topic structures. For instance, if we want to get a rough overview on correlation between implementation and documentation, one possible solution would be to apply LDA on both and compare packages with chapters in the recovered topic space. Contrary, our work assumes that a comparison of topics, recovered for documentation and implementation in separation, uncovers a completely different insight into a relationship between two types of resources.

We motivate such a special comparison by the following points: (i) We expect to gain additional insight into the nature of a topic. A topic that exists in documentation, but that is not comparable to any implementation topic, might bear

different cognitive insight into a system compared to a topic that is also present in implementation. Finally, this can lead to a classification of knowledge, like already manually done for documentation fragments in [11], and automatically in [12, 13]. (ii) If such comparison on topic level is well understood, it can be used to determining the fitness of a topic as an alternative to internal quality metrics like cohesion and separation. (iii) We use these recovered relationships between topics in our efforts to develop a comprehensive model [14–16] for EMF on the basis of suitable 'ground truth' and quality indicators, e.g., in terms of the model to cover existing documentation. On one side the comparison provides us with a scale of topic importance, encouraging us to concentrate modeling on the most prominent topics. On the outer sides, it gives us insight into the topics relatedness to implementation which is important for our models, since we are primary interested in environments of software systems and not in fine grained implementation details.

We propose measures to characterize a topic's distribution over a second topic structure, i.e., average, standard deviation, max and a measure representing the internal importance of the topics in the underlying data. We use these measures as features to manually find clusters of very similar characteristics that we call regions. Measures and regions are then used to classify topics into a taxonomy taken from [11].

Contributions of the paper

- We present a method for a topic comparison between implementation and documentation for frameworks.
- We define a list of measures, relationship characteristics, and topic classifications to analyze a topic comparison.
- We validate the method for the Eclipse Modeling Framework (EMF) and Steinberg et al.'s textbook on EMF [17].

Road-map of the paper Section 2 develops the method of topic comparison including means of analysis in detail. Section 3 applies the method to EMF implementation and documentation. Section 4 discusses related work. Section 5 concludes the paper. A data-set for the paper is available online.[1]

2 The Method of LDA Topic Comparison

In explaining the method, we refer to the EMF study's artifacts, the underlying domain, etc. for illustrative purposes. We explain the method as a sequence of processing steps as follows.

2.1 Decomposition

Documentation (text) and implementation (source code) are split into semantically coherent plain text fragments. Possible decomposition methods for documentation can follow the hierarchical structuring of a document into chapters,

[1] http://softlang.uni-koblenz.de/emf-analysis

sections and paragraphs. We opt for fine-grained paragraph fragments to abstract from the textbook's modular structure. In the EMF study, the paragraphs were tagged by hand; all figures, source code and XML was excluded.

The decomposition of source code is syntax-aware and uses each (Java) method as a fine-grained fragment by extracting its name, all identifiers combined with the names of used types in the parameters, the result, and the method body. In the EMF study, the sub-packages *ecp*, *edapt*, *change*, and *emfstore* are excluded, as they are not involved in EMF standard use-cases.

2.2 Preprocessing

Unified text normalization is applied to all fragments. It is focused on normalizing conventions that match on both, documentation and implementation. The stage filters stop-words, removes everything which is not a letter, splits camel-case, rewrites letters into lower-case, and applies stemming. The filtered words come from an English stop-word list where domain-specific words are added taken from the list of most frequent words (e.g., get, set, object, e, and type). This list is available on-line.

2.3 Recovering Topics by LDA

We constructed a vector space model for implementation and documentation and apply LDA separately to recover the relevant topics sets. We use the same configurations for both topic-models. We set k (number of topics) to 30, alpha on 1.01, beta to be 3.0 and iteration number is 300. We use the Spark ML implementation of LDA.

2.4 Asymmetric Topic Correlation Matrix

The central element of the method is a comparison of the two topic sets that are recovered by separately applying LDA on implementation and documentation. The cell's color reflects the similarity between the topic's term vectors in the asymmetric topics correlation matrix. The matrix for the EMF study can be found in Fig. 1. We defined following measures for a topic in such comparison where the topic-topic similarity is defined to be the cosine between the term vectors:

- **Average** The average measure of a topic, represented as row or column in the matrix, is the mean similarity between this topic and all topics out of the compared topic structure.
- **Maximum** Maximum measure represents the maximal similarity in a row or column.
- **Standard Deviation** Standard deviation measure reflects the deviation of one topic's similarity compared with the topics of the second topic structure.
- **Importance** The importance of a topic is given by the number of the underlying data source's documents where this topic is nearest. For Steinberg et al.'s textbook it is given in number of paragraphs.

