ABSTRACT – The challenge in the creation of RE/SE (Requirements Engineering/Software Engineering) models automatically from NL (Natural Language) description of requirements is to discover the knowledge model within the language model. In this article, we present a new approach for analyzing and processing these two models. Our approach combines two technologies: The first is NLP (Natural Language Processing) which we use to construct a graphical model of the language and of the knowledge it simultaneously carries within it. Once the model has been built, we can extract structural analogies with a DM (Domain Model). The second is FCA (Formal Concept Analysis), with which we process these structures. FCA helps us in two ways: as an analytical tool to formalize the concepts, and as a technology to structure and visualize them. A case study is provided to demonstrate the applicability of our approach.

KEY WORDS – Knowledge Representation, NLP, AI approach to Requirements Engineering, FCA, Domain Model

1 Introduction

Several technologies exist for the automatic building of graphical models from Textual User Requirements (TURs). The importance and benefits of such tools is evident. They save time, they serve to proofread the specifications and they facilitate communication between the client who defines the TURs and the specialist who has to implement them. In other words, these models fill the gap between the informal NL description of the problem and the formal representation of the knowledge appropriate for implementation.

There are as many approaches to solving the problem of automatically creating a graphical model from a textual description as there are projects. One way to summarize them would be to consider the existing theories from the points of view of knowledge and language. The main aim of the automatic translation of TURs into an SE model can be defined as discovering the knowledge model within the language model. However, achieving this is not easy, for two reasons: i) there is a widely used non-automatic means for creating an SE model, which is based on an intuitive understanding of the language and takes into account only the knowledge it represents; and ii) the NL model is very often inappropriate for discovering the model of the knowledge necessary to build target SE models. For instance, how can FOPL (First Order Predicate Logic) be used to generate a graphical SE model? Alternatively, how can we apply the well-known syntactical analysis tree of NL, and how could it help us?

Since the language model is different from the knowledge model necessary to create a graphic SE model, the first processing step is most often to “extract” the desired knowledge from the language description. Depending on the way this phase is realized, we separate the existing theories into two main groups:

The focus of the theories in the first group is on the knowledge itself and on modeling it. These theories consider specific knowledge for a specific target model, and their main purpose is to find this knowledge in the syntactic language structures. Most often, these theories consider controlled NL (with a limited number of syntactical structures) which represents the appropriate knowledge for the end-model. These theories are applicable to special texts and for a single target model only [4,6,7].

The focus of the theories in the second group is on the language model. Knowledge is extracted by applying language processing theories and technologies. This knowledge could be: a) previously defined and specific, appropriate for a preselected target model; or b) general – independent of any target model. The knowledge referred to in a) differs from that considered by the first group of theories in that the examined language is informal. This means that theories deal with the wider and deeper processing of NL; while the models obtained in b) need further formatting in order to acquire some of the standard SE models (e.g. UML).

The position of the knowledge “extracted” from the language (TextK) is shown in Fig. 1a. This knowledge contains some of the characteristics of the language model, staying closer to the language, and, in order to be suitable for the SE model, they often go through additional formatting, as a result of which GraphK is obtained. This second form of knowledge is closer to that of the graphic SE models, and its transformation into a standard end form, i.e. UML, is almost direct. The sequence in which knowledge is transformed corresponds to the processing phases in many of the systems. For example, in [4], there are two models between the NL and the VDM (Vienna Development Method) class representations: a contextual knowledge base (tree-like data structure) and TLG (Two Level Grammar) formalism. The authors of [6] offer two intermediate
models between the textual description and the end-model: linguistic patterns and graphical conceptual patterns. The project in [7] uses the following intermediate models of knowledge: the use-case model with defined sets of statement structures, and the graphical model of actions. The authors of [8] propose building a model similar to the case-grammar model with 12 verb classes for distinguishing 5 phrase categories (primitives). Rules for the integration of primitives are developed from which the graphical model, a cooperation-type schema, is derived. The same research group, in a different project [9], uses the SN (Semantic Network) to achieve another type of RE model, the OO-class diagram. In [5], two models were developed, the Conceptual Prototype Language (CPL) and graphical models for static and dynamic knowledge.

