Asking the right questions to elicit product requirements

Min Wang *; Yong Zeng *

* Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Montreal, Quebec, Canada

First Published: April 2009
Asking the right questions to elicit product requirements

Min Wang and Yong Zeng*

Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Montreal, Quebec, Canada

(Received 5 December 2007; final version received 2 May 2008)

Eliciting precise and comprehensive product requirements from customers is of critical importance for the success of product development. In this paper, a generic process is proposed for eliciting product requirements by asking questions based on linguistic analysis. The linguistic analysis transforms a text into a graphic language called recursive object model (ROM). Two types of questions are asked in the process. One type of question, generated according to the topological structure of the ROM diagram, is domain-independent whereas the other relies on the domain of product development. A generic template is developed for generating the questions and for determining the sequence in which those questions are asked. The answers to the questions can be sought on the internet, in text books, the dictionary, the designer's own knowledge and experience, the customers and other partners involved in the product development, and/or nature itself. The generation of new questions may be based on the answers that are obtained. A software prototype is developed to support the proposed process. A case study of a rivet-setting tool design is used to illustrate the process of generating questions.

Keywords: requirements elicitation; question asking; question generation; engineering design

1. Introduction

Product requirements are descriptions of the desired solution to a design problem. In engineering design, just as in all other design problems, the efficient, precise, and complete specification of design requirements is critical if designers are to deliver a quality design solution within a reasonable range of cost and time. As a result, the nature of design requirements and the design process have been the subject of a wide range of research. Recently, some requirements for modeling approaches have been proposed, such as methodology-based design and language-based design (Darlington and Culley 2002). In the category of methodology-based design, certain methods (Darlington and Culley 2002) have been applied to the development of design support mechanisms, including quality function deployment (QFD) (Clausing 1998, Akao and Glenn 2003), a taxonomic approach (Gershenson and Stauffer 1999), key characteristics (Lee and Thornton 1996), and functional decomposition (Andersson et al. 2000). Darlington and Culley classify the research in this category into two kinds of noticeable design theories (Darlington and Culley 2002). One comes from Wootton, who made an analysis of the design requirement in terms of the stakeholders and the information sources involved in the complexity of developing new products as corporate activity (Wootton et al. 1998). The theory provides a foundation for a prescriptive guide to the requirement-capturing process for industrial use. The other one, a science-based approach to product design theory, comes from Zeng and Gu (1999a, 1999b). They have proposed a set theory-based representation scheme for the representation of the design objects that evolve during the design process. As a continuation of the efforts in the science-based approach to product design theory, Zeng has proposed a new design methodology, environment-based design (EBD), which is a step-by-step approach to solving poorly defined design problems and which can assist designers in delivering creative and innovative design solutions (Zeng 2004).

For the language-based design, language is viewed as playing an important role in the communication of the ideas involved in the design requirement. Leceouche discussed natural language constraints as a means of guiding the elicitation between a user and a support system (Leceouche et al. 1998). The constraints were provided by means of a theory that characterises the structure of natural language. On the one hand, natural language allows non-specialists to discuss design requirements easily and naturally with enormous semantic richness. However, natural language descriptions carry a lot of noises, ambiguities,

*Corresponding author. Email: zeng@ciise.concordia.ca

ISSN 0951-192X print ISSN 1362-3052 online
© 2009 Taylor & Francis
DOI: 10.1080/09511920802232902
http://www.informaworld.com
and contradictions, as pointed out by Meyer (1985). On the other hand, formal language promotes precision and systematism. It provides a foundation for formal verification and validation; however, it is limited in expression and it can be difficult to understand. Both of these two representations have their respective disadvantages and limitations. The combination of the two approaches could greatly improve the practice of requirements engineering. However, not much significant research has been reported in the literature regarding the generation of formal design specification from natural language.

The major focus of this present research is to identify the customer’s real intent and to elicit the completed requirements from a poorly defined design problem. Much effort has been expended concerning requirements elicitation, such as interviewing (Zeroual et al. 1989), issue-based information system (IBIS) (Christel and Kang 1992), joint application design (JAD) (Wood and Silver 1995), accelerated requirements method (ARM) (Hubbard et al. 2000), quality function deployment (QFD) (QFD-Institute 2005). Among various requirement elicitation approaches, interviewing plays a significant role by allowing the customer’s unexpressed requirements to be discovered. However, the success of the interview approach relies largely on the interviewer’s knowledge and experience in asking the proper questions. In the literature, very few efforts have been reported regarding how to ask the right questions in the interview process.

