ABSTRACT
This article describes graphical language for the representation of textual user requirements. We need such a language in order to approach the automatic construction of graphical software engineering (SE) models (in particular UML diagrams) as a translation from one graphical language into another. In order to fulfill this goal, we have included universal notation in graphical language, which is equally valid for the presentation of linguistic knowledge and the problem domain knowledge. Using unified graphical presentation of natural language and knowledge, we are literally able to draw text as a picture. This process is based on a deep analysis and a clear representation of the text, using a limited set of meaningful graphical symbols. As a result, we achieve a methodology for mining a huge volume of knowledge about relations and structures between entities: words in the text and objects/concepts in the problem domain. The extracted relations generalize knowledge and can be formulated in a presentation appropriate for different purposes. In our case - many target SE models.

KEY WORDS
Modelling, Knowledge systems, NLP, UML, Semantic Network

1. INTRODUCTION
The graphical language (GL) notation came up in our attempt to automatically translate textual user requirements into Unified Modeling Language (UML) diagram [14]. In order to resolve this issue, we face two tasks: the first is to understand the problem in the description and the second is to solve it using established or novel methods. The first task – “understanding the problem” – can be divided in two subtasks: understanding the language and understanding the knowledge expressed (presented) through it. If we juxtapose presentation and understanding we will obtain two types of presentation – presentation of the language, and presentation of the knowledge it carries. Although the linguistic knowledge serves as a basis for obtaining knowledge related to the problem, very often these two types of knowledge are presented differently and distinctly: linguistic knowledge with the corresponding presentation, and problem knowledge with the corresponding presentation. The gap between these two types of presentations is overcome by restricting Natural Language (NL) in such a manner that a limited volume of specific knowledge is arranged in a predefined structure. That is time consuming, leads to loss of information and limits the applicability of the SE model. The question arises: would it not be easier to consider the language, and the knowledge it carries, through a single graphic presentation? Then, the automatic construction of UML diagrams would change from translation of one graphic language into another. It is precisely this idea that is developed in the present article. To begin with, section 2 reviews related works. In section 3 a particular graphic notion is described, which presents the language and the knowledge it carries simultaneously. Text rewritten through GL notion is Semantic Network (SN), and in section 4 there is an example of that. In section 5 we compare our semantic network with the semantic networks summarized in [12], and we indicate future tasks as well.

2. RELATED WORK
Fig.1 presents the scheme which summarizes “understanding and representation” of textual knowledge. It consists of three different parts based on knowledge and processing procedures. First, we are concerned with Domain Knowledge (DK) in the text, but its automatic extraction cannot be effected without considering Linguistic Knowledge (LK). At this stage some systems offer processing for simplifying text and reducing ambiguity. Second, the central part gathers processings which make a model of NL and the knowledge carried in it. These are: syntax (Sy), logical form (LF) and Intermediate Model (IM). Third part of the scheme includes processings that transform the language model and the knowledge (general and domain) it contains into an application specific model. According to this scheme we can categorize the published systems into two types: The first type of systems follows the upper road of the schema in Fig.1. They prefer to analyse controlled NL because domain knowledge inside a form can be extracted easily and reliably. At each step of the processing they consider domain knowledge and obtain specific final models for specific domain knowledge. In [5] for example, the controlled text consists of knowledge about the types of data, operations over them and relations between them. For the presentation of this knowledge an intermediary graphic form (IM) is chosen which looks like tree data structure, with three types of nodes: data, function and context. This graphical model is translated into an end OO model, presented within VDM (Vienna Development Method) notation.

Fig1. Phases of automatic NL Requirements analyzing
In [8] the UCDA (Use Case driven Development Assistant) is presented, which uses as knowledge intermediate presentation (IM) – graph with four types of behaviour: request, validation, change, and response. This type of intermediary presentation fits well into the end graphic UML OO diagram.

In ProCasor project [6], the IM consists of three types of activity (emitted, absorbed and internal) and three types of connections (sequencing, alternative, repetition). The final presentation is a type of activity diagram.

In [7] linguistic patterns are extracted, which are subsequently presented into IM - conceptual patterns for an object model. The IM consists of nine graphic conceptual patterns from which is built a diagram of classes and the relations between them.

Among the second type of systems are those that attempt to make models more universal and independent from DK and the text. These systems follow the lower road of the scheme. They process unrestricted NL from which are obtained two consequative models: the first model obtained is LF, which adheres more closely to the text and the general knowledge (GK); the second model is IM which adheres more closely to SE model and DK.

