Automatic Issue Extraction from a Focused Dialogue

Koen .V. Hindriks[1], Stijn Hoppenbrouwers[2], Catholijn M. Jonker[1], Dmytro Tykhonov[1]

[1]Man-Machine Interagtion Group, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
[2]Faculty of Science, Radboud University Nijmegen, Toernooiveld 1, 6500 GL Nijmegen, The Netherlands

{k.v.hindriks, c.m.jonker, d.tykhonov}@tudelft.nl, stijn@cs.ru.nl

Abstract. Various methodologies for structuring the process of domain modeling have been proposed, but there are few software tools that provide automatic support for the process of constructing a domain model. The problem is that it is hard to extract the relevant concepts from natural language texts since these typically include many irrelevant details that are hard to discern from relevant concepts. In this paper, we propose an alternative approach to extract domain models from natural language input. The idea is that more effective, automatic extraction is possible from a natural language text that is produced in a focused dialogue game. We present an application of this idea in the area of pre-negotiation, in combination with sophisticated parsing and transduction techniques for natural language and fairly simple pattern matching rules. Furthermore, a prototype is presented of a conversation-oriented experimentation environment for cooperative conceptualization. Several experiments have been performed to evaluate the approach and environment, and a technique for measuring the quality of extraction has been defined. The experiments indicate that even with a simple implementation of the proposed approach reasonably acceptable results can be obtained.

Keywords: natural language processing, domain modeling, grammar parsing.

1 Introduction

Domain models (including domain ontologies) are now a common asset created and used in many contexts, perhaps most prominently in Knowledge Engineering and Information System Development (two increasingly related disciplines). The groups involved in the research reported in this paper are concerned with domain modeling from different perspectives ranging from supporting system development to supporting negotiators. For the moment, the chief context to which we apply our ideas and setup is that of *conceptual modeling in small, communication-oriented, volatile domains*. The main characteristic of modeling in such domains is that it cannot be solidly based on existing data (corpus, documents, reference models) since the concepts involved reflect knowledge of only a small number of individuals, which in addition may crystallize only in the course of the interaction between those involved (consensus-based modeling). A typical example of such a context would be *prenegotiation*, a process that among others involves establishing a conceptual common ground on the basis of which negotiations can take place, and *specification* of information system requirements and models in fast evolving environments [12] .

Only limited research has yet been done concerning the *process* of domain modeling (for example [1], [2], [6], [16]), and only some of it has an experimental character. In order to study and, in the longer run, support and improve domain modeling in general, we believe it is important to create controlled environments that enable an experimental approach to modeling *processes* and *strategies*. We believe that such environments can evolve into actual modeling environments that take modeling beyond mere "ad hoc model creation" (graphical or otherwise). Such environments will take the shape of cooperative software tools that actively support the participants in the domain description process and allow them to discuss the target domain in a focused and structured manner, and, consecutively, can present them with a clear domain model they can then validate and refine.

The research presented here concerns the design, deployment, and evaluation of a prototype of a conversation-oriented experimentation environment for cooperative conceptualization. Our focus is on the detailed succession of expressive actions taken by people involved in a conversation for domain description/modeling, and (crucially) on the patterns, rationale, and strategies underlying such actions ([6]).

Our approach involves two key steps: *focused elicitation* of a domain description in the form of a structured natural language dialogue (captured in written textual form), and *automated extraction* of the core domain concepts from that dialogue. In our approach, we assume that a predefined meta-model is available and the aim of extraction is to populate this predefined meta-model. The meta-model for the experiment was designed by us and is presented in this paper.

We also present data and results from an experiment that has been carried out in order to *evaluate* the combined focused elicitation and automated extraction approach. As part of this evaluation, we use a manually constructed domain model as a benchmark (see section 5) and apply a metric to calculate the success rate of the automatic model extractor.

We believe our approach is promising for a number of reasons. If one takes a complex text or document, not specifically created to render core concepts, as a basis for automated domain analysis, then there are two main problems:

- The Natural Language Processing (NLP) involved (parsing, semantic analysis) is highly complex, very likely beyond the point of realistic application;
- Text analysis usually renders a large number of concepts with strongly varying degrees of relevance. Separating relevant concepts from irrelevant concepts is a daunting task that cannot be automated (not without substantial material to "learn about the domain from", that is).

