# Automated Recommendation of Related Model Elements for Domain Models

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
henning.agt-rickauer@hpi.de

[2] Database Systems and Information Management Group,
Technische Universität Berlin, Berlin, Germany
ralf-detlef.kutsche@tu-berlin.de

[3] FIZ Karlsruhe & Karlsruhe Institute of Technology (KIT),
Karlsruhe, Germany
harald.sack@fiz-karlsruhe.de

**Abstract.** Domain modeling is an important activity in the early stages of software projects to achieve a common understanding of the problem area among project participants. Domain models describe concepts and relationships of respective application fields using a modeling language and domain-specific terms. Creating these models requires software engineers to have detailed domain knowledge and expertise in model-driven development. Collecting domain knowledge is a time-consuming manual process that is rarely supported in current modeling environments. In this paper, we describe an approach that supports domain modeling through formalized knowledge sources and information extraction from text. On the one hand, domain-specific terms and their relationships are automatically queried from existing knowledge bases. On the other hand, as these knowledge bases are not extensive enough, we have constructed a large network of semantically related terms from natural language data sets containing millions of one-word and multi-word terms and their quantified relationships. Both approaches are integrated into a domain model recommender system that provides context-aware suggestions of model elements for virtually every possible domain. We report on the experience of using the recommendations in various industrial and research environments.

**Keywords:** Domain Modeling, Recommender System, Semantic Network, Information Extraction, Knowledge-Based Modeling

## 1 Introduction

### 1.1 Motivation

Model-driven engineering (MDE) suggests the systematic use of models as primary development artifacts for software system building [50]. These models describe various aspects of a system at a higher level of abstraction using particular modeling languages (e. g., UML, entity relationship diagrams, or domain-specific languages). MDE aims to continuously refine models and generate source code. As a result, 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 [14, 16]. Its goal is to create models that reflect the conceptual structures of a business domain. These models include domain-specific terms and their relationships to improve the understanding of the problem area among stakeholders [6].

Domain modeling requires knowledge in model-driven software development, e.g., finding the right abstractions, creating meta-models, and the correct use of generalizations, specializations and aggregations. Assuming that software engineers have these skills, these techniques are typically applied to different areas of application and industries. Engineers must have detailed knowledge of the domain to build domain-specific models and derive corresponding refined implementation models. Building domain knowledge is a time-consuming manual process (such as talking to domain experts and reading specific documentation). Recent modeling environments (e.g., Eclipse Modeling Project, MagicDraw) provide sophisticated assistance for the correct use of modeling languages and verification of models, but support for the actual content and meaning of the model elements is very limited [43, 27].

### 1.2 Problem Statement

Domain modeling was and still is a challenging task [36, 46]. It involves gathering a lot of information that comes from different types of people, documents, and other sources of knowledge. Domain modeling is a knowledge-intensive process and requires intensive collaboration between engineers and domain experts. Automation of domain modeling was addressed by research [41], but support for this activity is still an open problem [19]. The key challenges of domain modeling and knowledge acquisition are as follows.

Solutions based on reusable domain information libraries such as Domain Engineering [42] suffer from a cold start problem. Reusable domain knowledge will only be available if enough solutions have already been developed using this methodology, while new projects already want to benefit from this domain knowledge. In the end, domain models often have to be developed from scratch [18].

Collaborations between technical stakeholders (modeling experts) and non-technical stakeholders (domain experts) require a time-consuming learning phase during a project[26]. Domain experts are often unfamiliar with modeling notations, and modeling experts usually need to develop a deeper understanding of domain concepts and terms in order to properly organize them into domain models. The greatest effort is at the modeler's desk, as it is usually more time-consuming to find, understand, and process all available domain information than to learn a few visual modeling concepts.

Domain information is contained in arbitrary sources. Structured information sources (e.g., databases, XML documents, knowledge bases, models) may be available, but unified access to all sources is in most cases not available and can only be facilitated by building additional search engines. Unfortunately, the amount of structured information sources is negligible compared to unstructured information. It is estimated that 80% of existing data is unstructured [4]. Domain information is often contained in natural language documents (e.g., textbooks, manuals, requirement specifications). Relevant facts must first be manually located and then interpreted.

Finally, the availability of large conceptual knowledge bases containing domain information is very limited. There are few handcrafted semantic databases (e.g., WordNet, ConceptNet, Wikidata) that are far from covering the diversity of possible domains. Most approaches of information extraction [9] and automatically created knowledge bases (e.g., DBpedia, YAGO) focus on factual knowledge at the instance level, which can not be used for domain modeling at the conceptual level. In addition, the core of many works (e.g., YAGO, BabelNet, DBpedia) is based on only one source of information: extraction from structured parts of Wikipedia (e.g., info boxes, categories).

### 1.3 Contributions and Outline

In this article, we present a domain modeling recommender (DoMoRe) system that contains a ready-to-use, extensive knowledge base of domain-specific terms and their relationships. DoMoRe also uses a set of existing knowledge bases to retrieve domain information, and is easily extensible with additional knowledge databases. Connected knowledge sources are automatically used during modeling to provide context-sensitive suggestions for model elements. The recommender system is integrated with a widely used modeling tool, the Ecore Diagram Editor of the Eclipse Modeling Project.

The rest of the paper is organized as follows. Section 2 introduces 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 describes in detail the implementation of the recommender system and how the delivery of contextual information and search-based suggestions works. In Section 5 we report on experiences with DoMoRe in different domains and scenarios. Related work can be found in Section 6, and Section 7 concludes the article and describes future work directions.

This paper extends [3], originally published in the proceedings of the MODELSWARD 2018 conference. In this extended version, we provide more detailed descriptions and examples of the semantic network, additional details on the recommendation generation and on the implementation of the recommender system, as well as additional related work.

[4] https://www.forbes.com/sites/forbestechcouncil/2017/06/05/the-big-unstructured-data-problem/ (Last accessed April, 2018)

## 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.

### 2.1 General Support Procedure

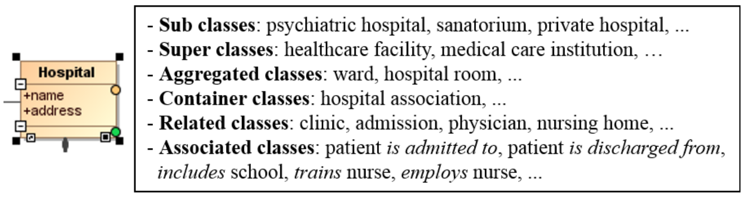
The semantic modeling support works as follows: (1) At some point of time during domain modeling a manual change in the model is made. This is usually referred to as model refinement, the activity in which a developer creates, modifies or deletes a model element (e.g., a new class). We concentrate on 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 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 [12] 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. Section 3 details both approaches. (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.

