# DoMoRe – A Recommender System for Domain Modeling

Henning Agt-Rickauer[1], Ralf-Detlef Kutsche[2] and Harald Sack[3]
[1]*Hasso Plattner Institute for IT Systems Engineering, University of Potsdam, Potsdam, Germany*
[2]*Database Systems and Information Management Group, Technische Universität Berlin, Berlin, Germany*
[3]*FIZ Karlsruhe & Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*

Keywords: Domain Modeling, Recommender System, Information Extraction, Knowledge-based Modeling.

Abstract: Domain modeling is an important activity in early phases of software projects to achieve a shared understanding of the problem field among project participants. Domain models describe concepts and relations of respective application fields using a modeling language and domain-specific terms. Detailed knowledge of the domain as well as expertise in model-driven development is required for software engineers to create these models. This paper describes DoMoRe, a system for automated modeling recommendations to support the domain modeling process. We describe an approach in which modeling benefits from formalized knowledge sources and information extraction from text. The system incorporates a large network of semantically related terms built from natural language data sets integrated with mediator-based knowledge base querying in a single recommender system to provide context-sensitive suggestions of model elements.

## 1 INTRODUCTION

**Motivation.** Model-driven engineering (MDE) proposes systematic use of models as primary development artifacts for software system construction (Whittle et al., 2014). These models describe different aspects of a system on a higher level of abstraction using particular modeling languages (e. g., UML, entity-relationship diagrams, or domain-specific languages). MDE aims at continuously refining models and generating source code. Consequently, the effort of manually creating code with programming languages is reduced and recurring tasks are automated.

An important activity in early phases of model-driven software development is domain modeling (Evans, 2004; Fowler, 2010). Its goal is to create models that reflect the conceptual structures of a business domain. These models contain domain-specific terms and their relations in order to improve the understanding of the problem field among stakeholders.

Domain modeling requires expertise in model-driven software development, such as finding the right abstractions, creating metamodels and correct usage of generalizations/specializations and aggregations. Assuming that software engineers have these competencies, these techniques are usually applied to different application areas and industrial sectors. The engineers are required to have detailed knowledge of the domain in order to build domain-specific models and derive appropriate refined implementation models. Assembling domain knowledge is a time-consuming manual process (e.g., talking to domain experts and reading specific documentation). Recent modeling environments (e.g., Eclipse Modeling Project, MagicDraw) provide sophisticated support with respect to the correct usage of modeling languages and verification of models, but assistance with regard to the actual content and meaning of the model elements is very limited (Segura et al., 2016; Kuhn, 2010).

**Problem Statement.** Domain modeling was and still is a challenging task (Mylopoulos et al., 1990; Störrle, 2010). It involves collecting a lot of pieces of information delivered by different types of persons, documents and other knowledge sources. Domain modeling is a knowledge intensive process and requires intensive collaboration between engineers and domain experts. The automation of domain modeling has been addressed by research (Reinhartz-Berger, 2010), but support of this activity is still an open issue (Frank, 2014). The main challenges of domain modeling and knowledge acquisition are as follows.

Solutions that build on reusable libraries with domain information, such as Domain Engineering, suffer from a cold start problem. Reusable domain knowledge will become only available if enough solu-



Agt-Rickauer, H., Kutsche, R-D. and Sack, H.
DoMoRe – A Recommender System for Domain Modeling.
DOI: 10.5220/0006555700710082
In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018)*, pages 71-82
ISBN: 978-989-758-283-7

tions have already been developed using that methodology while new projects already want to benefit from this domain knowledge. In the end, domain models frequently have to be developed from scratch (Frank, 2013).

Collaborations of technical stakeholders (modeling experts) and non-technical stakeholders (domain experts) require a time-consuming learning phase during a project (Ionita et al., 2015). Domain experts are often not familiar with modeling notations, and modeling experts usually have to develop a deeper understanding of the domain concepts and terms in order to arrange them properly in domain models. The biggest effort is at the modeler's desk, because it is usually more time-consuming to find, understand, and process all available domain information than to learn a few modeling concepts of a visual notation.

Domain information is contained in arbitrary sources. Structured information sources (e.g., databases, XML documents, knowledge bases, models) may be available, but uniform access to all sources is not available in most cases and may only be facilitated by building additional search engines on top of them. Unfortunately, the amount of structured information sources is vanishingly small compared to unstructured information. It is estimated that 80 percent of existing data is unstructured[1]. Domain information is often contained in natural language documents (e.g., text books, manuals, requirements specifications). Relevant facts have to be manually located first and then interpreted.

Finally, the availability of large conceptual knowledge bases containing domain information is very limited. Only a few handcrafted semantic databases exist (e.g., WordNet, ConceptNet, Wikidata) that by far do not cover the variety of possible domains. Most approaches of information extraction (Banko et al., 2007) and automatically created knowledge bases (e.g., DBpedia, YAGO) focus on factual knowledge on instance level that cannot be used for domain modeling on conceptual level. Additionally, the core of many works (e.g., YAGO, BabelNet, DBpedia) relies on a single source of information only: extraction from Wikipedia's structured parts (e.g., infoboxes, categories).

**Contributions and Outline.** In this paper, we present a domain modeling recommender (DoMoRe) system that incorporates a ready-to-use large-scale knowledge base of domain-specific terms and their relationships. DoMoRe also uses a set of existing knowledge bases to retrieve domain information and is extensible with additional knowledge bases at low effort. Connected knowledge sources are used automatically during modeling to provide context-sensitive suggestions of model elements. The recommender system is integrated into a widely used modeling tool, the Ecore Diagram Editor of the Eclipse Modeling Project.

The remainder of the paper is organized as follows. Section 2 presents the general approach and details the model refinement steps that our system supports. In Section 3 we describe how existing knowledge sources are used and how the knowledge base of related terms was created. Section 4 details the implementation of the recommender system and shows how provision of contextual information and search-based suggestions operate. In Section 5 we report on experiences using DoMoRe in different domains and scenarios. Related work is given in Section 6, and Section 7 concludes the paper and describes future work directions.