If we rearrange the flat form of Fig. 1a, as shown in Fig. 1b, we can visually explain the distinctive features and advantages of our approach.

Other systems consider indirect correspondence between a text and a graph going through knowledge (lines 1 and 2). Such systems, in which knowledge is separated from its source, has the following shortcomings: i) only particular types of knowledge are processed, and there is always the possibility of missing important elements; and ii) a given type of knowledge, appropriate for a certain graphical model, would require a certain type of text to contain it. Both types of system work when applied on preselected texts.

We find a direct correlation (line 3) between the source of knowledge (text) and the target model (graph), and we do so through a graphical representation of the text. By representing the text graphically, we also represent the knowledge it contains. Comparisons between the two graphical representations, source model (NL) and target model (SE), reveal analogies and show us a new way to automatically create an SE model from TURs through translation of one type of graph into another. This idea is schematically represented in Fig. 1c. The main feature of our approach is the fact that it represents the entire text graphically, and not only the different types of knowledge it carries. The advantages of our approach are the following: i) with a small number of graphical symbols, we can represent an arbitrary text. Our theory is independent of the source text and the knowledge contained in it; and ii) since we represent specific and general knowledge in a unified, graphical manner, our method is applicable for obtaining various target models.

The nature of the graphical representation of a text is described in section 2. Section 3 proposes a graphic DM based on analogies discovered between a text graph and a DM graph. In section 4, we briefly describe and apply Formal Concept Analysis (FCA) as an analytical tool to formalize concepts and a technology to structure them. In section 5, we summarize the results and propose ideas for further research. The different parts of the methodology are demonstrated with examples taken from a case study, which is presented in section 2.

### 2 Graphic representation of TURs

We claim that, if we represent a text graphically, then we also represent the knowledge it contains. Our claim is based on the following:

1) The definition of knowledge: “…explicit knowledge is exactly that kind of knowledge that can be encoded and is transmittable in language…” [1]. “Descriptive knowledge, also declarative knowledge or propositional knowledge, is the species of knowledge that is, by its very nature, expressed in declarative sentences or indicative propositions.”

2) The nature of graphical modeling languages: In the same way that the graphical models represent concepts and the relationships between them, so does the text. As is the case with graphical SE models, the concepts from the text can also be represented with an appropriate graphical symbol, and they can be connected with relations in a similar manner.

The concepts in a text are nouns, and the relations between them are the following:

- predicative – expressed with a verb;
- prepositional – expressed with a preposition;
- attributive – expressed with an adjective-noun modifier or is-a;
- structural – which can be has-a, an enumerative (key word) or a complex word (noun-noun modifier).

In [11, 12], we have briefly described and used the main relations in the graphical notation, but we have also

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Graphic presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicative</td>
<td>Team recruits members</td>
<td></td>
</tr>
<tr>
<td>prepositional</td>
<td>a group of teams</td>
<td></td>
</tr>
<tr>
<td>Attributive</td>
<td>i) Typical meet</td>
<td></td>
</tr>
<tr>
<td>Objectivet-noun</td>
<td>ii) competition is a series of events</td>
<td></td>
</tr>
<tr>
<td>complex word</td>
<td>A gymnastic scoring system</td>
<td></td>
</tr>
<tr>
<td>enumerative</td>
<td>the women’s competitions</td>
<td></td>
</tr>
</tbody>
</table>

---

*Fig. 1 A schematic summary of approaches to the automatic creation of a graphic model from TURs (a) Text, Knowledge, SE (b) Text, Knowledge (c) Text, Knowledge, SE.*
represented them in a graphical language [2]. Table 1 provides examples of main relations.