Motivated by psychology, social cognition, and education as well as artificial intelligence, Ram developed a theory of questions and of question asking in the area of understanding and learning, which includes the process of question generation, question management, and question interaction (Ram 1991). Knuth proposed that relevant questions can be generated based on a Boolean lattice of logical statements (Knuth 2005). However, this approach is limited to questions to which possible answers exist in advance. The questions are used only to identify the proper answers. A recent investigation has been Eris’s work on the role of effective inquiry in the innovative engineering design process (Eris 2003, 2004). Eris has also provided the taxonomy of questions, which classifies questions as deep reasoning questions, generative design questions, and others. Along the same line, Aurisicchio et. al. have refined the classification of questions used during the design process (Aurisicchio et al. 2006). However, their work is focused on the linkage of questions and decisions at the conceptual level and falls short of a methodology on how to make effective inquiries.

In the present paper, a systematic iterative question-asking approach is proposed to elicit product requirements. This approach aims at identifying the customer’s real intent and at capturing the complete product requirements by asking questions based on a semantic analysis of the requirements text. To the authors’ best knowledge, this is the first questioning strategy that has been reported in the literature of requirements engineering. The rest of the present paper is organised as follows: Section 2 provides the theoretical foundation for the question-asking approach. The iterative question-asking approach is given in Section 3. A software prototype and a case study are given in Section 4 to illustrate the effectiveness of the proposed approach. Section 5 concludes this paper.

2. Theoretical foundation for question asking

This section introduces the theoretical foundations for question asking in requirements elicitation. These foundations are based on the logic of design (Zeng and Cheng 1991), the classification of product requirements (Chen and Zeng 2006), and the recursive object model (ROM) (Zeng 2008). The logic of design provides an underlying reasoning mechanism for question generation. The classification of product requirements defines a roadmap for the identification of complete requirements. The ROM provides a linguistic tool for capturing the semantics of the requirements text.

2.1. Logic foundation for question asking in design

The traditional view of the design process is that design evolution goes through the following stages: specification of design requirements, design synthesis, and design evaluation. These three stages iterate until a satisfying design solution is found.

Zeng and Cheng (Zeng and Cheng 1991) indicated that design is a recursive process in which a satisfying design solution must pass an evaluation defined by the design knowledge that is recursively dependent on the design solution to be evaluated. Since the design knowledge, which implies the design criteria, is part of the design problem, the generation of design solutions does indeed change the original design problem. This observation leads to the proposal of recursive logic as the logic of design (Zeng and Cheng 1991). Based on this logic, the design process can be described as a series of design states defined by both product descriptions and product requirements (Zeng and Gu 1999b), as is shown in Figure 1 where design requirements and product descriptions co-evolve throughout the design process. Therefore, it is fundamentally impossible to distinguish design problems and design solutions.
On the one hand, when a customer approaches a designer, s/he often asks for a product with certain functions under certain constraints. As is implied in the recursive logic of design, the required functions and constraints indeed come with the product that the customer refers to. On some occasions, the product described by the customer may not be the real product that s/he is looking for. The functions carried out by the product may be mistaken for the product itself. Hence, it is important for a designer to identify the customer’s real intent. Based on Figure 1, the customer’s real intent can be better seen in the earlier design state. Accordingly, designers need to find a way to help the customer to go back to those earlier design states, which are often implicit for customers.

On the other hand, moving from the current design state to the next one usually leads to refined product descriptions and thus expands the design requirements. According to the recursive logic of design and Figure 1, product requirements cannot be completely defined until the final product descriptions are generated. This forward process can lead to more complete product requirements.

### 2.2. Roadmap to complete product requirements

As new and latent product requirements have to be identified in the design process, a roadmap is indispensable for a complete set of product requirements. Such a roadmap depends on the establishment of an effective taxonomy of the product requirements. In the literature, a number of criteria have been proposed to classify product requirements (IEEE 1983, Southwell et al. 1987, Bahill and Dean 1999, Schach 2002). Some of these criteria are the following: functional and non-functional requirements, performance/reliability, interfaces and constraints (Southwell et al. 1987), mandatory requirements and preference (Bahill and Dean 1999).

