Models Derived from Automatically Analyzed Textual User Requirements

M. G. Ilieva, O. Ormandjieva
Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec, Canada
E-mails: {magda,ormandj}@cse.concordia.ca

Abstract

Requirements Engineering is an important area of software engineering concerned with the extraction and presentation of knowledge from user requirements. There is a considerable gap between the various types of presentation of the same kinds of knowledge – those of the user written in Natural Language (NL) and those of Requirements Engineering (RE) depicted using diagrams (de facto standard UML diagrams, for example). Our research is aimed at filling that gap. In this paper, we introduce a universal formalism based on the basic building unit of NL, which is the relation triad. Through the definition of these basic relations in NL, we create three RE models: the Use Case Path model, the Hybrid Activity Diagram model and the Domain model. These models are abstractions of the knowledge contained in the text, and serve as the basis for deriving UML diagrams.

1. Introduction

In order to create effective and useful software specification models, it is necessary, first, to have the ability and talent to formalize and present knowledge in an appropriate way from the point of view of a computer, and second, to know, in detail, the specific domain environment in which the system is to function.

While a domain expert would not know much about this type of modeling and would tend to express his or her knowledge in NL, the RE specialist would not be entirely familiar with the problem domain and would tend to express knowledge in some formal way, a graphical model, for example. There are methodologies and tools available for the automatic translation of knowledge from text into models in order to fill the gap between the two types of knowledge and competency. The models are created automatically and might not be as perfect as those created by the RE expert, but they speed up the process of understanding the problem domain and quickly create a solid base from which the model can be built and perfected.

Related Work. In an effort to find a way to map the NL presentation to the SE model presentation, an intermediate model has been developed which combines the two types of presentation: linguistic specificities with elements of the target model. The difficulty in translating NL into any type of model is considerable, and so, in order to do this, NL is limited (formalized) and the processing is carried out according to the following scheme: NL > Formal NL > Intermediate MODEL > Target (RE) MODEL.

In [6], for example, NL is limited to scenario specification, while the intermediate model is an activity diagram of three types of activities, which are related to three types of connections. The target model is a kind of UML [15] activity diagram. In [9], NL is limited to simple, one-verb phrases, and the intermediate model is a diagram depicting four types of behavior. The target model is a class model diagram. In [5], NL is limited to a context-free grammar, and the intermediate model is a tree data structure with three types of nodes representing different types of knowledge. The target model is a class model in VDM notation. In [3], NL is limited to functional grammar, and the Static Object and Event Model are used as intermediate models. In [4], NL is limited to a generative grammar (similar to case grammar), which produces 5 phrase categories. From these phrases, an intermediate cooperation-type schema is built which easily maps to the final activity diagram. In [7], a limited NL is offered with defined syntax and semantics, the intermediate presentation is made up of linguistic patterns and the final model is a UML class diagram. In [2], NL is limited to use-case scenarios, the intermediate presentation takes a flat logical form and the final step generates the graphical model.
The disadvantage of all these methodologies is that they require correspondence between the NL format, the intermediate model and the target model. Moreover, their application is limited to descriptions which strictly follow the limitations of NL for a single target model.

**Our Approach.** Our system differs from those described above in the following way: we consider unlimited NL, and, while the intermediate models keep their NL characteristics, we rewrite it in a different format. Our intermediate models – tabular text presentation and a semantic network – are universal, in that they each have their own meaning and they can be used independently for different linguistic and modeling purposes. The NL processing (NLP) and the knowledge are connected through the searching and matching of “element knowledge” v.s. “element presentation” from the NL language description and a target model. For the purposes of searching and matching we use the NLP methodology; the architecture of our scheme is presented in the following figure:

![Figure 1. Main processing phases](image)

Textual analysis and the construction of models are conducted in four stages: syntax categorization, tabular modeling of the text, semantic processing of the text, and, finally, interpretation of the text for diagrammatic modeling. The various stages correspond to the traditional processing types in NL analysis – syntax, semantics and pragmatics. These terms do not have a single definition, and, for our purposes, we stick to the definitions of Charles Morris [8]: syntax – examining the relationships between signs; semantics – tending to focus on the actual objects or ideas to which a word refers; pragmatics – attempting to understand the relationships between the signs and their interpretation. For us, the important key words extracted from Morris’ definitions that characterize the phases, are: syntax – the differences between the signs (caused by the different relations); semantics – the relationships between the signs; and pragmatics – the interpretation of the relationships.

