# Preprocessing of Requirements Specification

**Conference Paper** · January 2000

DOI: 10.1007/3-540-44469-6_63 · Source: DBLP



**1 author:**

Petr Kroha
Technische Universität Chemnitz
**75** PUBLICATIONS **303** CITATIONS

# Preprocessing of Requirements Specification

Kroha,P.

Technical University of Chemnitz,
Department of Informatics, 09107 Chemnitz, Germany

**Abstract.** In this paper, we discuss the part of the requirements specification process which is located between the textual requirements definition and the semi-formal diagrams of the requirements specification. It concerns the acquisition and the refinement of requirements and the consensus improvement between the analyst and the user. We argue that the existence of a textual description of requirements that represents the analyst's understanding of the problem will improve the efficiency of the requirements validation by the user. A method and a tool are introduced that support the acquisition, refining, and modeling of requirements, and finally the automatic generation of a textual description of the model for the validation.

## 1 Introduction

To get a contract for a large software project, the software house has to present a requirements definition. Its purpose is to describe all features of the new system that are necessary for the customer to sign the contract. Using a natural language is necessary because a customer would not sign a requirements definition written e.g. in the Z notation.

Once a contract has been awarded and after a feasibility study has been approved, a requirements specification must be written that describes the functionality and constraints of the new system in a more detailed way. It will usually be written in some semi-formal graphical representation given by the used CASE tool. Since software engineers are not specialists in the problem domain, their understanding of the problem is immensely difficult, especially if routine experiences cannot be used. It is a known fact [2] that projects completed by the largest software companies implement only about 42 % of the originally-proposed features and functions.

We argue that there is a gap between the requirements definition in a natural language and requirements specification in some semi-formal graphical representation. The analyst's and the user's understanding of the problem are usually more or less different when the project starts. The step from the requirements definition towards requirements specification will be done only in the brain of the analyst without being documented. Usually, the user cannot deeply follow the requirements specification. The first possible time point when the user can validate the analyst's understanding of the problem is when a prototype starts to be used and tested.

We offer a textual refinement of the requirements definition which can be called requirements description. Working with it, the analyst is forced by the supporting tool TESSI to complete and explain requirements and to specify the roles of words in the text in the sense of the object-oriented analysis. During this process, a UML model will automatically be built by TESSI driven by the analyst's decisions. This model will be used for the synthesis of a text that describes the analyst's understanding of the problem, i.e. a new, model-derived requirements description will automatically be generated. Now, the user has a good chance to read it, understand it and validate it. His/her righting comments will be used by the analyst for a new version of the requirements description. The process repeats until there is a consensus between the analyst and the user. This does not mean that the requirements description is perfect, but some mistakes and misunderstandings are removed.

We argue that the textual requirements description and its preprocessing by TESSI will positively impact the quality and the costs of the developed software systems because it inserts additional feedbacks in the development process.

The rest of the paper is organized as follows. In section 2, we discuss the related work. In section 3, the presented method will be described in detail. In section 4, we mention the XML-interface to the tool Rational Rose. Section 5 describes advantages of our method for the software development. The last section discusses the implementation and conclusions.

## 2 Related work

There are many good books on requirements engineering, e.g. [15]. Specialized papers related to our topic try either to filter some semantic information from a text (analysis) or to generate text from some diagrams (synthesis). We have not found any papers that combine these concepts to build a feedback in requirements specification.

To obtain semantic information directly from a text, manual or automatic methods can be used. There are many papers and books [1] that support manual methods. For example, the method of grammatical inspection of a text can be mentioned. It analyzes the narrative text of use cases, identifies and classifies the participating objects [5]. In this method, nouns will be held for candidates for objects or classes, adjectives will be held for candidates for attributes, and verbs will be held for candidates for methods or relationships. The Object Behaviour Analysis [16] belongs to this group of methods, too. Nevertheless, we have not had any opportunity to use a CASE tool which supports these methods.

There are some tools helping in requirements management (DOORS, Requisite Pro) that transform unstructured texts of requirements into structured texts. They neither build any models nor generate refining questions like our tool TESSI.