## 2 SEMANTIC MODELING SUPPORT

In this section we introduce the concept of semantic modeling support and detail our approach. *Modeling*: The activity of creating and refining models. In our case these models are domain models that focus on concepts and relationships of various application areas. *Support*: Modeling activities are assisted with context-sensitive pieces of information. Tool support is completely automated in contrast to guidelines or methodologies. *Semantic*: Modeling support focuses on the domain-specific terms and their relationships in domain models in contrast to syntactic modeling language assistance.

**General Support Procedure.** The semantic modeling support works as follows: (1) At some point of time during domain modeling a manual change in the model is made. This is usually referred to as model refinement, the activity in which a developer creates, modifies or deletes a model element (e.g., a new class). We concentrate on supporting the modifications that add new content to the model. All detailed scenarios are described in the next paragraph. (2) Based on the current state of the model, domain knowledge is acquired automatically. Knowledge acquisition is based on the terms that are used to name the elements (e.g., class names or association names). We pursue two strategies: First, we

[1] https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/ (Last accessed July, 2017)

exploit existing structured knowledge sources to acquire the required domain terms and their relations. We employ mediator-based querying for a uniform access to this knowledge. Secondly, it is a well known problem (Colace et al., 2014) that existing knowledge bases (often created manually) do not contain enough information or do not exist at all for respective target domains. We address this issue by the automated creation of own semantic terminology networks from natural language datasets that cover a variety of domains. Both approaches are detailed in Section 3. (3) Acquired knowledge is transformed automatically into appropriate suggestions (e.g., related classes, possible sub- or super-classes) and presented to the user. It is the goal to present semantically related model elements that support the developer's decisions on what to include in the model and how to connect the elements. After that the procedure starts all over again.

**Modeling Support Scenarios.** Many opportunities exist to create and manage domain models. Domain modeling is not necessarily bound to using one specific modeling language. For example, UML class diagrams, ER models, and ontologies can be used. All approaches have in common that the respective modeling language is used to express conceptual structures of a domain using specific terms to improve understanding of the problem field. Since our semantic modeling support concentrates on the terms in domain models, the methods presented in this paper are applicable to several modeling languages. Nevertheless, we had to exemplarily choose one approach to illustrate our work, namely UML-like class diagrams, because they are the most widely used modeling paradigm in industry (Reggio et al., 2014; Hutchinson et al., 2014).

During domain model development the user has several options to change the model. In the following we itemize for which modeling activities what kind of support will be accomplished. We distinguish between two different kinds of support. First, contextual information will be provided if an element of a domain model is selected by the developer (Scenarios 1 and 2). Context information includes possible related model elements with all kinds of relationships the modeling language offers. Second, if a new element is created, automated suggestions will be provided on how to name the element (Scenarios 3 to 9). The support depends on the type of connection between the new element and existing elements of the model.

*Scenario 1 – Selection of a class.* The goal of providing contextual information is the recommendation of possible connected model elements together with their types of relations for a selected domain model element. In case a class is selected (c.f., Figure 1) possible generalizations/specializations, aggregations (containers and parts), and associations are shown. Related classes are either unconnected classes or classes that are connected with an association that has no name.

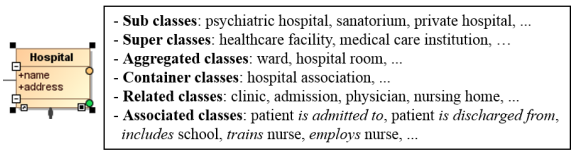



Figure 1: Contextual information for a selected class (Scenario 1).

*Scenario 2 – Selection of an association.* If an association is selected, alternative association names, and possible other connected classes for each association end will be shown (c.f., Figure 2). Note that if nothing is selected, contextual information for every element of the model will be shown in a summarized form.

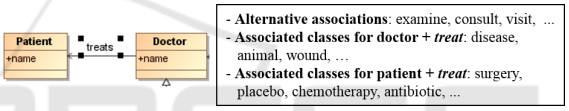



Figure 2: Contextual information for a selected association (Scenario 2).

*Scenario 3 – Creation of a class (no connection).* Modeling tools usually offer the creation of new classes in a model without any connection. Typically, this happens, when classes are added to the diagram and the respective connections are drawn afterwards (c.f., Figure 3). In this case, class name suggestions are dependent on all existing class names in the model. Particularly, in the list of suggestions class names should appear that are related to all of the existing classes ordered by relevance.

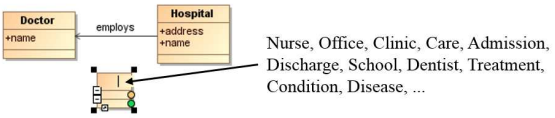



Figure 3: Related class name suggestions while adding a class without a connection (Scenario 3).

*Scenario 4 – Creation of a sub class.* A sub class will be created, when the developer uses the specialization link starting from an existing class to empty space in the diagram (c.f., Figure **??**). In this case, class name suggestions are dependent on the linked super class. In the example, different types of doctors are shown (different kinds of medical specialists).

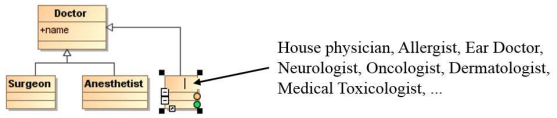



Figure 4: Sub class name suggestions while adding a specialization (Scenario 4).

*Scenario 5 – Creation of a super class.* Analogous to the sub class creation, a super class will be created when using the generalization link. The example shows the recommendation of more general terms for doctor (c.f., Figure 5).

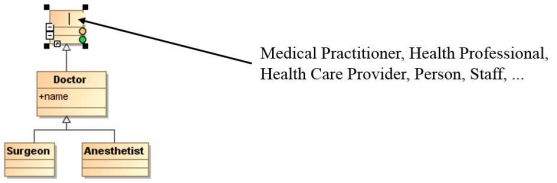



Figure 5: Super class name suggestions while adding a generalization (Scenario 5).

*Scenario 6 – Creation of an aggregated class.* In case the developer uses a composition or aggregation link starting from an existing class, an aggregated class will be created in the diagram. The example in Figure 6 shows possible parts of a hospital.