The KCPM (Klagensfurt Conceptual Predesign Model) described in [3,4], in the part for the LF presentation contains 12 distinct verb classes, from which are extracted 5 phrase categories. In the part for the IM presentation, these 5 phrase categories assign 3 types of primitives: condition, operation, and cooperation. From the graphic elements which presents these primitives a cooperation type scheme is built, which later is mapped into the UML activity diagram. KCPM is a basic model of the system NIBA [2]. However, in order to make an OO class diagram, NIBA transfers LF into another IM, namely Semantic Network. This graphic consists of things types and connection types. The translation from SN into OO class diagram is not entirely automatic – the designer chooses potential candidates for the conceptual scheme.

Colour-X [1] builds LF as conceptual prototype language which replaces “three central representations, i.e. (structured) natural language sentences, lexicon knowledge and conceptual models”. This system doesn’t aim to obtain UML or a similar final model. The philosophy of the Colour-X project is to provide the user with a static view and a dynamic view of general knowledge.

Our approach differs from the two approaches described above. We resemble the systems from the second type, obtaining more general and independent information from text and DK presentation. We differ, however, by creating a common model of linguistic and domain knowledge in order to achieve one general graphical IM which aims to reach different target UML diagrams. Colour-X remains on stage LF; it has a unique presentation of different knowledge extracted from the text, which needs to be additionally processed to obtain target SE model. KCPM and NIBA use different IM for different target models. Our approach aims to be more universal. It analyses uncontrolled NL by building a common presentation of LF and IM, which we use for building various final models. The following section describes the notation of our graphic representation.

3. GRAPHICAL LANGUAGE NOTATION

Our graphical language is based on tabular presentation of sentences, built after syntax analysis. We divide the sentence into three basic groups (columns in the table) according to the function each one performs: Su bject), Pr edicate) and Ob ject). The subject and the object are noun groups; they can be simple (nouns and attributes) or complex (consisting of simple nouns connected with operators – prepositional, conjunctuional). The predicate is a verb group consisting of a main verb, and its corresponding adverb, modalty, infinitive, and auxiliary verbs. These three basic types form a sentence. Several sentences can be connected with conjunctions or relative pronouns in order to produce a complex/compound sentence. It is vital that each main and subordinate sentence contains this triplet (Su, Pr, Ob). Some of the positions in it can be left unfilled for various reasons: i) syntactic peculiarity (inverse phrase, relative phrase following the predicate), ii) understanding from the context or iii) use of the passive verb form. To fill the empty positions we use heuristics (see section 4/Heuristics) or the help of an analyst in interactive mode.

Concepts: The basic building blocks of our graphic language are concepts (entities) and the relations between them. The concepts are nouns/names in NL taking on the role of subject or object. They are characterised by name, gender, singular/plural, definite/indefinite article. In the graph we present only the name of the concept, while the other information is kept in the tabular presentation of the text, which serves us as a knowledge base (KB) for all kinds of syntax and semantics knowledge extracted during the analysis. The concepts are presented in the manner of an oval form containing the name (see Tab.1 line 1), while the relations between the concepts are presented as a pointed and labelled arc. The following paragraphs describe different relations in which concepts are involved.

Predicative: This relation connects two concepts with a verb. It is presented by an arc labelled with the verb, which points from the subject towards the object (Tab.1 line 6). When the action is transferred from the object (Ob) towards the indirect object (iOb) the label of the connecting arrow also contains the preposition which transfers the action. For example: RTPS charge driver at a gate (Tab.1 line 8). We use the active voice of verbs (already changed and saved into KB). Attributes of verbs are adverbs, modality, and time. They can be added in brackets to the label of the arc. This information is contained in the tabular KB, but is not used for all applications.

Prepositional: This relation is between two concepts connected with a preposition. Unlike the predicate arc, the prepositional arc is a dashed line (Tab.1 line 7).

Attributive: This type of relation presents two kinds of grammatical constructions: “adjective(s) followed by noun” or “noun is adjective(s)”. For example: nice blue car or the car is blue. More adjectives can be connected through commas or conjunctions. This information we define dur-
ing the phase of syntax analysis. The graphic presentation of the attributive relation is presented in the form of a solid line oval for the concept (noun) and an oval with a dashed line for the attributes (adjectives) attached to the noun (Tab.1 line 9). One concept can enter into predicative relations alone or with some of its attributes. In the first case the arc of the relation connects to the concept and in the second case to the attribute.