So, if we cannot rely on high quality bulk input that can be effectively analyzed (indeed we assume we cannot), then instead we prefer to start with the creation of a *simple* text that is *purpose created* to contain core domain concepts and show that such texts can be *analyzed* using *simple, robust NLP techniques*. In order to obtain such natural language input, we use focused dialogue games. Formal dialogue games are interactions between two or more players, where each player acts by making utterances, according to a set of rules (cf. [14]). A dialogue game has a clear goal shared by the participants in the dialogue. As a consequence, it is reasonable to expect that the task of "filtering out" relevant concepts happens *as the text is created*, based on human intelligence in description/production rather than reading/interpretation afterwards. This "filtering" effect may be enhanced by structured/guided elicitation, i.e. by introducing additional rules in the dialogue setting that should be adhered to by the participants. This approach thus is based on an alternative *method* for domain modeling: a guided elicitation process, which aims at the production of focused texts including primarily relevant, core domain concepts in a structured environment, to which the automated, and therefore repeatable, extraction procedure is then applied.

An additional advantage of our approach lies in our use of a basic meta-model that requires minimal categorization effort on behalf of the extractor. This reduces the sensitivity to errors in the extraction process. The structure we use matches the basic structures in many comparable but more elaborate meta-models (ontological meta-models), suggesting that, for example, extending this approach to more negotiation-specific and complex meta-models (such as a negotiation description language [3], or to more generic widely-used ontology specification languages such as OWL [18], should not be too challenging. Later refinement of the domain model is possible if required (both of the meta-model and of the elicitation procedure).

In many applications, including prenegotiation, extraction of a domain model instance with relations exclusively between specific objects defined in the meta-model is required (bound variables). The main bulk of the domain independent knowledge can be pre-defined in the meta-model, by knowledge engineers. Thus, language constructions such as quantification or complex anaphoric references, which are particularly difficult in view of NLP, can be omitted in the *automated extraction* stage of our approach.

We propose a method to automatically extract a (partial) domain model from a focused dialogue of natural language. The effectiveness of the extraction method has been empirically validated by means of a series of experiments. The results of the experiments were validated against manually built models using a validation metric. The metric calculates the distance between the "ideal" model extracted manually by a human domain modeling expert and the atomically extracted model.

In the next section, we present our domain extraction model. Section 3 briefly introduces the NLP techniques used in the extraction tool. The extraction approach itself is introduced in Section 4. The results of the experiments with human dialogues are used to validate the extraction approach in Section 5. Our conclusions are presented in Section 6.

2 The Domain Extraction Approach

The extraction approach proposed here consists of two phases: (i) Focused Elicitation and (ii) Automated Extraction. The goal of the first phase is to organize collaboration of the domain experts on model elicitation with a specific focus on the domain: the natural language input for the domain extraction system should have a reasonable fit with the meta-model that is used. To ensure such a fit we propose to use variants of a *dialogue game*. The main advantage of dialogue games is that the users can be manipulated to keep their sentences simple.

The second phase automatically extracts a model from the elicited domain description in terms of a given domain meta-model. The method that is proposed here for extracting a domain model instance from natural language is a combination of *robust, wide coverage parsing techniques* and what we call *concept extraction rules*, which are used by a pattern matching algorithm to process the parser results. In two steps, the automatic domain extraction system transforms the natural language input into a domain model, an instance of the given meta-model. The effectiveness of this method relies on the assumption that the natural language utterances have a reasonable "fit" with a predefined, given meta-model. Effective concept extraction rules can then be derived from this meta-model and the output format of the parser.