### 2.2 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 [40, 25].

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.

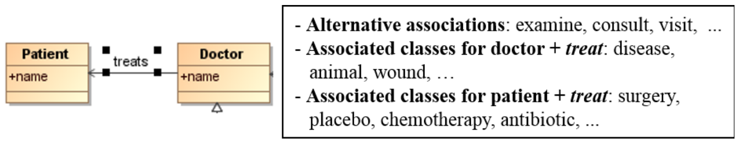
*Scenario 1 – Selection of a class.* The goal of providing contextual information is the recommendation of possible connected model elements together with their types of relations for a selected domain model element. In case a class is selected (c.f., Figure 1) possible generalizations/specializations, 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.



- **Sub classes**: psychiatric hospital, sanatorium, private hospital, ...
- **Super classes**: healthcare facility, medical care institution, …
- **Aggregated classes**: ward, hospital room, ...
- **Container classes**: hospital association, ...
- **Related classes**: clinic, admission, physician, nursing home, ...
- **Associated classes**: patient *is admitted to*, patient *is discharged from*, *includes* school, *trains* nurse, *employs* nurse, ...

**Fig. 1.** Contextual information for a selected class [3].

*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.



- **Alternative associations**: examine, consult, visit, ...
- **Associated classes for doctor + *treat***: disease, animal, wound, …
- **Associated classes for patient + *treat***: surgery, placebo, chemotherapy, antibiotic, ...

**Fig. 2.** Contextual information for a selected association [3].

*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.

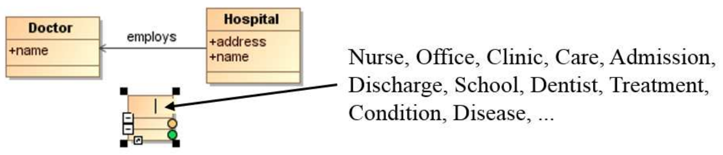

Doctor
+name
employs
Hospital
+address
+name
Nurse, Office, Clinic, Care, Admission, Discharge, School, Dentist, Treatment, Condition, Disease, ...

**Fig. 3.** Suggestions of related class names when adding a class without a connection [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 4). 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).

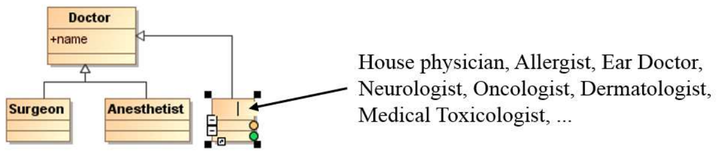

Doctor
+name
Surgeon
Anesthetist
House physician, Allergist, Ear Doctor, Neurologist, Oncologist, Dermatologist, Medical Toxicologist, ...

**Fig. 4.** Suggestions of sub class names when adding a specialization [3].

*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).

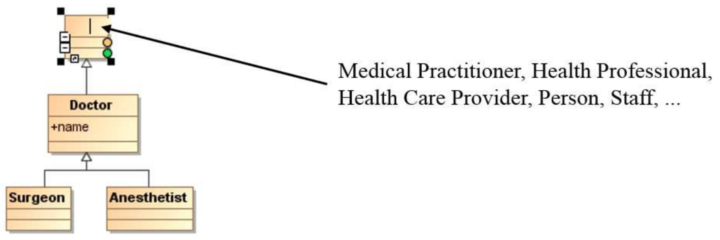

Medical Practitioner, Health Professional, Health Care Provider, Person, Staff, ...
Doctor
+name
Surgeon
Anesthetist

**Fig. 5.** Suggestions of super class names when adding a generalization [3].

*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.

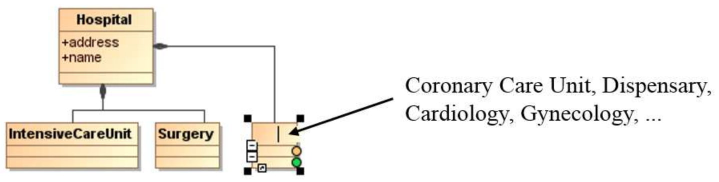


**Fig. 6.** Suggestions of aggregated class names when adding an aggregated class [3].

*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.

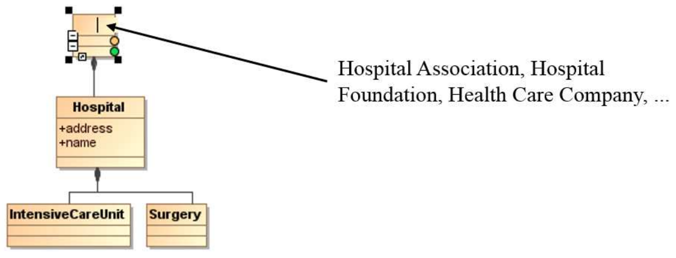

Hospital Association, Hospital Foundation, Health Care Company, ...
Hospital
+address
+name
IntensiveCareUnit
Surgery

**Fig. 7.** Suggestions of container class names when adding a container class [3].

*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.

*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.

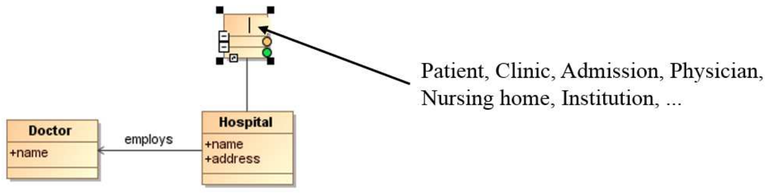

Patient, Clinic, Admission, Physician,
Nursing home, Institution, ...
Doctor
+name
employs
Hospital
+name
+address

**Fig. 8.** Suggestions of associated class names when adding an associated class [3].

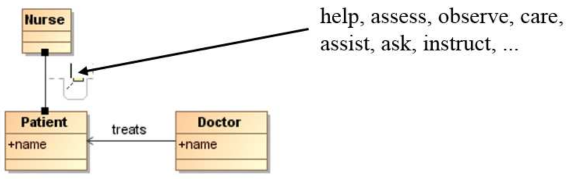


**Fig. 9.** Suggestions of association names when adding an association [3].

## 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.

### 3.1 Mediator-Based Knowledge Base Querying.

As described in the introduction, only a few knowledge bases exist that contain conceptual knowledge. WordNet [15] is the most widely used lexical database for the English language. Other important resources are BabelNet [37], a multilingual encyclopedic dictionary, and Cyc [29] and ConceptNet [44], 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. These facts describe entities on instance level, hence, most of the content cannot be used for domain modeling. Nevertheless, the schemata of these knowledge bases 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.

