Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2001 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2001

Conversation Systems for Requirements Engineering

Alexander Hars
University of Southern California

Jiangfan Zhong
University of Southern California

CONVERSATION SYSTEMS FOR REQUIREMENTS ENGINEERING

Alexander Hars
Marshall School of Business
University of Southern California
hars@bus.usc.edu

Jiangfan Zhong
Marshall School of Business
University of Southern California
jiangfaz@marshall.usc.edu

Abstract

Requirements analysis is one of the most critical problems in information systems development. While many approaches have been proposed, it has proven difficult to bridge the semantic gap between information technology specialists and users who are familiar with the domain. In this paper a new approach to requirements analysis will be presented that uses information technology to communicate with users in natural language. The paper describes the architecture of a prototype system that is based on natural language processing technology. The goal of the system is to engage the user in a productive dialogue about information systems requirements. The system generates questions about systems requirements, parses the users' answer and then uses the answer to generate additional questions. This approach has two main advantages: dialogues can be adapted to the level of knowledge and the expertise areas of the user. In addition each session results in a knowledge structure that provides a strong foundation for the subsequent steps in requirements specification.

The Semantic Gap

As the capability of information technology matures it becomes more and more difficult to match the reality of an organization to the opportunities of technology. ERP systems, for example, provide an abundance of functionality and promise to make information instantly available throughout the corporation. However, ERP implementations frequently encounter problems because critical needs or unique aspects of a company have not been recognized in the configuration process. Similarly, object-oriented programming languages and component-based technologies have made the job of assembling software much easier. This has reduced development costs and durations. It has led to the emergence of prototype-based development approaches such as RAD and has increased the amount of communication between development teams and users. However, all development approaches have limitations when knowledge is spread among many users and when attention needs to be spent on detail. In these cases it is crucial to systematically collect requirements by interviewing many parties. This can be costly, error-prone and time-consuming. In this article we will provide an alternative approach where requirements are collected by a computer-supported tool.

This article is organized as follows: After a brief review of the problems in requirements elicitation, the feasibility of a language-based approach will be explored and current natural language technologies will briefly be discussed. Subsequently, the architecture of a language-based system will be presented.

Approaches for Requirements Engineering

Requirements engineering is an important foundation for developing software systems. Requirements engineering is the branch of software engineering concerned with the real-world goals for functions of and constraints on software systems (Zave 1997). Based on this point of view, "software requirements engineering is defined as all the activities devoted to identification of user requirements, analysis of the requirements to drive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against the actual user needs" (Saiedian and Dale, 2000, p.420). The main objective of requirements engineering is that "customers and designers must develop a shared understanding of the work problems and the impact of technical solutions on the work" (Holtzblatt and Beyer 1995). This is a major problem because developers

frequently find it hard to understand users' needs and they often lack familiarity with domain. On the other hand, users are often not aware of technological advances and their business potential (Saiedean & Dale, 2000, p.422). Thus a 'semantic gap' (Kaiser & Bostrom, 1982) exists between users and developers who conceptualize requirements in very different ways.

Requirements elicitation has a long history of research. Churchman and Schainblatt (1965) provide one of the early pieces focusing on requirement analysis. Thereafter it has attracted special and increasing attention by many researchers (Holtzblatt and Beyer, 1995; Keil and Carmel, 1995; Hutchings and Knox, 1995; Brun-Cottan and Wall, 1995; Beyer and Holtzblatt, 1995; Holtzblatt and Beyer, 1995). Key requirements elicitation techniques are shown in Table 1.

Table 1. Requirements Elicitation Techniques

Scenario analysis	Concept generation, organization and analysis	Nominal group techniques
Work process analysis	Requirements clustering	Mind mapping
Task analysis	Structured interviews and surveys	Brain storming
Decision process tracking	Interactive observation	Documentation using non-traditional artifacts

Conversation-Based Approaches

Hallman (1988) was one of the first to point out that the main problem in requirements engineering is to achieve harmony between a well-defined basic model and a convenient language. Users and domain experts are most familiar with natural language. Therefore several requirements engineering methods have focused on analyzing natural language terms. Several researchers have described methods for manually analyzing the language used for describing requirements statements.

McDavid (1996) proposed an approach for business language analysis for object-oriented information systems. Drawing on lexical semantic and category theory, the paper recommends specific activities and work products to produce a model of business language and suggests use of the language model in various requirements activities. The author shows how the language model can contribute to database design and offers key support for the development of user interfaces. Furthermore, language models can reveal the variations in importance of the same concept from one domain to another. It follows that language models have special value for requirement definition. Language-based heuristics are also used in many object-oriented analysis methodologies. Kristen (1994), for example, provides a detailed account of linguistic rules for the identification of classes and properties during object-oriented analysis.

