Completeness of Knowledge in Models Extracted from Natural Text

Viktorija Gribermane[a] and Erika Nazaruka[b]
Department of Applied Computer Science, Riga Technical University, Setas Street 1, Riga, Latvia

Keywords: Natural-language Requirements, Domain Modeling, Model Extraction, Natural Language Processing.

Abstract: Requirements given in the form of text in natural language are a widely used way of defining requirements for software. Various domain modeling approaches aim to extract domain models from the given natural text with different goals and output models. The article focuses on evaluating 17 approaches for domain model extraction based on the completeness of the extracted knowledge of the resulting target models. Criteria for the evaluation have been defined and a comparison has been given, which highlights the importance of including all three - functional, behavioral and structural information, in order to retain the most complete extracted knowledge.

1 INTRODUCTION

During the requirements analysis phase of software development, the system structure and behavior usually need to be specified. These requirements are often given in the form of text in natural language (hereinafter "natural text"), which is time consuming and costly to read, maintain and use.

Various methods exist for extraction of domain models from software requirements given in natural text. These output models can be used for various goals, for example – for visualization of the defined problem domain; further generation of source code from the model; validation or assessment of the quality of the requirements used as input (Ferrari et al., 2014).

In the given article various approaches that extract functional, behavioral and structural data have been overviewed and compared based on comparison criteria defined by the authors. These criteria focus on evaluating the completeness of extracted knowledge of the target domain models according to the mentioned aspects with the goal of finding approaches that retain the most knowledge in the target model.

Section 2 introduces various approaches for model extraction from natural text, section 3 defines comparison criteria and the comparison itself with a discussion of its outcome that is given in section 4. The last section summarizes the results and future research areas.

2 APPROACHES FOR DOMAIN MODEL EXTRACTION

In this section we briefly introduce 17 different approaches for domain model extraction from natural text that show the advancements in this field of research. These approaches were selected from those of found in IEEE and ACM publication bases, since they satisfy the following requirements: natural language processing of text for constructing the domain model for software development and a publishing year from 2005 till 2020. Twelve of these approaches deal with unrestricted natural language text, the final five are suitable for controlled natural language text. All of them will be used for comparison.

In the scope of this overview, we mainly focus on the source models used by the approaches describing a domain AS-IS (today's reality) and/or a domain TO-BE (customer expected reality) as well on whenever a mapping between them is created if both are used; the form in what source models are given; what is the target model; what is used for the transition (intermediate) model and what techniques are used for knowledge extraction.

[a] https://orcid.org/0000-0002-8368-9362
[b] https://orcid.org/0000-0002-1731-989X

Gribermane, V. and Nazaruka, E.
Completeness of Knowledge in Models Extracted from Natural Text.
DOI: 10.5220/0010454301140125
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 114-125
ISBN: 978-989-758-508-1

2.1 Unrestricted Language

2.1.1 TFM4MDA

The Topological Functioning Model (TFM) for Model Driven Architecture (MDA) is an approach described in detail in (Asnina, 2006). The TFM is represented in the form of a topological space (X, Θ), where X is a finite set of functional characteristics of the system under consideration, and Θ is a topology among functional characteristics of set X (Asnina, 2006; J. Osis, 1969; J Osis & Asnina, 2011). Visually it can be represented in the form of a directed graph.

The TFM4MDA approach uses textual descriptions of functionality as a source of knowledge for constructing the domain AS-IS model and software requirements specification as a source of knowledge for constructing the domain TO-BE model. Conformity between the models is provided by continuous mapping between them. Textual description must be given in plain text form and be pre-processed to avoid incompleteness of sentences. Requirements could be declared in the form of "The system shall do...".

Text processing is manual and includes part of speech recognition and determination of dependencies between words in the sentences.

Target models are the Use Case model and specifications, conceptual class diagram, and a set of potential interfaces. The TFM is used as a transition model, but can be used also as a target model.

2.1.2 TopUML

The approach presented in (Donins, 2012) and (J. Osis & Donins, 2017) introduces a new modeling language and approach called Topological UML (TopUML). It is aimed at establishing traces between artifacts of domains AS-IS and TO-BE. The author considers that "without these traces the acceptance process of developed software gets meaningless since the customer cannot fully verify the delivered solution" (Donins, 2012).

The approach uses textual descriptions of functionality as a source of knowledge for constructing the domain AS-IS model, software requirements as a source of knowledge for constructing the domain TO-BE model and establishes a continuous mapping between the models. Similarly to the TFM4MDA, requirements descriptions must be given in plain text form and be manually pre-processed to avoid incompleteness of sentences, as well as software requirements can be given in the form of "The system shall do...". The target model is TopUML and UML models. The TFM is used as a root model for further transformations, i.e. as an intermediate model.

Text processing is manual and uses the same activities as TFM4MDA.

2.1.3 Ilieva and Ormandjieva's Approach

The approach described in (Ilieva & Ormandjieva, 2006) uses plain text descriptions of software requirements in unlimited natural language as a source of knowledge for constructing the domain TO-BE model with the aim to obtain requirements engineering models such as a Use Case Path model, a Hybrid Activity diagram model and a domain model. Tabular presentation and Semantic Networks are used as transition models. For extraction, the approach uses the feature of the Natural Language Processing (NLP) called Part of Speech (POS) recognition and semantic analysis of text.