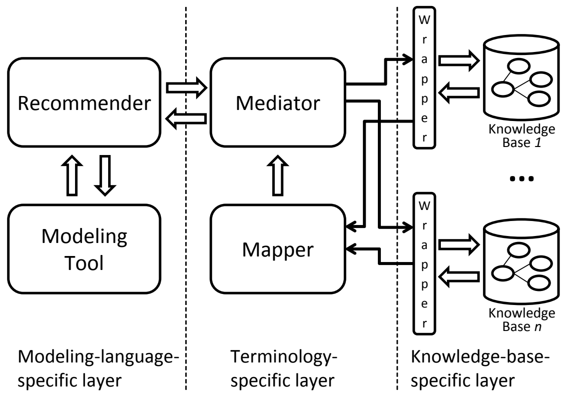

Recommender
Mediator
Modeling Tool
Mapper
Wrapper
Knowledge Base 1
...
Wrapper
Knowledge Base n
Modeling-language-specific layer
Terminology-specific layer
Knowledge-base-specific layer

**Fig. 10.** Three layer mediator-wrapper architecture to retrieve terminological information from heterogenous knowledge bases [3].

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 [51]. 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.

In the *modeling-language-specific level*, the developer uses the modeling tool and interacts with the recommender. This layer treats elements such as classes and associations, and the recommender proposes these types of elements based on the content of a model.

The mediator and the mapper are in the *terminology-specific level*. Domain-specific terms used in a model are relevant in this layer (e.g., nouns and their related terms). The mediator is responsible for translating terminology-specific content into the modeling layer and vice versa. It also manages a set of knowledge bases and their corresponding wrappers and sends queries to them as needed. 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 must handle different query languages and formats (such as OWL, RDF, SPARQL, JSON) and different types of modeling (e.g., graphs, concepts, synsets). The DoMoRe recommender system supports the **automated integration** of these types of data models without any development effort:

- Ontology schemata: concepts and relationships modeled using OWL or RDFS classes and object properties.
- SKOS-based vocabularies: terms modeled with concepts and broader, narrower, and related relationships [35].
- Lemon-based knowledge bases: a specific vocabulary for modeling lexicons of ontologies [32].

If none of these data models are present, we support **semi-automatic integration** of any knowledge base that offers a SPARQL endpoint. The effort to add a new knowledge base to the system is relatively small, it is only necessary to specify a small set of queries for taxonomic, part/whole, related and verbal relationships.

### 3.2 Extraction of Semantically Related Terms

The automated proposal of related model elements requires a comprehensive lexicon that covers almost all possible domains and their domain-specific terms. On the one hand, existing conceptual knowledge bases are valuable sources of structured information, but on the other hand they are not extensive enough to do that. For this reason, we use natural language processing techniques on a large textual dataset to identify conceptional terms and their relationships.

The approach relies on syntactic properties of sentences and statistical features of text corpora to perform a *domain-independent* extraction. Large collections of texts contain a lot of redundancy and paraphrases [8]. That is, the same facts are repeated in several documents and formulated differently. In addition, natural language has the property that certain lexical elements tend to co-occur more often than others. This implies that words with similar meanings occur in similar contexts known as the distribution hypothesis [48].

We use these features to automatically create a large database of semantically related terms. Our methods are applied to Google Books N-Gram Corpus [33], a dataset derived from 5 million books with over 500 billion words. It covers a wide range of domains because it contains scientific literature from many areas as well as fiction and non-fiction.

The dataset provides the information on how often words and phrases occurred 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 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 [52] and sum up how often concept terms co-occur in different contexts (e.g., *doctor – patient – 418,711 times*, c.f., Figure 11b). We exclude proper nouns and named entities (e.g., city names, people). With this information, we obtain related terms and their frequencies for each term and build a semantic network (c.f., Figure 11c).

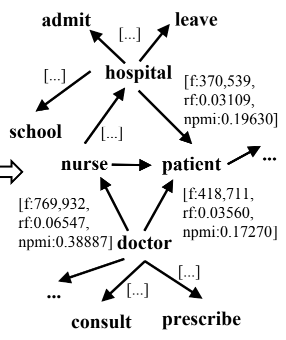
The first version of SemNet was published in [1]. In the following, we describe several improvements that have been made in comparison to the original version. SemNet contains binary **noun-noun relationships** as shown in the previous paragraph. Besides that, the semantic network now features **verbal**

| 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

**Fig. 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) [3].

**relationships**. That means, the analysis also records how often noun terms co-occurred with verbs (e.g., *doctor – consult – 173,786 times*). These relationships allow to suggest association names for domain models. SemNet now also includes **ternary relationships** for nouns and verbs (e.g., *obesity – hypertension – diabetes – 4,372 times / pregnancy – induce – hypertension – 2,365 times*). Ternary noun relationships record simultaneous occurrence of three technical terms. This information allows improved suggestions of related class names for multiple input terms. Ternary noun-verb relationships record the connection of one verb with two technical terms. These relationships are used to recommend names for classes connected with an association. Finally, for the current version of SemNet several heuristics were applied to extend the limited five word context of the Google N-gram dataset. As a result, we were able to extract terms consisting of more words and almost five times more relationships.

Table 1 shows examples of SemNet divided into the respective types of relationships. Three terms where queried: *pregnancy*, *software*, and *movie*. Each category shows the two strongest relationships of the respective query term.

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. SemNet includes 5.7 million unique one-word terms and multi-word expressions and 222 million relationships. Each relationship is quantified by the absolute frequency of co-occurrence, a calculated relative frequency, and the pointwise mutal information (PMI) measurement (see Section 4.4 for more details on this associativity value between terms). While the text analysis and extraction requires sophisticated hardware and runtime, the semantic network only needs 14 GB of storage space and can therefore be used on standard PC hardware. We provide an online web interface to browse and query SemNet: `http://www.bizware.tu-berlin.de/semnet`.

