Explanation Component Of Software Systems

by Bruce A. Wooley

Abstract

Explanation is an important feature that needs to be integrated into software products. Early software that filled the horizontal software market (such as word processors) contained help systems. More specialized systems, known as expert systems, were developed to produce solutions that required specific domain knowledge of the problem being solved. The expert systems initially produced results that were consistent with the results produced by experts, but the expert systems only mimicked the rules the experts outlined. The decisions provided by expert systems include no justification, thus causing users to doubt the results reported by the system. If the user was dealing with a human expert, he could ask for a line of reasoning used to draw the conclusion. The line of reasoning provided by the human expert could then be inspected for discrepancies by another expert or verified in some other manner.

Software systems need better explanations of how to use them and how they produce results. This will allow the users to take advantage of the numerous features being provided and increase their trust in the software product.

Introduction

Providing explanations is key to ensuring the future success of software products in all domains of applications. Webster's Dictionary offers ``giving meaning or interpretation to, or to make understandable'' as a definition of explanation. For explanation systems, the goal is to make some piece of knowledge clear to the user. The explanation is complete when the user is satisfied with the explanation and understands the concept being explained.

In horizontal software applications, this explanation will be viewed in the form of intelligent tutoring and help systems that will lead the user through the process of accomplishing their desired results.

Expert systems will need to provide justification for their results in a format that the individual user can comprehend, with appropriate facts to convince the user of the validity of their results [2, 11]. In other support software such as decision support systems, the system needs to provide instruction on its use and explanation of its results in a format appropriate for the individual user.

Training based systems consist of both critiquing systems and intelligent tutoring systems. Both of these systems need to provide some explanation in the form of help to aid the user in the use of the system.

The system must also explain the process leading to a solution. The critiquing system should find flaws in the solution provided by the user and explain why these items are flaws. The intelligent tutoring system should lead the user to a solution via a logical sequence of steps. It should be able to justify each step's existence and the ordering of each step.

There are four basic techniques used in explanation systems. **Canned text** is a technique by which the designer of the system identifies keywords and explanations for these keywords at the time the product is being designed and implemented. **Rule trace** is a process applied to expert systems that identifies the individual rules that were used and the order in which they were used. **Domain knowledge** adds a level of abstraction to the rule trace by providing the reason the expert put this rule into the system. **Task-based explanations** break an expert system into logical components known as tasks. This is basically a hierarchical breakdown of the software components leading to a solution [12]. This allows explanation to be tied directly to the individual objects or tasks that are solving that piece of the problem. A special form of task based explanation is known as generic tasks. The task structure for task based explanation applies to different domains of knowledge. The user must also be considered in the process of explaining any topic. For example, if the user is already knowledgeable, they should get a different level of explanation than a novice.

There are numerous software systems available that use a variety of these techniques for generating explanations. A brief discussion of a few of these software products will be presented as examples at the end of this paper.

Types and Uses of Explanation Systems

There are three basic uses for explanation systems:

1. Help systems used to describe how to use a specific product or command
2. Belief justification systems to increase user confidence in the results created by the system
3. Training or tutoring systems

Help systems use three basic kinds of explanations. The simplest form of help is a keyword lookup, in which an explanation is tied to specific keywords. A relatively simple enhancement to this kind of help is context sensitive help. Context sensitive help still uses keyword lookups; however, it tries to give the appropriate keyword help based on the current state or configuration of the software, as well as supplying cross-references within the explanation. A further enhancement of help systems is to use mental models in the development of the help system. The goal is to get the user to develop this same mental model of a topic that an expert in the field would have.

Belief justification is becoming a normal part of an expert system. The objective is to explain how and why the system arrived at a specific solution to a problem [11]. This may be implemented as a trace of rules that are used in obtaining the solution. This trace of rules may not provide sufficient explanation because of the rules' cryptic nature and hidden logic in the sequence of the process. Domain knowledge

can be added to each rule to provide an additional level of abstraction by providing the reason the rule is included in the expert system.