The target model is presented as three models:

- Use Case Path (UCP) model presents the route of one action through the different actors responsible for its implementation. UCP can be transformed to Hybrid Activity Diagram;
- Hybrid Activity Diagram (HAD) model presents a use case scenario and the notation of this diagram. HAD can be transformed to UCP;
- Domain model has structural relations among concepts in Semantic Network, where "different elements are presented only once".

2.1.4 Relative Extraction Methodology

The approach presented in (Krishnan & Samuel, 2010) uses user requirements or problem statements as a source of knowledge for constructing the domain TO-BE model and UML Class diagram as the target model. A dependency graph is used as the transition model. For the extraction NLP (POS) is used with Breadth First Search (BFS) and Depth First Search (DFS) algorithms for processing the graph as well as algorithmic structures for concept, value and action identification and class diagram generation.

Their obtained class diagram lacks advanced relationships like aggregation and dependency between classes, as well as multiplicities between the classes.

2.1.5 DAA4BPM

The Description Analysis Approach for Business Process Management (DAA4BMP) presented in (Friedrich et al., 2011) uses informal textual descriptions of processes (the domain AS-IS model) for Business Process Model and Notation (BPMN) models creation as a target model. The approach uses a dependency graph as a transition model.

The approach uses the following NLP techniques for knowledge extraction: syntax parsing (factored model of Stanford Parser, action filtering by example indicators), semantic and flow analysis (FrameNet and WordNet; lists of indicators - Condition, Parallel, Exception and Sequence) as well as a custom anaphora resolution algorithm.

The four main elements of their World Model are Actor, Resource, Action, and Flow.

2.1.6 DAA4BPM v.2

The authors of DAA4BMP have modified the approach and present it in (Leopold et al., 2017). In the second version information is taken from business process models given in BPMN or Event-driven Process Chain (EPC) notations together with corresponding informal textual descriptions in plain text. This information is used as a source of knowledge for constructing the domain AS-IS model. Resource Description Framework (RDF) notation is used as the target model. The approaches use the following NLP techniques: syntax parsing (Stanford Parser) and semantic analysis (Stanford Parser, WordNet and predicates).

They identify the grammatical entities (subject, object, predicate and adverbial) and the relations between them. As the result they identify the behavioural pair-wise relations.

2.1.7 DoMoRe

Domain Modeling Recommender (DoMoRe) introduced in (Agt-Rickauer, 2020) is author's own implementation of the DoMoRe system that "generates context-sensitive modeling suggestions using SemNet Nad OntoConnector".

The approach uses a large text corpus in plain text as input and Semantic Term Network as the target model. N-Grams constructs are used as an intermediate model. It employs Stanford NLP toolkit, syntactic POS patterns for 5-Grams; and statistics of term occurrence.

As the result this approach allows identification of concepts and semantic relationships between them on a conceptual level.

2.1.8 Mirończuk's Approach

The approach described in (Mirończuk, 2020) uses fire service reports as the source model, which were given in the combination of structured data and unrestricted text. The target model was a database with structured data as the records in it.

The author employed the following techniques for the approach: classification by using supervised machine learning, creation of terms DB, manually created taxonomy, and extraction rules based on manually created patterns.

As the result the author extracts concept values from text using predefined extraction patterns.

2.1.9 AR2AA

Automated Requirements to Assertions Analyzer (AR2AA) uses plain text Requirement specifications as a source of knowledge for constructing the domain TO-BE model (Anwar et al., 2020). The target model for the approach is a triplet <Requirements, Actions, Conditions>. The approach employs the following techniques: NLP for identification of nouns, verbs and adjectives, and rules to identify actions and conditions.

2.1.10 Kashmira and Sumathipala's Approach

The approach introduced in (Kashmira & Sumathipala, 2018) is targeted at generation of Entity Relationship (ER) diagrams from requirements specification using NLP. The domain TO-BE model is Use case specifications that are given in plain text form, but must be specifically semi-structured with keywords, and ER diagram as the target model. The approach uses NLP, machine learning, ontology, and web-mining to achieve its goals.

As the result this approach identifies entities, attributes, and relationships from a text, as well as relationships between entities/sub entities-attributes, entities-entities (association), entities-sub entities (generalization), attributes-attributes, cardinalities (One to One, One to Many, Many to Many).

2.1.11 AnModeler

AnModeler is a tool for generating domain models from textual specifications (Thakur & Gupta, 2017). The domain TO-BE model is Use case specifications given in semi-structured unrestricted plain text. The target models are UML Class and Sequence diagrams. The approach employs NLP, sentence structure and transformation rules.

As the result this approach allows discovering the following domain elements: domain objects, their attributes and operations, and interactions between objects.

2.1.12 A Domain Model Extractor

The authors of the Domain Model Extractor (Arora et al., 2016) indicate that it is capable of extracting UML Class diagrams as a target model from software requirements given in unrestricted natural language form. It uses NLP features and extraction rules.

As the result this approach allows extracting concepts, associations and generalizations, cardinalities, and attributes.

2.2 Controlled Language

2.2.1 IDM

The Integrated Domain Modeling (IDM) approach explained in (Slihte, 2015) uses Use Case specifications as the domain TO-BE model. The specification must be given as plain text with steps in the form of "subject verb object" (SVO). The approach produces a TFM as the target model, employing NLP features and ontology bank for this task.