Zeng has derived a theorem of the source of product requirements based on the Axiomatic Theory of Design Modeling (Zeng 2002). The theorem posits that all the product requirements in a design problem are imposed by the product environment in which the product is expected to work (Zeng 2004). Chen and Zeng classify product requirements by listing all the environment components with which a product interacts (Chen and Zeng 2006). On the one hand, the product requirements can be classified in terms of the product life cycle. Different kinds of products may have different life cycles. For example, for a mechanical product, the life cycle can be divided into seven kinds of events, which are design, manufacture, sales, transportation, use, maintenance, and recycling, as shown in Figure 2 (Chen and Zeng 2006). All the requirements are classified according to those seven kinds of events so that various requirement providers are able to concentrate on their respective parts, which are associated with their relevant environments.

On the other hand, by classifying environments into human, natural, and built environments (Zeng 2004), Chen and Zeng have categorised product requirements into eight levels: natural laws and rules, social laws and regulations, technical limitations, cost, time and human resources, basic functions, extended

![Figure 1. Evolution of the design process.](image1.png)

![Figure 2. Seven events in product life cycle (Chen and Zeng 2006).](image2.png)
functions, exception control, and human-machine interface, as is shown in Figure 3 (Chen and Zeng 2006). In this pyramid-like model, those requirements at the lower levels have higher priority in the development of a design solution. And those products meeting the requirements at the highest level are said to be called high usability products.

In this model, higher level requirements can be considered after lower-level requirements are satisfied. Basically, this pyramid-like model can be divided into two major groups: non-functional requirements, and functional requirements. The lower four, natural laws and rules, social laws, regulations, technical limitations, cost, time, and human resource level are usually non-functional requirements. The upper four, basic functions, extended functions, exception control, and human-machine interface are usually functional requirements. Among those requirements, the highest four levels of product requirements come from the human environment. They pertain to the purposes of the human use of the product; the lowest level of product requirements comes from the natural environment; and the rest are the result of the built environment.

In summary, a complete list of product requirements can be defined from two perspectives: one is based on the partition of the environment in terms of the product lifecycle whereas the other is the partition of the environment into the human environment, natural environment, and built environment.

2.3. Semantic foundation for question asking in design

As shown in Figure 1 and Section 2.1, a design process may evolve from the current design state forward to a more refined one or backward to an old one. The former serves for the purpose of finding the complete list of product requirements whereas the latter aims to identify the real intent behind the design problem. Researchers have found, through experiments or observations, that engineering design is a question-driven process (Eris 2003, 2004). As the recursive logic of design implies, a design problem and the design solutions are coupled throughout the entire design process (Zeng and Cheng 1991). Questions set up goals for each design stage, thereby leading to a new design state.

Question-asking is strongly dependent on the understanding of the semantics of a text (Ram 1991). ROM is a graphic language that carries the semantic information implied in a natural language-based design problem description (Zeng 2008). Based on the Axiomatic Theory of Design Modeling (Zeng 2002), ROM can represent the linguistic structure of a free text through syntactic analysis only. Table 1 shows the graphic symbols in the ROM. The ROM diagram provides the semantic foundation for the question-asking in the design process.

3. Asking the right questions to elicit product requirements

In this section, a generic process is proposed for gathering product requirements and transforming them from natural language description into formal specification. How to ask proper questions is critical for collecting right product requirements. These questions can help to gather precise and complete requirements for the product that is to be designed.

3.1. Procedures for eliciting product requirements

Any design process starts with a description of a design task, which customers often describe in natural language. In other words, the design task is customer oriented and may be ill-defined whereas the requirements elicitation process should generate formal and structured descriptions, which are engineering oriented. Asking the questions is an effective way of identifying the customer’s real intent and of defining a relatively more complete list of product requirements.

We have proposed a generic inquiry process for eliciting product requirements (Wang and Zeng 2007). It is shown in Figure 4. The process can be divided into eight steps as follows. An algorithmic description is given later in this section followed by a detailed case study.

Step 1: Create a ROM diagram.
The designer transforms the original design problem described by natural language into a ROM diagram using a ROM analysis tool. This enables the designer to understand the design problem more clearly.
Table 1. Types of symbols in ROM (Zeng 2008).