Training systems consist of both critiquing systems and intelligent tutoring systems. **Critiquing systems** allow the user to enter a proposed solution to a problem. They system then tries to find flaws in the solution, which are then communicated to the user. One example of a critiquing system is a floor planning system for the deck of a ship. Possible flaws it could identify include weight imbalance, lack of access paths to the different parts of the ship, and signal interference with communication equipment. **Intelligent tutoring systems** are used to guide a user through the process required to obtain a solution. There are many tutoring systems already in use; however, a large number of these systems would not fit into the category of intelligent tutors. They are more likely to provide some concepts or examples, and then ask predetermined questions pertaining to the examples or concepts just presented. Based on the correctness of the answers (which may be keystroke dependent) they will continue to the next concept or force a review of the current concept. These tutors do not take into account the user's ability and knowledge; they have only limited answers to predetermined questions. An intelligent tutor requires domain knowledge that is relevant to the solution process being explained to the user. The tutor also needs to build a model pertaining to the individual user. As this user asks questions the tutor can relate its answers to prior concepts about which the user has demonstrated proficiency.

Techniques

Canned text is the lookup of a definition associated with a keyword. The advantage of keyword lookup is the simplicity of concept and implementation. In addition to traditional indexing lookups for the definitions, software tools such as hypertext can be used to enhance the initial lookup with cross-reference links to other keywords. There are limitations in using canned text to implement help or explanation systems. First, canned text requires a large number of keywords to be associated with messages. Second, all of the messages must be established in advance, so the designer must anticipate all of the questions the user could ask. Third, the explanation does not necessarily follow the logic of the program, so the explanation does not have a concrete tie-in with the actual functions being performed.

Rule trace can greatly enhance the explanation of an expert system. Rule trace involves listing all the rules that were used, along with the order in which they were used. This provides a dynamic explanation of the program's logic. Limitations of this process result from the missing abstraction associated with the detail of the individual rules and the abstract sequence used in applying these rules. To say a specific rule was used because of a certain relationship between two attributes does not indicate why that rule was included in the system, or how those two attributes relate to each other. In addition, following the detail of the execution of a program rarely gives a description of the thought process behind the sequence of events taking place.

Adding domain knowledge enhances the rule trace by adding a layer of abstraction in the form of an expert's reason for including individual rules in the expert system. The ability to include canned text with each rule and sequence of rules makes this extension easy to implement.

Task-based explanation ties the explanation to individual modules that are used in the expert system [12]. This allows a hierarchical approach to be used to create the explanations, and allows modifications to a portion of the system without affecting the remaining system components. Each module may use any combination of techniques best suited for that component to generate effective explanations, encouraging a hybrid of techniques for the overall system. This also encourages the designer to structure the solution in terms of tasks or objects, which is a parallel area of research known as object oriented methodology. Each individual object may use any technique it chooses for implementing explanation as long as it provides a standard interface for the request being submitted and the explanation being returned.

A special application of task-based explanations are referred to as generic tasks. Generic tasks are designed to be domain independent. An example of a problem that applies this technique is diagnosis [12]. This problem is generally solved by examining the hierarchical structure of the object of interest. This generic task may be built from other generic tasks such as a hierarchical classifier and a data inferencer. The hierarchical classifier is a generic task that is a subtask of the diagnosis. An optimized tree can be built by starting at the root and iteratively working down it. Each child is considered and matched with a known task pruned from the tree.