The result of text processing is a TFM. The approach allows determining such elements as functional features (action, object, actor, preconditions) and causal dependencies between them (according to a sequence of steps a text specification). However, this approach provides only manual adding of logical operators AND, OR, XOR and manual detection of cycles of functionality.

2.2.2 Nassar and Khamayseh's Approach

Research described in (Nassar & Khamayseh, 2015) targets the construction of Activity diagrams from Arabic user requirements using NLP tools. Authors believe that the UML Activity diagram is the most important diagram to be generated from user requirements. The domain TO-BE model is given as user requirements in Arabic that must follow strict writing rules (formal short statements SVO and VSO – "subject, verb, object" and "verb, subject, object"). The approach uses manually detected tag patterns to generate UML Activity diagrams as the output model.

As the result this approach allows extracting actions, domain objects without additional details and actors.

2.2.3 AGER

The authors of the research (Ghosh et al., 2018) propose an Automated E-R diagram Generation System that can generate ER diagram from plain text. User statements are given in VSO ("verb, subject, object") form. It uses E-R diagrams as the target model and employs NLP, detection of Entities, Attributes and Relations for this task.

As the result this approach allows extracting actions, domain objects and attributes and relationships between objects without cardinalities.

2.2.4 Shweta, Sanyal and Ghoshal's Approach

The approach introduced in (Shweta et al., 2018) uses Use Case specifications as a source of knowledge for constructing the domain TO-BE model. The source software requirements must be given in a semi-structured plain text form with keywords specified. The target model is UML Class diagrams. The approach uses NLP features and rules for extraction of classes, attributes, and methods.

As the result this approach allows extracting actions, domain objects and attributes and relationships between objects without additional information.

2.2.5 ReDSeeDS

Requirements Driven Software Development System (ReDSeeDS) introduced in 2010-2012 by the international researchers teams (Kalnins, 2010; Kalnins et al., 2011; Smialek & Straszak, 2012) is a project that uses domain vocabulary and use case specifications in the Requirements Specification Language (RSL) as a source of knowledge for constructing the domain TO-BE model. The source model type is Semi-formal equivalent of the domain class model with links to WordNet entries, use cases for scenarios in controlled language. The source model format is plain text with hyperlinks.

The target model is a platform independent model based on UML profile, which includes static structure as classes, components and interfaces. The target model does not represent the final version, the authors call it "draft". Draft behavior is represented as a sequence diagram.

2.3 Summarization of Findings

As it can be seen from the approach descriptions, most of them use BMPN, UML Class and Sequence diagrams as well as TFM as output. NLP is most often used for processing the system requirements. The main aim of this processing is syntactical analysis of sentences and extraction rules application to the outcome. The extraction rules are defined by the authors themselves and vary from the simplest (as noun or verb processing) to the complex (as semantical analysis of sentences). The complex processing may include also using ontologies and machine learning models. A use of simple and complex rules does not depend from the approach publishing year.

3 COMPARISON CRITERIA

Section 2 illustrates various approaches and methods for extraction of domain models from natural text. In this section criteria are defined for evaluating the domain knowledge completeness of target models extracted by these approaches.

From the target models of the overviewed approaches it can be noted that UML Class diagrams focus mostly on structural information, BPMN and UML Sequence diagrams on behavioral information, and TFM-based TopUML on structural, behavioral and functional information. We grouped criteria (Figure 1) accordingly to these three types of information. Additionally, as we speak about models and modeling, ability to determine levels of abstraction is also included as a criteria group.

The criteria are defined based on understanding of necessary conformity between problem and solution domain models, as well as, the more complete domain knowledge are extracted, the closer IT industry is to automatic creation of source code from domain models.

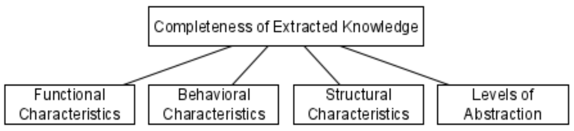
Completeness of Extracted Knowledge
Functional Characteristics
Behavioral Characteristics
Structural Characteristics
Levels of Abstraction

Figure 1: Criteria Categories for Comparison of Completeness of Extracted Target Models.

3.1 Functional Characteristics

Functional characteristics come from the system theory and are important for understanding the process of functioning of the system (where functioning is holistic representation of system's dynamic properties):

- **Signal** (stimulus, input) from the external environment, which provokes the system reaction;
- **Reaction** of the system to the external environment;
- Functioning **cycle** is the main causal cycle in the system and its structure joins functional parts of the system those of vital for its successful and long life;
- **Holistic view** (systemic) indicates whether the model represents the domain holistically (“+”) as a space of functional parts without any isolated functionality or fragmentary (“-”) as just an isolated part of the whole functional space;
- **Affiliation** to the system indicates to which system in the domain functional part element belongs to. If the approach considers only a system without the external environment then value is “-”;

3.2 Behavioral Characteristics

Behavioral characteristics are based on interaction of the system’s elements caused by a signal or signals from the external environment:

- **Causal dependencies** between behavioral characteristics link a **cause** (which generates the effect) and its **effects** (generated by the cause). The cause-and-effect relations describe causal implication between system’s functional parts and allow introducing a chronology as well as following up a process of functionality;
- **Logical dependencies** (sequential) indicate a fact that one action happened before another according to a business or logical rule. Logical dependencies not always match the causal dependencies;
- **Action** (Operation or Method) **name**, **parameter**, and **result**. An action represents an activity that may occur in the domain under certain conditions;
- **Object** is a domain entity that is **responsible** for the action execution;
- **Control flow** is an order in which actions are processed;
- **Logical operation** between control flows indicates branching, i.e. whether AND, OR, XOR relations exist between the flows;
- **Preconditions** are obstacles which satisfaction allows starting execution of the action;

- **Post-conditions** are system's characteristics that must be set after the execution of the action.