Aside from these basic relations (represented through simple sentences), there are relations between relations (compound sentences). Examples are: i) IF-THEN-ELSE, where the graphical representation consists of three predicative relations, as the relation after IF is first connected to the relation after THEN and again with the relations after ELSE. With a solid and transparent diamond, we annotate the difference in the connections (see Table 2a); and ii) a subordinating conjunction as in: Sue thinks that Bob believes that a dog is eating a bone, which would be represented as three main predicative relations connected through their predicates (see Table 2b).

To summarize complex relations, we can say that these are the cases where one whole predicative relation is an argument for another. Not all simple sentences in a compound sentence are functionally related. Those that are, however, are connected through their predicates. Examples are presented graphically in Table 2.

In order to make graph of the whole text, which is a type of semantic network (SN), we need to restructure the text into a set of basic relations. The tabular representation is very appropriate for this purpose. There are four main columns: Su (object), Pr (predicate), Ob (object), Con (conjunction), but the table could also include additional columns, depending on the application and the type of extracted knowledge. Su and Ob are noun groups, containing noun(s) and possible modifier(s), conjunction(s) and preposition(s). Pr is a verb group, containing a verb and possible adverb(s). Every compound sentence consists of more than one simple sentence, i.e. it has more than one verb, i.e. more than one triplet, Su, Pr, Ob, arranged in a separate row in the table. The positions of Su and/or Ob can remain empty during the restructuring of the compound sentence and the conversion of the passive voice into an active one, but we propose heuristics to fill them up [2,10] which can be verified and corrected in an interactive mode. Table 3 presents the restructured text, and Fig. 2a its corresponding graphical representation.

We have published the methodology for building the tabular and graphical representation of TURs in [11,12]. Here, we show only the end-result after applying the technology to an example taken from [3]. Briefly, the scheme that we follow in creating the graphical model of the text and the knowledge it carries, includes the following phases:

NL $\Rightarrow$ POS (Part of Speech tagger/parser/chunker output) $\Rightarrow$ Table Representation $\Rightarrow$ XML $\Rightarrow$ SVG $\Rightarrow$ SN (graph)

### 3 Domain Model revealing from graphically represented TURs

In carefully exploring the graphical representation of text, we notice structures which are similar to those in the block representation of the DM. By domain model, we mean a hierarchical structure of concepts extracted from the description of user requirements.

We take one part of the graphically represented text (PART from Fig. 2a) and redraw it in the block diagram shown in Fig. 2b. Once we have the graphical representation of the text and the knowledge in it on one side, and the graphical DM on the other, we immediately notice the analogies. By examining different texts and their corresponding representations, we reach the conclusion that, in order to build a DM, the following linguistic structures are important: enumerative, complex word, attributive and prepositional relations.

<table>
<thead>
<tr>
<th>#</th>
<th>Con(pre)</th>
<th>Su</th>
<th>Pr</th>
<th>Ob</th>
<th>Con(post)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A gymnastics scoring system will be developed SYSTEM to automate the definition, registration, scoring, and record-keeping of a gymnastics season.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A gymnastics league is a group of teams that compete against each other.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Each team recruits members</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A typical meet consists of several contests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SEVERAL CONTESTS held in the course of one day.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>For example</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>When a team enters a meet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>each team enters the same number of members who</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Table 2, Complex relations a) IF-THEN-ELSE When a team enter, it enters all the competitions. b) Subordinating Sue thinks that Bob believes that a dog is eating a bone.
MEMBERS must compete in all parts of the competition. Each competition is a series of events. EVENTS run on different equipment. For example, the women’s competitions involve balance beam, vault, high bar, floor exercise. All pieces of equipment are in operation at the same time; each team’s competing gymnasts perform on one piece of equipment and then rotate to the next.