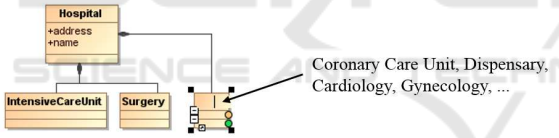



Figure 6: Aggregated class name suggestions while adding an aggregated class (Scenario 6).

*Scenario 7 – Creation of a container class.* If the opposite direction of a composition or aggregation relation is used, a container class will be created. In the example used in Figure 7, suggestions are provided what a hospital can be part of.

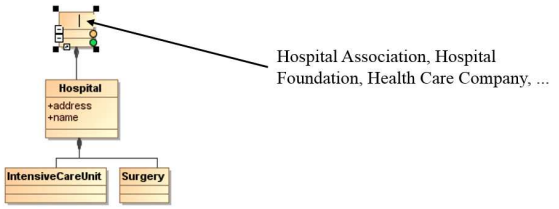



Figure 7: Container class name suggestions while adding a container class (Scenario 7).

*Scenario 8 – Creation of an associated class.* An associated class will be created, if the developer draws an association link from a class to empty space in the diagram (a new class and an association without a name will be created). Names for the new related class will be recommended (c.f., Figure 8). This scenario is very similar to Scenario 3, but the suggestions are dependent on the linked class only.

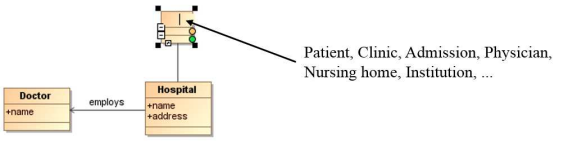



Figure 8: Associated class name suggestions while adding an associated class (Scenario 8).

*Scenario 9 – Creation of association.* If the developer creates an association link between two classes, association names (verbs) will be provided. The suggestions are dependent on both class names. In case the association does not have a direction, verbs are suggested that apply to both directions.

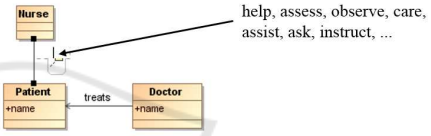



Figure 9: Association name suggestions while adding an association (Scenario 9).

## 3 DOMAIN KNOWLEDGE SOURCES

Our intended modeling support requires a large body of background knowledge in order to provide model element suggestions for nearly every possible domain. Since the support focuses on the terms used in the models, we concentrate on knowledge sources that provide lexical information.

We pursue two strategies: First, we exploit existing structured knowledge sources to acquire the required domain knowledge. Knowledge bases and ontologies are automatically queried for terms of a model to retrieve related terms. Secondly, we target the lack of conceptual knowledge bases by the automated creation of a semantic network of terms from natural language datasets.

**Mediator-based Knowledge Base Querying.** As described in the introduction, only a few knowledge bases exist that contain conceptual knowledge. WordNet (Fellbaum, 1998) is the most widely used lexical database for the English language. Other important resources are BabelNet (Navigli and Ponzetto,

2012), a multilingual encyclopedic dictionary, and Cyc (Lenat, 1995) and ConceptNet (Speer and Havasi, 2012), both common sense knowledge bases. Most of the other large publicly available knowledge bases (e.g., DBpedia, YAGO, Wikidata) consist of a relatively small ontology schema describing the model of the data and a large body of factual knowledge. Nevertheless, these schemata can be used for modeling suggestions.

The greatest challenge in using these knowledge sources is the unavailability of uniform access to lexical information. Heterogenous data models prevent querying the knowledge bases in a consistent way. Lexical information of terms and their relationships exist on schema level, intermediate proprietary data models and on instance level.

Our approach proposes a mediator-wrapper solution. A mediator allows the interaction of a user or system with heterogeneous data sources in a uniform way. Knowledge bases remain as they are, a wrapper is responsible for content translation, and the mediator provides a single point of access to the information for the modeling recommendations. Figure 10 shows the architecture of our approach. We differentiate between three different layers.

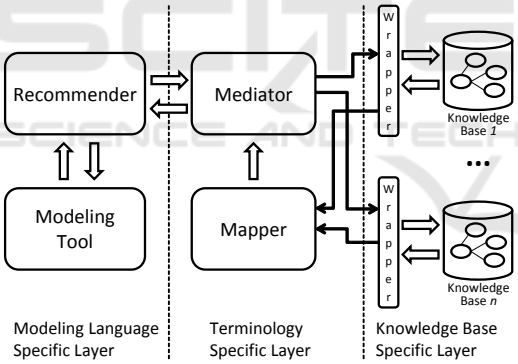



Figure 10: Three layer mediator-wrapper architecture to retrieve terminological information from heterogenous knowledge bases.

In the *modeling language specific layer* the developer uses the modeling tool and interacts with the recommender. This layer deals with elements such as classes and associations, and the recommender suggests these types of elements depending on the content of a model.

The mediator and mapper are located in the *terminology specific layer*. In this layer domain-specific terms used in a model are relevant (e.g., noun terms and their related terms). The mediator is responsible for translating from terminology specific content to the modeling layer and vice versa. It also manages a set of knowledge bases and their corresponding wrappers and submits queries to them whenever necessary. The mapper collects and integrates results of the wrappers and provides the information to the mediator.

In the *knowledge base specific layer* the wrappers communicate with the knowledge bases. Each wrapper has to deal with different query languages and formats (e.g., OWL, RDF, SPARQL, JSON) and different ways of modeling (e.g., graphs, concepts, synsets). We support the automated integration of three types of knowledge base data models without the need of any development: (1) Ontology schemata: concepts and relationships modeled using OWL or RDFS classes and object properties, (2) SKOS-based vocabularies: terms modeled with concepts and broader/narrower/related relationships, and (3) Knowledge bases using the lemon model: a specific way of modeling lexicons of ontologies (McCrae et al., 2012).

In case none of these data models apply, we support any knowledge base that offers a SPARQL endpoint. The effort of adding a new knowledge base to the system is comparatively low, it is only necessary to specify a small set of queries for taxonomic, part/whole, related and verbal relationships.