The paper is organized as follows: sections 2 to 5 present the four processing phases. Section 6 presents a solution which illustrates, in practice, the phases of the processing and the creation of the three final models: use-case path, hybrid activity diagram and domain model. Section 7 summarizes the results and provides directions for future research work.

### 2. Syntax categorization

The goal of this stage is to obtain the syntax category for each word in the text. For this, we use some of the available Part Of Speech (POS) taggers [16]. Take, for example, the sentence **Opening an account can be performed only via a teller or the Internet.** After treating it with POS tagging, it looks like this:

```
Opening/NN an/DT account/NN can/MD be/VB performed/VBN only/RB via/IN a/DT teller/NN or/CC the/DT Internet//NN ./.
```

After each word, there is a slash followed by the respective grammatical category of the word. For example, NN stands for noun, singular, and RB for adverb. The complete tag-set terminology for the above mentioned POS tagger is available in [17].

### 3. Tabular text model

The information extracted in the first stage is used to discover the roles of the words in the sentence and to arrange them in a table. There are three roles in a sentence: subject (Su), predicate (Pr), object (Ob). Su and Ob are noun groups and Pr is a verb group. In the noun group can be found determinants, adjectives, numbers and nouns. Two or more noun groups can be joined using prepositions and conjunctions. For example, the phrase **account statements to all clients** consists of two noun groups connected by the preposition to form a complex noun group. In the verb group are included the main verb, auxiliary verbs and adverbs. Two or more verb groups connected using a conjunction form a complex verb group; for example, **can read and write slowly.**

The compound sentence consists of more than one triad (Su, Pr, Ob) connected with conjunctions or relative pronouns, or nothing. Some of the positions that can be filled with Su, Pr, Ob may remain empty in some compound sentences. To demonstrate how the roles in the sentence are defined, as well as how they are arranged in tabular form, we will examine the first sentence of the second paragraph in the case study presented in section 6.1.

To label the roles in the sentence, we use the outcome of the POS tagger. The verb group forms Pr, the noun group to the left of Pr is labeled Su and the noun group to the right of Pr is labeled Ob. We use grammatical rules and heuristics to label the roles in a compound sentence. The sentence that has already been
partitioned into roles is arranged into a table. The table has four main columns, each of which representing a role, which appear in the same order as the roles appear in the sentence. There is also a column for the connection between the simple sentences that make up a compound sentence. This connection is important, as it signifies the role of the sub-sentence in the main sentence (its relationship to the main sentence).

<table>
<thead>
<tr>
<th>№</th>
<th>Su</th>
<th>Pr</th>
<th>Ob</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The</td>
<td>are</td>
<td>recorded</td>
<td>, because</td>
</tr>
<tr>
<td>2</td>
<td>----</td>
<td>----</td>
<td>at the end of each month</td>
<td>,</td>
</tr>
<tr>
<td>3</td>
<td>the system</td>
<td>sends out</td>
<td>the account statements of all clients</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>showing</td>
<td>all the transactions performed on their accounts during the last period ;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>the system</td>
<td>sends</td>
<td>the statements to the printer from where</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>the junior clerk</td>
<td>posts</td>
<td>them .</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Tabular presentation of text

Let us take a closer look at the empty Su and Pr boxes in second record. This is the way we represent apposition, the role of which is Ob (adverbial modifier for time). The 4th record is a subordinate sentence, where we find an ellipsis (containing the missing Su). The easiest (and in many ways the most correct) way to find the missing part is to look for it in the preceding record in the same column. Similarly, we can also resolve the reference, as, for example, *them* in column Ob – the 6th record. *Them* refers to *the statements*, which is the main noun in the preceding record in that column.

In this example, we can see how a compound sentence of any complexity can be presented in a unified and formal way. The record preserves the natural state of the sentence and is clearly arranged. The tabular presentation has other advantages:

– direct access to the parts of each sentence, and, as a result, easy implementation of inference and question/answer mechanisms;
– better possibilities for future automatic processing, due to its formal nature; for example, writing in XML format [13], transformation into SVG [14] and visualization.