Fig. 2 Revealing the Semantic Network and Domain Model analogy

We are not particularly selective with regard to the text. All texts are of interest to us, as long as they are considered by different theories. There are texts, however, where the graphical representation very clearly shows structural similarity with the DM. In Fig. 3, for instance, we have shown examples (from [2] and [12]) in which, with only a slight rearrangement of the graphical elements, a direct correspondence between the SN and the DM is obvious.

Fig. 3 Structural analogies between the SN and the corresponding DM

Having defined which language elements are important for building a DM during the TUR analysis phase, we will now seek a way to implement this idea. FCA would be suitable in two ways: i) to formally represent concepts and the relations between them. This makes us work within tight formal limits, which in turn deepens the level of the formalization; ii) to take advantages of FCA application: • it is a relational model which describes structural relationships between concepts; • it is a precise algorithm which builds a structured hierarchy; • it enables visualization of the ordered structures.

4 FCA technology

Brief description:
The methodology of FCA is based on the formal understanding of a concept as a unit composed of two parts: i) extension, uniting all objects which belong to the concept; and ii) intention, uniting all attributes of the objects integrated into the concept. (1)

Using the language of mathematics, this understanding can be written in the following manner: If \( O \) represents all objects, \( A \) all attributes and \( P \) the relations between them, i.e. \( P \subseteq O \times A \), we arrive at the mathematical formula of context, \( C = (O, A, P) \), which is read as follows: formal
context is a triple: a set of Objects, a set of Attributes and a relation \( R \) between them.

Definitions (1) and (2) could be represented in a third manner – a table. The table, which is similar to the context, creates a binary relation: the rows contain the objects and the columns contain the attributes. The relation between an object and an attribute is represented by a check mark in the corresponding row and column. This is demonstrated in Fig. 4.

\[
\begin{array}{c|cccc}
\text{Object} & A1 & A2 & A3 & A4 \\
\hline
O1 & x & x & & \\
O2 & x & x & x & \\
O3 & & & & \\
O4 & x & & & \\
O5 & x & & & \\
\end{array}
\]

Fig. 4 Context represented as a table and a duality of relations

The representation of the table in two ways, as shown in Fig. 4b, demonstrates the duality of relations, meaning that they can be read either as: I. The Object \( O1 \) has attributes \((A1, A2)\), etc., or II. Attribute \( A1 \) belongs to the objects \((O1, O2, \ldots)\), etc.

To order the objects into a hierarchy according to their corresponding attributes, we will need a set of attributes which belong to equal sets of objects. From Fig. 4b, in cases I.1 and I.2, a pair of sets is \( <\{A1, A2\}, \{O1, O2\}> \) (its field is shaded in Fig. 4a). This pair of sets is called a formal concept, written as \((X,Y)\).

The rule for ordering a formal concept in a hierarchy is:

\[
(X1,Y1) \leq (X2,Y2) \iff X1 \subseteq X2 \land Y2 \subseteq Y1,
\]

where \((X1,Y1)\) is a subconcept and \((X2,Y2)\) is a superconcept.

The hierarchical order of the concepts is in the form of a lattice, which, in our context from Fig. 4a, looks as shown in Fig. 5.

Now, let us take a look at the application aspects of FCA, i.e. FCA technology. It is made up of two main processing phases, the order and grouping of the various steps of which are shown in the following scheme.

Fig. 6 Contents of FCA technology

The second part (executive) is more explicit and easily applicable, as the objects and attributes are written in the form of relations, from which, in an algorithmic way, realized and existing as readily available software, a hierarchical order in a lattice is obtained. Having in mind the end-result of the FCA technology, an ordered hierarchy, it is natural for this technology to be used in the following classes of applications:

- classification of words with related semantics based on context: (words) appearing in (special surroundings in the phrases) [17];
- classification of documents with context: (key words or phrases) which are parts of (documents) [15];
- object-oriented class hierarchy with context: (class) has-a (operational method) [18];
- modular hierarchy of programs using context: (global variables) used in (procedure) [19]; etc.