**Table 1.** Examples of automatically identified binary and ternary relationships for the terms "pregnancy" and "software". Each paragraph shows the top two relationships with the highest degrees of relatedness.

| **Noun-Noun Relationships** | | | **Noun-Verb Relationships** | | |
|---|---|---|---|---|---|
| pregnancy | lactation | | pregnancy | terminate | |
| pregnancy | childbirth | | pregnancy | occur | |
| software | hardware | | software | use | |
| software | piece | | software | install | |
| movie | television | | movie | see | |
| movie | TV | | movie | watch | |
| **Ternary Noun Relationships** | | | **Ternary Noun-Verb Relationships** | | |
| pregnancy | delivery | labor | pregnancy | carry | term |
| pregnancy | nausea | vomiting | pregnancy | feed | breast |
| software | hardware | data | software | require | hardware |
| software | hardware | system | software | allow | user |
| movie | television | radio | movie | watch | television |
| movie | magazine | book | movie | win | award |

## 4 Recommender System

In this section we describe in detail the implementation of the Domain Modeling Recommender (DoMoRe) system. We describe how domain information from a set of knowledge bases and our self-created semantic network of terms are used and transformed into recommendations of model elements according to our nine modeling support scenarios.

In essence, the task of the recommender system is as follows. For a given model element, it is necessary to determine a set of related model elements associated with a particular relationship type (e.g., all possible subclasses of the class "Doctor" or all related classes of "Doctor" and "Hospital"). In the following, we first provide a mapping of semantic relationships between different representations of knowledge. Then the architecture of the recommendation system is presented. After that we explain the features of the recommendation system and how the proposals are ranked by relevance.

### 4.1 Semantic Relationships

Domain models describe concepts and relationships of an application domain using a modeling language. Although the modeling community still discusses [22, 7] how real-world concepts can be represented correctly using modeling languages, UML class diagrams are the most commonly used modeling paradigm in the industry to do that [40, 25]. In this section, we analyze the semantic relationships of UML class diagrams from a lexical perspective and their representations in other knowledge sources (see Figure 10 for the three levels).

We reviewed literature from database research [45], linguistics [31, 10], information systems [39, 20], and semantic web research [5, 24] and relate the various types of relationships. Table 2 provides an overview, details are given below.

**Table 2.** Corresponding semantic relationship types of different modeling paradigms [3].

| **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 (WN)* |
| Aggregation (Whole) | Holonymy | PartOf (SPW), *Holonym (WN)* |
| Association (named) | Agent-Action | Object Property |
| Association (unnamed) or group of classes | Semantic Relatedness | Related Term |

*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).

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

*Aggregation* is used to specify a part-of relationship between two UML classes. We summarize both aggregation (parts can exist independently of each other) and composition (parts can not exist independently of each other) under the term aggregation. In linguistics, part-whole relationships are called meronymic relationships (meronyms are the parts and holonyms are the wholes). Part-whole relationships are not supported directly in the thesaurus definition nor in the RDF / OWL ontology specification. There is a W3C Best Practice specification "Simple Part Whole" (SPW) that includes *hasPart* and *partOf* relationships. However, there are knowledge bases that contain part-whole relationships but use a non-standard vocabulary (such as WordNet).

*Association* is the third kind of conceptual relationship that we have analyzed in terms of other representations. We distinguish two types: unnamed associations, to express a simple dependency between two domain model classes and named associations that further specify the kind of association (usually with a verb). In linguistics, named dependencies fall into the category of case relationships [10], more specifically in our case in *agent-action* relationships. To a certain extent, RDF/OWL *object properties* with domain and range constraints can be compared with named associations. In lexical semantics, the unnamed association is referred to as *semantic relatedness*, an associative relationship that describes any functional relationship between two words. The relation *related term* of thesauri is assigned to this relationship. From a lexical point of view, the unnamed association is similar to a group of classes (the diagram is the container).

In summary, taxonomic relationships in domain models can be well mapped to other structured knowledge sources such as thesauri and ontologies. Other domain model relationships are not fully represented in these resources. As a result, they are a good source for acquiring knowledge for our modeling support, but they are not enough. All domain model relationships and their inherent conceptual relationships are rooted in various linguistic theories. Thus, the combination of knowledge base queries and natural language analysis allows retrieving related domain model elements for all our modeling support scenarios (see Section 2.2).

## 4.2 Components of the Recommender System

Figure 12 shows the architecture of the DoMoRe recommender system. DoMoRe is integrated into the Eclipse environment with a number of plug-ins. The *Model Listener* monitors changes in Ecore models developed with the Ecore Diagram Editor. When a change is made to a model, the current content of the model is retrieved along with the newly added or changed model element and its relationships.

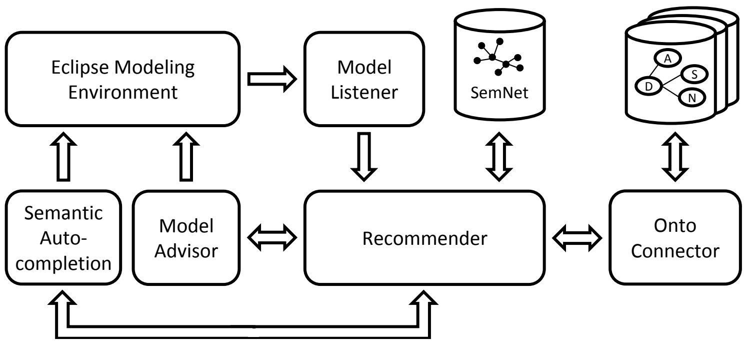


**Fig. 12.** Architecture of the DoMoRe recommender system [3].

The *Recommender* is notified and coordinates all subsequent steps of the 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 2). Based on this representation, the *Semantic Network* is queried for related terms and directly provides ranked lists with related terms. The *Ontology Connector* manages the set of linked knowledge bases and is also queried. It contains the mediator and the mapper (see Figure 10) and runs the translation of the terminological queries into knowledge-base-specific queries and the integration of results. The recommender controls two components that the user interacts with. The *Model Advisor* is a view in the Eclipse environment that displays contextual information about 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 dragging suggested elements into the diagram. The corresponding relationships are created automatically. *Semantic Autocompletion* is triggered when a new element is named in the model or the name of an existing element is changed. This function behaves like a search engine. A context-sensitive pop-up list of names for the item will be displayed and suggestions will be filtered as you type.