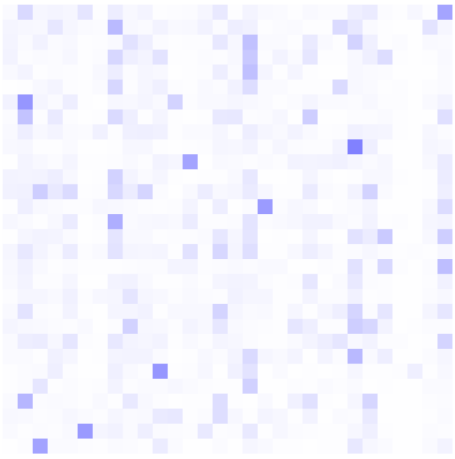

Fig. 1. Visualization of the asymmetric topic correlation matrix: Steinberg et al.'s textbook topics are located on horizontal and EMF implementation's topics are placed on the vertical.

2.5 Topic Types

The following list gives the topics types we use in this work and is taken from [11] where it is used to classifying API documentation. We adapted their taxonomy in that it matches our high level classification purpose. The list explains the original types and, if this is necessary, our adapted version. Type that can not be detected in the EMF topics are highlighted. Italic parts are citations form [11].

- **Functionality and Behavior** *describes what the API does (or does not do) in terms of functionality or features.*
- **Concept** *explains the meaning of terms used to name or describe an API element.*
- **Directives** *specifies what users are allowed / not allowed to do with the API element. Directives are clear contracts.* We interpret this type more generally and focused on the user to be concerned with what users is allowed to do with the API or framework e.g. developing uml models.
- **Purpose and Rationale** *explains the purpose of providing an element or the rationale of a certain design decision. Typically, this is information that answers a why question.* We did not find the purpose type in the topics recovered from Steinberg et al.'s textbook on EMF.
- **Quality Attributes and Internal Aspects** specifies *non-functional requirements* and *... APIs internal implementation that is only indirectly related to its observable behavior.*
- **Control-Flow** *describes how the API (or the framework) manages the flow of control, for example by stating what events cause a certain callback to be triggered...* This type is too detailed and we did not detect it.

– **Structure** *describes the internal organization of a compound element...* We detected structure in one topic describing a set of primitive types.
– **Patterns** *describes how to accomplish specific outcomes with the API,...* We focused on design patterns.
– **Code Examples** We generalized this type on Examples.
– **Environment** *describes aspects related to the environment in which the API is used, but not the API directly, e.g., compatibility issues, differences between versions, or licensing information.*
– **References** We did not detect references since references are usually very specific and are not separated as individual topic.
– **Non-information** *provides only uninformative boilerplate text.*

2.6 Relation Characteristics

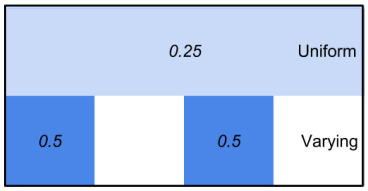

Fig. 2. Extreme cases in a correlation matrix

The asymmetric topic correlation matrix shown in Fig. 2 depicts extreme cases in a relation that appear only in a less extreme manner in Fig. 1. The characteristics found here are the starting point for a classification of documentation topics into the taxonomy defined in the last section. The basic idea is that a documentation topics can better be classified, if one knows that this topic it is also reproduced in the implementation. The figure shows two possible distributions that a topic can produce over the compared structure. For a documentation topic, we examine its distribution over all implementation topics. Possible relation characteristic shown in the plot are called uniform or varying.

Varying One documentation topic spreads over several implementation topics, but not over all. This distribution produces significant peaks in a row or column of the asymmetric topic correlation matrix. A Varying distribution of a documentation topic over the implementation is found, when the semantic aspects are partially reproduced in implementation. This is the case for implementation near topic types, i.e. (1) Functionality and Behavior that represents the main semantics of implementation and must be reflected in the topic structure, (5) Quality Attributes and Internal Aspects as they are meaningful for implementation in terms of non-functional requirements, (7) Structure since the shape of data plays a significant role in implementation and (8) Patterns that push a regularity into unstructured code.