The more implicit part is an analytical part of the technology, in which objects and attributes, and the relations between them, must be discovered. These are very often not explicitly given, and must be revealed through serious analytical work. Other technologies could be used in the analysis as well, but the goal in the end is to discover the concepts and order them into a lattice.

An example of combinations of processing, first NLP and then the concept lattice, is the project RECOCASE [13]. Before reaching a final concept lattice model, it passes through disambiguation of the text, Link Grammar parsing and semantic representation of each sentence in Flat Logical Form. The relational table of the FCA model contains, as objects, the sentences from the use case scenario specification, and the attributes are the various phrases in these sentences. Control NL is processed, and the end-model of this system is a use-case model represented with a concept lattice.

Analytical part: FCA as a means for formalization of concepts and the relations between them

The use of FCA obliges us to present a relation between concepts and attributes. Our concepts are nouns and adjectives in selected phrases of the text; the attributes are the groups (real nouns and adjectives from the text, or abstract groups) to which these concepts belong, and the relation is an affiliation of concepts into groups. To summarize, according to the terminology of FCA, our context is: (concepts) affiliated with (groups) [Fig. 7a].

This might seem an easy task, but in practice it is not. We consider unlimited NL, in which the concepts are not simple nouns (consisting of one word), but compound words. If a simple word is met in different phrases and is a part of different compound words, what would its place be in a final, summarized group? In order to solve this problem, we adopt the nested format of concepts to formally represent the chosen linguistic structure. We also define a merge operation between the nested structures. A description of this algorithm follows.

Step 1: Find and represent the main and subordinate concepts.

- **Adjective-noun attachment**: We accept that the main concept is a noun, and the adjective is its instance. For example, senior competition means that competition has an instance, which is senior. Adjectives with a common purpose, like several, different, typical, etc., can be ignored.
- **Noun-noun attachment**: We assume that the different words in a compound noun are encountered in an order which determines their seniority – the noun
Before we define the rules, however, we introduce the common structure which contains the substructures. The aim of merging two structures is to achieve a terms that we will use: structure, head of a structure, body of a structure, terminal concept and nonterminal concept.

**Prepositional attachment**: prepositions, which express place and possession, are important for the DM; for example, in, for, of, etc. If we represent a prepositional phrase as “Su preposition Ob”, we accept that the object (Ob) is the main concept, and it contains the subject (Su). For example, in “parts of the competition”, the main concept is competition, which contains parts. Another example would be “driver in vehicle” – where vehicle is the main concept, which contains driver.

**Key-word attachment**: we summarize this type of phrase with the following structure: Su key word Ob. Key word could be: has_a, and similar words which express structures (“composition”, according to the terminology of UML), for example: consist of, involve, type of, part of, etc. We accept that the subject (Su) of the key word is the main concept in the relation and contains the object (Ob). For example, in “A typical meet consists of several contests”, the main concept is “A typical meet” (Su of the phrase), and it contains “several contests” (Ob of the phrase).

Main and attached concepts are written in a nested format in the following intuitive way: i) enumerative structure: mainConcept (subConcept1, subConcept2, …); ii) for the other structures: mainConcept (subConcept1(subConcept2(…))). The following table contains all the phrases important for the DM extracted from the text, and their corresponding nested representation.

<table>
<thead>
<tr>
<th>Table 4. Concepts, their textual and nested representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Definition, registration, scoring and record keeping of gymnastics season</td>
</tr>
<tr>
<td>2. A gymnastics scoring system</td>
</tr>
<tr>
<td>3. A gymnastics league is a group of teams</td>
</tr>
<tr>
<td>4. Each team's competing gymnast</td>
</tr>
<tr>
<td>5. A typical meet consists of several contests; a Meet (typical_competition(event,Gr,team,vehicle) Group(women's(all-round) men's_all-round))</td>
</tr>
<tr>
<td>6. men's_all-round, women's_all-round</td>
</tr>
<tr>
<td>7. Junior and senior</td>
</tr>
<tr>
<td>8. in all parts of the competition</td>
</tr>
<tr>
<td>9. Each competition's series of events</td>
</tr>
<tr>
<td>10. The women's competitions involve balance beam, vault, high bar, and floor exercise</td>
</tr>
<tr>
<td>11. Operation (equipment (pieces))</td>
</tr>
<tr>
<td>12. All piece of equipment are in operation</td>
</tr>
<tr>
<td>13. The first piece of equipment</td>
</tr>
</tbody>
</table>