**Compositional:** This relation type presents the following syntax constructions:

- **Compound noun** (noun-noun modifiers), for example: “account statement” or “toll gate sensor”. The other reading of a compound noun is: the account has a statement; the statement belongs to an account; statement of account. We present the compound noun as an oval for each concept participating in the combination, and we place the ovals adjacent to one another (Tab.1 Line 9). With experience, however, we have developed heuristics to keep the graphic simple and clear. Thus, if a part of the compound noun does not participate on its own in relations, we prefer to express the sentence within a single oval, for example, special line (Tab1 line 9). If different parts of the compound noun participate in different relations, the arc of the relation is attached to the last concept of the combination, for example: “toll gate” – attached to gate or “toll gate sensor” – to sensor (see SN in fig2). In that way we keep the order of inheritance.

- **Key-word structure:** There is one characteristic category key word in the texts, which present structures from data or objects. Such as: type, kind, consists, include, part, have… etc. The concept related to the key word forms the head of the structure, while the concept/s (after verb be or enumeration) - its body. In the example: There are three types of toll gates: single toll, entry and exit toll, the head is toll gate, and the body - single toll, entry and exit toll. Graphically the head of this structure is depicted as a concept, and the body in two ways: i) rectangular callouts, with label the key word and content the concepts from the body of the structure. This representation is used when the concepts from the body don’t participate in other relations in the SN (Tab.1 line 11); ii) fork in the tips of which are placed the concepts from the body of the structure. This representation is used when the concepts from the body participate in other relations in the SN (Tab.1 line 10).

**Compression info:** In order to achieve precision and eliminate ambiguity, often in written language are used clarifications and redefinitions. This is reached through the use of punctuation, explicatory relative sentences, apposition, etc. We extract this information during the syntax analysis and through heuristics we decide whether or not to include it in the semantic network, and if so in what form. The compressed form is depicted through a numbered pin attached to the concept, which it is related to, and under this number in a legend we write the “compressed” text (Tab. 1 line 17).

**Synonyms:** For example: device (gizmo). We recognize by the punctuation that a synonym is introduced. We write the synonym after the name of the main concept in the oval form, and divide the two names with a perpendicular line (Tab.1 line 3).

**Relations between relations:** Graphically these relations are defined with an arc which is between the concept and arc or between two arcs.

**IF – THEN – ELSE.** In technical texts are often seen conditional sentences which have the following syntax structure: if Re1 then Re2 else Re3. In place of IF one can use: when, whenever, etc; In place of THEN: comma, etc. Re1, Re2 and Re3 can be any of the relations described above, concept or attribute, as the presence of Re1 is a condition for the presence of Re2, while the failure of Re1 leads to Re3. This dependence is expressed through two arcs: one connects Re1 with Re2 as in IF-THEN, the other – Re1 with Re3 as in IF-ELSE (Tab.1 line 15). In order to distinguish the two conditional connections, the beginning of one of them (truth) is shaded diamond, and the other (false) is empty. One of the relations can be a result of various conditions, coming from various sentences in various parts of the texts.

**Result of the predicative relation.** Let’s consider the following two sentences, which are consequential in a given text: The sensor reads the gizmo. The information read is stored by the system. We notice that the verb from the first sentence becomes an attribute of concept in the next sentence, which most probably means that the concept in the second sentence is a result of the action of the first. This relation is graphically expressed with a pointed arc from the predicate of the first sentence to the resulting concept, as shown in Tab.1 line 12.

**Predicative relations connected with conjunctions or relative pronouns.** Often one and the same subject fulfills two actions in a single sentence. We can present this as we unite the two predicative arcs with a graphic sign for conjunction/disjunction attached to the subject (Tab.1 line 13).

Let’s also consider the example: Sue thinks that Bob believes that a dog is eating a bone. The graphic of this sentence is as in Tab.1 line 16.

Other examples demonstrating predicative relations: The client activates a gizmo using an ATM; the system used read info to debit account (Tab.1 line 14).