Uniform One topic matches the complete second data source's topics allover with constant similarity. Intensity of this continuous alignment is not of relevance for uniform distribution. A Uniform distribution bears fundamental differences since the semantic separation found in the documentation topics is not reflected in the implementation topic structure. There are two possible cases, either the implementation misses the semantics completely or the implementation is systematically affected by the semantics implying an unchanged similarity allover the implementation topics. We expect this characteristic in the topic types: (2) Concept, since the implementation is constantly bound to it; (3) Directives and (9) Code Examples, since those consider the framework or API as a whole; (10) Environment, as the implementation is constantly bound to it; and (12) Non-information.

For detecting the uniform and varying characteristics of a relationship we take the relation between standard deviation and average. Both measures for the sample can be found in Table 1.

Extreme	Average	Standard Deviation	Stdev. / Average
Uni.	$\frac{4}{4} \cdot 0.25 = 0.25$	$\sqrt{\frac{4}{4} \cdot 0^2} = 0$	0
Var.	$\frac{2}{4} \cdot 0.50 = 0.25$	$\sqrt{\frac{4}{4} \cdot 0.25^2} = 0.25$	1

Table 1. Sample measures for the extreme cases in Fig. 2

3 Analysis of EMF implementation and documentation

3.1 Methodology

To unfold the capabilities of our method, we first show the different measures for the documentation topics over implementation in the Fig. 3. By hand, we detect very similar topics with comparable measures and group them into regions. We use these strongly similar topics and the measures to recover topic types according to uniform and varying distribution characteristics, that are depicted in Fig. 4.

3.2 Regions

We defined the following regions for the topics recovered from the textbook paragraphs by our method. Each region is defined so that the measures of the contained topics are comparable. In Fig. 4 and Fig. 3 the regions are highlighted. Additionally, we bring up a small description if this region is important for our modeling purposes.

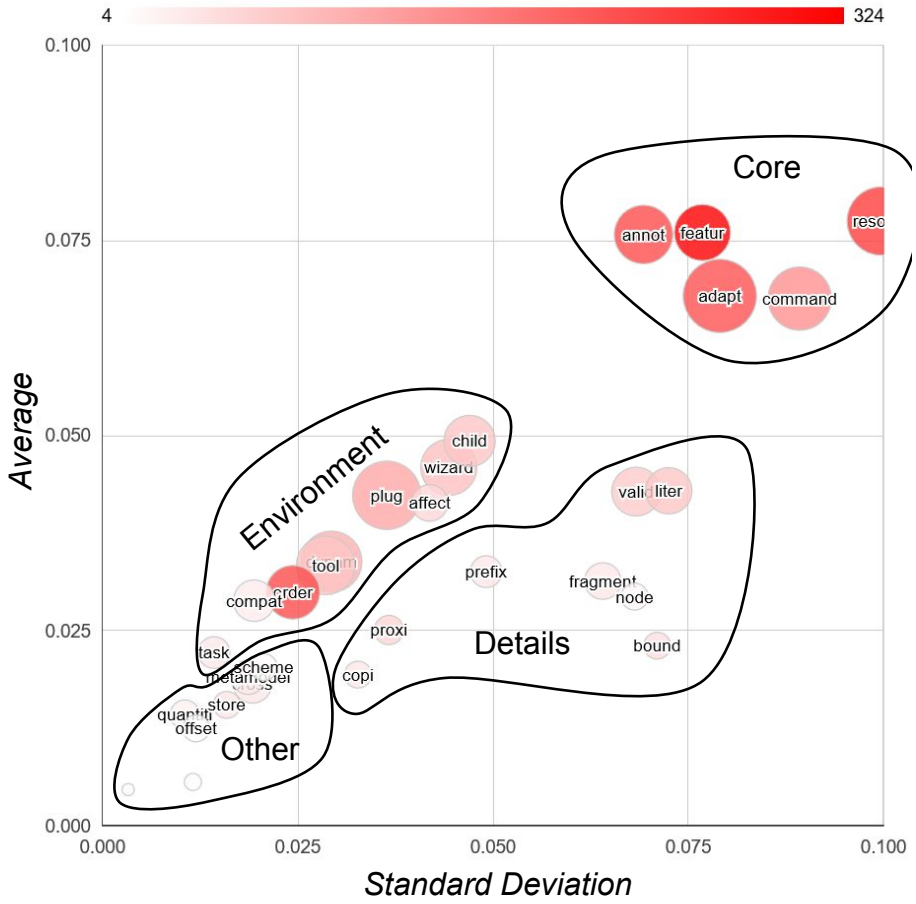

Fig. 3. Documentation topics arranged by average and standard deviation measure: The size of the circles reflects the maximum measure, i.e., the highest correlation with an implementation topics and the color show importance, i.e. the the number of paragraphs that are assigned to this specific topic.