4. Semantic Network text model

The tabular presentation shows the connections between the elements (words) at the sentence level. In order to include the interconnections between the elements within the whole text, they need to be presented in a semantic network. The basic elements of this network are the relations (triads): predicate, prepositional, structural, attributive, possessive and enumerative. Examples of these relations are given in Fig. 3.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Example of usage</th>
<th>Graphical notion</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicate</td>
<td>the bank offers accounts</td>
<td><img src="bank-offer-account" alt="predicate" /></td>
</tr>
<tr>
<td>prepositional</td>
<td>services on these accounts</td>
<td><img src="service-on-account" alt="prepositional" /></td>
</tr>
<tr>
<td>structural</td>
<td>account types which fall into two categories: saving and checking</td>
<td><img src="account-saving-checking" alt="structural" /></td>
</tr>
<tr>
<td>attributive</td>
<td>last period</td>
<td><img src="last-period" alt="attributive" /></td>
</tr>
<tr>
<td>possessive</td>
<td>account statement of bank’s client</td>
<td><img src="statement-of-account" alt="possessive" /></td>
</tr>
<tr>
<td>membership (enumerative)</td>
<td>services on these accounts: deposit, withdraw, transfer, get balance</td>
<td><img src="deposit-withdraw-transfer-get-balance" alt="membership" /></td>
</tr>
</tbody>
</table>

Figure 3. Main relationships in a semantic network

In the semantic network, different elements are presented only once, no matter how many times it is encountered in the text. All relations in which it participates are attached to it. In this way, we obtain a full and comprehensive, yet compact, picture of each element in the network: the activities in which it participates (the arcs that it sends, in its role of Su, and the arcs that it receives, in its role of Ob); the prepositional and/or structural relations in which it participates; its attributes.

5. Interpretation of relationships extracted from the text

At this stage, we interpret the different ways in which to present the knowledge in NL, and we look for
the corresponding presentation in the RE model. We are interested in the main participants, the actions, the conditions for the actions and the communications (messages) between participants. The objective is to arrange this information and to model it in the form of a diagram.

The main component of language models (tabular and semantic) is the relation. It has three elements, collectively referred to as the triad which has the following features:

i) It consists of two participants and the relation between them. Some of the triad elements can be omitted for purposes of simplification, i.e. when the element is clear and there is no need for repetition. However, clarity in NL is relative, which is why it can be ambiguous. If we take a closer look at the relations defined in Fig. 3, we discover that the last three are not presented as a triad. They are triples, but the relation is omitted. For example, in the “membership relation”, the predicate contains or consists of, or something similar, is omitted and replaced by a semi-colon. Account statement, which is an example of a possessive relation, is also a triple and can be represented as statement of account. The attributive relation last period signifies that that period is last. The following sentence is another example of an attributive relation: The transactions are recorded. For our purposes, we prefer to consider this relation in its active form: Someone records transactions. But, to whom does someone refer? Incomplete relations can be encountered at any level of the text, and it is incompleteness that causes ambiguity. The logical way to resolve such ambiguity (i.e. filling in the missing part) is to use the principle of the closest or corresponding part (see the example in Fig. 2.). The meaning of the relations is different, and that is why the presentation is different (see Fig. 3).

ii) Direction of the relation. The direction very often matches the direction of the sentence, i.e. left to right: Su > predicate > Ob. (The predicate is directed from Su towards Ob), or Noun1 > preposition > Noun2 (Preposition is directed from Noun1 towards Noun2). However, that is not always the case. For example, the passive voice reverses the direction, as in the following: “A separate...” defines the direction from A, but “A is separated by...” defines the direction towards A. “A to B” defines the direction from A to B, but "A from B" defines the direction from B to A. In “Client accesses BAT via ATM”, we interpret the sequence of the actors as follows: Client, ATM, BAT. Direction is valid for the space as well as for the time. Direction is important in the semantic network, and evidence that this is true is the existence of prepositions expressing direction; for example, from...to...; if... then... Usually, in the if–then relation, we consider the meaning of each word individually; however, if we look at if–then as a pair, these two prepositions signify direction from one relation to another: the first relation is after if, and is a condition for execution of the second (after then).

iii) Heuristics: Extraction and presentation of implicit knowledge is of great importance in processing unrestricted NL. For example:

– Relations with an implicit reverse direction, such as possessive relations. The bank’s client can also be interpreted as the client of the bank. Similarly for account statement, which can be interpreted as statement of account. We use the first part of the possessive relation (client, statement) if such a group participates in the role of Su or Ob. The relation between the two also means that the bank has a client and the account has a statement.