### 4. CASE STUDY

In order to demonstrate the application of our graphic language we propose a solution for an example taken from [18]. In the source, that example is chosen to show extraction of statistical information helping the work of the analyst, and not an automatic analysis of NL. The example is interesting in that the text is uncontrolled and contains various language constructions. The text of the case study follows; the semantic network with described notation of Table 1 is presented in Fig.2.
<table>
<thead>
<tr>
<th>№</th>
<th>Notation</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image" alt="driver" /></td>
<td>Concept with name <strong>driver</strong></td>
<td>Nouns, proper nouns</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="authorized" /></td>
<td>Attribute with name <strong>authorized</strong></td>
<td>Adjectives</td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="device" /></td>
<td><strong>Gizmo</strong> is synonym of <strong>device</strong></td>
<td>Device(gizmo); device named gizmo; device called gizmo</td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="store" /></td>
<td>Predicate with name <strong>store</strong></td>
<td>All verbs</td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="at" /></td>
<td>Prepositional connection <strong>at</strong></td>
<td>All prepositions</td>
</tr>
<tr>
<td>6</td>
<td><img src="image" alt="system" /> <img src="image" alt="account" /> <img src="image" alt="debit" /></td>
<td>Predicative relation <strong>debit</strong> between <strong>system</strong> and <strong>account</strong></td>
<td><strong>Subject Verb Object</strong> – sentence “<strong>system debit account</strong>”</td>
</tr>
<tr>
<td>7</td>
<td><img src="image" alt="driver" /> <img src="image" alt="gate" /></td>
<td>Prepositional relation <strong>at</strong> between <strong>driver</strong> and <strong>gate</strong></td>
<td>Noun preposition <strong>Noun “driver at gate”</strong></td>
</tr>
<tr>
<td>8</td>
<td><img src="image" alt="RTPS" /> <img src="image" alt="charge" /> <img src="image" alt="gate" /></td>
<td>The action <strong>charge</strong> is transited from <strong>driver</strong> to <strong>gate</strong></td>
<td><strong>Su Ve Ob iOb – sentence RTPS charges driver at gate</strong></td>
</tr>
</tbody>
</table>
| 9  | ![tool](image) ![gate](image) ![authorized](image) ![sensor](image) ![special lane](image) | Compositional relations: i)compound noun containing 3 nouns ii)attribute + noun (grouped and ungrouped) | i) Tool gate sensor  
ii) Authorized vehicle  
Spacial lane |
| 9a | ![man](image) ![owner](image) ![professor](image) | Compositional relations: Noun- verb **be** –noun                        | The man is an owner  
Smith is a professor |
| 10 | ![toll](image) ![gate](image) ![type of](image) ![single](image) ![entry](image) ![exit](image) | Key word structure - fork representation kind of, type of, has a, consist, include, sort of... | There are three types of tool gates: single, entry and exit. |
| 11 | ![registration](image) ![bank account number](image) ![vehicle details](image) | Key word structure - callout representation                              | Registration includes: owner’s personal data, bank account number, vehicle details |
| 12 | ![sensor](image) ![read](image) ![gizmo](image) ![read info](image) | One relation gives a result                                              | i) The sensor reads a gizmo. Read info...  
ii) Client activates a gizmo. Gizmo activation... |
| 12a| ![book](image) ![flight](image) ![city](image) ![booked](image) | Prepositional phrase attached to the predicate                          | (He booked a flight to the city) for me                                |
| 13 | ![system](image) ![turn on](image) ![yellow light](image) ![take photo](image) | Conjunction/disjunction of 2 relations with the same **Su**             | System turns on yellow light and takes a photo.                         |
| 14 | ![system](image) ![debbit](image) ![account](image) | One relation causes another                                             | System used read info to debit account.                                 |
| 15 | ![g lane](image) ![vehicle](image) ![g light](image) ![turn on](image) | Conditional relation: **If** rel1 **then** rel2                           | If vehicle passes green lane, system turn on green light.               |
| 16 | ![Sue](image) ![think](image) ![Bob](image) ![believe](image) ![dog](image) ![eat](image) | Relative sentence connections                                            | Sue thinks that Bob believes that a dog is eating a bone.               |
| 17 | ![photo](image) | Compression info                                                         | **photo, used to find the owner of the vehicle**                        |
In a road traffic pricing system, drivers of authorized vehicles are charged at toll gates automatically. The gates are placed at special lanes called green lanes. A driver has to install a device (a gizmo) in his/her vehicle. The registration of authorised vehicles includes the owner’s personal data, bank account number and vehicle details. The gizmo is sent to the client to be activated using an ATM that informs the system upon gizmo activation. A gizmo is read by the toll gate sensors.

The information read is stored by the system and used to debit the respective account. When an authorised vehicle passes through a green lane, a green light is turned on, and the amount being debited is displayed. If an unauthorised vehicle passes through it, a yellow light is turned on and a camera takes a photo of the plate (used to fine the owner of the vehicle). There are three types of toll gates: single toll, where the same type of vehicles pay a fixed amount, entry toll to enter a motorway and exit toll to leave it. The amount paid on motorways depends on the type of the vehicle and the distance traveled.