### 4.3 Recommendation Generation

This section provides an insight into the features of the recommender system using examples. In the following, we demonstrate the retrieval of knowledge and the generation of recommendations for Scenario 3: A few classes already exist in a diagram, and in this diagram, a new unconnected class is created. Figure 13a shows the domain model that contains two classes connected with a named association. After creating the new class, the model listener triggers the recommender, and the lexical representation of the domain model is created (see Figure 13b). The information need depends on the model refinement step. In this case, the unconnected class requires the system to retrieve nouns that are semantically related to both *Hospital* and *Doctor* (c.f., Figure 13c).

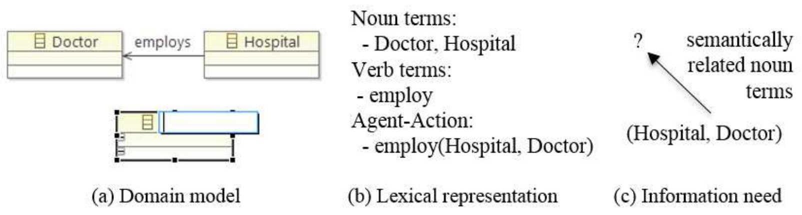


**Fig. 13.** Lexical preparation during the recommendation generation process [3].

In the following, the information need is broken down into separate lexical queries for each term and relationship type (c.f., Figure 14d). The main reason for separately retrieving the information is that there is virtually no conceptual knowledge base that contains n-ary relationships. In contrast, our semantic network directly supports ternary relationships that allow more accurate results for pairs of terms. First, ternary relationships are retrieved from SemNet. Second, separate binary relationship queries are executed for each term (c.f., Figure 14e). Each connected knowledge base is also queried for each term (see Figure 14f).

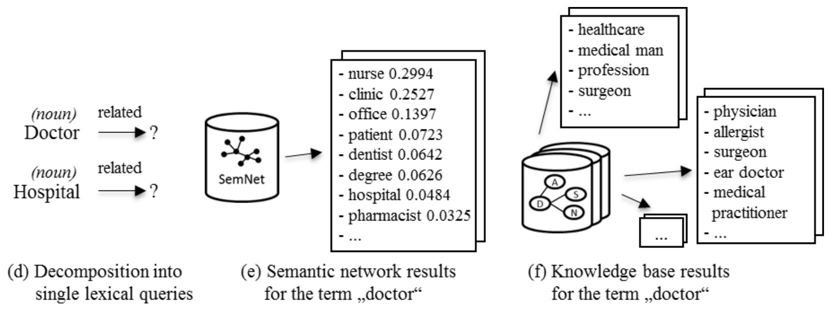


**Fig. 14.** Retrieval during the recommendation generation process [3].

Up to now, separate lists of related terms were determined for each term of the original domain model and for each relationship type and for each knowledge source[5]. Results from the knowledge bases are integrated based on the following principle. First, for each query term, it is recorded in how many knowledge bases (e.g., WordNet, BabelNet, ConceptNet) each related term occurred. Second, the distinct union of all intermediate results is generated for each query term (c.f., Figure 15g). The resulting lists have a tentative order indicating that more important terms appear first.

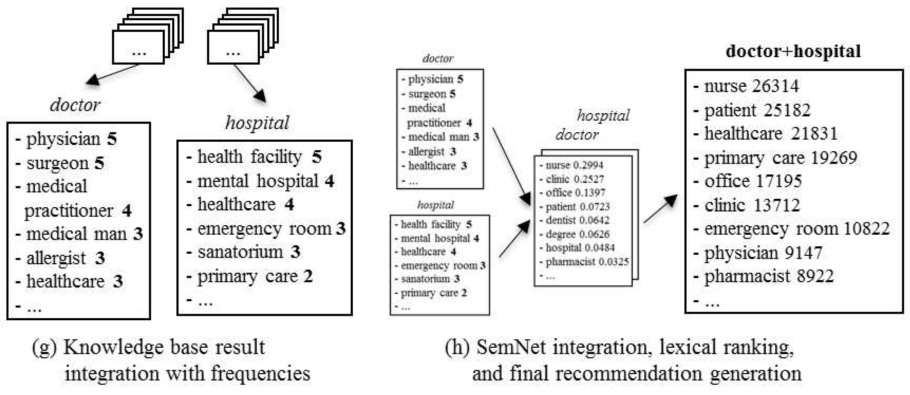


**Fig. 15.** Integration and ranking during the recommendation generation process [3].

[5] For instance, for two terms, one relationship type and five connected knowledge bases, 10 intermediate result lists are generated.

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 are joined for each term. The occurrence frequencies ensure that terms found in many knowledge bases occupy a more prominent position in the final ranking. In our example, this results in one list of related terms for *Doctor* and one for *Hospital*. Second, a final list of recommended terms is created. Separate results are intersected and the relative frequencies of common terms are multiplied. The final list is divided into $n$ segments: Terms related to $n$ query terms appear first. Next are the terms related to $n-1$ query terms, and so on. Finally, sorting by relevance is achieved by applying the pointwide mutal information score (c.f., Figure 15h). This measurement is explained in Section 4.4. For the other scenarios, preparation, retrieval, integration and ranking are similar. They differ only in the requested relationship type (for example, subclasses / narrower terms instead of related terms).

The recommendation generation process also ensures that correct suggestions are provided even for ambiguous terms. Imagine a single query for the term *table*. Without additional contextual information, the proposals contain related terms for both the furniture and tabular array meaning. If there is a second term *database* in the domain model, the result integration ensures that all terms related to furniture are ranked low.

## 4.4 Ranking

It is likely that queries to our semantic network and connected knowledge bases will yield many related terms (up to a few thousand for each request). The ranking implemented in the recommendation component is responsible for presenting the most relevant model elements first. Thus, when retrieving a list of related terms, it is ordered and the most important terms are displayed 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 (see Figure 11b). For each term in the network we compute *relative frequencies* with respect to the set of related terms. This normalization makes it possible to compare the relationship between different terms. Both measures allow for a basic ranking of terms, but they have a deficiency: very common terms (e.g., time, man, year) that occur in almost all contexts are likely to be ranked in prominent positions.

To overcome this disadvantage, we implement an information theory measurement: *Pointwise mutal information* (PMI) and its normalized form (c.f., Equations 1). It measures the dependency between the probability of coincident events and the probability of individual events (first introduced in lexicography by [11]).

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

The application of 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 that takes into account their individual corpus frequencies, so very common and general terms get lower values. Unfortunately, this measurement also has a drawback: although very general terms are ranked lower, very rare terms that co-occur with other terms only a few times tend to get high values.