– Business rules. In order to recognize the conditions in which the actions are executed, we look for key words in sentences, e.g. possible, requires, except, only, no; the modality of the verb, e.g. must identify, can be performed, to be sent; quantifiers, e.g. all ...; if – then sentences, e.g. if a client opens an account, then....

– Pattern expressions. Exploration of pattern language constructs is important for the discovery of missing parts. In our text example, these phrases are:
  o all transactions and queries are possible via a teller
  o all transactions and queries are possible except deposit via an ATM
  o all...except deposits and withdrawals via the Internet

We take “transactions and queries are possible” as the missing part. We look at the text for another phrase that matches ours, and find ..., if a client opens an account... We assume that Su = client. The rules for matching are defined in [12].

– Discovering synonyms. We explore belong/not belong to discover synonyms. Let us look at the following phrases:
  o services on accounts: deposit, withdraw, transfer, get balance
  o all transactions and queries except deposit
The first phrase states that deposit belongs to the set of services, and the second that it is excluded from all transactions and queries. We can accept with a high level of confidence that the two sets are synonymous, i.e. services = all transactions and queries.

The above are examples of the heuristics that we apply, and on which we continuously expand.

This interpretation gives rise to several static and dynamic models which are presented as diagrams. The
diagrams chosen here are the Domain model (static view), the Use Case Path model (dynamic view) and the Hybrid Activity Diagram model, which shows the relation between the static and the dynamic view in a comprehensive manner.

The process of interpretation of these diagrams by means of the semantic network and the process by which they are constructed are explained below.

6. Illustration on a case study

In this section, we illustrate the stages of our methodology on a simple case study.

6.1. Defining the problem

We have chosen a plain text description which is not aimed for automatic processing [18]. This is a text created for training purposes, and the objective is to perform scenario specification through the analysis of the text.

The Bank Accounts and Transactions (BAT for short) system is to be built for the Big Bank Corporation. It must handle clients' bank accounts and the (standard) services on these accounts: deposit, withdraw, transfer, get balance.

The transactions are recorded, because at the end of each month, the system sends out account statements to all clients showing all transactions performed for their accounts during the last period; the system sends the statements to the printer from where a junior clerk posts them.

The system is accessed by the bank's clients only indirectly, i.e., either via a teller, an ATM, or the Internet. All transactions and queries are possible via a teller; all transactions and queries are possible except deposits via an ATM; and all except deposits and withdrawals via the Internet.

Opening an account can be performed only via a teller and the Internet; however, if clients open an account via the Internet they must identify themselves with a teller to have their account activated (this is government policy to avoid money laundering, e.g.).

Closing an account can only be performed by a teller, and it requires a final statement to be sent out to the client. The Bank offers various account types, which fall into two categories: savings and checking. Savings accounts cannot be overdrawn. There can be a credit limit, subject to agreement by the bank, on checking accounts; a checking account cannot be overdrawn beyond this limit.

6.2. Semantic Network

We start with syntactical analysis of the text, and then build a tabular presentation and a semantic network. The methodology for this procedure is described in [11].

The resulting semantic network diagram (see Fig. 4) reveals its features unambiguously. The diagram is compact, and clearly presents the objects and the relations among them. The advantage of the semantic network is that it connects the objects to the actions occurring between them, even though they appear at different places in the text. In that way, repetition is avoided, which is the objective: each object appears only once with all its relations with the other objects in the network. This presentation is useful when we want to map it to concepts and their relations in a Domain model, as explained later. In order to build a sequence or activity diagram, or any other type of diagram where the time sequence is important, we need to unfold/unwrap the semantic network. We do this using the algorithm described in next section.

6.3. Transition from the Semantic Network to models

6.3.1. Use Case Path model – generation rules. The Use Case Path (UCP) presents, consecutively, the route of one action through the different actors responsible for its implementation. In order to generate routes for possible actions, we use the following algorithm:

1. Look for sentences (in the table presentation of text) with the following structure: Subject -- Pr -- direct_object -- preposition for direction --
indirect Object. This structure shows the direction of the action. The action Pr is performed by Su on dOb and is directed towards iOb. We are guided by the heuristics that participate as actors in this type of sentence, and discovering them is the first step. In the text of the case study, the sentences of that type are the following: The system sends out account statements to all clients; The system sends the statements to the printer from where a junior clerk posts them; Client closes an account by a teller.