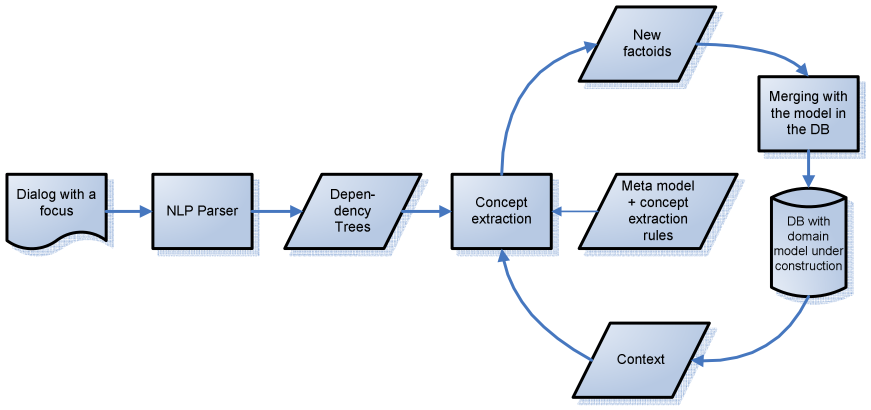

New factoids
Merging with the model in the DB
Dialog with a focus
NLP Parser
Depen-dency Trees
Concept extraction
Meta model + concept extraction rules
DB with domain model under construction
Context

Fig. 1. Method for Automatic Model Extraction

The method for automatic domain model extraction is shown in Figure 1. A transcript of a dialogue is provided as input to the system. A robust *dependency parser* is used to transform the utterances into so-called *dependency trees* (see below for an explanation). The dependency trees are input to a pattern matching module which is able to take the context of a tree (representing one or more factoids) into account, e.g. for resolving pronoun references. Finally, so-called *concept extraction rules* are used to extract a concrete instance of a domain model. These rules are fairly simple pattern matching rules derived from the generated parser output and the meta-model.

3 Dependency Trees and Dependency Triplets

All utterances are parsed using the EP4IR grammar of English [7], [8], normalized, transduced to dependency trees, and unnested to dependency triplets. By a dependency tree (DTree) we mean a graph (a tree with possibly some confluent arcs) whose nodes are marked with words and whose arcs are marked with certain syntactic relations. A dependency tree obtained from an utterance represents the most important syntactic relation in the utterance: SVOC (Subject/Verb/Object/Complement) trees and NP (Noun Phrase) trees. The SVOC trees correspond to the *factoids* (who is said to do what to whom under what circumstances) expressed by the utterance. The following dependency tree shows the typical structure of the attributed noun and of the SVOC-sentence.

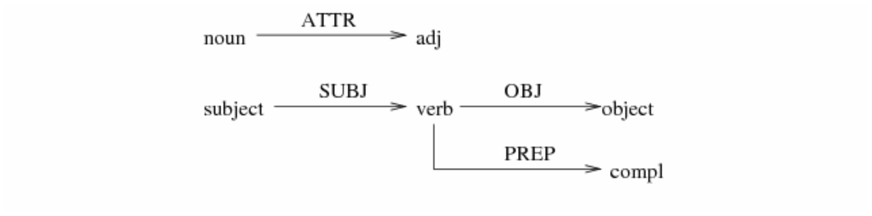


Fig. 2. Example of a dependency tree

By a dependency triple (DT) we mean a triple (word, relation, word), which forms part of a dependency tree, from which it can be obtained by *unnesting* the tree. DT's are the building-stones that constitute factoids. There is a long history of the use of DT's and the related *head/modifier pairs* [9] in Information Retrieval.

A dependency tree gives an abstract view of the structure of a sentence in terms of well defined syntactic word relations from which semantic relations can be derived relatively easily. A dependency tree is much more compact and abstract than a constituent tree (parse tree).

The parsing process takes into account the subcategorization frames of verbs, nouns and adjectives, as well as the verb valences. The words occurring in the DTs are lemmatized. The following table shows the most important dependency relations, together with their concrete notation as a DT and an example:

Table 1. Dependency relations

subject relation	[noun,SUBJ verb]	[picture,SUBJ show]
object relation	[verb,OBJ noun]	[show,OBJ view]
attrib relation	[noun,ATTR noun]	[theatre,ATTR movie]
attrib relation	[noun,ATTR adje]	[monument,ATTR large]
predicative relation	[noun,PRED noun]	[Louvre,PRED museum]
prepos relation	[noun,PREP noun]	[sword,IN hand]
prepos relation	[verb,PREP noun]	[sit,ON chair]
prepos relation	[adje,PREP noun]	[full,OF arrows]
modification	[adje,MOD advb]	[green,MOD intensely]
modification	[verb,MOD advb]	[cause,MOD not]
quantification	[noun,QUANT number]	[horse man,QUANT three]
determination	[noun,DET determiner]	[scene,DET whole]