Syntax analysis of the text precedes the building of a semantic network. We use the following processing stages of the text: Part-of-Speech (POS) tagger [15] to obtain syntax category of words; morphological analysis in order to identify inflexion of the words; parser [16] and syntactic chunking [17] in order to identify the three main groups in the sentence – Su, Pr, Ob; general knowledge glossaries for key words; heuristics.

Algorithm for defining the triplet Su, Pr, Ob: The sentences with the verb in a passive form are processed by transferring the verb into active form and changing the positions of the object and the subject. For example: The gates are placed at special lanes; A gizmo is sent to the client. After turning the passive form into active this sentences becomes: Someone places the gates at special lane; Someone sends the gizmo to the client. Since the examples lack a subject we can keep the position of the subject empty and the analyst will fill it in interactive mode. However, we use heuristics and propose a solution. The heuristics are based on: i) knowledge for agent (system is an agent by default); ii) “head” of the phrase (prepositional, passive or active); iii) patterns; iv) the principle of the closest neighbour.

There are three main algorithm steps:

**Step 1: Filling in the empty Su positions in passive sentences.** We explored many examples of complex sentences with sub sentences in passive voice of the verb. We summarized the cases of using verbs in passive voice into the following three categories: i) conjunction ‘and’ between two ‘passive’ sub sentences introduces repetitiveness which leads us to assume that the subject from the first sentence is repeated in the second; ii) be+passive in the second sub sentence takes the subject from the previous sentence; iii) pure passive verb is attached to the nearest neighbour (either subject or object from the previous sentence) that it supports. In this case the neighbour is accepted as subject.

**Step 2: We transfer the passive sentences into active.** The concept from the subject position becomes direct object, Su position remains empty, indirect object preserves its position, the verb transforms from passive into active.

**Step 3: We fill in all Su positions that are left empty.** The possible candidates to fill in the empty Su position are: i) an agent, if any is introduced in the previous sentence; ii) system by default. We support the glossary with possible agents. Application example of the algorithm follows.

| 1 | The gizmo is sent to the client |
| 2 | The gizmo to be activated |
| 3 | using an ATM that informs the system upon gizmo activation |

Fig. 2. Semantic Network of textual requirements specification
Step1 ii) defines the subject position of the second sentence, the gizmo. Step 2 transforms the passive voice into the active voice and we obtain:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Agent</th>
<th>Object</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sent</td>
<td>the gizmo</td>
<td>to the client</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>activate</td>
<td>gizmo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>using</td>
<td>an ATM</td>
<td>that</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>informs</td>
<td>the system</td>
<td>upon gizmo</td>
<td>activation</td>
</tr>
</tbody>
</table>

Step3: An agent is introduced in the first sub sentence – client, which becomes the subject of the second and third sentences. ‘ATM’ becomes Su of the fourth sentence since it is introduced by a relative pronoun. The only position left empty is the Su position in the first sentence which by default becomes ‘system’. The processed sentences become: 1System sends gizmo to the client; 2Client activates gizmo; 3Client uses ATM; 4ATM informs the system upon gizmo activation.

In tabular view similar to the example illustrated above we inscribe the entire text, from which we later build a SN. Constructing table and its advantages are described in [9].

**Heuristics for knowledge compression:** In order to preserve a clear picture of SN we prefer to compress some parts of the sentences. Good candidates for compression are explanations to concepts: 1) Business rules. In order to find them we seek i) key words, e.g. possible, request, except, depends, only...ii) modality of the verb: must identify, can be performed...iii) quantifiers: all, every, nobody... 2) Appositions and inversion 3) explanation attached to the enumeration by brackets or relative pronoun (see the second to last sentence in the case study).

We continue to develop and improve the graphic language for presentation of textual user requirements. We seek the most appropriate and readable graph, as well as heuristics for extraction and presentation of all relations.

**POS and referential ambiguity:** Table presentation of text gives good visualization of text for easy verification of these two problems. The position of the concept/word in the table must correspond to its syntax category. For example, a common mistake is considering verb as a noun and vice-versa. With a quick proofread of the predicate column this mistake can be identified. Similar is the resolution for substitution of reference with real concept – it is found in close proximity in the Su or Ob column.