Finally, to achieve a balanced ranking, our recommendation system uses the lexicographers mutual information (LMI), which is the NPMI score multiplied by the absolute co-occurrence frequency [34].

## 4.5 Eclipse Plug-ins

Two extensions for the Eclipse Ecore Diagram Editor have been implemented to allow the user to interact with the recommender system. The *Semantic Autocompletion* function of the recommender system relies directly on the ranked lists of terms that are generated based on the current content of a domain model. When the name of a class or association is edited, the user can trigger the display of a context-sensitive pop-up list of related terms that contains the most relevant terms at the top. It behaves like a search engine and provides filtering as you type (c.f., Figure 16).

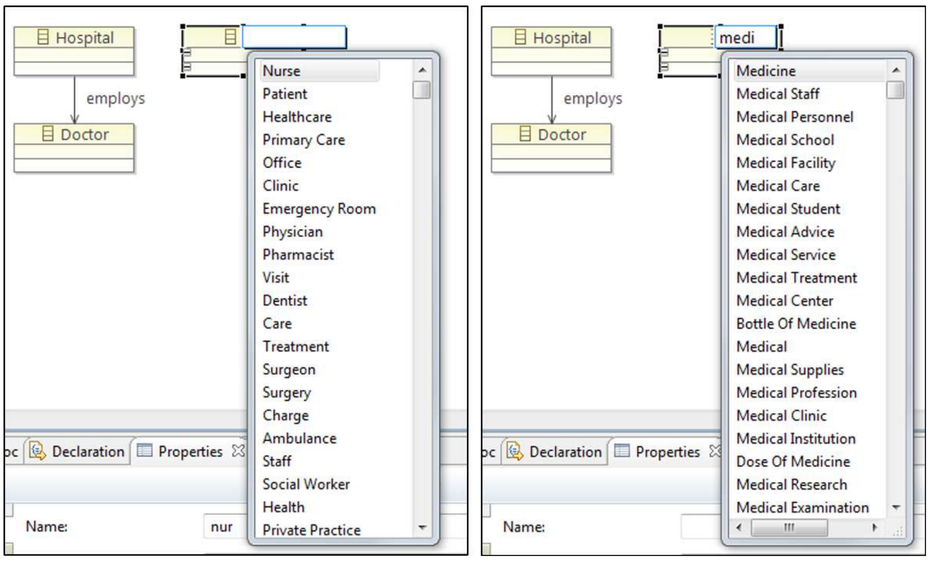


**Fig. 16.** Semantic Autocompletion of the recommender system: context-sensitive name prediction and infix search [3].

The second extension, the *Model Advisor*, is a view plug-in that displays contextual information about the currently selected model element. It queries the semantic network and knowledge bases for just one term, but with multiple relationship types. The information is aggregated and grouped into related elements, possible generalizations, specializations, aggregations, and associations (c.f., Figure 17).

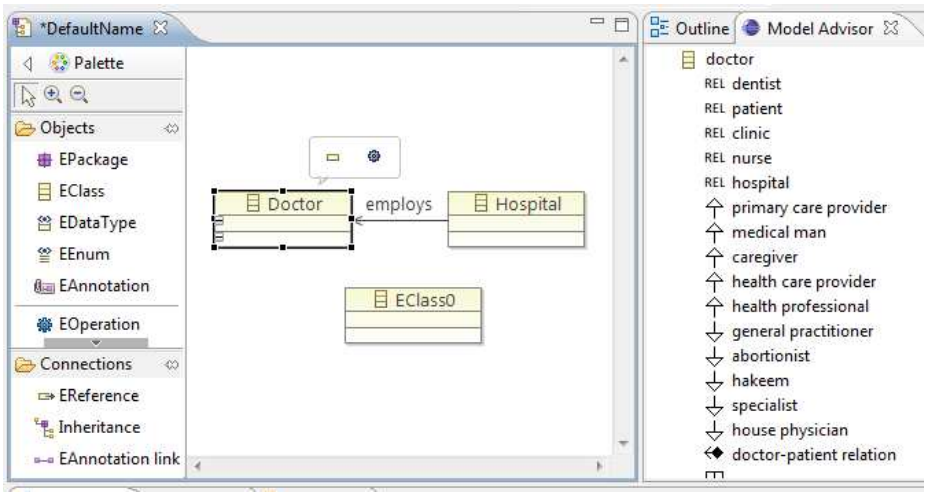


**Fig. 17.** Model Advisor of the recommender system: Suggesting possible related classes, superclasses, subclasses, and aggregations [3].

## 5 DoMoRe in Practice

In this section, we summarize the experience of using our recommender system in various settings and domains. A first prototype of the system was used in the context of the BIZWARE project [2, 1]. BIZWARE explored the potential of domain-specific languages (DSLs) and model-driven engineering for small and medium-sized enterprises in a variety of sectors including healthcare, manufacturing / production, finance / insurance, publishing and facility management. The actors in the setting were software developers from the respective companies who were working towards the introduction of DSL-based workflows to improve their development tasks. Modeling experts from research worked closely with the software engineers and provided the recommendation system. The companies planned to use DSL in customer projects, but the project found that using DSL to modernize their own software products and development infrastructures was more effective. Because the software engineers had little experience with DSLs, DoMoRe mainly supported the domain analysis phase to identify and agree on domain-specific terms that were later used in DSL's meta-models. In particular, the suggestions in the abstraction process helped to correctly distinguish between class and instance levels. The analysis of the modeling sessions showed that the ranking had to be improved (too general terms in the top positions) and that the software engineers missed suggestions for relationships between domain-specific terms (extraction of verbal relationships was not available at the time).

DoMoRe has also been used in the dwerft project [4], a collaborative research effort to apply Linked Data principles for metadata exchange across all stages of the media value chain. The project successfully integrated a number of film production tools based on the Linked Production Data Cloud, a technology platform for the film and television industry, to enable the interoperability of software in the production, distribution and archiving of audiovisual content. One of the key tasks of the project was to develop a common data model that would convey all the metadata from the different production steps (e.g., screenplay, production planning, on-set information, post-production, distribution). The actors in the environment were domain experts of the respective tasks (most of them with no technical background) and modeling experts who created the domain models and metadata schemas. Many interviews with domain experts had to be conducted to gather domain-specific knowledge and to discuss drafts of domain models. DoMoRe mainly supported post-meeting modeling and interview preparation with more extensive models that allowed for more efficient agreement on the necessary metadata. For these tasks, the recommender system was adapted to ontology schema development to use it in a Linked Data context.