- **EMF Core** contains the most important EMF knowledge with the highest standard deviation and average measures over the implementation. Here uniform and varying characteristics can be found. The two topic types present are Pattern and Concept. For our modeling, these topics might be relevant, despite some of them come close to implementation.
- **EMF Environment** contains mostly contextual information with an high average but lower standard deviation over implementation, since the topics have a uniform influence over implementation. Mostly Environment topics are located in this region, followed by Directives, Examples and Concepts.

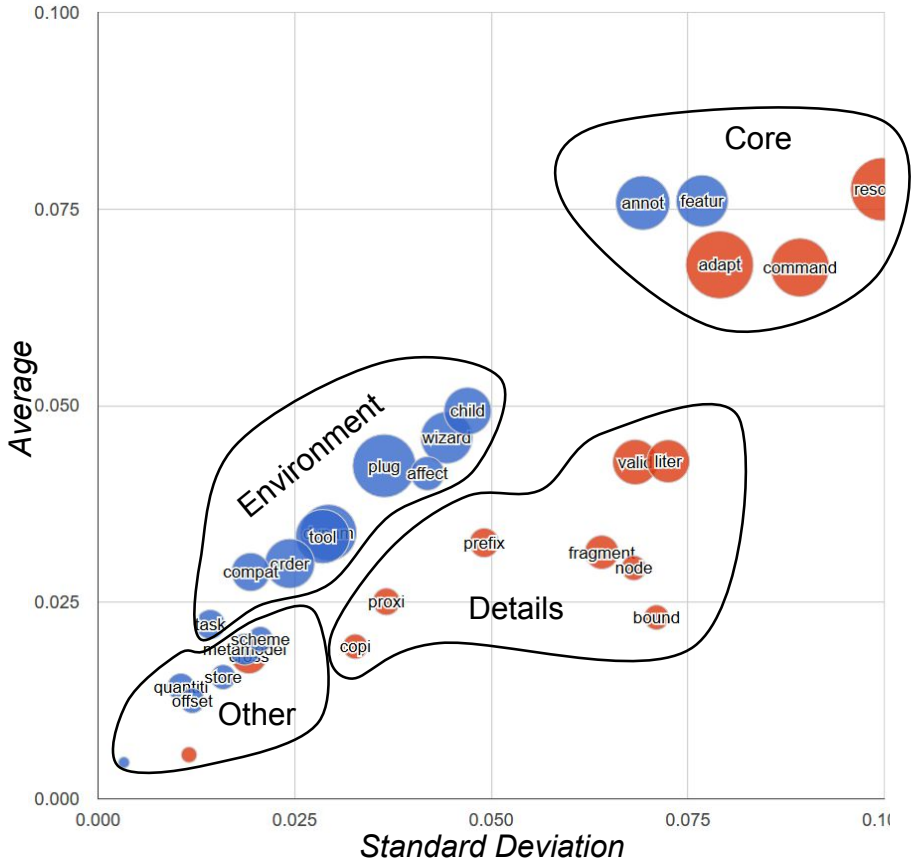

Fig. 4. Characteristic of the topics: Blue topics are of uniform distribution over implementation, red ones are varying. The uniform characteristics is decided by the threshold of: $\frac{StandardDeviation}{Avarage} < 1.05$

In terms of our efforts to develop a comprehensive model of EMF, we are very interested in this region since we want to describe EMF's interaction and embedding into it's environment.

– **EMF Details** contains EMF implementation knowledge of minor important with a low average but high standard deviation over code. All topics bear a varying characteristic. Dominant topic type is Functionality and Behavior followed by Structure and Quality Attributes and Internal Aspects. For our modeling purposes this regions is not interesting.

– **EMF Other** contains the remaining topics that are far away from the EMF implementation with a low average and low standard-deviation. Topic types are mixed but most topics can be assigned to Non-Information. Non-

Information is not relevant for our modeling purposes but still this region bears some interesting topics that might further be examined in a generalized context.