<table>
<thead>
<tr>
<th>Type</th>
<th>ROM Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>$O$</td>
<td>Everything in the universe is an object.</td>
</tr>
<tr>
<td>Compound Object</td>
<td>$O$</td>
<td>It is an object that includes at least two objects in it.</td>
</tr>
<tr>
<td>Relations</td>
<td>Constraint Relation</td>
<td>It is a descriptive, limiting, or particularising relation of one object to another.</td>
</tr>
<tr>
<td></td>
<td>Connection Relation</td>
<td>It is to connect two objects that do not constrain each other.</td>
</tr>
<tr>
<td></td>
<td>Predicate Relation</td>
<td>It describes an act of an object on another or that describes the states of an object.</td>
</tr>
</tbody>
</table>

Figure 4. Generic inquiry process for requirements elicitation.
Step 2: Generate generic questions.
In this step, the designer analyses each object in the ROM diagram to ascertain the objects that need to be identified or clarified further and then generates some candidate questions based on a set of predefined rules and a question template. Then some of these questions will be selected for asking. These questions help customers to understand and to clarify their real intent.

Step 3: Collect answers.
The designer collects the answers to the questions that are generated in step 2 by consulting a dictionary or knowledge base, by searching on the internet, or by collecting information from the customer.

Step 4: Repeat Steps 1 to 4 until no more generic questions can be asked.
The answers collected in step 3 are analysed iteratively as new ROM diagrams in step 1. Then these new ROM diagrams are identified or clarified by asking questions in step 2. Thereafter, based on a set of predefined rules, the ROM diagrams are merged into the original ROM diagrams in the ROM tool.

At the end of the step, the explicit design problem description is extended further. The recursive process will be shown more clearly in the algorithmic description of the procedures.

Step 5: Generate domain specific questions.
The explicit design problem description above may not exactly and completely define the design problem. The designer needs to elicit more implicit environment information about the design problem. In this step, the designer analyses the relationships between the objects in the updated ROM diagram and generates a set of questions that should be answered at each stage of the design. Meanwhile, the sequence for asking these questions should be determined automatically or manually based on a set of predefined rules.

Step 6: Collect answers to the questions generated in Step 5.
This step is similar to step 3.

Step 7: Repeat steps 1 to 7 until no more domain questions can be asked.
The iterative process given in steps 1 to 7 may ensure that the domain-dependent product requirements are elicited accurately.

Step 8: Output the updated design problem description.
The process presented above can be alternatively represented by the following algorithmic process described in pseudocode. The entire process is made up of two sub-processes, which are design problem analysis and design problem extension.

FUNCTION generic inquiry process for requirements elicitation
// Generic Process - Input: text; Output: ROM
ROM GenericProcess( text) {
  // Analyse design problem through generic questions
  ROM ← DesignProblemAnalysis (text); 
  // Analysed design problem through domain specific questions
  ROM ← DesignProblemExtension (ROM); }

Design problem analysis is a recursive process that consists of asking the generic questions to get the real intent from the customer. This process analyses a text written in natural language to generate the questions to be asked and to collect answers, which are analysed iteratively within the process. Finally, a merged ROM diagram is created with a more detailed requirements description. The following pseudocode describes the process.

FUNCTION sub-function of design problem analysis by asking generic questions
// Input: text; Output: ROM
ROM DesignProblemAnalysis (text) {
  // Generate the ROM diagram for the design problem text
  ROM ← ROMAnalysis (text); 
  // Object analysis to determine objects to be asked
  ROM ← ObjectAnalysis (ROM); 
  // Generate question list
  Q_list ← GenericQuestionGeneration (ROM); 
  // Ask the questions
  if (Q_list is not empty)
    forall (Q in Q_list)
      // Collect answer to question Q
      answer ← GetAnswer(Q); 
      // Call ROM analysis recursively
      ROM1 ← DesignProblemAnalysis (answer); 
      // Merge ROM1 to ROM
      ROM ← ROMMerge (ROM, ROM1); }
  else return ROM; }