Currently, DoMoRe is being used as part of the AdA project[6], an interdisciplinary research group in which film scholars collaborate with computer scientists to support empirical film studies using tool-based semantic video annotation and automated video analysis. The goal of the project is to reduce the burden of elaborate, manual annotation routines in order to accelerate the film-scientific analysis of audiovisual motion patterns at the level of larger data sets. All annotation data and analysis results are published as Linked Open Data using the project's semantic vocabularies. Collaboration on domain modeling is similar to the project mentioned above. We expect similar support by using the recommender system for the domain expert interviews. One of the main findings so far has been that the recommender system requires more complex support for concept-value relationships (e.g., Recording Playback Speed – slow motion, timelapse, freeze).

## 6 Related Work

**Modeling Assistance.** Modeling support systems provide additional information and functionality during the modeling process to help model development. They typically focus on two areas: (1) creating model libraries or similar content; and (2) developing assistance frameworks and functions using these libraries. The largest known *model repository* of UML models and meta models is the Lindholmen UML dataset [21]. It contains links to over 93,000 UML diagrams collected from GitHub repositories. A similar effort, the Gothenburg UML Repository contains over 20,000 models crawled from the Internet, images and GitHub (only a

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

collection of nearly 1,000 models is publicly searchable). Not surprisingly, most of the models are implementation models rather than domain models (for example, the search for "hospital" or "doctor" yielded 7 models, while a search for "interface" returned 90 models ). Other important resources are ReMoDD [17], MOOGLE [30], the AtlanMod Metamodel Zoos[7] (containing a total of several hundred models). EMFStore[8] and the Eclipse Model Repository[9] are tools for maintaining model repositories. There are works that propose certain *recommendation features*: SmartEMF [23] uses reasoning in Prolog for consistency checking in DSL development. Kuhn proposes a concept for recommending method names in source code and UML models [27].

The HERMES project a framework for creating model *recommendation systems* to support the reuse of software models [13]. The main objective is to provide tool support for building model libraries and providing the deployment infrastructure to create recommenders that use the contents of these libraries. The EXTREMO assistant [43] is a similar tool to facilitate meta-model development with unified model element search in a model repository. We share the same goals, but both systems are facing a cold start problem: Reusable content will only be available if enough solutions have already been developed or converted to the repository, but new projects already want to benefit from domain knowledge. To some extent, this challenge is solved by using WordWeb / WordNet, which we also use, but these databases contain about 150,000 concepts, in contrast to our semantic network of 5.7 million terms.

**Knowledge-Based Modeling.** There is a variety of work on how model-driven engineering can benefit from formalized knowledge. At the *conceptual level* there are approaches to unify ontological and software modeling paradigms [28], approaches to adopt modeling concepts from each other [22], and to extend MDE languages with ontological foundations [20].

The OntoDSL framework [49] uses *ontology technologies* at the meta-model level (such as reasoning) to help DSL users identify model-level inconsistencies. The CoCoViLa tool [38] generates metamodels of domain-specific languages from OWL descriptions and [47] use ontologies in the analysis phase of DSL development, but both works require manual development of the respective ontologies.

There are several *knowledge sources* from other research areas that can be used for domain modeling. The most popular resource is WordNet [15], a lexical database for the English language that most other knowledge-based approaches also use (as we do). It contains about 82,000 noun synsets and 100,000 noun relationships. OpenCyc [29] is a common sense ontology with around 230,000 classes and 300,000 relationships. ConceptNet [44] is a multilingual semantic graph containing approximately 415,000 English concepts and 900,000 relationships. Linked Open Vocabularies[10] is a data set that stores vocabulary specifications. To the best of our knowledge, BabelNet [37] is the largest semantic

[7] http://web.emn.fr/x-info/atlanmod/index.php?title=Zoos
[8] http://www.eclipse.org/emfstore/
[9] http://modelrepository.sourceforge.net
[10] http://lov.okfn.org/dataset/lov/

dictionary available with 3 million concepts. It is based 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 is based on a large semantic network of related terms that has 5.7 million distinct nodes and 222 million binary and ternary weighted relationships. DoMoRe also integrates multiple existing knowledge bases using mediator-based information retrieval of lexical information. This allows context-sensitive information to be provided during domain modeling (Model Advisor) and to propose semantically related names for model elements ordered by relevance (Semantic Autocompletion).

In our future work, we'll cover more modeling support scenarios (such as suggesting attributes, operation names, relationship types) that require other types of information extraction. We will also examine how lexical information can be used to detect semantic inconsistencies in domain models.

Currently we are working on a controlled experiment to quantitatively measure the efficiency of our recommender system as opposed to the qualitative feedback we received during the practical application of DoMoRe. Participants are introduced to a modeling tool and asked to perform multiple domain modeling tasks. Subjects are randomly subdivided into a treatment group using the tool with recommendation of related model elements and a control group modeling without the 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

1. Agt, H., Kutsche, R.D.: 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, *Lecture Notes in Computer Science (LNCS)*, vol. 7908, pp. 610–625. Springer (2013)
2. Agt, H., Kutsche, R.D., Natho, N., Li, Y.: The bizware research project. In: Model Driven Engineering Languages and Systems-Exhibition Track, 15th International Conference, MODELS (2012)
3. Agt-Rickauer, H., Kutsche, R., Sack, H.: Domore - A recommender system for domain modeling. In: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2018, Funchal, Madeira - Portugal, January 22-24, 2018., pp. 71–82 (2018)
4. Agt-Rickauer, H., Waitelonis, J., Tietz, T., Sack, H.: Data integration for the media value chain. In: International Semantic Web Conference (Posters & Demos) (2016)