Automatically generated natural-language description of software models is not a new idea. The first system of this kind was described in [18]. More recent projects include the ARIES [6] and the GEMA data-flow diagram describer [17].

Even if it is widely believed that the graphical representation is the best base for the communication between the analyst and the user, there are some serious doubts about it. It has been found in [7] that even a co-operation between experienced analysts and sophisticated users who are not familiar with the particular graphical language (which is very often the case) results in semantic error rates of about 25 % for entities and 70 % for relations. Similar results brings [14]. Some systems have been built which transform diagrams, e.g. ER-diagrams, into the form of a fluent English text. For example, a system MODEX has been described in [12], [13] which can be used for this purpose. Differently to our system, input to MODEX (MODel EXplainer) is based on the ODL standard from the ODMG group.

The first version of our system TESSI has been introduced in [10]. Additionally to MODEX, it supports the manual analysis of the requirements description by the analyst and the semi-automatic building of an OO-model in UML. Differently from MODEX, we have used simpler templates for generating the output text, because our focus was not in linguistics. The current version of TESSI generates refining questions and uses an XML-interface to Rational Rose.

## 3 Preprocessing of Requirements

The acquisition of requirements is a very important part of software engineering. It mostly consists of interviews with users about the problem domain and their use cases. We argue that

- this process should be supported by tools,
- the best way to represent the obtained information at this level of knowledge is not a diagram as commonly accepted but a text.

After the first version of requirements description has been written by the analyst, it will be analyzed by him/her (supported by our tool TESSI - TExtual aSSIstent) to build an object-oriented model and to validate it in a co-operation with a user.
This process can be divided into:

- model identification
  - identification of static features supported by generated questions,
  - identification of dynamic features supported by generated questions,
- model validation
  - automatic generation of a text that corresponds to the identified object-oriented model,
  - validation of this generated text by the user,
  - revision and correction of the last version of the requirements description by the analyst,
  - evaluation whether the whole process should be repeated.

During the model identification that is based on the grammatical inspection, TESSI generates complementary questions to force the analyst not to forget to describe some specific features of the system. The validation of requirements supported by the automatic generation of the model-derived requirements description is the next step [10]. The generated requirements description represents the analyst's understanding of the problem.

We argue that the comparison of the analyst's understanding and the user's understanding provides an important feedback. The generated text will be read by both the user and the analyst. It will be compared with the original requirements description in order to approve it or to correct/complement it. The corrected original requirements description (resp. its corrected and newly appended parts) will be then analyzed again by using the same method.

This process will be repeated while the user and the analyst see any lacks and suspicious formulations in the generated requirements description. After consensus, the current UML model represents the starting point for development of the functional requirements specification and the prototype.

### 3.1 Identification of the model

The analyst's decision about the role of words in the textual requirements description is one of the focal points of the presented method and it is supported by interface menus of TESSI. We do not believe that such a decision can automatically be made without a human interaction. The semantic vagueness of natural languages and the context dependence are too hard problems.

TESSI supports object-oriented modeling in UML. For modeling static features, we use class diagrams and the concepts like class attribute, class operation, class cardinality, generalization, association, and aggregation. For modeling dynamic features, statechart diagrams and sequence diagrams have been supported until now. In a requirements description, a word can be irrelevant, it may denote a candidate for an object, a class, an attribute, a relation, a method, a constraint [1], [10].

Each decision of the analyst causes changes (usually APPEND) in internal data structures representing the model. They have bi-directional pointers to the corresponding source text segments and to templates for generating refining questions (see section 3.2).

The process of identification of dynamic features is based in principle on Object Behaviour Analysis [16].

### 3.2 Questions and their templates

Observing, asking questions and looking for answers is in principle the only method how to investigate. In common, it is a creative activity, but in a specific scope of requirements acquisition, there are many known standard questions that can mechanically be asked and that must be answered to discover an essential subset of classes, their attributes, methods, relations, etc. We distinguish two kinds of questions that differ in how the answer is intended to be used.