There have been few attempts, however, to leverage natural language processing techniques for requirements analysis. Nanduri and Rugaber (1996) have shown that syntactic analysis can be used to extract initial database requirements from natural language texts. Hars (1998) utilized a semantic approach based on a large dictionary to extract requirements from natural language process descriptions which were subsequently translated into process models. All of these approaches, however, had significant limitations. Many approaches were domain-dependent and could not easily be generalized. Other approaches required perfect specifications and disregarded the fact that the user undergoes a learning process and will often change his ideas about requirements during a specification session.

In this paper, therefore, a new approach will be proposed which overcomes both shortcomings by focusing on the dialogue occurring during domain analysis rather than the production of a finished 'perfect' requirements model. Several systems have recently been developed which support the creation of natural language dialogues.

In the following, systems that support natural language dialogues will be called conversation systems. A major goal of such systems is to maintain a conversation with a user for a significant time period. Many conversation systems have been developed that focus on this goal. They have been particularly spurred through the Loebner Prize (1994) that will be awarded to the first system that mimics human conversation capability. Examples of such systems are Weizenbaum's Eliza (1976) which was developed in the Sixties as well Alice (www.alicebot.org), a chatterbot that has won several annual Loebner contests. For other conversation systems, see http://bots.internet.com/search/s-chat.htm. Nevertheless, conversation systems are also used in commercial applications. For example, some web sites use conversation systems to provide product information or to help make purchasing decisions. An example is Buy.Com where conversation system assists customers in selecting the appropriate desktop or notebook computer system.

Most of these systems, however, require considerable configuration and maintenance efforts. Chatterbots such as Alice, Brian (http://www.strout.net/info/science/ai /brian/), and others have the disadvantage that they do not keep track of issues raised in a conversation. Most answers are discarded immediately and don't surface again in later parts of a conversation. The user thus perceives that it is difficult to keep a conversation going because the system has difficulties following the user through a topic.

The main problem for conversation systems is that they need to mimic a knowledgeable person. Providing systems with common-sense knowledge, however, has proven extremely difficult (Lenat, 1998). For conversation systems focusing on requirements analysis, however, the problem is different. In requirements analysis knowledge is distributed asymmetrically. The user is the primary source of information. The purpose of the interviewer is to provide the user with adequate questions that lead to valuable information. Thus a requirements elicitation system does not need to convince the user that it is aware about the specifics of a problem. It merely needs to be able to ask useful questions. Showing ignorance is not necessarily perceived as a weakness but is a required feature of any interviewing system. Thus requirements elicitation may be an ideal field to apply conversation systems as it is able to use the strengths of such systems – being able to engage in a dialogue in natural language, being able to generate questions and to react to utterances by the user. At the same time, the weaknesses of current conversation systems – their lack of common-sense knowledge – does not weigh as heavily.

In the following the architecture of a domain analysis system will be presented. The functions of each component and the challenges during their implementation will be discussed. An initial version of such a system has been developed by the authors.

System Architecture

At the core of all conversation systems from Eliza onwards are two key functions – a function that analyzes input coming from the user and a function that generates a response. In most cases, several strategies are followed. The HeX system, for example, which won the Loebner Competition in 1996, breaks user responses into keywords which are then matched to canned responses in a database. If no match is found, the program performs checks the answer for additional conditions. If this part also fails, the program switches the topic or provides other default answers (Hutchens, 1996). Alice, another chatterbot, uses a similar approach based on pattern templates which translate user input into a system response (Wallace, 2000). The developers of Alice have developed a markup language to specify such patterns and their transformation.

Therefore both functions (answer parsing and question generation) need to be part of a requirements elicitation system (see Figure 1). In the simplest case, a dialogue starts with a predefined question. The user's answer is analyzed for keywords for which follow-on questions have been defined in a question database. For example, if the answer contains the keyword 'report', a question database might contain follow-on question that ask what kind of information the report contains, what other reports might be relevant and who would use the report. Or, if the answer contains a planning verb, the database might contain a request to describe the planning activities.