5. Almeida, M., Souza, R., Fonseca, F.: Semantics in the semantic web: a critical evaluation. Knowledge organization **38**(3), 187–203 (2011)
6. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software & Systems Modeling **7**(3), 345–359 (2008)
7. Atkinson, C., Kühne, T.: In defence of deep modelling. Information & Software Technology **64**, 36–51 (2015). DOI 10.1016/j.infsof.2015.03.010. URL `http://dx.doi.org/10.1016/j.infsof.2015.03.010`
8. Banko, M.: Open information extraction for the web. Ph.D. thesis, University of Washington (2009)
9. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: IJCAI, vol. 7, pp. 2670–2676 (2007)
10. Chaffin, R., Herrmann, D.J.: The similarity and diversity of semantic relations. Memory & Cognition **12**(2), 134–141 (1984)
11. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational linguistics **16**(1), 22–29 (1990)
12. Colace, F., De Santo, M., Greco, L., Amato, F., Moscato, V., Picariello, A.: Terminological ontology learning and population using latent dirichlet allocation. Journal of Visual Languages & Computing **25**(6), 818–826 (2014)
13. Dyck, A., Ganser, A., Lichter, H.: On designing recommenders for graphical domain modeling environments. In: Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on, pp. 291–299. IEEE (2014)
14. Evans, E.: Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional (2004)
15. Fellbaum, C.: WordNet : An Electronic Lexical Database. The MIT Press, Cambridge, MA (1998)
16. Fowler, M.: Domain-specific languages. Pearson Education (2010)
17. France, R.B., Bieman, J.M., Mandalaparty, S.P., Cheng, B.H., Jensen, A.: Repository for model driven development (remodd). In: Software Engineering (ICSE), 2012 34th International Conference on, pp. 1471–1472. IEEE (2012)
18. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Domain Engineering, pp. 133–157. Springer (2013)
19. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Software & Systems Modeling **13**(3), 941–962 (2014)
20. Guizzardi, G.: Ontological foundations for structural conceptual models. CTIT, Centre for Telematics and Information Technology (2005)
21. Hebig, R., Quang, T.H., Chaudron, M.R., Robles, G., Fernandez, M.A.: 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, pp. 173–183. ACM (2016)
22. Henderson-Sellers, B., Gonzalez-Perez, C., Eriksson, O., Ågerfalk, P.J., Walkerden, G.: 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, pp. 72–77 (2015). DOI 10.1109/MiSE.2015.20. URL `http://dx.doi.org/10.1109/MiSE.2015.20`
23. Hessellund, A., Czarnecki, K., Wasowski, A.: Guided development with multiple domain-specific languages. Model Driven Engineering Languages and Systems pp. 46–60 (2007)
24. Huang, C.r.: Ontology and the lexicon: a natural language processing perspective. Cambridge University Press (2010)

25. Hutchinson, J., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Science of Computer Programming **89**, 144–161 (2014)
26. Ionita, D., Wieringa, R., Bullee, J.W., Vasenev, A.: Tangible modelling to elicit domain knowledge: an experiment and focus group. In: Conceptual Modeling, pp. 558–565. Springer (2015)
27. Kuhn, A.: On recommending meaningful names in source and uml. In: Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering, pp. 50–51. ACM (2010)
28. Kühne, T.: 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, pp. 95–102. ACM (2016)
29. Lenat, D.B.: Cyc: A large-scale investment in knowledge infrastructure. Communications of the ACM **38**(11), 33–38 (1995)
30. Lucrédio, D., Fortes, R.P.d.M., Whittle, J.: Moogle: a metamodel-based model search engine. Software & Systems Modeling **11**(2), 183–208 (2012)
31. Maroto Garca, N., Alcina, A.: 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 (2009). DOI http://dx.doi.org/10.1075/term.15.2.04mar. URL `http://www.jbe-platform.com/content/journals/10.1075/term.15.2.04mar`
32. 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.: Interchanging lexical resources on the semantic web. Language Resources and Evaluation **46**(4), 701–719 (2012)
33. 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., Aiden, E.L.: Quantitative Analysis of Culture Using Millions of Digitized Books. Science **331**(6014), 176–182 (2011)
34. Milajevs, D., Sadrzadeh, M., Purver, M.: Robust co-occurrence quantification for lexical distributional semantics. ACL 2016 p. 58 (2016)
35. Miles, A., Bechhofer, S.: Skos simple knowledge organization system reference. W3C recommendation **18**, W3C (2009)
36. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: Representing knowledge about information systems. ACM Transactions on Information Systems (TOIS) **8**(4), 325–362 (1990)
37. Navigli, R., Ponzetto, S.P.: Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence **193**, 217–250 (2012)
38. Ojamaa, A., Haav, H.M., Penjam, J.: Semi-automated generation of dsl meta models from formal domain ontologies. In: Model and Data Engineering, pp. 3–15. Springer (2015)
39. Olivé, A.: Conceptual Modeling of Information Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
40. Reggio, G., Leotta, M., Ricca, F.: Who knows/uses what of the uml: a personal opinion survey. In: Model-Driven Engineering Languages and Systems, pp. 149–165. Springer (2014)
41. Reinhartz-Berger, I.: Towards automatization of domain modeling. Data & Knowledge Engineering **69**(5), 491–515 (2010)

42. Reinhartz-Berger, I., Cohen, S., Bettin, J., Clark, T., Sturm, A.: Domain engineering. Springer (2013)
43. Segura, Á.M., Pescador, A., de Lara, J., Wimmer, M.: An extensible meta-modelling assistant. In: Enterprise Distributed Object Computing Conference (EDOC), 2016 IEEE 20th International, pp. 1–10. IEEE (2016)
44. Speer, R., Havasi, C.: Representing General Relational Knowledge in ConceptNet 5. In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). Istanbul, Turkey (2012)
45. Storey, V.C.: Understanding semantic relationships. The VLDB Journal **2**(4), 455–488 (1993). DOI http://dx.doi.org/10.1007/BF01263048. URL `http://portal.acm.org/citation.cfm?id=615179.615182`
46. Störrle, H.: Structuring very large domain models: experiences from industrial mdsd projects. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pp. 49–54. ACM (2010)
47. Tairas, R., Mernik, M., Gray, J.: Using ontologies in the domain analysis of domain-specific languages. In: International Conference on Model Driven Engineering Languages and Systems, pp. 332–342. Springer (2008)
48. Turney, P.D., Pantel, P.: From frequency to meaning: vector space models of semantics. J. Artif. Int. Res. **37**(1), 141–188 (2010)
49. Walter, T., Parreiras, F.S., Staab, S.: An ontology-based framework for domain-specific modeling. Software and Systems Modeling pp. 1–26 (2014)
50. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. Software, IEEE **31**(3), 79–85 (2014)
51. Wiederhold, G.: Mediators in the architecture of future information systems. Computer **25**(3), 38–49 (1992)
52. Williams, S.: An analysis of pos tag patterns in ontology identifiers and labels. Tech. rep., Technical Report TR2013/02, Department of Computing, The Open University, UK (2013)