3.3 EMF Core

Terms	Avg.	Stdev	Max.	Imp.	Char.	Type
annot element schema ecor kei attribut name entri metadata xml	0.076	0.069	0.068	227.0	uniform	Concept
featur attribut class refer interfac properti defin java oper gener	0.076	0.077	0.066	324.0	uniform	Concept
adapt chang factori method item return call notif creat provid	0.068	0.079	0.101	220.0	varying	Pattern
command edit provid item label descriptor execut domain viewer emf	0.068	0.089	0.077	142.0	varying	Pattern
resourc uri option load serial save xml registri factori regist	0.077	0.1	0.089	246.0	varying	Pattern

Table 2. Topics of the EMF Core region

The EMF Core region, listed in Table 2, contains the most prominent topics defined by the high average and high standard deviation. The five topics differ respective uniformity and variety. Compared to other topics in this region, the uniform ones, i.e., *annot* and *feature*, constantly effect the EMF implementation semantics. Both topics can be seen as representatives for the bridge that is drawn by EMF between XML, Java and Ecore. We define these topics to be of type **Concept** since they form a very macroscopic view on EMF naming the important modeling elements. The second set of topics, i.e., *adapt*, *command*, and *resource* are varying over the implementation. Semantics of the topics can be seen to cover the adaptation of EMF models, the command infrastructure enabling the modification of models and the resource framework that is responsible for persistence. We defined those topics to be of type **Pattern**. When looking at the importance and the max measure, all topics are very close to each other. This can be seen as motivation for this particular EMF Core region.

3.4 EMF Environment

The EMF Environment region, listed in Table 3, contains topics where the semantic is not mixed into implementation but average coverage is still high. The characteristics of the topic's distribution over code is uniform. The most topics are of type **Environment** as a consequence of EMF embedding into eclipse: *Child* describing EMF's UI contribution, *affect* covers the semantics of eclipse projects, *plug* the plugin mechanism, *compat* for version compatibility, and *task*

Terms	Avg.	Stdev	Max.	Imp.	Char.	Type
child children action displai select parent view creation menu bar	0.049	0.047	0.055	75.0	uniform	Environment
wizard select rose project click page box file model figur	0.046	0.044	0.063	82.0	uniform	Directives
affect test categori plug directori project properti code edit variabl	0.041	0.042	0.033	54.0	uniform	Environment
plug eclips applic file templat plugin in manifest run platform	0.042	0.036	0.088	113.0	uniform	Environment
tool uml develop diagram model emf project eclips java languag	0.033	0.029	0.069	75.0	uniform	Directives
dynam api reflect emf code gener framework runtim model memori	0.033	0.029	0.069	75.0	uniform	Concept
order purchas po custom exampl supplier address look figur like	0.03	0.024	0.059	228.0	uniform	Example
compat version backward sinc flag complianc older reserv properti exist	0.029	0.019	0.038	25.0	uniform	Environment
task ant script applic headless eclips build java develop user	0.022	0.014	0.027	33.0	uniform	Environment

Table 3. Topics of the EMF Environment region

describes Java ant builds. Further the type **Directive** can be found: describing the usage of wizards in the *wizard* topics and the process of developing uml diagrams in the *tool* topic. One topic, *order*, covers a running **Example** of the textbook. Another, *dynam*, which describes the **Concept** of dynamic emf models. The EMF Environment region is not that uniform in its features compared to the Core region. Topics *order* has a very high importance since it represents the running example in the textbook and spans over several paragraphs. We are not interested in the order topic for building a comprehensive EMF model. Further, the max measure has a wide range from 0.27 to 0.88.

3.5 EMF Details

The EMF Details region, listed in Table 4, includes the topics with a low average but high standard deviation. The distribution characteristics over implementation is varying. Most topics are of type **Functionality and Behavior**: Topic *valid* described the EMF validation and diagnostics, *copi* covers a utility mechanism for copying objects, *proxi* shows the EMF's proxy resolution, *prefix* the usage of uris, *fragment* the indexing of EMF objects and *bound* the behavior of EMF generics. Further, the *node* topic can be seen as **Quality Attributes**

Terms	Avg.	Stdev	Max.	Imp.	Char.	Type
valid constraint invari regener tag merg diagnost javadoc comment gener	0.043	0.068	0.052	68.0	varying	Functionality and Behavior
liter field enum enumer int primit constant virtual arrai boolean	0.043	0.073	0.047	74.0	varying	Structure
copi origin equal copier wrapper object util facil algorithm helper	0.019	0.033	0.022	32.0	varying	Functionality and Behavior
proxi resolv invers resolut refer bidirect cross contain remov automat	0.025	0.037	0.023	55.0	varying	Functionality and Behavior
prefix n packag uri subpackag lowercas name capit letter charact	0.033	0.049	0.027	31.0	varying	Functionality and Behavior
fragment path prioriti index standard iter segment move prune maintain	0.031	0.064	0.032	32.0	varying	Functionality and Behavior
node dom w queri x xalan tree evalu middl c	0.029	0.068	0.022	14.0	varying	Quality Attributes and Internal Aspects
bound except upper lower throw unbound unsupport role thrown stub	0.023	0.071	0.023	46.0	varying	Functionality and Behavior

Table 4. Topics of the EMF Details region

and Internal Aspects type covering the document object model as underlying persistence mechanism. Last, the topic *liter* representing types, is defined to be a **Structure** type. The topics of this region are very similar in max and importance measures except the two topics *liter* and *valid*.