**Step 2: Rules for merging compound concepts**:
The aim of merging two structures is to achieve a common structure which contains the substructures. Before we define the rules, however, we introduce the terms that we will use: structure, head of a structure, body of a structure, terminal concept and nonterminal concept.

Merging of the concept structures is based on their matching.

**Rule 1**: Two structures are matched when their heads are equal.

**Rule 2**: The matched structures are merged and the result is a single structure with a common head and a body containing the joined bodies of the two structures.

**Rule 3**: Joining the structures’ bodies together:

1) body1, which consists of terminal(s) and/or nonterminal(s), is joined to body2, which consists of nonterminal(s) in the following way: both are concatenated and separated by a comma. The same holds for the reverse case, or when body2 is added to body1.

2) If body1 consists of terminal(s) and is added to body2, which also consists of terminal(s), two cases can arise: a) if the added terminals are of the same type as the ones they are being added to, then the two groups are concatenated and separated by a comma; b) if the terminals being added and being added to are of a different type, they have to be grouped before being joined.

In Rule 3-2, we apply a different understanding for concepts that originate from: i) nouns and adjectives; or ii) enumerative and nonenumerative concepts. They are grouped with the purpose of keeping concepts of a similar nature in the same group. For example, the nonterminals [senior, junior] are added to [parts, events, women’s] immediately after they have been grouped: Gr(senior, junior). The name assigned to the group is “Gr” and it is indexed for uniqueness. The following are examples of merging nested structures:

<table>
<thead>
<tr>
<th>X(Y)X(Z)X(YZ)X(YAZB)X(YAB)X(YB)X(YBD)X(YBD)X(YB)X(YB)</th>
</tr>
</thead>
</table>

In the example from our case study: the 5th and 6th structures from Table 4 above are merged, as shown:

<table>
<thead>
<tr>
<th>competition(parts)</th>
<th>competition(parts, events(series))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meet</td>
<td>competition(women’s(Gr1(all-round, individual), Gr2(balance beam, vault, high bar, floor exercise), men’s_all-round, Gr3(junior, senior), parts, event(series))))</td>
</tr>
<tr>
<td>Operation</td>
<td>equipment(pieces)</td>
</tr>
<tr>
<td>Gymnastics</td>
<td>season(definition, registration, scoring, record-keeping), scoring_system, league(team(group, gymnast(competing))))</td>
</tr>
</tbody>
</table>

As a result of the application of the rules for merging and grouping the structures from Table 4, we obtain the following three main complex structures:

**Executive part of FCA: Technology for structuring and visualization**

The relational table (or context in the FCA discourse) contains the relations between the CONCEPTS and the GROUPS in which they participate (see Fig. 7a). The relation is defined as: concept c participates in group G.

In the columns of the tables, the groups are inscribed, and in the rows the concepts are inscribed.
The presence of a concept in a group is indicated by filling in the intersection field between the row of the concept and the column of the group in which it participates. It is important not to forget to add all the subgroups of a concept when it is added to a supergroup. More formally, the algorithm for the addition of a concept to a supergroup can be presented through a recursion in the following way:

\[
\text{Process the subgroups of the concept } \mathcal{C}\text{ in a super-group } \mathcal{G} \text{ (attribute column).}
\]
\[
\begin{cases}
\text{Go in group } \mathcal{C} \text{ (attribute column) of the object } \mathcal{E}.
\text{For every element } x \in \mathcal{C},
\quad \text{Inscribe the element } x \text{ in super-group } \mathcal{S}.
\text{Process the subgroups of the concept } x.
\end{cases}
\]