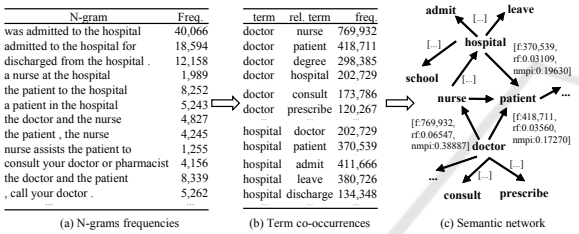
**Extraction of Semantically Related Terms.** We apply natural language processing techniques on a large text dataset to identify terms on conceptual level and their relationships. The approach relies on syntactical properties of sentences and statistical features of text corpora to perform a *domain-independent* extraction. Large text collections include a lot of redundancy and paraphrasing (Banko, 2009). That means equal facts are repeated in several documents and are formulated differently. Additionally, natural language has the property that certain lexical items tend to co-occur more often than others. That implies that words with similar meanings occur in similar contexts, known as the distributional hypothesis (Turney and Pantel, 2010).

We exploit these features to automatically build a large database of semantically related terms. We processed the Google Books N-Gram Corpus (Michel et al., 2011), a dataset derived from 5 million books with over 500 billion words. It covers a wide range of domains, because it contains scientific literature of lots of areas as well as fiction and non-fiction books.

The dataset provides the information how often a certain word or word sequence occurs within the original text corpus (an n-gram is a sequence of *n* consecutive words). For example, *"the doctor and the patient – 8,339"* is one example of the 700 million 5-grams in the dataset (c.f., Figure 11a). We apply part-

of-speech (POS) tagging on all n-grams to identify technical terms (Williams, 2013) and record how often concept terms in different contexts co-occur (e.g., *doctor – patient – 418,711 times*, c.f., Figure 11b). We exclude proper nouns and named entities (e.g., city names, persons). Using this information, we are able to obtain for every term a set of related terms and their frequencies and construct a semantic network (c.f., Figure 11c).

A detailed description of the construction is given in (Agt and Kutsche, 2013). Several improvements have been made in comparison to the original version. The semantic network now covers verbal and ternary relations in order to suggest associations in domain models, several heuristics were applied to extend the length of extracted terms, and a more sophisticated ranking of terms was developed.




| N-gram | Freq. |
|---|---|
| was admitted to the hospital | 40,066 |
| admitted to the hospital for | 18,594 |
| discharged from the hospital . | 12,158 |
| a nurse at the hospital | 1,989 |
| the patient to the hospital | 8,252 |
| a patient in the hospital | 5,243 |
| the doctor and the nurse | 4,827 |
| the patient , the nurse | 4,245 |
| nurse assists the patient to | 1,255 |
| consult your doctor or pharmacist | 4,156 |
| the doctor and the patient | 8,339 |
| , call your doctor . | 5,262 |

(a) N-grams frequencies

| term | rel. term | freq. |
|---|---|---|
| doctor | nurse | 769,932 |
| doctor | patient | 418,711 |
| doctor | degree | 298,385 |
| doctor | hospital | 202,729 |
| doctor | consult | 173,786 |
| doctor | prescribe | 120,267 |
| hospital | doctor | 202,729 |
| hospital | patient | 370,539 |
| hospital | admit | 411,666 |
| hospital | leave | 380,726 |
| hospital | discharge | 134,348 |

(b) Term co-occurrences

(c) Semantic network

Figure 11: Information extraction procedure to construct a large-scale semantic network from co-occurring terms in n-gram natural language datasets (*f* – absolute frequency, *rf* – relative frequency, *npmi* – normalized point wise mutual information).

In essence, the semantic network is a large-scale graph in which each term is a node and each directed edge denotes a weighted relationship between the terms. It comprises 5.7 million unique one-word terms and multi-word expressions and 114 million relationships. Each relationship is quantified with the absolute co-occurrence frequency, a computed relative frequency, and the pointwise mutal information (PMI) measurement (see Section 4 – Ranking – for more details on this associativity score between terms). While the extraction required sophisticated hardware and runtime, the semantic network requires just 7 GB disk space and thus is usable on standard PC hardware.

# 4 RECOMMENDER SYSTEM

In this section we detail the realization of the domain modeling recommender (DoMoRe) system. We describe how the domain information of a set of knowledge bases and a self-created semantic network of terms is used and transformed to recommendations of model elements according to our nine modeling support scenarios.

Essentially, the task of the recommender system is as follows. For a given model element a set of related model elements has to be determined that are connected with a specific type of relationship (e.g., all possible subclasses of the class ”Doctor” or all related classes of ”Doctor” and ”Hospital”). In the following we first discuss a mapping of semantic relationships across different knowledge representations. After that, the architecture of the recommender system is presented. Then, we detail the features of the recommender system and describe how suggestions are ranked by relevance.

**Semantic Relationships.** Domain models describe concepts and relationships of an application domain using the means of a modeling language. Although there is an ongoing foundational discussion in the conceptual modeling community (Henderson-Sellers et al., 2015; Atkinson and Kühne, 2015) on how to properly represent real-world concepts with modeling languages, UML class diagrams are the most widely used modeling paradigm in industry (Reggio et al., 2014; Hutchinson et al., 2014). In this paragraph we analyze the semantic relationships of UML class diagrams from a lexical viewpoint and their representations in other knowledge sources (c.f., Figure 10 for the three layers).

We reviewed literature from database research (Storey, 1993), linguistics (Maroto Garca and Alcina, 2009; Chaffin and Herrmann, 1984), information systems (Olivé, 2007; Guizzardi, 2005), and semantic web research (Almeida et al., 2011; Huang, 2010) and relate the different types of relationships to each other. Table 1 provides an overview, details are given as follows.

*Specialization and Generalization* are hierarchical abstraction mechanisms in UML to refine abstract classes to more specific ones and to group specific classes to more abstract ones. In lexical semantics these conceptual relationships are referred to as *hyponymy* and *hypernymy* between words or phrases. They are mapped to *subClassOf*-relationship (and its inverse) in RDF/OWL ontologies and to the *broader term* and *narrower term* relation in thesaurus specification (e.g., based on ISO 25964).

