From Requirements Documents to System Models: a Tool for Interactive Semi-Automatic Translation

Leonid Kof
Fakultaet fuer Informatik, Technische Universitaet Muenchen,
Boltzmannstr. 3, D-85748, Garching bei Muenchen, Germany
kof@in.tum.de

***Abstract*—Natural language is the main presentation means in industrial requirements documents. This leads to the fact that requirements documents are often incomplete and inconsistent. Despite the fact that documents are mostly written in natural language, natural language processing (NLP) is barely used in industrial requirements engineering.**

The presented paper shows, how a natural language processing (NLP) approach can be integrated in a CASE tool. This enables the integrated tool to learn on the fly, which grammatical construction represents which model element. A valuable side effect of the proposed integration is tracing between the textual document and the constructed model. The presented integrated tool was evaluated in several case studies that have shown practical applicability of the tool.

***Keywords*-requirements analysis, system modeling, natural language processing**

I. Introduction: from Requirements to Models

At the beginning of every software project, some kind of requirements document is usually written. The majority of these documents are written in natural language. This results in the fact that the requirements documents are imprecise, incomplete, and inconsistent. It is one of the goals of requirements analysis, to find and to correct the deficiencies of requirements documents. A practical way to detect errors in requirements documents is to convert informal specifications to system models. In this case, errors in documents would lead to inconsistencies or omissions in models, and, due to more formal nature of models, inconsistencies and omissions are easier to detect in models than in textual documents.

The goal of the presented paper is to show, how a natural language processing (NLP) approach can be integrated in a CASE tool (AutoFOCUS, http://af3.in.tum.de/) and enriched with machine learning techniques, so that the integrated system provides text-to-model translation without constraining the allowed natural language. The main benefit of the integration of an NLP approach is not the extracted model itself, but the extraction procedure: the inmplemented procedure is interactive, which contributes to thorough exploration of the requirements document.

II. Interactive Text-to-Model Translation

A. Training Data Collection: Manual Modeling

The prerequisite for text-to-model translation is a training data set, which is gathered by the means of manual modeling. From the point of view of user interaction, the tool distinguishes two types of model elements:

- entities, characterized solely by their names, and
- relations, characterized both by their names and the entities involved in the relation.

For example, a state of an automaton is an entity, as it is characterized solely by its name, but a state transition is a relation, as it is characterized by its name, and two states involved in the transition.

To create a model element, the tool user has to mark a word sequence in the text field showing the requirements document, and then to select the model element type from the context menu. The marked word sequence becomes the name of the newly created model element. The tool creates the model element and asks the user to select the container for the newly created model element. For relations, additionally to the container, the tool asks the user to select the entities involved in the relation to be created. For example, for a state transition the tool asks the user to select the source and the target state.

When creating a new model element, the tool creates an explicit trace between the model element and the previously marked word sequence. Every trace specifies the sentence, the exact position of the word sequence used to create the model element name, and the model element itself. These traces are used later as training data items for the automatic text-to-model translation.

B. Automatic Extraction of Entities

From the point of view of the tool user, extraction of entities looks similar to the manual creation of model elements: the user selects a document section, and then chooses from the context menu, which model element type should be extracted. Then, the tool proposes the names of potential new model elements to the user. The user selects, which names should be used to create new model elements, and, analogously to the manual creation, selects the container for the newly created model elements.

Internally, the tool keeps track of the user decision and augments the training data: for every model element name selected by the user, the tool creates a trace from the name occurrence in the text to the newly created model element, and makes the created trace to a positive example (new training data item). Every not selected name becomes a negative example and is not presented to the user again, in order that the user is not bothered with already declined suggestions. The tool does not

This work was partially funded by the Deutsche Forschungsgemeinschaft (German Research Foundation), grant “Inserve III, BR 887/19-3”

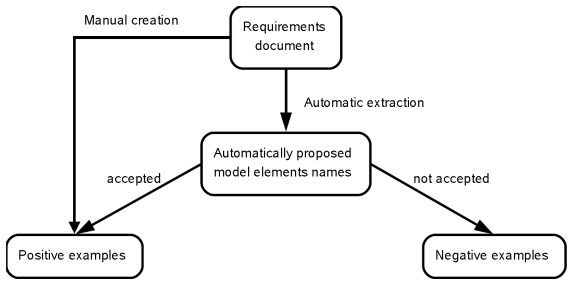

Figure 1. Training data management

distinguish between manually created and automatically extracted model elements and puts them in the same sets of positive/negative examples, as shown in Figure 1. Thus, the tool does not force the user to create all the training data first, but, instead, allows for arbitrary interleaving of manual modeling and automatic extraction.