As an example, the sentence 'the picture shows a view of Ravenna from the air' corresponds to the following dependency tree:

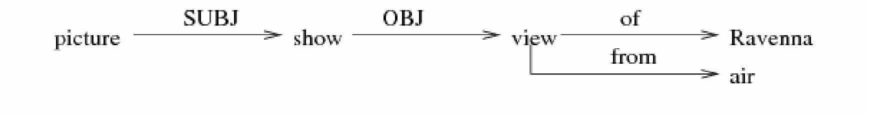


Fig. 3. Dependency tree for 'the picture shows a view of Ravenna from the air'.

The example 'the picture shows a view of Ravenna taken from the air' is transduced to two (connected) Dependency Trees [10]:

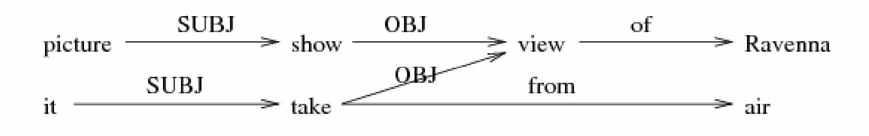


Fig. 4. Dependency tree for 'the picture shows a view of Ravenna taken from the air'.

The subject 'it' in the second DTree is just a handle for anaphora resolution. During the transduction, extensive normalizations are performed in order to map equivalent phrases onto a common representative: variations in word order, time and modality are eliminated, questions and passive sentences are translated to active form (see [9], [10]). Finally, the words in the DT's are lemmatized. The EP4IR parser/transducer was developed for application in Information Retrieval [11]. Our paper shows that it can also be used successfully for Domain Modeling.

4 Extracting a Domain Model

In general, it will not be possible to match the dependency tree output of the parser one-on-one with a given meta-model. The natural language parser, however, does provide a well-structured and well-defined output that can be used in a final domain extraction phase. The key idea of this final phase is to match parts of a dependency tree with parts of the desired domain model.

The meta-model determines the structure of the desired domain model as well as that of the extraction rules that are used in the extraction phase. The meta-model consists of the key concepts that need to be extracted from the natural language text. Of course, the meta-model should have a reasonable fit with the natural language text. As discussed above, a reasonable fit can be obtained by using structured dialogue games to produce the text.

In the prenegotiation domain, which provides the running example of this paper, a meta-model of the domain of negotiation needs to be instantiated in order to fix the negotiation issues. As Raiffa discusses in [16], parties are advised to prepare a negotiation template in this prenegotiation phase. Such a template has a simple structure. It consists of a list of issues that need to be resolved, and, for each issue, an agreed-upon set of possible resolutions.

In a negotiation about multiple issues, the result of the domain extraction method should be an instance of the meta-model depicted in Fig. 5. Basically, objects and their properties need to be extracted from the dependency trees.

The rules for extracting domain elements have to capture those patterns present in a dependency tree that with a high probability indicate that the text is about an object or a property (or both). By inspection of the relations listed in Table 1, and dependency trees (cf. Fig. 3 and Fig. 4) that result from typical dialogue games, various patterns are readily suggested.