Expanding questions will be used when writing requirements, i.e. at the begin of the iterative process. At this level of knowledge we hope to find new relevant words in answers that would help us to identify new entities that are not contained in the existing text. We have used templates like "What is the purpose of <placeholder> in the system?", "What is the role of <placeholder> in the system?", "What is the responsibility of <placeholder> in the system?". These questions will be triggered after the context of a sentence is left for classes identified in the sentence. The analyst can ignore the question if he/she can find the answer in the next text.

Example:
Starting text: An information system for a library shall be written.
The analyst identifies LIBRARY as a candidate for a class and will be asked: "What is the purpose of the LIBRARY in the system?" There are many possibilities how to answer. An experienced analyst answers, for example, "In a LIBRARY librarians lend books to readers." Even if this is an answer at a children level it helps to refine the system. An unexperienced analyst could answer: "A LIBRARY improves the cultural level of the nation." This is not completely wrong, but it is absolutely useless for requirements acquisition.

(End of example)

Completing questions will be generated to complete the existing text of requirements description. As a last step of the identification process, TESSI checks whether all attributes of the internal modeling data structures containing the object-oriented model are completed. Every not completed attribute triggers a question generation. Because of the bi-directional pointers, the context of such a question can be found and shown. The analyst checks the reason, inserts (usually) an additional sentence (or more) to the requirements description that obtains the missing information, and completes the model.

Of course, the generated questions do not guarantee any completeness of the model, because even if TESSI generates many questions, it cannot check the correctness of the answers. This means that the generated questions remind the analyst on some, maybe forgotten, modeling features.

As far as the implementation is concerned, the question templates build a set of fixed pre-built questions that are parameterized by names taken from the associated data modeling structures. Templates for generating questions and templates for generating the text description of the model are associated with internal data structures in which TESSI stores the identified entities. Templates are strings with placeholders in the current implementation of TESSI. Before generating, placeholders will be substituted by names of entities described in the associated data structures.

In MODEX [13], templates are complex structures representing not only the contents of the sentence but also its grammatical structure in English.

### 3.3 Transformation of the model into a structured text

After the analyst's processing of the requirements description has been finished, there is a complete UML model available in the internal data structures. TESSI can automatically generate the textual description of this model using a one-to-one mapping between entities of the model and textual templates. This generated version of the requirements is equivalent to the UML model and describes how the analyst understands the problem. Denoting the UML model as a component of requirements specification, the generated text can be denoted as a textual description of this component of requirements specification.

The text generation starts at the root (roots) of the class hierarchy. After the structure of the class hierarchy has been used for the text generation, the description of specific classes and their static features will continue. The description of each statechart diagram will be generated as a part of a class description, the description of sequence diagrams will be generated separately as use cases.

The generated text sounds sometimes unusual, but it contains all features important for understanding. All words which were not marked in the original text will be held for irrelevant and will be omitted in the generated text. New irrelevant words come from the used templates.

A set of templates can be constructed for any natural language. It is also possible to define pairs of templates in two (or more) natural languages and generate the requirements description in more than one language. In this case, names of entities (classes, attributes, etc.) should be international because they will not be changed by the templates.

Some linguistic problems occurred that concern the complexity of different natural languages, e.g. irregular plurals, suffixes of adjectives, articles, etc. in the German language. The language-dependent parts have been programmed in separated units. Our focus is not in linguistics, so we did not try to achieve a linguistic perfection.

The generated text will be used for validation of the requirements description by the user as stated above.

### 3.4 Feedbacks in the requirements description

The first feedback will be provided during the identification of static and dynamic features of the UML model. Each identified concept has a corresponding list of question templates that forces the analyst to complete the necessary details, e.g. cardinality of relations, constraints of attributes, etc.. As a result the analyst inserts some sentences into the text of requirements as an answer to the generated question.

The next feedback is represented by the validation of the generated text by the user. In co-operation with the analyst, the user compares the requirements description, his/her knowledge, and the generated text. Discrepancies will be discussed, noticed, and some corrections will be done in the last version of requirements description.

Example:

A sentence in a requirements description: "A table has legs." The analyst recognizes that "table" and "leg" can be candidates for classes. An experienced analyst can recognize that between the class "TABLE" and the class "LEG" could be an aggregation relation "Consists-Of". If he/she does not recognize it, TESSI generates a question: "Is there some relationship between TABLE and LEG?". Such questions will always be generated for names of classes that are included in one sentence without having an identified relationship.