3.3 Structural Characteristics

Structural characteristics are based on relationships between domain objects or concepts:

- **Actor** is an entity that initiates an action;
- The **role of object** in relation (beneficiary, for attitude, affected, changing, disappear);
- **Object name** (Class) is a unique identifier of the object in the domain;
- **Attribute name** is a unique identifier of the object's structural property;
- **Object cardinality** is a number of objects participating (optionally or mandatory) in relations;
- **Attribute type** is an attribute classifier;
- **State of object** encompasses all properties and their current values;
- **Structural relations** consist of **aggregation** or is-part-of relation (UML aggregation or attribute as a separate class); **generalization/specification** or is-a (UML generalization); and **association** (other types of relations).

3.4 Levels of Abstraction

Levels of abstraction include separation into detailed or simplified actions, processes, systems and sub-systems:

- **Action** is an atomic activity;
- **Process** defines a part of system functionality (e.g. scenario of execution) as a set of linked actions;
- **Sub-system** is a finite part of the **system**.

4 APPROACH COMPARISON

In this section criteria defined in section 3 are used for the comparison of approaches introduced in section 2.

Extracted knowledge on functional, behavioral, structural characteristics and levels of abstraction is analyzed in the considered approaches

During comparison it is possible only to note the fact that needed knowledge is extracted. It is impossible to analyze the quality of this extraction. Therefore, the presence of such facts are denoted as "+" in tables and a lack of them as "–".

4.1 Functional Characteristics

Comparison of functional characteristics (Table 1) shows that signals from and reactions to the external environment are presented in 8 from 17 approaches. In these approaches dynamic properties of the system are taken into consideration.

Thus, elements of the use case paths (**TFM4MDA** and **Ilieva and Ormandjieva's** approach), activity diagrams (**Ilieva and Ormandjieva's** approach and **Nassar and Khamayseh's** approach), sequence diagrams (**RedSeeDS** and **AnModeler**), BPMN model

Table 1: Extracted Knowledge on Functional Characteristics ("+" is presented, "-" is not presented).

Approaches	Signals	Reaction	Cycle	Holistic view	Affiliation
2.1.1 TFM4MDA	+	+	+	+	+
2.1.2 TopUML	+	+	+	+	+
2.1.3 Ilieva and Ormandjieva	+	+	-	-	-
2.1.4 Relative Extraction Methodology	-	-	-	-	-
2.1.5 DAA4BPM	+	+	-	-	+
2.1.6 DAA4BPM v.2	-	-	-	-	-
2.1.7 DoMoRe	-	-	-	-	-
2.1.8 Mirończuk	-	-	-	-	-
2.1.9 AR2AA	-	-	-	-	-
2.1.10 Kashmira and Sumathipala	-	-	-	-	-
2.1.11 AnModeler	+	+	-	-	+
2.1.12 A Domain Model Extractor	-	-	-	-	-
2.2.1 IDM	+	+	+	+	+
2.2.2 Nassar and Khamayseh	+	+	-	-	-
2.2.3 AGER	-	-	-	-	-
2.2.4 Shweta, Sanyal and Ghoshal	-	-	-	-	-
2.2.5 ReDSeeDS	+	+	-	-	+

(**DAA4BPM**) and TFM (**IDM**) allow representing these properties.

The cycle and holistic view are used only in the approaches, which use the TFM as the target or the transition model. The possibility to define the affiliation to one or another system exists in BPMN models, Use Case models where it is possible to show different systems verbally or graphically, as well as in the TFM where it is one of the mandatory elements in the functional feature tuple. Therefore, affiliation is represented in 6 approaches, which use these models. It means that all mentioned functional characteristics are presented only in those approaches, which use the TFM as the target model or the transitional model.

4.2 Behavioral Characteristics

Comparison of behavioral characteristics illustrates discovering of causal and logical dependencies, as well as demonstrates presence of such behavioral elements as an action (its name, parameters, and result), an object which is responsible for the action, control flows and logical operations on them, and also preconditions and post-conditions (Table 2).

Causal dependencies (determined by causes and effects) are represented in those approaches, which use the TFM as the target model or the transitional model. The **DoMoRe** approach also uses causal dependencies but between the facts, not between behavioral characteristics. Logical dependencies are defined in those approaches, which use behavioral diagrams (and models), as well as RDF triplets, as the target model.

Definition of actions includes extracting their names, parameters, and results (Table 2). Most of approaches define action names, but **Mirończuk's** approach does not, because it is oriented on extraction of structural characteristics.

Parameters are defined only in 4 approaches: **Ilieva and Ormandjieva's** approach, **ReDSeeDS**, **AnModeler** and **AR2AA**. **Ilieva and Ormandjieva's** approach takes a direct object (dOb) as a parameter in their diagrams. In **ReDSeeDS** approach parameters are shown in the sequence diagram and are also transferred into class diagrams as attributes. The authors of the **AnModeler** approach detect action parameters by their defined transformation rules. In the **AR2AA** approach parameters can be taken from the ActionOutput

Table 2: Extracted Knowledge on Behavioral Characteristics ("+" is presented, "-" is not presented).