**5. CONCLUSION**

**Summary:** The goal of our project “graphical language for presentation of textual user requirements” is to create a universal graphic model of NL text and the knowledge it carries. We consider this model to be a basis for obtaining graphical models for requirements specification, for example UML models. Instead of developing a unique graphic presentation of the knowledge appropriate for each different UML model we follow this approach: NL is universal as a method for presentation. If we have a graphic model of the language, we can extract necessary knowledge from it and proceed to other graphic models through automatic transformations. We present this idea through the following scheme: NL → GL → UML models.

The transitions between the different phases of the transformation of NL into UML diagrams can contain different treatments according to the chosen technologies for implementation, but in our system the treatments follow the order: NL → POS tagging/parsing/chunking → Table presentation (visualisation, verification) → XML presentation → Visualisation → Semantic Network → Translation into other graphical models.

Until now we have proposed translation of GL into the following UML models: OO model in [9], and domain model, activity diagram, use case path in [10]. The translation is based on finding the correlation between the graphic templates from source GL and target models.

**Comparison with the other Semantic Networks:** Our graphic language is a type of semantic network [11, 12]. It has graphical power to present all examples given in [12] which demonstrates the possibilities of a different SN. The philosophy of our graphical model is that there is a relation between the basic building blocks of language and knowledge. This relation consists of 3 elements; it has direction and can influence or is influenced by other relations. For example, Definitional Network [12] is a hierarchy of concepts that is easily presented in GL notation through compositional relation.

Implications are the basic elements in the Existential Graph of Peirce [13] and Implicational Networks [12]. Implications are presented easily in GL notation as shown in line 15, Tab.1. Fig. 3 illustrates three types of SN for the example: If a farmer owns a donkey, then he beats it. Our GL graph is in the central position. However, our graphic language is an Assertional Network [12]. Our representation of asserton differs from the relational graph (RG) of Peirce in the sense that we consider entities independently, while in RG they are united around lines of identity. Assertion for us is not an indivisible unit, but a relation built by the entities/concepts, which can participate in more than one assertion.

This independence of concepts allows us the possibility to connect them in different relations and in this way to present a text as a whole, rather than different sentences. This independence of the concepts differentiates our SN from Dependency Graph [12], and other graphs similar to it (SNePS, Schank’s conceptual dependencies, Conceptual graphs of Sowa) [12], which present one sentence/relation as a frame; while one entity enters into different relations it is repeated in each of them. Fig. 4 presents the example Sue thinks that Bob believes that a dog is eating a bone realized in 3 different SN. Our GL graph is in the central position.
The advantage of our SN is in attaching to each entity all relations which it participates in, and this is of major importance for the construction of various models. For example, in OO model it is very important to have for some object/concept all actions which it executes, the attributes it possesses. Through GL presentation of text we have the possibility to compare superficial graphic characteristics, which leads to the discovery of different heuristics for possible inclusion of different concepts in different roles in the SE models. For example: the active nodes (the ones which send more predicate arcs than they receive) are serious candidates for objects in OO models. The passive nodes can have different “swimming lanes” in the sequence message chart, if they receive messages (predicate arcs) from more participants in the network, or else they can be presented as an addition to the activity of a basic/active “swimming lane”. Our SN is unique also because it uses different graphic notations and positions for different language elements and roles, and in this way implicitly includes semantics without having to use apparent semantic labels. For example, agent is this concept, which gives the beginning of an arc. However, one and the same entity can play a different role in different relations – that of an agent and that of a patient as well. Knowledge supporting and aiding the semantics is saved in tabular presentation, which can include different numbers of columns for the different knowledge appropriate for different applications. The tabular presentation also maintains the order of different assertions (sentences) in the way they appear in the text. In other words the tabular presentation is nothing other than the text kept in full, but structured in three basic columns (Su,Pr,Ob) and with sentences transformed from the passive into the active voice. Other examples of tabular presentation can be seen in [9, 10].

GL can also be considered as Executable Networks [12], because it easily maps into an activity or Use Case Path diagram, which is nothing else but a series of actions and conditions, executed by actors. Finally, our GL can be called Hybrid Networks [12], because it presents different knowledge with different syntax constructs. Its main purpose is to automatically translate textual user requirements written in uncontrolled NL into UML, or other types of SE diagrams.

Future work: We continue to develop and improve our graphic language, seeking the most precise and readable graphic presentation of linguistic, general and domain knowledge all together. We are working on automatic transformation of GL into different types of SE models. We are working on the visualisation and improvement of the interface with the analyst. We are developing a prototype of Integrated Framework for Automatic Analysis of Textual User Requirements, to which GL serves as a basic module.

REFERENCES