After the analyst identifies the relationship between TABLE and LEG as "Consists-Of", TESSI generates a question about the cardinality of the identified relationship "How many LEGs does a TABLE consists of?". Now, the analyst either knows it or asks the user. Then he/she inserts a sentence "A table consists of four legs." into the text of requirements specification. In this sentence, "four" will be identified as a cardinality "1 to 4" of the relationship "Consists-Of" between the classes TABLE and LEG. It will be noted in internal structures of TESSI that describe the model.

The validation by a user represents the second feedback. This time the user (accompanied by the analyst) compares sentences: "A table has legs" in the original text of requirements description with "A table consists of four legs" in the generated text. The result of this comparison may be that the user recognizes that this description is not complete because a table has also some other parts, e.g. a desk. However, it may be that the existence of a desk is not relevant. Suppose that it is relevant and that the analyst has again forgotten to describe the cardinality and that the new, improved version of the text will be: "A table consists of four legs and a desk." The analysis of this sentence signals an existence of a class DESK. If the analyst does not identify a relationship "Consists-Of" between TABLE and DESK then questions follow: "Is there some relationship between TABLE and DESK?", "How many DESKs does a TABLE consists of?", "Is there some relationship between LEG and DESK?". The last question does not bring anything in this case but it will mechanically be generated, because TESSI does not understand semantics of our world. In this way the iterative process of requirements description develops.

(End of example)

## 4 XML-Interface to Rational Rose

We did not intend to substitute any professional CASE tool. Our motivation was to document the phase of the analyst's thinking during requirements acquisition and modeling. TESSI helps only during the preprocessing of the requirements description and builds a model. It can generate a skeleton of a program corresponding to the UML model in C++ and POET, but it does not draw any diagrams.

As the next step in requirements specification we transfer the validated UML model in XML from TESSI to Rational Rose to design a prototype using all possibilities of Rational Rose.

Currently, an interface to Rational Rose has been implemented [4]. There are some pre-built extensibility possibilities in Rational Rose 4.0 available. The model built by Rational Rose can be accessed through the RREI (Rational Rose Extensibility Interface) using a script language. For Rational Rose, a script has been written that generates (for Rational Rose output) or interprets (for Rational Rose input) the UML model in XML format. In TESSI, the XML format description of the UML model will be generated or accepted, too. This makes the exchange of models between TESSI and Rational Rose possible. The interpretation of the XML format in TESSI has been appended as a separate phase. This interface makes it possible to send the TESSI-model that represents the validated UML model to Rational Rose to draw diagrams and to build a prototype. It is also possible to send UML diagrams (including all corrections) from Rational Rose back to TESSI and to generate the corresponding textual description.

## 5 Advantages for the software development

The preprocessing of requirements not only improves the quality of the analysis (requirements will better be understood) and the quality of the design (main program structures will automatically be created during the analysis and are directly bound to entities of the analysis). All aspects of the software system evolution (adaptive maintenance, new black-box test cases, etc.) will also profit when using the method described above.

### 5.1 Metrics as a side effect

At the very beginning of every project, the customer asks about the costs and delivery time. The preprocessing of requirements brings the possibility to use metrics at this phase of the development. Currently, the most simple object-oriented metric has been used in TESSI. Corresponding to every version of the requirements description, the analyst knows how many classes, methods, attributes, relationships, etc. will probably be used in the description of the system. The costs of the product development, its time schedule, the volume of tests, personal resources, etc. can be derived from this knowledge if enough experience has been collected during similar projects. Additionally, the possible impact of every change in the requirements can be followed during the discussion with the user.

### 5.2 Traceability of requirements

In software evolution and adaptive maintenance, traceability of requirements is very important. When we move from analysis to design, we assign requirements to the design elements that will satisfy them. When we test and integrate code, our concern is with traceability to determine which requirements led to each piece of code. The tool TESSI builds and maintains some important relationships between requirements and parts of the resulted system, e.g. bi-directional

links between the identified entities in sentences of textual requirements and the corresponding entities of the model. These links will be followed during an adaptive maintenance. This helps to hold requirements and programs consistent and supports the concept of software evolution in which every change in the software system should start with the change of the requirements specification and follow the life-cycle of development.