Approaches	Causal dependency		Logical dependencies	Action			Responsible object	Control flow	Logical operation	Precondition	Post-condition
	Cause	Effect		Name	Parameter	Result					
2.1.1 TFM4MDA	+	+	+	+	-	+	+	+	+	+	-
2.1.2 TopUML	+	+	+	+	-	+	+	+	+	+	+
2.1.3 Ilieva and Ormandjieva	-	-	+	+	+	-	+	+	+	+	-
2.1.4 Relative Extraction Methodology	-	-	-	+	-	-	+	-	-	-	-
2.1.5 DAA4BPM	-	-	+	+	-	+	+	+	+	+	-
2.1.6 DAA4BPM v.2	-	-	+	+	-	-	-	+	+	+	-
2.1.7 DoMoRe	-	-	-	+	-	-	-	-	-	-	-
2.1.8 Mirończuk	-	-	-	-	-	-	-	-	-	-	-
2.1.9 AR2AA	-	-	-	+	+	-	-	-	-	+	-
2.1.10 Kashmira and Sumathipala	-	-	-	+	-	-	+	-	-	-	-
2.1.11 AnModeler	-	-	+	+	+	-	+	+	+	+	+
2.1.12 A Domain Model Extractor	-	-	-	+	-	-	+	-	-	-	-
2.2.1 IDM	+	+	+	+	-	+	+	+	+	+	+
2.2.2 Nassar and Khamayseh	-	-	+	+	-	-	-	+	+	+	-
2.2.3 AGER	-	-	-	+	-	-	+	-	-	-	-
2.2.4 Shweta, Sanyal and Ghoshal	-	-	-	+	-	-	+	-	-	-	-
2.2.5 ReDSeeDS	-	-	+	+	+	+	+	+	+	+	-

descriptions. However, parameters themselves do not exist in the extracted view. Some of the approaches discover action results as separate elements: **DAA4BPM** approach illustrates results in the form of domain objects, which are sent between lanes; **ReDSeeDS** approach presents results as a type of a return value in an operation of a class in class diagrams and a return message in sequence diagrams; approaches, which use the TFM specify a result as an element in a functional feature tuple. Basically, almost all approaches that discover an action also have an object responsible for the action, excluding those of using the semantic term network (**DoMoRe**), activity diagram (**Nassar and Khamayseh** approach) and RDF triplets (**DAA4BPM v.2** and **AR2AA**).

Control flows are presented in those approaches, which model behavior of systems in the domain, i.e., which use such diagrams/models as a use case path model, activity and sequence diagrams, a TFM, a BPMN model and RDF triplets (Table 2). Logical operations that express branching can be presented between control flows.

Logical operations are used in all the approaches, which discover control flows, excluding 2 approaches, namely, **ReDSeeDS** and **DAA4BPM v.2**. The **TFM4MDA** approach and **ReDSeeDS** use only OR (conditionals), but it is not clear is it possible to discover the logical operation AND explicitly. While approaches, namely, **Ilieva and Ormandjieva's** approach, **DAA4BPM**, **Nassar and Khamayseh** and **AnModeler** approaches use OR and AND. In their turn, **DAA4BPM v.2**, **TopUML** approach and **IDM** approach use OR, AND and XOR. Preconditions are extracted in those approaches which discover OR but **AR2AA** approach which uses predicates and does not define control flows. Post-conditions are discovered in the **TopUML** approach, **IDM** approach, **AnModeler**. The **IDM** and **AnModeler** approaches extract post-conditions from Use Case specifications where they are specifically indicated. The **TopUML** approach extracts post-conditions from text if they are explicitly defined.

In most of the approaches causal dependencies are replaced by logical dependencies, i.e., mostly the latter are analyzed. Causal dependencies among functional characteristics are discovered only in those approaches which use the TFM. Behavior is determined by actions and control flows, herewith in most cases only action names and rarely when action parameters and results are extracted from text. However, objects responsible for action execution are defined almost in all cases. Regarding control flows that include branching and preconditions, they are determined when a target model is a dynamic model. Post-conditions are defined only when they are explicitly indicated in the text and in most approaches authors disregard them.

4.3 Structural Characteristics

Structural characteristics extracted by the approaches (Table 3) include entities responsible for execution of an action (actor), different properties of entities (role, name, attributes and their types, cardinality, state) and structural relations (is-part-of, is-a and other relations that can be grouped as associations).

Actors are extracted in those approaches, where they are defined in a target model or a transitional model. Actors are entities that directly initiate some action. They can be named as actors, responsible entities or defined by predicate names.