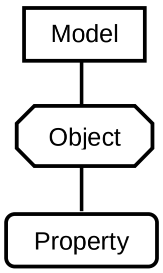
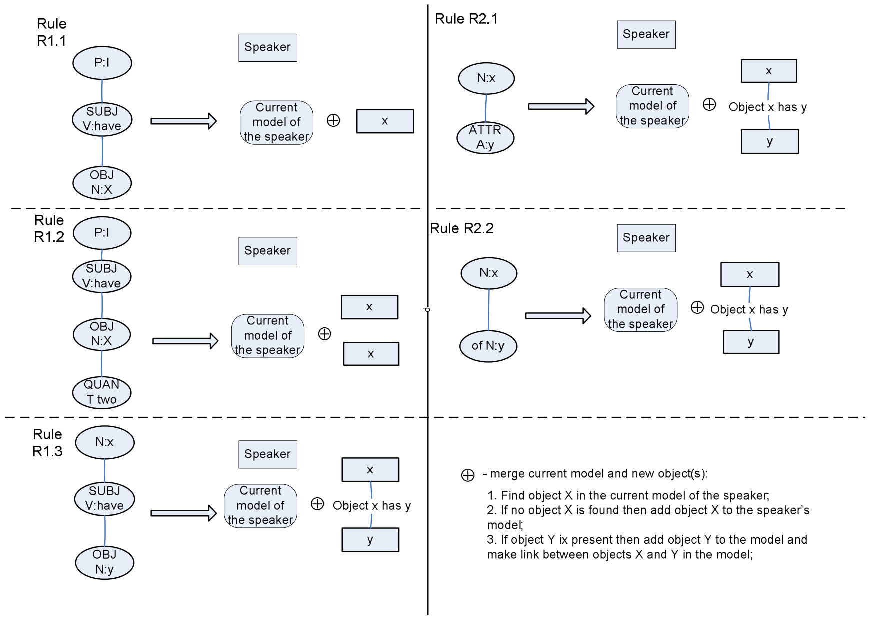


Fig. 5. Structure of the Negotiation Meta-Model

In the dialogue games that we have used in our experiments, typical patterns are, for example:

1. [pro: I, SUBJ, verb: have, OBJ, noun: *x*],
2. [noun: *x*, SUBJ, verb: have, OBJ, noun: *y*],
3. [noun: *x*, ATTR, adje: *y*].

An instance of the first pattern is, for example, a sentence such as *I have a daisy*. It is clear that such a pattern requires the addition of the object named *daisy* to the domain model. The first pattern is also a sub-pattern of the slightly more complicated sentence *I probably have a daisy*, which is an instance of the pattern: [pro: I, SUBJ, verb: have, OBJ, noun: daisy, MOD, advb: probably]. Even though this sentence indicates that there is a chance the object is *not* a daisy, the pattern is processed by



⊕ - merge current model and new object(s):

1. Find object X in the current model of the speaker;
2. If no object X is found then add object X to the speaker's model;
3. If object Y ix present then add object Y to the model and make link between objects X and Y in the model;

Fig. 6. Domain Extraction Rules

adding the object named *daisy* to the domain model.[1] An instance of the second pattern is e.g. *The cup has a handle*. Finally, an instance of the third pattern is *The cup is blue*. In the latter case, a property of being *blue* needs to be added to the model.

The conception extraction rules should map such patterns onto domain elements, where the domain structure is given by the meta-model. The basic structure of a conception extraction rule therefore is defined as:

<subpattern of dependency tree> à <update instruction(s) for domain model>.

The rules code instructions for extracting domain elements from a dependency tree in case the left-hand side of a rule matches with a sub pattern of the tree.

The process of domain extraction can be summarized as follows (cf. also Fig. 5, and 6). The pattern matching module of the domain extraction system tries to match the left-hand side of each concept extraction rule. For each match, the resulting bindings of the matching process are retrieved and the instructions (properly instantiated) on the right-hand side of the rule are executed. These instructions consist of *adding a new node to the domain model*, *adding a property together with the related object to the domain model*, and *merging the extracted information with the domain model* (in case a property of an object needs to be added but the object is already present in the model). The primitive operations that are performed on a domain model are *add_node* and *add_edge* operations. The domain extraction module thus also performs merging of overlapping models that are extracted from different sentences of a single dialogue.