5 Conclusion

Summary: In order to build the graphical RE/SE model from the NL description automatically, we have developed a technology based on the following processing phases:

1. Formal representation of NL with the relation as a basic building block;
2. Graphic representation of relations;
3. Obtaining analogies from two graphical models – a source model of NL and a target RE/SE model;
4. Appropriate translation of analogical forms (from source to target).

Following these phases we have thus far applied our methodology to the automatic building of the following models: the OO-class diagram [11], the Hybrid Activity Diagram [12] and the Use Case Paths model [10]. In this article, we deal with building the DM. Various technologies could be used for each stage of the implementation, but we have chosen to proceed as follows: Stages 1 and 2 involve automated language processing. We have implemented these stages using NLP tools (POS tagging [21], parsing, chunking [14]) and software technologies for structural representation and visualization (XML, SVG). Stage 3 is not automated, rather it is based on visual analysis, and it requires good knowledge of the target graphical models. Having discovered the analogies, we can create algorithms for translation of the source language structure into a target model structure, and the task of SN (Semantic Network) can be considered completed. Stage 4 provides an algorithm for translating one type of graphical element (SN) into another (UML model). The algorithms are different for each unique target graphical model. Here, for the DM, we use the options readily available from FCA.

While the input data for FCA constitute context (in general, context means “concepts have attributes”, and in our case “concepts participate in groups”), we had to transform our graph elements into flat-nested form and regroup them in order to obtain appropriate input for the concept lattice.

Lesson learned: Using different technologies for the same class of problems is valuable, because it provides for different points of view and helps in the achievement of a higher level of understanding and formalization of the problems at hand. How FCA helped us is outlined below:

It requires formalization of the problems in a relational and hierarchical manner. The world is relational, and so
FCA, which has a built-in relational mechanism, is very useful for representing types of relations. FCA does not, however, offer us a methodology for discovering relations, objects and attributes, in other words, the context. FCA does not tell us which context is the most useful for solving a problem. This is why we divide the FCA technology into two parts (see Fig. 6): the analytical part, the main goal of which is to discover the context; and the executive part, the goal of which is to represent the context. Various methodologies could be used in the analytical part – practically the entire science, as well as intuition and talent. In the second or executive part, a precisely defined representation based on a precise algorithm is used. Hidden in the link between the independent, at-first-sight activities – discovery and representation – is the use of FCA, and the effect of its application. There are two approaches to this: i) the results of the discovery activity are available and the goal is to represent them; or ii) the concept lattice representation is the goal and a discovery is made. In our research (Natural Language → UML model), we made use of both advantages of the FCA technology. With the lattice-representation of the FCA technology in mind and working on discovering the context, we achieved the following:
i) We formalized the static structures of NL (flat-nested structure). FCA inspired us and forced us to keep track of detail, while at the same time maintaining clarity and relativity: once within the framework of the language model, and then within the framework of the domain model;
ii) By using FCA, we obtained, in fact, two new models:
a) a model of the language – linguistic structure / knowledge structure; and b) a model of the problem – DM. The concept lattice helped us to reveal these two models and to connect them.

Further work: Language is structured, and it also carries within it knowledge for structured relations. Using FCA to represent these relations brings positive outcomes. Our intention is to continue the research in two directions: to refine the formalization of NL and to apply FCA to knowledge modeling. We propose to do this in the following ways:
- Apply the knowledge included in the DM to compare documents on a deeper semantic level and mine knowledge from the text;
- Devise rules for making connections between the structures on a horizontal level;
- Develop a supporting tool;
- Use the “weights of nodes” method in the concept lattice to incorporate more knowledge into the relational model.

6 References