Entities (domain objects) in their mutual relationships can take certain roles. These roles are determined in **ReDSeeDS** (by default from UML, correspondingly to an object type, i.e. class) and in the **Mirończuk's** approach manually. Entities are defined in all the approaches, wherein the structural characteristics are extracted. If an entity is extracted, then it has a name, which is determined by the corresponding noun. In those approaches, which have target models such as class diagrams or structure in the form of semantic networks, attributes are also determined. The authors of the **Domain Model Extractor** approach mentioned that it is not always possible to detect a semantically correct name of the attribute from text one-to-one and a domain expert participation may be necessary. However, in the **TFM4MDA** and **IDM** approaches attributes are not extracted, because they apply a TFM, where the internal structure of the object is not defined. Though, the authors of those approaches mention that attributes can be extracted from text and specified in data vocabulary or in some other specification of domain objects. Regarding extraction of attribute types, most approaches just skip this step. In the **Relative Extraction Methodology** attribute types are not explicitly defined, but it seems that the authors of this approach add them to a target model manually. In **ReDSeeDS** approach authors take attribute types as they are mentioned in software requirement text. Generated classes are distinguished by author-defined stereotypes such as *entity*, *form*, *list*, *button*, *gridLink*, *link* and others. Object cardinalities also are rarely when extracted. The **AnModeler** approach analyzes singular and plural forms of nouns directly from text, as well as numerical values assigned to nouns. **Kashmira and Sumathipala's** approach uses a machine learning module for detecting cardinalities. A state of object after execution of the action as a separate element is presented only in the **TopUML** approach, but in other approaches the state can be represented as an attribute and requires additional analysis.

Structural relations between objects are aggregation, generalization/specialization, and other relations that can be logically grouped under associations. In all the approaches where objects are extracted, associations are also determined. The exception is the **IDM** approach in which a target model is the TFM, where structural relationships between objects are not modeled. Though, ontology can be used for this purpose. Majority of the approaches which extract associations also determine aggregation and generalization. There are two ways how they can be determined: by keywords and key phrases in the text or relations in the semantic network. **Ilieva and Ormandjieva's** and **DoMoRe** approaches use the semantic network. Other approaches (**ReDSeeDS**, **A Domain Model Extractor**, **AnModeler**, **DAA4BPM v.2**) use transformation rules with key phrases such as "contain", "is made up of", "include", "identified by", "recognized by" for aggregation and "is a", "type of", "kind of", "may be" for generalizations. Among the considered approaches, the **AGER** and **Kashmira and Sumathipala's** approaches, with the E-R diagram as a target model, determine aggregations. However, the **AGER** approach does not determine generalization. Though, **Kashmira and Sumathipala's** approach uses the machine learning module to find generalization between entities and sub-entities.

Domain objects and associations between them are extracted actually in all the approaches. In most approaches attributes are also determined, but their types mostly are ignored, as well as object cardinalities in associations. Detailed analysis of structural relations such as aggregation and generalization is less presented in the approaches.

The main path to determine these structural relations is analyzing keywords in text. Thereby, during text analysis it is necessary to pay attention to extraction of object cardinalities, attributes, attribute types and structural relations such as aggregations and generalizations. As well it is necessary to pay attention to detecting a role of an object in those cases, when objects can participate in multiple relations. The approaches which use the TFM should envisage an additional specification for detailed

Table 3: Extracted Knowledge on Structural Characteristics ("+" is presented, "-" is not presented).

Approaches	Actor	Role of object	Object name	Attribute name	Attribute type	Object cardinality	State of object	Structural relations		
								Aggregation	Generalization	Association
2.1.1 TFM4MDA	+	-	+	-	-	-	-	-	-	+
2.1.2 TopUML	+	-	+	+	-	-	+	-	-	+
2.1.3 Ilieva and Ormandjieva	+	-	+	+	-	-	-	+	+	+
2.1.4 Relative Extraction Methodology	-	-	+	+	+	-	-	-	-	+
2.1.5 DAA4BPM	+	-	-	-	-	-	-	-	-	-
2.1.6 DAA4BPM v.2	+	-	+	+	-	-	-	+	+	+
2.1.7 DoMoRe	+	-	+	+	-	-	-	+	+	+
2.1.8 Mirończuk	+	+	+	+	-	-	-	-	-	+
2.1.9 AR2AA	-	-	-	-	-	-	-	-	-	-
2.1.10 Kashmira and Sumathipala	-	-	+	+	-	+	-	+	+	+
2.1.11 AnModeler	+	-	+	+	-	-	-	+	+	+
2.1.12 A Domain Model Extractor	-	-	+	+	-	+	-	+	+	+
2.2.1 IDM	+	-	+	-	-	-	-	-	-	-
2.2.2 Nassar and Khamayseh	+	-	-	-	-	-	-	-	-	-
2.2.3 AGER	-	-	+	+	-	-	-	+	-	+
2.2.4 Shweta, Sanyal and Ghoshal	-	-	+	+	-	-	-	-	-	+
2.2.5 ReDSeeDS	+	+	+	+	+	-	-	+	-	+

description of extracted objects and their internal structures, as well as structural relations between them.

4.4 Levels of Abstraction

Table 4 illustrates levels of abstraction determined in the approaches. Most approaches, excluding **DoMoRe** and **Mirończuk**, extract knowledge at the level of actions. Those approaches which define the business processes (execution scenarios) as a result also extract knowledge at the level of processes. These processes usually are defined for one concrete system in the domain and correspondingly it is possible to assume that these approaches assign those processes to a certain system and can also indicate sub-systems. Those more complete approaches are the **TFM4MDA**, **DAA4BPM**, **TopUML**, **IDM** and **AnModeler**.

4.5 Discussion

As it can be seen from comparisons presented in this section, none of the reviewed approaches extract enough knowledge to fulfill all of the defined criteria.