*Aggregation* is used to specify a part-of relationship between two UML classes. We subsume both aggregation (parts can exist independently) and composition (parts cannot exist independently) under the term aggregation. In linguistics part-whole relations are referred to as meronymic relationships (meronyms being the parts and holonyms being the wholes). Part-

Table 1: Corresponding semantic relationship types of different modeling paradigms.

| **Modeling Language Relationship** | **Lexical-Semantic Relationship** | **Knowledge Source Relationship** |
|---|---|---|
| Specialization | Hyponymy | Subclass, Narrower Term |
| Generalization | Hypernymy | Subclass (inv.), Broader Term |
| Aggregation (Part) | Meronymy | HasPart (SPW) *Meronym (WordNet)* |
| Aggregation (Whole) | Holonymy | PartOf (SPW) *Holonym (WordNet)* |
| Association (named) | Agent-Action | Object Property |
| Association (unnamed) or group of classes | Semantic Relatedness | Related Term |

whole relationships are neither directly supported in thesaurus definition nor in RDF/OWL ontology specification. There is a W3C best practice specification Simple Part Whole (SPW) with *hasPart* and *partOf* relationships that we support. Nevertheless, there are knowledge bases that contain part-whole relationships but use a non-standard vocabulary (e.g., WordNet).

*Association* is the third kind of conceptual relationship we analyzed with respect to other representations. We differentiate between two types: unnamed associations to express a simple dependency between two domain model classes and named associations further specifying the kind of association (usually with a verb). In linguistics named dependencies fall into the category of case relationships (Chaffin and Herrmann, 1984), more specifically, in our case, *agent-action* relationships. To some extent RDF/OWL *object properties* with domain and range restrictions can be compared to named associations. In lexical semantics the unnamed association is referred to as *semantic relatedness*, an associative relationship describing any functional relationship between two words. The *related term* relationship of thesauri is mapped to this relationship. From a lexical point of view the unnamed association is similar to a set of classes grouped by the diagram itself.

Based on these mappings we are able to retrieve related domain model elements from different knowledge sources for all modeling support scenarios.

**Components of the Recommender System.** Figure 12 depicts the architecture of the DoMoRe recommender system. DoMoRe is integrated into the Eclipse environment using a set of plug-ins. The *Model Listener* observes changes in Ecore models that are developed with the Ecore diagram editor. Whenever a change in a model is made the current content of the model is retrieved together with the newly added or changed model element and its relationships. The *Recommender* is notified and coordinates all subsequent steps of modeling suggestions. First, the domain model is transformed into a lexical-semantic representation using the domain-specific terms and semantic relationship mappings (c.f., Table 1). Based on this representation the *Semantic Network* is queried for related terms and directly delivers ranked lists of related terms to the recommender. The *Ontology Connector* manages the set of connected knowledge bases and is queried as well. It incorporates the mediator and mapper (c.f., Figure 10) and operates the translation of the terminological queries to knowledge base specific queries and the integration of results. The recommender controls two components with which the user interacts. The *Model Advisor* is a view in the Eclipse environment that displays contextual information of the model elements. It shows possible generalizations, specializations, aggregations, associations, and related elements. The developer can use this view to easily add new content to a domain model by drag & drop of suggested elements to the diagram. The appropriate relationships will be created automatically. *Semantic Autocompletion* is triggered in case a new element in the model is named or the name of an existing element is changed. This feature behaves like a search engine. A context-sensitive pop-up list with names for the respective element is displayed and the suggestions are filtered while typing.

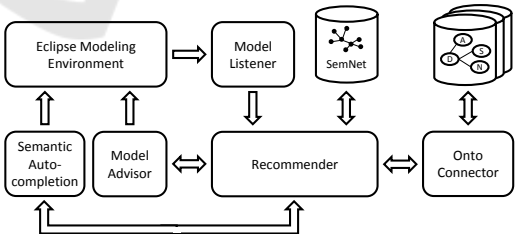



Figure 12: Architecture of the DoMoRe recommender system.

**Recommendation Generation.** In this paragraph we provide insights into the features of the recommender system by means of examples. We demonstrate the knowledge retrieval and recommendation generation for Scenario 3: A new unconnected class is created and names for that class are suggested. Figure 13a shows a small domain model example containing two classes connected with a named associ-

ation. After creating the new class the model listener triggers the recommender and the lexical representation of the domain model is generated (c.f., Figure 13b). The information need depends on the model refinement step. In this case, noun terms are required that are semantically related to both *Hospital* and *Doctor* (c.f., Figure 13c).

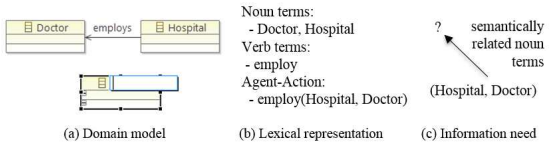



Figure 13: Lexical preparation in the procedure of the recommendation generation.

In the following, the information need is decomposed to separate lexical queries for each term and relationship type (c.f., Figure 14d). The main reason for retrieving the information separately is that practically no conceptual knowledge base exists that contains n-ary relationships. In contrast, our semantic network directly supports ternary relationships allowing more precise results for term pairs, but for more than two terms separate queries have to be executed in any case. In the next step, the semantic network is queried for each term (c.f., Figure 14e) and each connected knowledge base is queried for each term (c.f., Figure 14f).

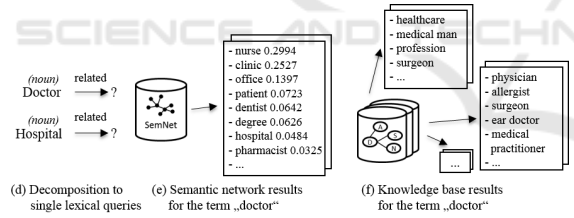



Figure 14: Retrieval in the procedure of the recommendation generation.

So far, for each term of the original domain model, and for each relationship type, and for each knowledge source separate lists of related terms have been determined[2]. First, results from the knowledge bases are integrated based on the following principle. Per term the distinct union of all intermediate results (e.g., related terms of ”doctor” from WordNet, BabelNet, ConceptNet, etc.) is created it is recorded how often each term occurred. The resulting lists have preliminary order indicating more important terms appear first (c.f., Figure 15g).