2. We underline Su and iOb, as it is likely that they are candidate actor, and, as such, the actions occurring between them will change. “System” performs actions by default. The underlined knots are darker.

3. We choose a starting point. The options are the following:
   – a knot in the network shown by the analyst;
   – client, user, etc. often play the role of a trigger for the system;
   – after exploring the knots in the network, a trigger is chosen which will take us along the longest path.

4. We follow all possible paths which start from the trigger. We maintain the direction of the actions, i.e. we move in the same direction as the action arcs, not against them. The labeling notation is the following: the actors are referred to by the names of the underlined knots; each action is presented along with a directed arc and labeled: the action (Pr) is above, and the object (dOb), if there is one, is below.

5. We assign the business rules (callouts) to each of the routes, and we connect them with the related objects/participants.

The result of applying this algorithm is a presentation of all possible paths extracted from the semantic network of textual requirements (see Fig. 5).

6.3.2. Use Case Path model – operations.

Merge: UCPs with similar parts are candidates for this operation. The objective is to make one UCP out of a number of them that have repeating parts. From all these identical parts, a single UCP is formed. Where the difference begins, a point of separation can be inserted, which could be of either the alternative or parallel type. Every UCP receives its identifier, which is checked at the point where the separation occurs. For example, the first three UCPs could be merged as follows: the hatched parts are the identical paths, i.e. they are replaced with a single UCP. The first hatched group is replaced with “client > access”. The separator follows, and identifies the UCP. If it is a UCP1, then it is processed along the first path; if it is a UCP2, then it is processed along the second path, and so on. Where identical parts appear, the paths are merged. The second hatched group is replaced with “access > BAT System > send > printer > junior clerk > post”. The entire operation is explained in Fig. 6.

Add: A new action or a sequence of actions can be added to any knot of the existing action sequence. The added sequence can be processed in parallel or as an alternative to the main sequence. The point of connection is qualified by a sign to denote an alternative or a parallel path (see legend in Fig. 6).

Cut: This operation is the opposite of add. Incorrect or unnecessary parts can be cut if cutting does not affect the logic of the connections (whether alternative or parallel).
through a Hybrid Activity Diagram (HAD), as shown in Fig. 7.

![Figure 7. Hybrid Activity Diagram](image)

To present HAD we use the graphical notation shown in Fig. 8 below. Here, only one of all possible use case scenarios is shown as an HAD. The others are handled in a similar way, using the algorithm in 5.3.1.

### 6.3.4. Use Case Path and Hybrid Activity Diagram models – a comparison

We consider the UCP to be a simplified activity diagram. It saves space and looks more like a path. The source for building both types of diagrams is the semantic network. The UCP notation is similar to writing a simple text sentence, containing actors and the actions occurring between them. There are three arguments associated with the action – direction, name, object. There are extra features, too – conditions and branches. The advantages of UCP are these: the route of one use case can be tracked from its starting point to its end point within the semantic network; all possible paths are easily traced; each path stands for one use case and follows its action sequence; the logic of each path can be traced quickly and easily; paths can be easily corrected through the defined operations (see 6.3.2.); and the notation is similar to the text presentation, and therefore user-friendly, efficient and clear.

In the HAD notation, the actors are presented with an actor’s lane, and the actions between them are presented in one of the ways described in Fig. 8. The advantages of HAD are these: it is compact; and its notation is similar to that of UML and other block charts, which are popular in the software community.

Transforming UCP into HAD, and vice versa, is easy. The main difference is in the spatial presentation of the actors. Apart from “decision”, the actions are presented in the same way. In UCP, decision is represented as a circle.

The automated extraction of knowledge from unrestricted text is a difficult job, one that has to be performed in many stages. At each stage, a small part of knowledge understanding, extraction and presentation is achieved. This constitutes a hierarchy of knowledge and processing, which could take the following form:

Text R ➤ tabular presentation ➤ semantic network ➤ use case path ➤ hybrid activity diagram

![Figure 8. Hybrid Activity Diagram notation](image)

### 6.3.5. Domain model

The knowledge from the semantic network can be easily translated to the Domain model. What we are looking for in the semantic network are the structural relations. In our example, there is a clear relation of that type: various account types, which fall into two categories: savings and checking. To present the relation as such, we used the following heuristics: the presence of the key words types and categories and the subsequent listing that of “saving and checking” gives us reason to accept this as a structural relation. We continue by looking at the knot account. At the same level, it is connected with services (which is also a structure – enumeration), and on a upper level BAT System, where the connection is of the predicate type (handle) with an abstract sense meaning subordination (not a concrete action). The BAT System is also connected to ATM, Internet and Teller by means of prepositions. We define these as its subparts (from the direction of the relation). Similarly, these three concepts are connected to account, although system has a higher priority and is located at a higher level. The


**BAT System** is also connected to **Big Bank Corporation** with a preposition (**Big Bank Corporation** is not connected to anything else), at which point the sequence of structural connections ends.