3.6 EMF Other

The EMF Other region, listed in Table 5, contains the topics that are far away from the implementation. Those still bear some relevant **Concepts**, **Functionality and Behavior** and **Environment** information. Concepts are given in topic *scheme* talking about URIs and in topic *meta-model*. Functionality and Behavior is given in topic *cross* with semantics covering the resolution of cross references in EMF. Environment can be found in topic *quantiti* giving some information about SAX parser pools and topic *store* described the EMF store. The rest of these topics can be considered as **Non-information** also motivated by the low count of importance.

3.7 Threads to Validity

The results presented in the EMF study are influenced by several parameters that have not been systematically studied in our work, i.e., decomposition functions, preprocessing, LDA parameters, similarity definitions, and orchestration

Terms	Avg.	Stdev	Max.	Imp.	Char.	Type
scheme dir author network myfil devic segment c part ur	0.02	0.021	0.023	9.0	uniform	Concepts
metamodel emof convers mof ecor xmi omg version latest interchang	0.019	0.019	0.03	32.0	uniform	Concepts
cross referenc find search usag consol util refer unresolv guarante	0.018	0.019	0.035	34.0	varying	Functionality and Behavior
store expos client intern d extrins ap uui intrins em	0.015	0.016	0.023	35.0	uniform	Environment
quantiti price parser pool sax appl product xerc percent less	0.014	0.011	0.025	22.0	uniform	Environment
offset assign sequenti d supertyp local unadjust ensur difficult post	0.012	0.012	0.021	7.0	uniform	Non-information
held pickup mean former interpret distinguish unset latter suppos case	0.006	0.012	0.014	5.0	varying	Non-information
possibli claus lengthi evid correct sign sure absenc leav benefit	0.005	0.003	0.012	4.0	uniform	Non-information

Table 5. Topics of the EMF Other region

of the data processing. Most options were adopted from cited (related) work, but several original ones were defined by ad-hoc exploration of computed topics and classifications.

Another issue concerns the validation of classifications. We tried to confirm the classification in an objective and transparent manner by using a taxonomy proposed by related work. An experiment with real software engineers would be a significant improvement for future validation. We kept the methodology simple to justify validity of classifications and we used a limited number of discussed topics in the interest of a comprehensive discussion in this paper.

4 Related Work

Natural Language Processing A comparison of topics, topic types and categories extracted by LDA on Twitter and New York Times data is done in [7]. With this comparison focused on topics, this work comes closed to ours. It differs in still falling back considering the underlying documents, our work only defined the importance of a topic in terms of document count and exclusively compares topics. Further, our work differs in that it does not compare the topics to gain insights into the different channels or media, we used the comparison for strengthening the classification and understanding of a topic since we expect to know what the compared data is about.

In [18], a knowledge base, including Hierarchy, Relations and Facts, is build using LDA on a text corpus. They apply the method on stack-overflow to recover a software knowledge base. Since a long time goal of our approach is to validate very specific software engineering knowledge, this work is comparable.

Classification A manual classification of knowledge that occurs in API reference documentation is done in [11]. This classification distinguishes between (1) Functionality and Behavior, (2) Concepts, (3) Directives, (4) Purpose and Rationale, (5) Quality Attributes and Internal Aspects, (6) Control-Flow, (7) Structure, (8) Patterns, (9) Code Examples, (10) Environment, (11) References and (12) Non-information. We try to classify automatically and use this taxonomy.

In [12] this taxonomy and data is used to develop a method for automatically extract the knowledge types of a given API documentation. This is done by a supervised machine learning approaches training a classifier on already classified data. At this point our approach is superior, since it uses the difference in perspective to classify the content. No manually tagged data is needed for the given domain. Further, we classify topics and not API documentation fragments.

In [19], LDA is used for a classification of software systems. Contrary to our fine-grained decomposition into classes, in this case up to 43 complete systems are taken as document units. LDA's topics are then merged until some threshold is met. The aggregated topics are defined to be a categories and assigned to the systems respective their similarity.