For automatic extraction of model element names, the tool performs two steps: (1) for every training data item, the tool decides, which of the built-in heuristics (named entity, sentence predicate, subject, ...) is most suitable for the extraction of this data item, and then (2) applies the heuristics, that is used for the most training data items, to the whole document section. For a training data item, the heuristic that extracts a string most close to this data item is considered most suitable.

C. Automatic Extraction of Relations

In order to extract relations (for example, state transitions), it is necessary not only to extract the name of the relation, but also to determine the entities involved in the relation. (E.g., in the case of state transitions, it is necessary to determine the involved states.) In order to find the entities involved in a relation, the tool uses a simple discourse model. The prerequisite for the usage of this discourse model is a LaTeX-like structure of the input requirements document: the document must consist of sections, that, in turn, consist of subsections, etc.

For training, the tool takes a set of relations, and for every relation it considers the involved entities. Then, for every relation and every involved entity, the tool determines the shortest path from the relation name to the entity name. "Shortest path" means that the structured requirements documents is considered as a graph (tree), and the tool searches for the shortest path in this graph. The leaves of this graph are single sentences, and its inner nodes are sections and subsections.

In order to extract relations, the tool extracts relation names first, using the procedures presented in Section II-B. Then, for every relation name that was accepted by the user, the tool looks for entities potentially involved in the relation: It follows text paths obtained from the training data and searches for the names of entities already existing in the model. The found entities, potentially involved in the relation, are presented to the user. The user selects, which entities are really involved in the relation, and after that the tool creates the relation, using the extracted relation name and the selected entities.

	Question	Answers
1	Are all the model element names suggested by the automatic extractor represent genuine model elements?	9×no
2	Does the extractor suggest model element names representing genuine model elements that you overlooked when reading the text?	6×yes 1×maybe 2×no
3	In your opinion, does the extractor speed up the modeling process?	1×no answer 7×maybe 1×no
4	What model elements are explicitly specified in the text, but not contained in the model?	Different entities and relations listed (subject-dependent).

Table I
EVALUATION RESULTS

III. EVALUATION

The tool was evaluated in two stages: firstly, the tool author used the tool in order to translate four specifications to system models. The Steam Boiler specification [1], Autopilot [2], Bay Area Rapid Transit (BART) [3], and Instrument Cluster [4] were used for this purpose. For all four specifications, the tool kept up with the expectations of the tool author.

Secondly, the tool was evaluated with independent subjects. The subjects (9 PhD students in computer science) were asked to use the tool and to model the Steam Boiler and the Instrument Cluster, as these two specifications contain concepts that can be directly modeled in AutoFOCUS. After the modeling session, every subject was individually (not in a group) asked to answer the questions about the tool. The questions and answers are presented in Table I.

The models produced by the subjects were highly different, but every model represented a valid interpretation of the corresponding requirements document. It shows that it is not possible to produce *the* model in the early modeling phase. To summarize, the evaluation showed that the translator fulfills its main purpose, namely to support the exploration of the requirements document.

REFERENCES

[1] J.-R. Abrial, E. Börger, and H. Langmaack, "The steam boiler case study: Competition of formal program specification and development methods," in *Formal Methods for Industrial Applications*, ser. LNCS, J.-R. Abrial, E. Borger, and H. Langmaack, Eds., vol. 1165. Springer, 1996.

[2] R. W. Butler, "An introduction to requirements capture using PVS: Specification of a simple autopilot," NASA Langley Research Center, Hampton, VA, NASA Technical Memorandum 110255, May 1996.

[3] V. Winter, F. Kordon, and M. Lemoine, "The BART case study," in *Formal Methods for Embedded Distributed Systems*, F. Kordon and M. Lemoine, Eds. Springer Netherlands, 2004, pp. 3–22.

[4] K. Buhr, N. Heumesser, F. Houdek, H. Omasreiter, F. Rothermehl, R. Tavakoli, and T. Zink, "Daimler-Chrysler demonstrator: System specification instrument cluster," 2004, http://www.empress-itea.org/deliverables/D5.1_Appendix_B_v1.0_Public_Version.pdf.