From the general semantic network, we separate the part with structural connections (see Fig. 9a). We adjust the semantic network notation with a block diagram, which is widely accepted for presenting structures, and we obtain a Domain model (see Fig. 9b.) derived from the description in the example.

Comparing the two models, (a) and (b), in Fig. 9, we can see how naturally and easily the knowledge from the semantic network is mapped to the Domain model. The condition is that the knowledge be presented in an appropriate manner. The methodology that we propose in this research offers this possibility. Unlike the other formalisms mentioned in the introduction, which create special predefined relations and functions, we explore the structure as a type of relation that is discovered in the text itself and is presented in an appropriate manner.

Figure 9. Translation of a semantic network into a Domain model

The Domain model serves as basis for developing the structure of the OO class diagram in the design phase of software development.

#### 7. Conclusions and further work

In this paper, we presented a methodology for the automatic analysis of textual requirements and the formal presentation of the extracted knowledge into graphical models. We are currently developing a tool for implementing our methodology. Our system has the following advantages over similar systems: Analysis of unrestricted NL. To bridge the NL model of knowledge and the RE model of knowledge, we propose two models: the tabular presentation of text and the semantic network of knowledge. These models may be applied independently in other applications, as they are independent of the style and complexity of NL, and from them we build three different graphical models of the extracted knowledge. The relation has been chosen as the basic building unit of both the language and the models. This basic unit is on a lower structural level than the grammatical pattern ([10]), and therefore is more universal and independent of the rules for presenting the language and knowledge within the text.

In comparing the two solutions, the source [18] (solved by a human analyst and used for training purpose) and the case study presented here (solved automatically – see section 6), we reach the following conclusions: the original has 9 use case models, presented on 9 pages of double-spaced text. We note that terms are presented in these 9 use cases which are not in the textual requirements. Seven of the use cases are presented in our UCP, while 2 are not. Our solution contains 6 use case paths as a model of the system, all of which are described on one page. The activities in the original solution are grouped around “service”, while in our proposal they are grouped around “access to input device”. The difference derives from the fact that the goal in the original is the use case model, while we strictly follow the text, where the phrase “...access via...” is encountered 8 times (in almost 40% of the sentences in the text). This reveals another advantage of our system, which is that it allows the correctness of the textual user requirements to be verified.

In terms of the graphical models that present the knowledge extracted from text, our approach has a distinct advantage. As HAD combines and presents a variety of knowledge, such as actors, actions, business rules, decisions and changing messages, which is often presented in the textual requirements simultaneously and in an interconnected way, it makes sense for the modeling to correspond to this knowledge, thereby achieving a correlation between the knowledge in the text and the model. We can obtain different views of knowledge presentation by adding, cutting, merging and splitting over the model. The UCP model, also called a
simplified activity diagram, presents the actions and actors in interaction sequences, and can be regarded as a use case scenario. One use case path corresponds to the work expended by the system to process one request. A similar tracing of the request path can be observed in the Use Case Map (UCM) [1]. While [1] states that it is difficult for the user to find the starting point for defining the requirements with UCM terms, we regard the notation in our UCP model as more like the thinking of the user and the presentation of the description: use case paths constitute a sequence of requests transmitted between participants, each of whom specializes in the performance of a specific activity. UCPs are similar to graphs, but they contain a sentence like a notion, which is familiar to everyone. UCPs are generated automatically and immediately provide the user with a base for modeling extensions and corrections.

In our future work, we envisage extending the collection of examples for which solutions have been found, and exploring requirements in unrestricted NL. We will continue to collect texts and expand the database from which we extract, generalize and increase the number of rules and heuristics for mapping knowledge from text to an RE model. Our intention is to assist the developers in defining UML standard software models (for example, the Use Case Path model) through the application of operations on UCP only. We will investigate the possibility for an automatic updating of an existing model when modifications are required, based on their textual description.

10. References