Tractability Recovery Some traceability recovering techniques overcome the barrier of asymmetry in language by the usage of information retrieval techniques. In [1, 2], Antoniol et al. describe a method for the recovery of links between free-text documentation and source code using the VSM. It is applied in two case studies: (i) between C++ source code and manual pages and (ii) between Java code and functional requirements. The differences set out by language are handled using well-adjusted preprocessing. Our asymmetric preprocessing is highly motivated by this. Comparable work in [3, 20] exclusively differs in the usage of language-models as an alternative to a VSM. Adding LSI to the recovery process that handles asymmetry in language is described in [4]. The approach differs to our work in that it considers code as plain text. Both, Antoniol et al. and Marcus et al. do not compare topics structures but only used them to compute similarity between artifacts to recover tractability links.

Reverse Engineering In [21–23], the code's semantic groups recovered by hierarchical clustering are analyzed. A correlation matrix, comparable to ours, is introduced in [21] to depict the mutual relation between clusters of code fragments. Our work differs in language and data asymmetry but the mutual comparison of topics and not of artifacts comes close to our work.

Quality Assurance De Lucia et al. describe a tool and experiment using high level artifacts to assure the quality of code [24]. They propose to show similarity between the code currently worked on and the set of requirement documents in the IDE. Absence of similarity should lead the engineer in improving quality of identifiers and comments. Poshyvanyk et al. propose a tool that assesses and maintains the quality of software documentation by using traceability information [5]. This is done combining two different similarity types: A structural measure is computed bases on coupling of code (e.g., CBO coupling between

objects [25, 26]). A semantic similarity measure between documents is computed using LSI. A combination of both is capable to predict links in documentation paragraphs based on the coupling of code. Both works are interested in assuring or defining quality taking a second data source into account like proposed in our work. We differ in that we defined quality respective a second set of topics.

In [6], software documentation is searched for deficits with respect to question and answer pages like stack-overflow. They analyze the topics recovered by LDA in that they searched best covering stack-overflow and documentation piece for comparing both. The topics ordered by this distance uncover topic that are not covered in documentation of stack-overflow. The authors want to make a statement about topics in different media. Contrary, we expect to know what is described in the second data source and use it to build up a classification.

A topic based analysis of the Eclipse Modeling Framework is done in [27]. They apply LDA on the 30 most widely used Eclipse forums to gain insights on the prominent problems that users have to face. This approach is very related to ours despite we try to analyze the actual content and not the problems.

5 Conclusion

We have developed and demonstrated a method for the topic comparison of documentation and implementation of (OO) frameworks. We applied the method to the EMF implementation and a textbook by Steinberg et al. At both ends, data is decomposed into paragraphs (textbook) and methods (implementation) and LDA is applied on both sources separately. By examining the mutual relationships between the resulting documentation and implementation topics, we semi-automatically derived a classification taxonomy proposed in [11] and detected regions, i.e., very similar topics respective a list of defined measures that helped to further distinguish between the types. For our efforts of developing a comprehensive model [14–16] for EMF we gathered useful insights into the EMF domain: We detected most important topics respective implementation to be *adapt*, *command* and *resource* and the more documentation near topics *annot* and *feature* all located in the EMF Core region. We found environment topics that we are specially interested in, and implementation details that we like to neglect in our models. Further, a set of completely EMF unrelated topics is detected, like *metamodel* or *schema*, that encourages us for separate consideration outside to scope of EMF.

Future research is planed to improve the described method. In particular, we aim at a formalization and exploration of the parameter spaces that are used for topic comparison on which grounds we could explore a search-based justification of different options for classifications, topic model parameters, fitness, etc. The long term goal is to supply a flexible and partially self-optimizing method for getting a deeper understanding of topics uncovered by various topic models.