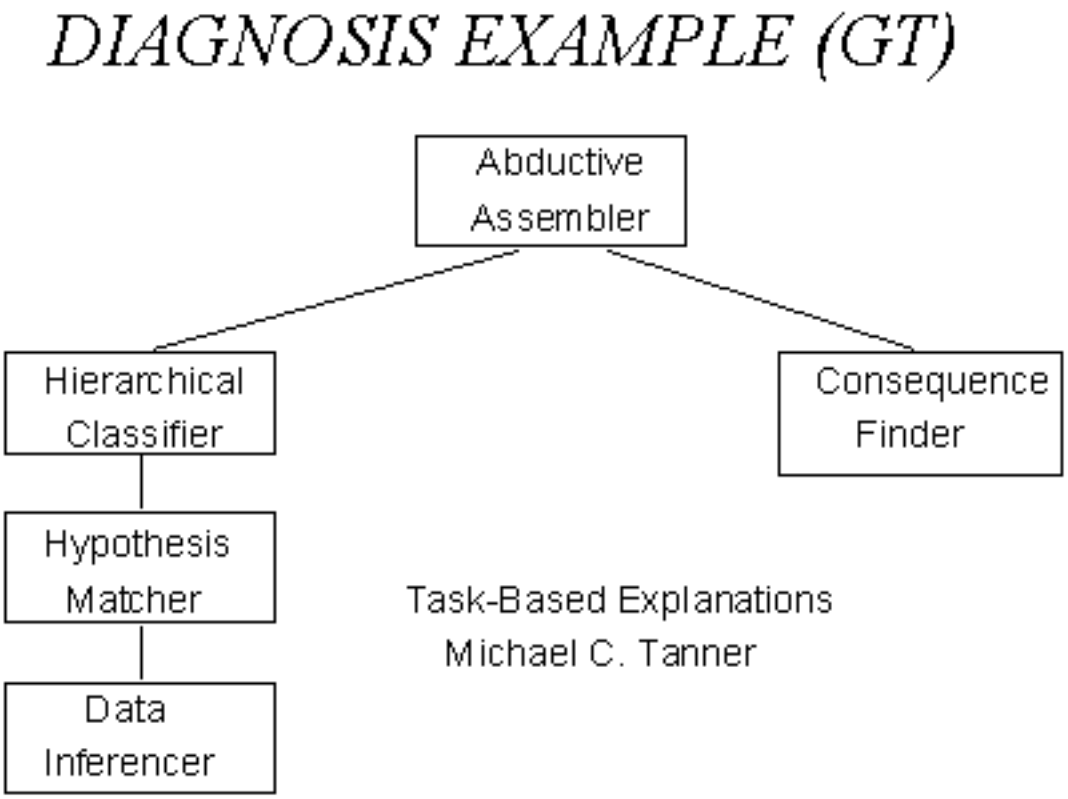

Figure 1

These techniques will become much more valuable if they take into account the user's abilities and limitations. The first step in improving explanations is to start identifying a model [9] for both the expert and the user. One element in a model is the identification of similarities between concepts being explained. In an explanation environment, the system needs to use concepts that have already been successfully explained as the basis for explaining new concepts. This can be accomplished by noting that a new concept is similar to a specific concept already defined, then listing the specific attributes that distinguish this new concept from the one that was explained previously. This process, which is similar to case-based reasoning, is believed to be one of the ways in which humans learn new concepts. This also helps overcome the limitations of short term memory (such as the inability of most humans to remember more than about seven items). Better results can be obtained by using a technique referred to as **chunking**, where several items are clustered under one concept. This is a hierarchical approach to short term memory, and follows the case-based process very closely. The interface can be used to help in the clustering process by using different communication features, where each feature may use about seven concepts in the communication of the explanation. An example is a GUI that breaks the screen down into seven different areas, each of which is describing a piece of the overall concept. Different colors could be used in each of these areas, where each of the colors has some specific meaning (such as red for things that are dangerous, dimmed writing for nonessential information or documentation, etc.).

The requirement of consistency is a paradox. Because we would like to create an explanation that fits the model the system has of the user and we also want this model to be dynamic, we need to be concerned with keeping the explanations consistent. If a system gives a user an explanation and the user later asks the same question, the user should get the same answer even if the system has updated its picture of the user's mental model. The only time the system should give the user a different explanation for the same question is if he specifically asks for an alternative explanation. After asking for alternative explanations, if the user asks the same question again, the system should pick one of the explanations already provided, preferably the explanation that was finally accepted by the user.

Example Expert Systems with Explanation Components

EDGE (Explanatory Discourse GEnerator) is a task-based explanation system that explains electronic circuit operation [3]. This system is interactive and communicates through textual explanations and user queries. The research focused on interaction between expert and novice with the analysis of dialogs between two humans and addressed many of the concepts already discussed in this paper. For instance, tailoring the explanation to the user's ability requires the system to build a model of the user, to know when and how interruptions occur and how to proceed after an interruption. The explanation planning is interleaved with execution; that is, the system only generates the part of the explanation requested then begins presenting this to the user and continues generating more explanation where the content is based on the interaction with the user.

RED is a diagnostic system that performs antibody identification activity for blood banks. The version

of RED discussed here is RED2 [12], and it is implemented as a generic task. The subtasks are also generic tasks, and each element contains explanation components for its individual piece of the solution.

NEOMYCIN [1] is a task-based explanation component used to explain the results MYCIN produces. NEOMYCIN is not a generic task explanation system. Its tasks are at a much lower level than those associated with generic tasks. NEOMYCIN is an intelligent tutoring system designed to teach diagnostic problem-solving to students. This is an area normally addressed in the student's apprenticeship.