[2]For example, if three classes exist in the model and broader and narrower relationships have to be retrieved from five knowledge sources, 30 intermediate result lists will be generated.

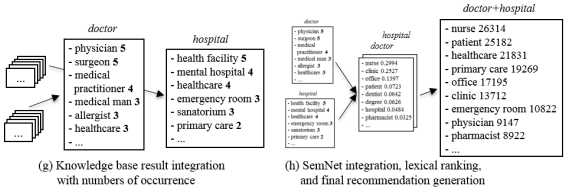



Figure 15: Integration and ranking in the procedure of the recommendation generation.

In the final step, the presorted knowledge base results are integrated into the semantic network results. First, the knowledge base result and the respective semantic network result for each term are joined using the importance weight. It assures that terms found in many knowledge bases receive more prominent positions in the final ranking. Secondly, one final list of recommended terms is created. Separate results are intersected and relative frequencies of common terms are multiplied. The final list is divided into $n$ segments: Terms that are related to $n$ query terms appear first. After that terms follow that are related to $n-1$ query terms, and so forth. Finally, a sorting by relevance is achieved by applying the pointwide mutal information score (c.f., Figure 15h). This measurement is explained in the paragraph ”Ranking”.

Generated lists of ranked terms are directly used in the *Semantic Autocompletion* feature of the recommender system. Whenever the name of a class or association is edited a context-sensitive pop-up list of related terms is shown. It behaves like a search engine and is filtered while typing (c.f., Figure 16).

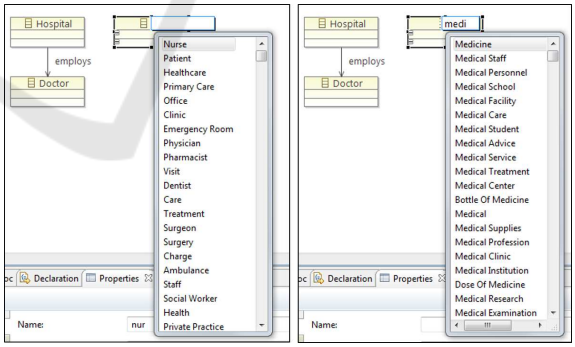



Figure 16: Semantic Autocompletion of the recommender system: Context-sensitive name predictions and infix search.

For the other scenarios preparation, retrieval, integration, and ranking is similar. They only differ in the queried relationship type (e.g, subclasses/narrower terms instead of related terms). To display contextual information several relationship types are queried and displayed in the *Model Advisor* view (c.f., Figure 17).

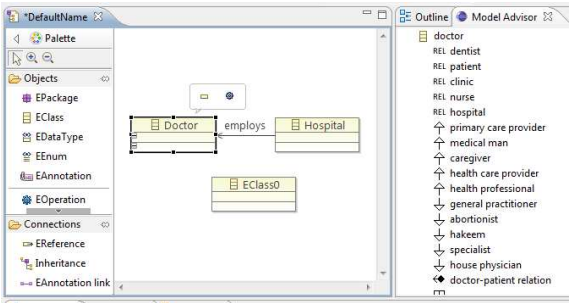



Figure 17: Model Advisor of the recommender system: Suggesting possible related classes, superclasses, subclasses, and aggregations.

**Ranking.** It is likely that queries to our semantic network and the connected knowledge bases deliver numerous related terms (up to a few thousand for each query). The ranking implemented in the recommender component is responsible for presenting the most relevant model elements first. Hence, if a list of related terms is retrieved, it will be ordered and the most important terms appear at the top. This is achieved by combining different relatedness measures.

From the construction of the semantic network we know *absolute frequencies* of co-occurring terms (c.f., Figure 11b). For each term in the network we compute *relative frequencies* with respect to each connected related term. This normalization allows to compare the relatedness among different terms. Both measures allow a basic ranking of terms, but they have a shortcoming: very general terms (e.g., time, man, year) that appear in almost all contexts are likely to be ranked on prominent positions.

To overcome this disadvantage we implement an information theoretic measurement: *Pointwise mutal information* (PMI) and its normalized form (c.f., Equations 1). It measures the dependency between the probability of coinciding events and the probability of individual events (first introduced into lexicography by (Church and Hanks, 1990)).

$$pmi(x,y) = \log\left[\frac{p(x,y)}{p(x)p(y)}\right] \quad npmi(x,y) = \frac{pmi(x,y)}{-\log[p(x,y)]} \tag{1}$$

Applying PMI to the semantic network means that $x$ and $y$ are terms, and PMI relates the probability of their coincidence $p(x,y)$ with the probabilities of observing both terms independently $p(x)p(y)$. PMI is an associativity score of two terms taking into account their individual corpus frequency, thus, very frequent and general terms receive lower scores. Unfortunately, this measurement also has a shortcoming: Although very general terms are ranked lower, very rare terms that co-occur only very few times with other terms tend to receive high scores.

Finally, in order to achieve a balanced ranking our recommender system uses the lexicographers mutual information (LMI), which is the NPMI score multiplied with the absolute co-occurrence frequency (Milajevs et al., 2016).

## 5 DoMoRe IN PRACTICE

In this section we summarize experiences of using our recommender system in different settings and domains. A first prototype of the system was used in the context of the BIZWARE project (Agt et al., 2012; Agt and Kutsche, 2013). BIZWARE investigated the potential of domain-specific languages (DSLs) and model-driven engineering for small and medium enterprises in different domains like healthcare, manufacturing/production, finance/insurance, publishing and facility management. The actors in the setting were software engineers of the respective companies aiming at the introduction of DSL-based workflows to improve their development tasks. Modeling experts from research closely worked together with the software engineers and provided the recommender system. The companies planned to used DSLs in customer projects, but in the course of the project it turned out that using DSLs for modernizing own software products and development infrastructures was more effective. The software engineers had little experience with DSLs, therefore, DoMoRe mainly supported the domain analysis phase to determine and agree on domain-specific terms that were later used in metamodels of the DSLs. Especially, the suggestions helped in the abstraction process to correctly differentiate between class and instance level. Analysis of the modeling sessions showed that the ranking had to be improved (too general terms at top positions) and the software engineers missed suggestions of relations between domain-specific terms (extraction of verbal relations was not available at that time).