References

1. G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia, "Information Retrieval Models For Recovering Traceability Links between Code and Documentation," in *Proc. of ICSM (2000)*. IEEE, 2000, pp. 40–49.
2. G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 970–983, 2002.
3. G. Antoniol, G. Canfora, A. De Lucia, and E. Merlo, "Recovering Code to Documentation Links in OO Systems," in *Proc. of WCRE (1999)*. IEEE, 1999, pp. 136–144.
4. A. Marcus and J. Maletic, "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing," in *Proc. of ICSE (2003)*. IEEE, 2003, pp. 125–137.
5. D. Poshyvanyk and A. Marcus, "Using Traceability Links to Assess and Maintain the Quality of Software Documentation," in *Proc. of TEFSE (2007)*, 2007, pp. 27–30.
6. J. C. Campbell, C. Zhang, Z. Xu, A. Hindle, and J. Miller, "Deficient documentation detection: a methodology to locate deficient project documentation using topic analysis," in *Proc. of MSR*. IEEE, 2013, pp. 57–60.
7. W. X. Zhao, J. Jiang, J. Weng, J. He, E. Lim, H. Yan, and X. Li, "Comparing Twitter and Traditional Media Using Topic Models," in *Proc. of Advances in Information Retrieval - ECIR (2011)*, ser. LNCS, vol. 6611. Springer, 2011, pp. 338–349.
8. N. Alhindawi, O. M. Al-Hazaimeh, R. Malkawi, and J. Alsakran, "A Topic Modeling Based Solution for Confirming Software Documentation Quality," *IJACSA (2016)*, vol. 7, no. 2, pp. 200–206, 2016.
9. R. Pandita, R. Jetley, S. Sudarsan, T. Menzies, and L. Williams, "TMAP: Discovering relevant API methods through text mining of API documentation," *Journal of Software: Evolution and Process*, 2017.
10. G. Petrosyan, M. P. Robillard, and R. D. Mori, "Discovering Information Explaining API Types Using Text Classification," in *Proc. of ICSE (2015)*. IEEE, 2015, pp. 869–879.
11. W. Maalej and M. P. Robillard, "Patterns of Knowledge in API Reference Documentation," *IEEE Trans. Software Eng.*, vol. 39, no. 9, pp. 1264–1282, 2013.
12. N. Kumar and P. T. Devanbu, "OntoCat: Automatically categorizing knowledge in API Documentation," *CoRR*, vol. abs/1607.07602, 2016.
13. S. Beyer and M. Pinzger, "A Manual Categorization of Android App Development Issues on Stack Overflow," in *Proc. of ICSME (2014)*. IEEE, 2014, pp. 531–535.
14. J. Härtel, L. Härtel, R. Lämmel, A. Varanovich, and M. Heinz, "Interconnected Linguistic Architecture," *The Art, Science, and Engineering of Programming*, vol. 1, 2017.
15. J. Favre, R. Lämmel, and A. Varanovich, "Modeling the Linguistic Architecture of Software Products," in *Proc. of MODELS (2012)*, ser. LNCS, vol. 7590. Springer, 2012, pp. 151–167.
16. R. Lämmel and A. Varanovich, "Interpretation of Linguistic Architecture," in *Proc. of ECMFA (2014)*, ser. LNCS, vol. 8569. Springer, 2014, pp. 67–82.
17. D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed. Pearson Education, 2008.

18. D. Movshovitz-Attias and W. W. Cohen, "KB-LDA: Jointly Learning a Knowledge Base of Hierarchy, Relations, and Facts," in *Proc. of ACL (2015)*. The Association for Computer Linguistics, 2015, pp. 1449–1459.
19. K. Tian, M. Revelle, and D. Poshyvanyk, "Using Latent Dirichlet Allocation for Automatic Categorization of Software," in *Proc. of MSR (2009)*. IEEE, 2009, pp. 163–166.
20. G. Antoniol, G. Canfora, G. Casazza, and E. Merlo, "Tracing Object-Oriented Code into Functional Requirements," in *Proc. of IWPC (2000)*. IEEE, 2000, pp. 79–86.
21. A. Kuhn, S. Ducasse, and T. Gîrba, "Enriching Reverse Engineering with Semantic Clustering," in *Proc. of WCRE (2005)*. IEEE, 2005, pp. 133–142.
22. A. Kuhn, O. Greevy, and T. Gırba, "Applying Semantic Analysis to Feature Execution Traces," in *Proc. of PCODA (2005)*. IEEE, 2005, pp. 48–53.
23. A. Kuhn, S. Ducasse, and T. Gîrba, "Semantic clustering: Identifying topics in source code," *Information and Software Technology*, vol. 49, no. 3, pp. 230–243, 2007.
24. A. D. Lucia, M. D. Penta, R. Oliveto, and F. Zurolo, "Improving Comprehensibility of Source Code via Traceability Information: a Controlled Experiment," in *Proc. of ICPC (2006)*. IEEE, 2006, pp. 317–326.
25. S. R. Chidamber and C. F. Kemerer, "Towards a Metrics Suite for Object Oriented Design," in *Proc. of OOPSLA (1991)*. ACM, 1991, pp. 197–211.
26. ——, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.
27. N. Kahani, M. Bagherzadeh, J. Dingel, and J. R. Cordy, "The problems with eclipse modeling tools: a topic analysis of eclipse forums," in *Proc. of MoDELS (2016)*. ACM, 2016, pp. 227–237.