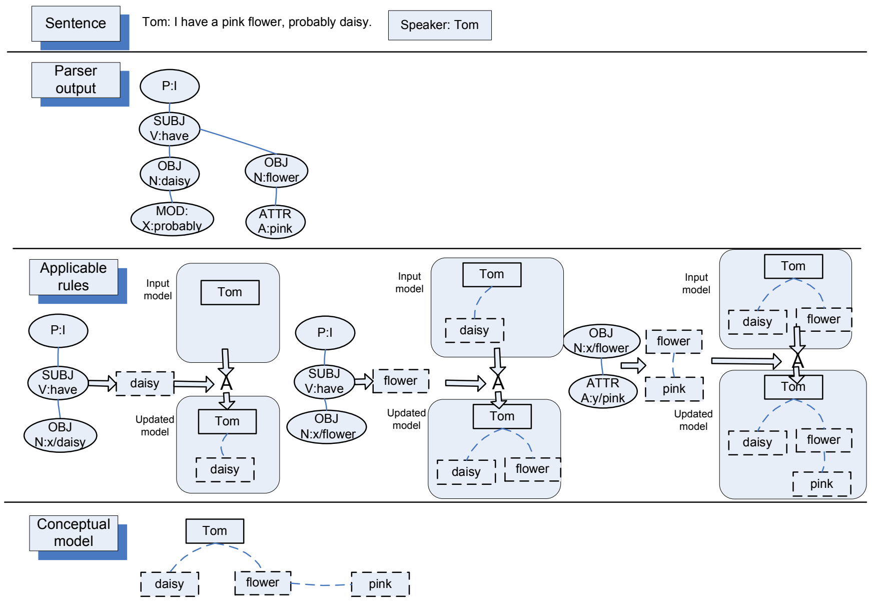


Fig. 7. Example of Domain Extraction with the Rules

[1] Depending on the application area such rules can be changed to not allow this.

5 Experimental Validation

The proposed domain extraction method has been designed in order to facilitate humans in the construction of a domain model. For the running example, a specific meta-model was used to illustrate the extraction method. In order to validate the method proposed in the previous section, a series of experiments with human subjects was performed to measure the effectiveness of the method. For negotiation and its corresponding meta-model, a dialogue game is needed that results in a descriptive natural language text that is focused on the naming of objects and the identification of properties of these objects. Such a game can be viewed as a model of a domain modeling task in which a knowledge engineer and a domain expert are trying to construct a domain model.

In line with a general view on domain modeling as expressed in [5], the experiment was organized as a dialogue game (taking the form of a chatbox) played by two participants seated in different rooms, who were each presented with a set of pictures on a screen (some identical, some different). Figure 8 presents a screenshot of the chat box software used to organize the experiment. The participants were asked to discuss the objects displayed in the pictures (each participant could only see his/her own set of pictures). The participants were given the task to find out which of the

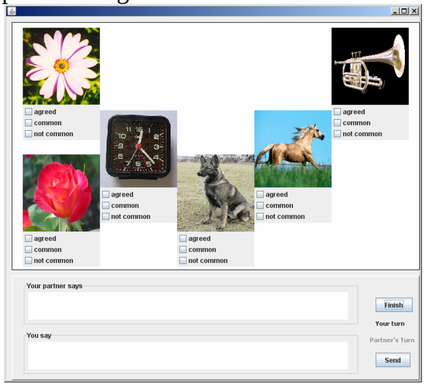


Fig. 8. Screenshot of the Chatbox Software used in the Experiment

objects are present on both sets of pictures (i.e., they had to identify the objects that are common, meaning that both participants see exactly the same pictures of those objects on their screens). This setup requires the participants to go through an elicitation phase as defined earlier. For the purpose of validation, the resulting dialogues were processed in two ways: by the automatic domain extraction tool and independently, by a knowledge engineer who manually created a domain model.

For manual domain modeling, the knowledge engineer was given a particular dialogue as a domain description, but the engineer had no access to the pictures that were presented to the participants in the dialogue. In this way, the engineer was limited to basing the domain model on the content of the dialogue. As a result, he added an object or its property to the model only if it was explicitly mentioned in the dialogue. For example, if a dialogue included a statement such as “Participant A: I have a pink flower” the knowledge engineer would add an object “flower” to the domain model and a property "pink" linked to the object "flower". The domain model obtained in this way has been used as the standard (or "ideal") domain model against which the results from automatic extraction were then compared.

To compare the ideal domain model and the automatically extracted model, the A* Algorithm for Error-Correcting Subgraph Isomorphism Detection [15] was used. Observe that domain models are graphs and thus can be provided as input to the algorithm. The algorithm calculates the similarity distance between two graphs and is based on the idea of compensating the distortions in one graph by means of edit operations that are applied to the second graph.

The edit operations include vertex deletion and insertion, edge deletion and insertion, and attributes and labels substitution. All edit operations have equal cost. The total cost of the transformation of the graph is the sum of the costs of each individual edit operation. The A* algorithm looks for a sequence of edit operations that would have the minimal total costs of the transformation.