DoMoRe was also used in the dwerft project (Agt-Rickauer et al., 2016), a research collaboration to apply Linked Data principles for metadata exchange through all steps of the media value chain. The project successfully integrated of a set of film production tools based on the Linked Production Data Cloud, a technology platform for the film and tv industry to enable software interoperability used in production, distribution, and archival of audiovisual content. One of the crucial tasks of the project was the development of a common data model conveying all metadata from the different production steps (e.g., film script, production planning, on-set information, post production, distribution). The actors in the setting were do-

main experts of the respective tasks (most of them with no technical background) and modeling experts that built the domain models and metadata schemata. A lot of interviews with domain experts had to be conducted to collect domain-specific knowledge and discuss drafts of domain models. DoMoRe mainly supported subsequent modeling steps after the meetings and preparation of interviews with more extensive models that allowed a more efficiently agreement on necessary metadata.

Currently, DoMoRe is used in the context of the AdA project[3], an interdisciplinary research group in which film scholars work together with computer scientists to support empirical film studies with tool-supported semantic video annotation and automated video analysis. The project objective is to reduce the effort of elaborate, manual annotation routines to accelerate the film-scientific analysis of audiovisual movement patterns on the level of larger data sets. All annotation data and analysis results will be published as Linked Open Data using the project's semantic vocabularies. The collaboration with respect to domain modeling is similar to the aforementioned project. We expect similar support by using the recommender system for the domain expert interviews.

## 6 RELATED WORK

**Modeling Assistance.** Systems for modeling assistance provide additional information and features during the modeling process to help the development of models. They usually concentrate on two areas: (1) Creation of model libraries or similar content and (2) developing assistance frameworks and features that use these libraries. Largest known *model repository* in the area of UML models and metamodels is the Gothenburg UML Repository (Hebig et al., 2016). It contains over 20,000 models crawled from the web, images, and from GitHub (only a collection of almost 1,000 models is publicly searchable). It is not surprising, that the majority of the models are implementation models rather than domain models (e.g., a search for "hospital" or "doctor" returned 7 models while a search for "interface" returned 90 models). Other important resources are ReMoDD (France et al., 2012), MOOGLE (Lucrédio et al., 2012), the AtlanMod Metamodel Zoos[4] (altogether containing a few hundred models).

The HERMES project a framework to build model *recommendation systems* in order to support reuse of software models (Dyck et al., 2014). Its main goal is to provide tool support for building model libraries and offering the implementation infrastructure to create recommenders that use content of these libraries. The EXTREMO assistant (Segura et al., 2016) is a similar tool to facilitate metamodel development with uniform model element search in a model repository. We are sharing the same objectives, but both systems face a cold start problem: Reusable content will become only available, if enough solutions have already been developed or converted to the repository, but new projects already want to benefit from the domain knowledge. To some extent this challenge is addressed by using WordWeb/WordNet, which we also use, but those databases contain roughly 150,000 concepts in contrast to our semantic network with 5.7M terms.

**Knowledge-based Modeling.** There are several works on how model-driven engineering can benefit from formalized knowledge. On the *conceptual level* there are approaches to unify ontological and software modeling paradigms (Kühne, 2016), approaches to adopt modeling concepts from each other (Henderson-Sellers et al., 2015), and to extend MDE languages with ontological foundations (Guizzardi, 2005).

The OntoDSL framework (Walter et al., 2014) leverages *ontology technologies* on metamodel level (e,g, reasoning) to assist DSL users with detecting inconsistencies on model level. The CoCoViLa tool (Ojamaa et al., 2015) generates metamodels of domain-specific languages from OWL descriptions and an approach by (Tairas et al., 2008) use ontologies in the analysis phase of DSL development, but both works require manual development of the respective ontologies.

Several *knowledge sources* from other research areas exist that can be exploited for domain modeling. The most famous resource is WordNet (Fellbaum, 1998), a lexical database for the English language, that most other knowledge-based approaches use (as we do). It contains around 82,000 nouns synsets and 100,000 noun relationships. OpenCyc (Lenat, 1995) is common sense ontology with around 230,000 classes and 300,000 relationships. ConceptNet (Speer and Havasi, 2012) is a multilingual semantic graph that contains approximately 415,000 English concepts and 900,000 relationships. Linked Open Vocabularies[5] is a dataset that stores vocabulary specifications. To the best of our knowledge BabelNet (Navigli and Ponzetto, 2012) is the largest available semantic dictionary with 3 million concepts. It is based

[3]http://www.ada.cinepoetics.fu-berlin.de/

[4]http://web.emn.fr/x-info/atlanmod/index.php?title=Zoos

[5]http://lov.okfn.org/dataset/lov/

on WordNet, integrates several other dictionaries and uses machine translation to achieve multilingualism. Our DoMoRe recommender system uses all of them to retrieve terms and domain information.

# 7 CONCLUSION AND FUTURE WORK

We presented DoMoRe, a recommender system that automatically suggests model elements for domain models. The system relies on a large-scale semantic network of related terms that features 5.7 million distinct nodes and 114 million binary and ternary weighted relationships. DoMoRe additionally integrates with several existing knowledge bases using mediator-based information retrieval of lexical information. This allows to provide context-sensitive information during domain modeling (Model Advisor) and to suggest semantically related names for model elements ordered by relevance (Semantic Autocompletion).

In our future work, we will address more modeling support scenarios (e.g., suggestion of attributes, operation names, relationship types) that require other types of information extraction. We will also investigate how lexical information can be used to detect semantic inconsistencies in domain models.

Currently, we are preparing a controlled experiment to quantitatively measure the efficiency of our recommender system in contrast to the qualitative feedback we received earlier during the practical application of DoMoRe. Participants will be introduced to a modeling tool and asked to perform several domain modeling tasks. Subjects are randomly divided into a treatment group using the tool with recommendation support and into a control group modeling without the recommender system. It is planned to measure the outcome variables time on task and model completeness.