## 6 Implementation and Conclusions

The first prototype without generation of questions was implemented in Turbo-Pascal and Turbo-Vision [10], the improved version in Delphi later on. Except of simple examples (50 - 100 sentences) TESSI has been used for processing requirements of its own which contained several hundreds of sentences. Using TESSI in practice we proved the very known facts:

- Textual requirements description needs some experience.
- Use cases should be organized into groups according to their actors with allocated functional requirements.
- Several iterations are necessary.
- In the generated text, attributes and methods are described, but not the purpose.
- The most important parts of the problem's semantics are usually not included in the first version of the requirements. The reason is that the software analyst doesn't know about their existence and the customer holds them for so much self-evident that he/she doesn't mention them at all.

The current version of TESSI has been implemented in Java [11] and includes the XML-interface between TESSI and Rational Rose 4.0 [4]. This combination of tools offers some very promising possibilities that will be used and tested in the future. The current version has just been finished, so that there was not enough time to get, collect, and evaluate some experience. Currently, we are looking for a co-operation in a project where TESSI will be used in parallel with classical technology in the sense that one group of analysts will directly use Rational Rose and the other group will use at first a preprocessing via TESSI. The goal will be to compare the necessary effort and the quality achieved in both cases.

## References


1. Coad, P., Yourdon,E.: Object-Oriented Analysis. Prentice Hall, 1991.
2. CHAOS Report, The Standish Group, 1995.
3. Emmerich, W.,Kroha, P., Schäfer, W.: Object-Oriented Database Management Systems for Construction of CASE Environments. In: Marik, V. et al (Eds.): Proceedings of the 4th International Conference DEXA'93, Lecture Notes in Computer Sciences, No. 720, Springer, 1993.
4. Gemeinhardt, L.: Connecting TESSI and Rational Rose by means of XML. Project Report, TU Chemnitz, 2000. (In German)

5. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.
6. Johnson, W.L., Feather, M.S., Harris, D.R.: Representation and presentation of requirements knowledge. IEEE Transactions on Software Engineering, pp. 853-869, 1992.
7. Kim, Y.-G.: Effects of Conceptual Modeling Formalism on User Validation and Analyst Modeling of Information Requirements. PhD Thesis, University of Minnesota, 1990.
8. Kroha, P.: Objects and Databases. McGraw-Hill, 1993.
9. Kroha, P.: Softwaretechnologie. Prentice Hall, 1997 (in German).
10. Kroha, P., Strauß, M.: Requirements Specification Iteratively Combined with Reverse Engineering. In: Plasil, F., Jeffery, K.G. (Eds.): Proceedings of SOFSEM'97: Theory and Practice of Informatics, Lecture Notes in Computer Science, No. 1338, Springer, 1997.
11. Leidenfrost, S.: TESSI in Java. Project report, TU Chemnitz, 1999. (In German)
12. Lavoie, B., Rambow, O., Reiter,E.: The ModelExplainer. In: Demonstration Notes of International Natural Language Generation Workshop, INLG'96, Harmonceux Castle, Sussex, UK, 1996.
13. Lavoie,B., Rambow, O., Reiter, E.: A Fast and Portable Realizer for ext Generation Systems. In: Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP'97, Washington, 1997.
14. Petre, M.: Why looking isn't always seeing: Readership skills and graphical programming. Communication of the ACM, Vol. 38, No. 6, pp. 33-42, 1995.
15. Robertson, S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley, 1999.
16. Rubin,K., Goldberg,A.: Object Behaviour Analysis. Communication of ACM, Vol. 35, No.9, pp. 48-62, September 1992.
17. Scott, D., de Souza, C.: Conciliatory planning for extended descriptive texts. Technical Report 2822, Philips Research Laboratory, Redhill.
18. Swartout,B.: GIST English Generator. In: Proceedings of the National Conference on Artificial Intelligence, AAAI'82.