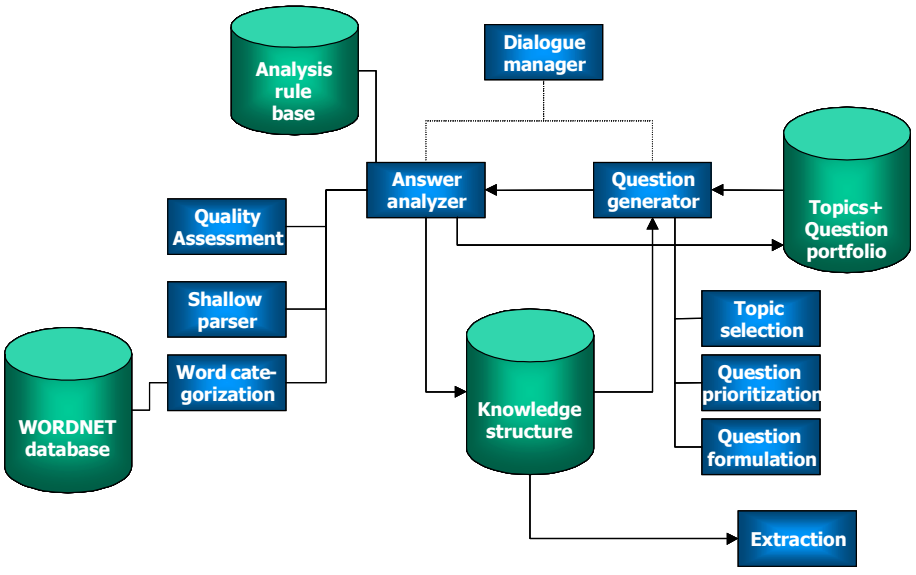

Figure 1. System Architecture

Such a direct keyword-matching approach has several limitations. While it generates questions easily, it does not provide direction to a conversation. Topics can change from sentence to sentence and discourse may turn in circles. Therefore it becomes necessary to organize a discussion around topics.

Therefore questions need to be structured into topics. The topic categories can be taken from current information systems development methodologies such as Rational Unified Process (Rational 2000), Architecture of Integrated Information Systems (Scheer, 1998) or from texts on requirements analysis (e.g. Urquhart, 2000). Table 2 shows top-level topics distinguished in these texts. For each of these topics many question templates can then be defined. If necessary, sub-topics can be added.

When the system processes an answer, it generates follow-on questions and places these questions into an internal portfolio of questions. From this portfolio it then selects the next question for the current topic. When the topic is completed or the time allotted for the topic has run out, it switches to the next topic and begins asking questions about these topics. This approach has the additional advantage that a single answer can generate several follow-up questions.

This approach is conceptually simple and provides much flexibility. The quality of such an interviewing system greatly depends on the question portfolio, on an adequate topic structure and on an effective prioritization of topics and questions. In any interview, the number of potential questions will greatly exceed the time frame available for discussing the requirements. Thus it is important, that a system is able to make some decisions about when to stop asking questions in a specific topic and which questions to prefer. Experimentation is can be used to provide adequate heuristics.

So far, this approach can ensure that similar questions are not unwillingly asked twice (although in some cases it may be useful to repeat questions on purpose). It cannot avoid, however, that questions are asked to which the answer has already been provided. For example, questions about business processes will often reveal the stakeholders of the planned system which would be asked in a different topic. Questions about information needs will frequently provide information about processes etc. It is desirable to take these nuggets of information into account. This can be achieved by storing such information in a knowledge structure that is gradually built up during the interview.

Table 2. Topics for Requirements Conversation

Topics for Requirements Conversations	Rational Unified Process	Architecture of Integrated Information Systems
Urquhart (2000)	Rational (2000, p.11)	Scheer (1998, pp.39f, 50)
Issues to be discussed	Vision	Initial strategic situation
Scope of current system	Stakeholder needs	Process view
Rapport building	Actors	Function view
Processes associated with the system	Use cases (processes)	Organization view
Keys in information system	Non-functional requirements	Information view
Future action		Product view
Information deficit in system		
Information output from system		
Analyst's understanding of processes		
Future solutions		

In its simplest version, such a knowledge structure would only contain categorized lists of items that have been found relevant for the system. It would contain a list of persons (either in the form of roles such as 'supervisor', 'clerk', 'customer' or in the form of names (e.g. John Doe or simply 'Jim'), a list of tangible objects (e.g. vehicles, accessories, motor etc. in an automatic setting), a list of information items (e.g. reports, <social security> numbers, <meeting> notes) and other categories such as organization units, process types etc. The knowledge structure then merely conveys that certain items are relevant but not in which way. Although this is very little information, it is useful for prioritizing questions, for generating follow-up questions and for aggregating the responses into a comprehensive requirements document. For example, it can group all answers that contain references to particular categories of items. It also represents the key elements of the users cognitive map about the application domain (Bartlett, 1932; Johnson-Laird, 1981; Walsh, 1995).