Only 8 approaches consider functional properties (**TFM4MDA**, **TopUML**, **Ilieva and Ormandjieva, DAA4BPM**, **AnModeler**, **IDM**, **Nassar and Khamayseh** and **ReDSeeDS**). Most of them still consider a system by its fragments and do not pay proper attention to formal (not intuitive) determination of system's borders and separation of the system from the space of its environment. The same limitation holds also for separation of sub-systems from the system.

Almost all approaches consider behavioral and structural properties, yet most are missing the extraction of details that will be necessary in the case the target model will further be converted to source code. This means that target models need to be manually supplemented by additional data.

One of the complexities of natural text descriptions is mixing levels of abstraction. It is usual for people to change the flow of a story from general to specific and then back to general. Just few approaches take this fact into account and try to extract levels of abstraction. The benefit of this is in decreasing the possibility of duplication and overlapping of structures or behavior in target models. Most of the overviewed approaches focus on a single aspect: functional, structural, or behavioral.

Authors believe that methods that focus on multiple aspects - **TFM4MDA**, **TopUML**, **Ilieva and Ormandjieva**, **AnModeler** and **ReDSeeDS** - are more valuable as these results in more retained knowledge in the target model and more information to use for transforming the model further into source code.

To better understand which approach can be applied for a certain case, both the source and target models must be checked. Weights must be given to each criterion based on its importance for the task and a comparison can be made based on that.
Regarding NLP features used for text processing, the authors must note that in most cases pre-processed POS tagged sentences are analyzed using

Table 4: Extracted Knowledge on Levels of Abstraction ("+" is presented, "-" is not presented).

Approaches	Action	Process	System	Sub-system
2.1.1 TFM4MDA	+	+	+	+
2.1.2 TopUML	+	+	+	+
2.1.3 Ilieva and Ormandjieva	+	+	-	-
2.1.4 Relative Extraction Methodology	+	-	-	-
2.1.5 DAA4BPM	+	+	+	+
2.1.6 DAA4BPM v.2	+	-	-	-
2.1.7 DoMoRe	-	-	-	-
2.1.8 Mirończuk	-	-	-	-
2.1.9 AR2AA	+	-	-	-
2.1.10 Kashmira and Sumathipala	+	-	-	-
2.1.11 AnModeler	+	+	+	+
2.1.12 A Domain Model Extractor	+	-	-	-
2.2.1 IDM	+	-	+	+
2.2.2 Nassar and Khamayseh	+	+	-	-
2.2.3 AGER	+	-	-	-
2.2.4 Shweta, Sanyal and Ghoshal	+	-	-	-
2.2.5 ReDSeeDS	+	+	-	-

extraction rules defined by the authors of the approaches. As mentioned in section 2, they can be simple and complex. Some approaches are more advanced since use ontologies and machine learning models for sematic analysis. However, analysis if published work illustrates that it is not common. The reason may be a lack of enough number of corpuses for training those models.

It must be noted that the comparison does not consider the quality of outputs of the approaches. The authors believe that quality of the target model is as important as the completeness of the extracted model, yet this is a separate research topic and is directly influenced by the tools and techniques applied. These results can change dynamically with innovations and updates of the tools and techniques. This means that results presented by authors of the reviewed approaches can quickly become outdated and not match a similar approach if implemented today.

5 CONCLUSIONS

The research has provided a short overview of 17 approaches for Domain Model Extraction according to 4 groups of comparison criteria. The criteria are based on functional, behavioral, and structural characteristics of systems as well as levels of abstraction.

The results showed that none of the approaches is able to extract a target domain model without losing any information from source models. The provided comparison lets a researcher define importance weight for each criterion according to the expected target model and source models and select appropriate approach for its case.

Construct validity: Since various methods for extraction of domain models from text exist, the question about completeness of extracted knowledge in the models also exists. Thus, this research reviews existing results of other author works. In order to evaluate the current state in the field, 4 groups of criteria have been presented: functional, behavioral, and structural characteristics of systems and levels of abstraction. They include basic elements which are needed for generation of source code from domain models. Since texts from which knowledge are extracted are different, we do not consider application specific and platform specific details. The quality of extraction of knowledge is not analyzed; just the presence or the lack of the extraction is noted.

Internal validity: Out research is limited by publication period (15 years) and publication databases (IEEE and ACM). The search of related works has been done using a limited number of keywords: “Knowledge Extraction”, “Domain model extraction”, “Natural Language Processing”. The result of the search was filtered by relevance to the research question and the target model. Multiple publications of the same authors were overviewed and the most complete publications were taken for analysis.

External validity: The presented results allow understanding which elements are more frequently extracted and which elements need greater attention. The more complete the model is, the more complete the source code will be. This means that researchers should focus on extracting the lacking knowledge.

Further research could focus on improvements of existing approaches as well as providing other criteria that are necessary for constructing the more complete domain model. Yet some aspects were out of scope of this article, e.g., evaluating the quality of the output of these approaches, transformability of the target model to other models, validation methods and metrics used by the original authors, and a level of automation of the approaches.

ACKNOWLEDGEMENTS

This research/publication was supported by Riga Technical University's Doctoral Grant programme.

REFERENCES

Agt-Rickauer, H., 2020. Supporting Domain Modeling with Automated Knowledge Acquisition and Modeling Recommendation. PhD Thesis. Berlin : Technischen Universit¨at Berlin.