The following formula determines the correctness of the extracted model:

$$c = \left(1 - \frac{d\left(g_{\exp ert}, g_{extracted}\right)}{d\left(g_{\exp ert}, \phi\right)}\right) \cdot 100\% \qquad (1)$$

where $d(g_{expert}, g_{extracted})$ is the distance between the domain model extracted by expert and the domain model automatically extracted by the tool, and
$d(g_{expert}, \varnothing)$ is the distance between the domain model extracted by the expert and the empty graph.

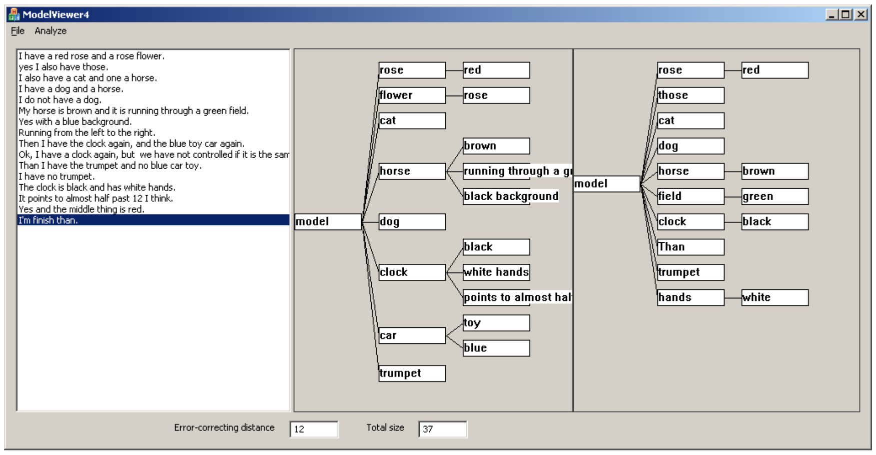
Table 2 presents the results of the validation of the series of experiment. Each of the eight pairs of the participants performed eight trials. Each trial has a set of six pictures. Two pictures out of six are common for the participants. We varied the sets of the pictures among the trials through the pairs of the participants to avoid any possible side-ways effects of the trials sequence.

Table 2. Experimental results – correctnes of the automaticaly extracted domain models

Experiment	Sets of pictures							
	1	2	3	4	5	6	7	8
Pair 1	40%	49%	46%	45%	51%	68%	69%	57%
Pair 2	50%	63%	58%	65%	77%	68%	67%	55%
Pair 3	68%	41%	43%	51%	68%	72%	49%	65%
Pair 4	56%	54%	41%	43%	63%	61%	65%	54%

The average percentage of the correctness of the extracted models is 57%.

The experimental results show that the precision of the model extraction still needs significant improvement. However, note that the models were extracted without any use of semantics. One way of improving the accuracy of the models is to use domain



```
I have a red rose and a rose flower.
yes I also have those.
I also have a cat and one a horse.
I have a dog and a horse.
I do not have a dog.
My horse is brown and it is running through a green field.
Yes with a blue background.
Running from the left to the right.
Then I have the clock again, and the blue toy car again.
Ok, I have a clock again, but  we have not controlled if it is the sam
Than I have the trumpet and no blue car toy.
I have no trumpet.
The clock is black and has white hands.
It points to almost half past 12 I think.
Yes and the middle thing is red.
I'm finish than.
```

Error-correcting distance 12   Total size 37

Fig. 9. Screenshot of the automatic domain extraction tool (from left to right: source dialogue, manually extracted reference model, automatically extracted model).

knowledge available, e.g., WordNet [4], CYC [13]. Accuracy might be improved by adding more rules to the dialogue game to structure the natural language produced. Another way is to make the modeling process interactive by presenting the updated instance of the domain model while the user continues his/her descriptions in natural language. Since the user immediately sees the interpretation of his words s/he can re-formulate his/her sentence if necessary.

6 Conclusions

This article presents an automatic domain model extraction method based on a predefined meta-model. Our method involves two basic steps: focused elicitation where domain experts describe the domain in a natural language dialogue and automated extraction based on an existing NLP parser and a set of pattern-matching rules to extract the basic concepts of the domain. The output of the proposed method has been validated against ideal models build manually by a domain expert using the dialogues received from the experimental setup.