It would be desirable to extend the knowledge structure to include relationships between different categories and potentially specific types of statements. Semantic networks or frame-based representation mechanisms would be adequate but require a substantial amount of coding. In the first version our tool is thus limited to category lists and custom frames.

To build the knowledge structure, it is necessary to parse and categorize the answer sentences. This can be accomplished by combining one of the readily available part of speech taggers with a lexicon-based tool such as Wordnet (Miller et al., 1993; Beckwith, Miller & Tengi, 1990). Wordnet is a large thesaurus developed at Princeton University that identifies many syntactic and semantic relationships between words.

Two additional components complete the architecture. A dialogue manager monitors the question-and-answer sessions and requests changes to the conversation where necessary (a user may request, for example, to change a topic). The extraction functionality is used to present the results of the analysis session. It uses the session log and the knowledge structure to categorize, format and summarize responses into a domain analysis document. For this module natural language techniques described by Park et al. (2000) may be particularly relevant. Their techniques measure the similarity between requirement sentences to identify possible redundancies and inconsistencies, and to extract possibly ambiguous requirements.

The architecture has been implemented in a prototype. The implementation is based on Borland's Delphi rapid application development environment. This shows that the architecture is feasible. The current prototype is based on individual dialogues. However, the system may exhibit additional strengths once several people are interviewed for the same project. In that case, questions and answers from previous participants can be used to build up the knowledge structure, to verify parts of these structures and to generate more effective questions.

Conclusion

This article has presented an architecture for conversation systems that support domain analysis. It is based on existing systems that engage users in open-ended conversations. We have argued that requirements elicitation is a more appropriate field for using such natural language technologies than open-ended conversation systems because here a) the burden of knowing rests with the user rather than with the system and b) there is a clearly established goal.

One might argue against the concept of asking a user to interact with a mindless system for requirements specification. While such a system can not replace human dialogue it can reduce the effort for collecting requirements, it can increase the number of persons whose requirements are being heard. In addition, it can lead to a better understanding of the issues at hand while developing an information system. It can promote the gradual learning and discovery process that is always a key part of requirements elicitation. (Keil and Carmel, 1995; Holtzblatt and Beyer, 1993, 1995; Beyer and Holtzblatt, 1995; Axel and Emmanuel, 2000). The system should not be visualized as an intelligent tool but rather as a preprocessor. In the same way that automated translation systems have found a very useful role in preparing rough drafts that are then refined by specialists, interactive requirements elicitation systems can provide initial drafts of requirements specification that are then elaborated, adjusted by specialists and additional human-to human requirements elicitation sessions. One way of using such a tool is to precede human-to-human interviews by computer-assisted requirements elicitation sessions.

The system should have several advantages (and a number of problems) which can be examined empirically. It is not likely that the quality of the requirements would be higher than those of human-to-human interviews. This is due to the fact that a skilled interviewer is able to identify critical issues much more rapidly. However the tool may be able to enlarge the number and breadth of requirements elicited. It should be able to identify a larger scope of problems and issue because it is able to involve more users. It may also increase the reuse of requirements statements because knowledge structures created during an interview can already be part of the next interview. Another advantage is that the system may exhibit greater consistency in interviewing. Depending on the priorities of each project, it may be more accurate in ensuring that specific points are raised during each interview. Another advantage of the tool may be that it can provide anonymity and thereby lead to more realistic requirements specifications.

References

Axel van, L., and L. Emmanuel (2000), "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Transactions on Software Engineering*, 26 (10), p.978-1005.

Bartlett, F. C. (1932), *Remembering*. Cambridge: Cambridge U Press.

Beyer, H., and K. Holzblatt (1995), "Apprenticing with the Customer", *Communications of the ACM*, 38 (5), p. 45-52.

Beckwith, R.; Miller, G.A.; Tengi, R.A. (1990), *Design and Implementation of the WordNet Lexical Database and Searching Software*.

Brun-Cottan, F., and P. Wall (1995), "Using video to Re-present the User", *Communications of the ACM*, 38 (5), p. 61-71.

Churchman, C. W., and A. H. Schainblatt (1965), "The Researcher and the Manager: A Dialectic of Implementation", *Management Science*, 11 (4), p.69-87.