The other sub-process, design problem extension, is also a recursive process, in which domain specific questions are asked to get the complete requirements. Those questions are generated from an all-around
classification of requirements embedded in the entire product life cycle. In the same way, the answers are analysed iteratively within the process of design problem analysis. The following pseudocode describes the process.

```c
// Sub-function of design problem extension by asking domain questions
// Input: ROM; Output: ROM
ROM DesignProblemExtension (ROM)
{
    // Call DomainQuestionGeneration to get question list
    Q_list ← DomainQuestionGeneration (ROM);
    // Ask the questions
    if (Q_list is not empty)
    {
        for all (Q in Q_list)
        {
            // Collect answer to question Q
            answer ← GetAnswer(Q);
            // Call ROM analysis recursively
            ROM2 ← DesignProblemAnalysis (answer);
            // Merge ROM2 to ROM
            ROM ← ROMMerge (ROM, ROM2);
        }
    }
    else return ROM;
}
```

3.2. Identification of customer’s real intent

The sub-process of asking generic questions is shown in Figure 5. This inquiry process is designed to get the customer’s real intent.

In this step, each object in a ROM diagram is analysed and categorised as a centre object or a constraining object that needs to be identified or clarified further. The whole ROM diagram is a directed graph with objects and relations from one object to another. The graph can be represented as $G = <V, E>$, where $V$ is a set of objects and $E$ is a set of edges, which are indeed relations, between the objects. This representation enables us to apply algorithms from graph theory to our problem under consideration. The rules listed in Table 2 can be applied to determine which objects should be extended first.

The pseudocode for the algorithm of asking generic questions is shown in the following.

![Figure 5. Asking the generic questions.](image-url)
in terms of environment components on their requirement level. A template is proposed to help generate the questions and to determine the sequence of questions. Table 4 shows the rules for asking the domain-specific questions.

4. Software prototype and case study

4.1. Software prototype

The quality of design can be improved by a better understanding of the requirements definition and the elicitation process. The generic questioning process is proposed to elicit the product requirements. Since the processing of ROM diagrams can be very time consuming, the process needs a design tool that allows designers to easily analyse design problems and generate questions. Meanwhile, a convenient communication platform is also essential for the smooth transitions between the questions asked by the designer and the answers given by the customer, especially in the case of collaborative product design. Computer-aided platforms and tools improve the precision and efficiency of the process.

A software prototype was developed to help designers build a generic formalised product system represented by the ROM language based on a design problem described in natural language. Meanwhile, the software prototype is used to validate the generic process proposed in the present paper. The input of this software system is a design problem described in English by customers whereas the corresponding ROM diagram is the output. Customers use the software to present the design problem and to answer the questions asked by the designer to help elicit the precise requirements. Designers use the software to analyse the design problem and to generate the final requirements.

The software system contains five modules: system management, design problem generation, design problem analysis, design problem extension, and document generation. The architecture of this system is shown in Figure 7.

The module of design problem analysis corresponds to the recursive process from Steps 1 to 4 as
shown in Figure 8. The module includes four core sub-modules: ROM generation, object analysis, question generation, and answer collection.

The module of design problem extension, as shown in Figure 9, corresponds to the recursive process from steps 5 to 8. The module includes environment analysis, question generation, and answer collection.

The two modules include core sub-modules, such as ROM generation, question generation, and answer collection.

Some primary interfaces in screen shots are given in the following to show the implementation process of the software prototype. The interface for design problem input is shown in Figure 10(a). The customer can create a design task in the edit box and save it. The interface for the ROM analysis is shown in Figure 10(b). The designer can transfer the text of the design problem descriptions or of the collected answers into ROM diagrams. Many functions such as ROM create, ROM edit, ROM merge, ROM save are included in the tool.

The question generation tool is shown in Figure 11. The designer uses the tool to analyse the ROM diagram and selects some questions from those candidate questions created by the system based on predefined rules and a template. The designer can also input questions manually and change the questions with flexibility.

4.2. Case study

A rivet-setting tool design example (Hubka et al. 1988) is used in this section to illustrate the generic process.
proposed in the present paper and in the software prototype. The task of this problem is to design a tool for riveting brake linings onto brake shoes for internal drum brakes.

In this case study, a customer first presents a design task in natural language as: design a tool for riveting brake linings onto brake shoes for internal drum brakes.

Then the software prototype introduced in Section 4.1 is used to identify the right product requirements for this problem.
Step 1: Create a ROM diagram.
The ROM diagram based on the design problem description is created in the tool of ROMA shown in Figure 12.

Step 2: Generate generic questions.
After analysing the ROM diagram created in step 1, some questions asked in this step are listed as followed:

- Q1: What kind of ‘brake’ is there in the design problem?
- Q2: What is the structure of the brake?