# ACKNOWLEDGMENT

This work is partially supported by the Federal Ministry of Education and Research under grant number 01UG1632B.

# REFERENCES

Agt, H. and Kutsche, R.-D. (2013). Automated construction of a large semantic network of related terms for domain-specific modeling. In *Advanced Information Systems Engineering, 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013*, volume 7908 of *Lecture Notes in Computer Science (LNCS)*, pages 610–625. Springer.

Agt, H., Kutsche, R.-D., Natho, N., and Li, Y. (2012). The bizware research project. In *Model Driven Engineering Languages and Systems-Exhibition Track, 15th International Conference, MODELS*.

Agt-Rickauer, H., Waitelonis, J., Tietz, T., and Sack, H. (2016). Data integration for the media value chain. In *International Semantic Web Conference (Posters & Demos)*.

Almeida, M., Souza, R., and Fonseca, F. (2011). Semantics in the semantic web: a critical evaluation. *Knowledge organization*, 38(3):187–203.

Atkinson, C. and Kühne, T. (2015). In defence of deep modelling. *Information & Software Technology*, 64:36–51.

Banko, M. (2009). *Open Information Extraction for the Web*. PhD thesis, University of Washington.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Chaffin, R. and Herrmann, D. J. (1984). The similarity and diversity of semantic relations. *Memory & Cognition*, 12(2):134–141.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Colace, F., De Santo, M., Greco, L., Amato, F., Moscato, V., and Picariello, A. (2014). Terminological ontology learning and population using latent dirichlet allocation. *Journal of Visual Languages & Computing*, 25(6):818–826.

Dyck, A., Ganser, A., and Lichter, H. (2014). On designing recommenders for graphical domain modeling environments. In *Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on*, pages 291–299. IEEE.

Evans, E. (2004). *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

Fellbaum, C. (1998). *WordNet : An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

France, R. B., Bieman, J. M., Mandalaparty, S. P., Cheng, B. H., and Jensen, A. (2012). Repository for model driven development (remodd). In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1471–1472. IEEE.

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

Frank, U. (2014). Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. *Software & Systems Modeling*, 13(3):941–962.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology.

Hebig, R., Quang, T. H., Chaudron, M. R., Robles, G., and Fernandez, M. A. (2016). The quest for open source projects that use uml: mining github. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pages 173–183. ACM.

Henderson-Sellers, B., Gonzalez-Perez, C., Eriksson, O., Ågerfalk, P. J., and Walkerden, G. (2015). Software modelling languages: A wish list. In *7th IEEE/ACM International Workshop on Modeling in Software Engineering, MiSE 2015, Florence, Italy, May 16-17, 2015*, pages 72–77.

Huang, C.-r. (2010). *Ontology and the lexicon: a natural language processing perspective*. Cambridge University Press.

Hutchinson, J., Whittle, J., and Rouncefield, M. (2014). Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, 89:144–161.

Ionita, D., Wieringa, R., Bullee, J.-W., and Vasenev, A. (2015). Tangible modelling to elicit domain knowledge: an experiment and focus group. In *Conceptual Modeling*, pages 558–565. Springer.

Kuhn, A. (2010). On recommending meaningful names in source and uml. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, pages 50–51. ACM.

Kühne, T. (2016). Unifying explanatory and constructive modeling: towards removing the gulf between ontologies and conceptual models. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pages 95–102. ACM.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lucrédio, D., Fortes, R. P. d. M., and Whittle, J. (2012). Moogle: a metamodel-based model search engine. *Software & Systems Modeling*, 11(2):183–208.

Maroto Garca, N. and Alcina, A. (2009). Formal description of conceptual relationships with a view to implementing them in the ontology editor protg. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 15(2):232–257.

McCrae, J., Aguado-de Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., et al. (2012). Interchanging lexical resources on the semantic web. *Language Resources and Evaluation*, 46(4):701–719.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, T. G. B., Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. (2011). Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 331(6014):176–182.

Milajevs, D., Sadrzadeh, M., and Purver, M. (2016). Robust co-occurrence quantification for lexical distributional semantics. *ACL 2016*, page 58.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems (TOIS)*, 8(4):325–362.

Navigli, R. and Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Ojamaa, A., Haav, H.-M., and Penjam, J. (2015). Semi-automated generation of dsl meta models from formal domain ontologies. In *Model and Data Engineering*, pages 3–15. Springer.

Olivé, A. (2007). *Conceptual Modeling of Information Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Reggio, G., Leotta, M., and Ricca, F. (2014). Who knows/uses what of the uml: a personal opinion survey. In *Model-Driven Engineering Languages and Systems*, pages 149–165. Springer.

Reinhartz-Berger, I. (2010). Towards automatization of domain modeling. *Data & Knowledge Engineering*, 69(5):491–515.

Segura, Á. M., Pescador, A., de Lara, J., and Wimmer, M. (2016). An extensible meta-modelling assistant. In *Enterprise Distributed Object Computing Conference (EDOC), 2016 IEEE 20th International*, pages 1–10. IEEE.

Speer, R. and Havasi, C. (2012). Representing General Relational Knowledge in ConceptNet 5. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Storey, V. C. (1993). Understanding semantic relationships. *The VLDB Journal*, 2(4):455–488.

Störrle, H. (2010). Structuring very large domain models: experiences from industrial mdsd projects. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, pages 49–54. ACM.

Tairas, R., Mernik, M., and Gray, J. (2008). Using ontologies in the domain analysis of domain-specific languages. In *International Conference on Model Driven Engineering Languages and Systems*, pages 332–342. Springer.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188.

Walter, T., Parreiras, F. S., and Staab, S. (2014). An ontology-based framework for domain-specific modeling. *Software and Systems Modeling*, pages 1–26.

Whittle, J., Hutchinson, J., and Rouncefield, M. (2014). The state of practice in model-driven engineering. *Software, IEEE*, 31(3):79–85.

Williams, S. (2013). An analysis of pos tag patterns in ontology identifiers and labels. Technical report, Technical Report TR2013/02, Department of Computing, The Open University, UK.