Anwar, M. W., Ahsan, I., Azam, F., Butt, W. H., & Rashid, M., 2020. A Natural Language Processing (NLP) Framework for Embedded Systems to Automatically Extract Verification Aspects from Textual Design Requirements. In *Proceedings of the 2020 12th International Conference on Computer and Automation Engineering*, pp. 7–12. https://doi.org/10.1145/3384613.3384619

Arora, C., Sabetzadeh, M., Briand, L., & Zimmer, F., 2016. Extracting domain models from natural-language requirements: Approach and Industrial Evaluation. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems - MODELS ’16*, pp. 76769

Asnina, E., 2006. Formalization of Problem Domain Modeling within Model Driven Architecture. PhD Thesis. Riga: Riga Technical University.

Donins, U., 2012. Topological Unified Modeling Language: Development and Application. PhD Thesis. Riga: Riga Technical University.

Ferrari, A., Dell'Orletta, F., Spagnolo, G. O., & Gnesi, S., 2014. Measuring and improving the completeness of natural language requirements. *Lecture Notes in Computer Science*, vol. 8396, pp. 23–38. Springer, Cham. https://doi.org/10.1007/978-3-319-05843-6_3

Friedrich, F., Mendling, J., & Puhlmann, F., 2011. Process Model Generation from Natural Language Text. In *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering (CAiSE 2011)*, pp. 482–496. https://doi.org/10.1007/978-3-642-21640-4_36

Ghosh, S., Mukherjee, P., Chakraborty, B., & Bashar, R., 2018. Automated Generation of E-R Diagram from a Given Text in Natural Language. In *Proceeding of the 2018 International Conference on Machine Learning and Data Engineering (ICMLDE)*, pp. 91–96. https://doi.org/10.1109/iCMLDE.2018.00026

Ilieva, M. G., & Ormandjieva, O., 2006. Models Derived from Automatically Analyzed Textual User Requirements. In *Proceeding of the Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, pp. 13–21. https://doi.org/10.1109/SERA.2006.51

Kalnins, A., 2010. A Model-Driven Path from Requirements to Code. *Computer Science and Information Technologies*, vol. 756, pp. 33–57.

Kalnins, A., Smialek, M., Kalnina, E., Celms, E., Nowakowski, W., & Straszak, T., 2011. Domain-Driven Reuse of Software Design Models. In Janis Osis & E. Asnina (Eds.), *Model-Driven Domain Analysis and Software Development*, pp. 177–200. IGI Global. https://doi.org/10.4018/978-1-61692-874-2.ch009

Kashmira, P. G. T. H., & Sumathipala, S., 2018. Generating Entity Relationship Diagram from Requirement Specification based on NLP. In *Proceeding of 2018 3rd International Conference on Information Technology Research (ICITR)*, pp. 1–4. https://doi.org/10.1109/ICITR.2018.8736146

Krishnan, H., & Samuel, P., 2010. Relative Extraction Methodology for class diagram generation using dependency graph. In *Proceeding of 2010 International Conference on Communication Control and Computing Technologies*, pp. 815–820. https://doi.org/10.1109/ICCCCT.2010.5670730

Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., & Reijers, H. A., 2017. Searching textual and model-based process descriptions based on a unified data format. *Software & Systems Modeling*, vol. 18(2), pp. 1179–1194. https://doi.org/10.1007/s10270-017-0649-y

Mirończuk, M. M., 2020. Information Extraction System for Transforming Unstructured Text Data in Fire Reports into Structured Forms: A Polish Case Study. *Fire Technology*, vol. 56(2), pp. 545–581. https://doi.org/10.1007/s10694-019-00891-z

Nassar, I. N., & Khamayseh, F. T., 2015. Constructing Activity Diagrams from Arabic User Requirements using Natural Language Processing Tool. In *Proceeding of 2015 6th International Conference on Information and Communication Systems (ICICS)*, pp. 50–54. https://doi.org/10.1109/IACS.2015.7103200

Osis, J., 1969. Topological Model of System Functioning (in Russian). *Automatics and Computer Science, J. of Academia of Siences*, vol. 6, pp. 44–50.

Osis, J., & Donins, U., 2017. *Topological UML modeling : an improved approach for domain modeling and software development*. Elsevier.

Osis, J, & Asnina, E., 2011. Model-Driven Domain Analysis and Software Development. In Janis Osis & E. Asnina (Eds.), *Model-Driven Domain Analysis and Software Development*. IGI Global. https://doi.org/10.4018/978-1-61692-874-2

Shweta, Sanyal, R., & Ghoshal, B., 2018. Automatic Extraction of Structural Model from Semi Structured Software Requirement Specification. In *Proceeding of 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pp. 543–558. https://doi.org/10.1109/ICIS.2018.8466406

Slihte, A., 2015. The Integrated Domain Modeling: an Approach and Toolset for Acquiring a Topological Functioning Model. PhD Thesis. Riga: Riga Technical University.

Smialek, M., & Straszak, T., 2012. Facilitating Transition from Requirements to Code with the ReDSeeDS Tool. In *Proceeding of 2012 20th IEEE International Requirements Engineering Conference (RE)*, pp. 321–322. https://doi.org/10.1109/RE.2012.6345825

Thakur, J. S., & Gupta, A., 2017. Automatic generation of analysis class diagrams from use case specifications. *ArXiv*. http://arxiv.org/abs/1708.01796