Hars, A. (1998), "Natural language-based data modeling: Improving validation and integration", *Journal of Database Management*, 9(2), p. 17-25.

Hallman, M. (1988), "An Operational Requirement Description Model for Open Systems", *10th International Conference on Software Engineering*, April 11-13, Singapore, p. 286-295.

Holtzblatt, K., and H. Beyer (1993), "Making Customer-Centered Design Work for Teams", *Communications of the ACM*, 36 (10), p.93-103.

Holtzblatt, K., and H. Beyer (1995), "Requirements Gathering: the Human Factor", *Communications of the ACM*, 38 (5), p. 31-32.

Holtzblatt, K., and S. Joens (1993), Contextual Inquiry: A Participatory Technique for System Design. In D. Schuler & A. Namioka (Eds.), *Participatory Design: Principles and Practices,* p. 177-210, Hillsdale, NJ:Lawrence Erlbaum Associates.

Hutchens, J. (1996), *How Hex Works*. Online article. http://www.amristar.com.au/~hutch/hex/How.html (3/9/2001).

Hutchings, A., and S. Knox (1995), "Creating Products Customers Demand", *Communications of the ACM*, 38 (5), p. 72-80.

Johnson-Laird, P.N. (1981), "Mental Models in Cognitive Science", in D.A. Norman (Ed) *Perspectives on Cognitive Science*, Hillsdale, NJ: Lawrence Erlbaum.

Kaiser, K.M., and Bostrom, R.P. (1982). "Personality Characteristics of MIS Project Teams: An Empirical Study and Action Research Design," *MIS Quarterly*, December, p. 43-59.

Keil, M., and E. Carmel (1995), "Customer-Developer Links in Software Development", *Communication of the ACM*, May, p. 33-47.

Kristen, G.(1994), *Object-orientation: The KISS method*. Wokingham.

Lenat, D.B. (1998), "CYC – A large-scale investment in knowledge infrastructure", *Communication of the ACM*, 38 (11), p. 33-38.

Loebner, H.G. (1994), "In Response", *Communication of the ACM,* 37 (6), p. 79-82.

McDavid D.W. (1996), "Business Language Analysis for Object-oriented Information Systems", *IBM Systems Journal*, 35 (2), p. 128-150.

McGraw, K., and K. Harbison (1997), *User-Centered Requirements*, Lawrence Erlbaum Associates, Inc., 10 Industrial Avenue, Mahwah, New Jersey 07430.

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K.(1993), *Introduction to WordNet: An online lexical database*.

Nanduri, S., and Rugaber, S. (1996), "Requirements Validation via Automated Natural Language Parsing", *Journal of MIS*, 12 (Winter), 1995-96, p. 9-19.

Park, S., H. Kim, Y. Ko, and J. Seo (2000), "Implementation of an Efficient Requirements-Analysis Supporting System Using Similarity Measure Techniques", *Information and Software Technology*, April, 42 (6), p. 429-438.

Urquhart, C. (2000), Strategies for conversation and systems analysis in requirements gathering: A qualitative view of analyst-client communication. [53 paragraphs]. The Qualitative Report [On-line serial], 4(1), Janauary. http://www.nova.edu/ssss/QR/QR4-1/urquhart.html (3/8/2001).

Rational (1998), *Rational Unified Process: Best practice for software development teams*. White paper TP-026-A rev.11/1998, Cupertino, CA.

Sadiedian, H., and R. Dale (2000), "Requirements Engineering: Making the connection between the software Developer and Customer", *Information and Software Technology*, 42, p. 419-428.

Scheer, A.-W. (1998), *ARIS – Business process frameworks*. 2nd ed. Berlin (Springer).

Svenson, O. (1979), Process Descriptions of Decision Making *Organizational Behavior and Human Performance*, 23, p. 86-112.

Wallace, R.S. (2000), *Don't read me: A.L.I.C.E. and AIML documentation* . http://www.alicebot.com/dont.html (3/1/2001).

Weizenbaum, J. (1976). *Computer Power and Human Reason*. W.H. Freeman and Company.

Yaung, A. T. (1992), "Design and Implementation of a Requirements Clustering Analyzer for Software System Decomposition", *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing* (Vol. II): Technological Challenges of the 1990's, March 1-3, Kansas City, MO USA.

Zave, P. (1997), "Classification of Research Efforts in Requirements Engineering", *ACM Computing Surveys*, December 29 (4), p. 315 - 321.