Validation results show a big deviation in the accuracy of the domain model extraction. The accuracy metric varies from 40% to 77% throughout the experiments, generally in correspondence to the “neatness” (complexity) of the sentences produced by the participants. In future work a sentence complexity evaluation algorithm will be developed using the parser output to assess quality of the domain elicitation. The accuracy of the approach will be improved: by involving the domain experts in a more direct way and by presenting them continuously with the models extracted. This allows the human to directly correct the system if necessary. Furthermore, the humans will be asked to reformulate if the parser has difficulties with the sentences produced. Finally, advanced pattern matching rules will be used, that are based on semantic knowledge obtained from the Internet, a specialized database or existing ontologies.

Acknowledgement. The authors thank Kees Koster (Radboud University Nijmegen) for his help in the production of this paper and in particular for his effort in making the EP4IR parser available for the reported research.

References

1. Anthony, S., Batra, D., and Santhanam, R.: The use of a knowledge-based system in conceptual data modeling. Decision Support Systems, 41:176–190,2005.
2. Batra, D. and Antony, S.: Consulting support during conceptual database design in the presence of redundancy in requirements specifications: an empirical study. IBM Journal of Research and Development, 54:25–51, 2006.
3. Elfatatry, A., Layzell, P.: A negotiation description language. Software—Practice & Experience, Vol. 35(4), pp. 323-343, 2005.
4. Fellbaum, C., WordNet: an Eleectronic lexical database, MIT Press, 1998.
5. Hoppenbrouwers, S.J.B.A., Proper, H.A., and van der Weide, Th.P.: “A Fundamental View on the Process of Conceptual Modeling". In: Proc. of 24th International Conference on Conceptual Modelling, Lecture Notes in Computer Science vol. 3716, 2005.
6. Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, Th.P.: Towards explicit strategies for modeling. In: Halpin, T.A., Siau, K. and Krogstie, J. (eds), Proc. of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD`05), p485-492. FEUP, Porto, Portugal, EU. ISBN 9727520774
7. Koster, C.H.A.: Affix Grammars for Natural Languages. In: H. Alblas and B. Melichar (Eds.),Attribute Grammars, applications and systems.SLNCS 545,Heidelberg,1991,p.469-484.
8. Koster, C.H.A., Verbruggen, E.: The AGFL Grammar Work Lab, Proc. FREENIX/Usenix 2002, pp 13-18.
9. Koster, C.H.A.: "Head/Modifier Frames for Information Retrieval". Proceedings CICLing-2004, Springer LNCS 2945, pp 420-432.
10. Koster, C.H.A.: Transducing Text to Multiword Units, Workshop on MultiWord Units MEMURA at the fourth International Conference on Language Resources and Evaluation, LREC-2004. Lisbon, Portugal, 8 pp.
11. Koster, C.H.A., Seibert, O., Seutter, M.: The PHASAR Search Engine. In: Proceedings of NLDB 2006, Springer LNCS 3999, pp. 141-152, 2006.
12. Krogstie, J., and Jorgensen, H.D.: Quality of Interactive Models. Conceptual Modeling - ER 2002, 21st International Conference. Lecture Notes in Computer Science, Vol: 2503, Pages: 351-363, Springer, Berlin, Germany, EU, 2002.
13. Lenat, D., Guha, R. V.: Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley, 1990.14
14. McBurney, P., Parsons, S.: Dialogue Games in Multi-Agent Systems, Informal Logic. Special Issue on Applications of Argumentation in Computer Science.22(3), pp.257-274, 2002.
15. Messmer, B.T., "Efficient Graph Matching Algorithms for Preprocessed Model Graphs," PhD thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, Switzerland, 1995.
16. Raiffa, H., Lecture Notes on Negotiation Analysis, Harward University, PON Books, 1996.
17. Shoval, P., Danoch, R., and Balaban, M.: Hierarchical ER Diagrams (HERD) - The Method and Experimental Evaluation. LNCS 2784, p264-274 (2003). Berlin/Heidelberg: Springer.
18. Web Ontology Language (OWL), http://www.w3.org/2004/OWL/.