Step 3: Collect answers.
The answers collected in step 3 are listed below.

- A1: The brake is an internal drum brake.
- A2: The structure of the brake can be found in a design handbook.

Step 4: Repeat steps 1 to 4 until no more questions can be asked.
Each answer should be analysed from steps 1 to 4, and the ROM diagrams are merged into the original one in Figure 12(b).

Step 5: Generate domain-specific questions.
In this step, the question ‘Q3: What is the life cycle of the tool’ is asked first to determine the life cycle the tool is involved in. And the answer to this question is collected as ‘A3: The life cycle includes design, manufacturing, sales, transportation, use and maintenance.’

Based on the classification of requirements in terms of product life cycle of the tool, the questions asked in step 5 are listed as follows:

- Q4: What standards should the tool conform to?
- Q5: What standards should be followed for the users?
Q6: Where should the tool be manufactured?
Q7: What is a reasonable cost for you?
Q8: How long will the tool’s service life be?
Q9: What are the basic functions of the tool?
Q10: What are the extended functions of the tool?
Q11: Who will use the tool?

Step 6: Collect answers to the questions generated in step 5.

The answers collected in step 6 are as follows:

- A4: The use of this tool should conform to the related industry safety standards.
- A5: The working height of the user should follow ergonomic standards.
- A6: It will be manufactured in a specific workshop, which has specified equipments.
- A7: The cost of this tool cannot be over $190.00.
- A8: The service life of this tool should be around 5 years.
- A9: The tool rivets brake linings onto brake shoes.
- A10: The tool should be easy for transportation and maintenance.
- A11: The user of this tool is a car mechanic.

Step 7: Repeat steps 1 to 7 until no more questions can be asked.
The ROM diagrams for answers A4 to A11 are listed in Figure 13.

After merging all the diagrams, the final ROM diagram is shown in Figure 14. A further merged diagram is shown in Figure 15.

Step 8: Output the updated design problem description.
The final requirements are described in the following paragraph:

A rivet-setting tool should be designed to rivet brake linings onto brake shoes. The structure of the brake can be found in a design handbook. The user of this tool is a car mechanic. The hand force, the foot
force, and the working height should follow ergonomic standards. The use of this tool should follow the related industry safety standards. The service life of this tool should be around 5 years. The tool should be easy for transportation and maintenance. It should be manufactured in a specific workshop that has specified equipment. The cost of this tool cannot be over $190.00.

5. Conclusion

In the present paper, a generic process is proposed to elicit precise and complete product requirements. The foundations of this generic process are the recursive logic of design, classification of product requirements based on environment and ROM. The logic of design provides an underlying reasoning mechanism for question generation. The classification of product requirements defines a roadmap for the identification of complete requirements. The ROM provides a linguistic tool for capturing the semantics of the requirements text.

An iterative question-asking approach is proposed to elicit precise and complete requirements by asking the right questions. This question-asking approach includes two major algorithms: one for asking generic questions and the other for asking domain specific questions. Both algorithms use predefined rules and templates to generate questions. The generic questions aim to identify the customer’s real intent whereas the domain specific questions aim to collect the complete list of product requirements.

A software prototype is designed and implemented as a collaboration platform and an analysis tool.
The software prototype helps designers to elicit the customer’s real intent and to collect complete product requirements. A case study based on a rivet-setting tool design is used to illustrate the concepts proposed in the present paper and in the software prototype.

The initial experiments show that the present approach for requirements elicitation is feasible and promising. However, the question-asking approach needs to be improved through more case studies. In particular, the following issues must be investigated before the approach can be put into industrial applications: the algorithms for asking domain-specific questions, the predefined rules for ROM analysis and product domain analysis, and the templates for question generation.

To improve the generic process, our research group is approaching that process from many perspectives, such as the processing of natural language, especially its ambiguities. The predefined rules for ROM analysis and for product domain analysis are being refined. The templates for generating questions are being...
supplemented. A more robust algorithm for determining the sequence of asking questions is under development. The software prototype can be extended by including an upgraded ROM tool and a knowledge-base server.

**Acknowledgements**

This work is partially supported by NSERC (Grant number RGPIN 298255). The authors thank Lei Chen for developing the ROM analysis system ROMA used in this paper.
References