XPLAIN [11] is a software product that helps users build expert systems containing explanation components. It contains an automatic programmer that uses two basic forms of input [11]. The first form of input is the domain model, which contains the facts of the domain. The second form of input is a collection of domain principles, which are the methods or algorithms that apply to the facts. This system refines the domain knowledge (preserving the knowledge) until it is at an appropriate level for the implementation of an expert system.

QEX (Quantitative EXplainer) is an expert system that was designed to explain the results of a scheduling system [10]. The distinguishing features of this system are its dependence on quantitative reasoning and its ability to translate the explanations for a user that does not have quantitative abilities. This system requires the translation of one set of concepts (those based on formal mathematics) into another set of concepts (those understood by users without formal mathematical training) that also describe the domain.

Summary

Explanation help systems and tutoring systems will no longer be an optional component of software systems since software is increasing in complexity. Any software that uses domain knowledge in arriving at a solution will be required to identify that domain knowledge for the user so the user is confident of the solution. As the functions in software packages increase in number and become more abstract in functionality, an effective help system becomes a requirement. The significant increase in required functionality demands that tutoring capabilities be included with software packages. The majority of users may overlook these new features if there is not some mechanism that periodically suggests the use of unused features and offers am explanation of the functionality that the user is not exploiting to its fullest potential. In addition, intelligent tutoring systems designed to teach non-computer applications should function similarly to any other explanation component in a software product. The domain knowledge must be represented in the explanation component of any explanation system. This domain knowledge not only builds belief in the results, but also increases the reliability of the system.

A combination of techniques will be the most useful in achieving good explanations in any system. Canned text will be a part of most systems and can handle the situations where the messages or information to be communicated to the user is tied to a keyword or a tangible object such as a rule or a

physical record. However, there must also be information about the process followed to arrive at a solution. This will be supplied through the rule trace and additional domain knowledge. The use of object-oriented paradigms will improve the integration of other techniques by allowing the explanation components to be tied directly to the object that is solving that piece of the problem. Generic tasks will provide a proven software path for a variety of domain independent problems. This software will not solve all problems, so there will continue to be specialized tasks designed for software that does not fit into the generic area that is domain independent. The fields that have well defined mathematical rules for obtaining solutions generally have a parallel solution that could be developed from a combination of common sense, experience, and intuition. To expand the market of the products that are based on mathematics, the system will need the ability to explain the solution in the domain of the user, which will include the mathematical domain and the domain consisting of common sense and experience. Translation of explanations between different domains is a viable alternative to maintaining multiple paths for solving problems through the alternative domains (such as the technique use by the REX system) [10, 13]. The explanation components will need to build a model that corresponds with the user's mental model, and generate explanations based on the domain described by this mental model. Interaction with the user will allow continuous maintenance of this mental model.

Explanation components are created by looking at the expert's knowledge and the techniques he uses to transfer this knowledge to individuals having a variety of background domains. A specific area that is lacking is the study of the individual's system of learning. There is some study in the development of mental models and how individuals change their mental model (through explanations to the user) to bring it closer to the mental model associated with the expert.

Expert systems currently address only problems that have a narrow domain. A large number of expert systems address problems concerning diagnosis in field of medicine. There are many opportunities for research in explanation, although they are all related; help systems, tutorial systems, and explanation systems all need to bring the user to the point of understanding a particular concept.

The knowledge representation that appears most suited to explanation is parallel case-based reasoning. This does not make any other representations that are currently in use less important. Many expert systems use some form of predicate logic, and these rules will be the central focus of the expert systems problem solving process. The suggestion that case-based reasoning is most suited is due to its apparent similarity to the way our mental model works.

References

1

Buchanan, Bruce G., and Edward H. Shortliffe. 1984. *Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison Wesley Publishing Company.

2

Cawsey, Alison. 1995. Developing An Explanation Component For A Knowledge-Based

System: Discussion. *Expert Systems with Applications* 8 (4): 527-31.

3

Cawsey, Alison. 1992. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues (ACL-MIT Press Series in Natural Language Processing)*. Cambridge, MA: The MIT Press.

4

Hollnagel, E. 1983. What Do We Know About Man-Machine Systems. *International Journal of Man-Machine Studies* (18): 135-43.

5

Kaplan, Randy, and Denny Rock. 1995. New Directions for Intelligent Tutoring. *AI Expert* 10 (February): 31-40.

6

Miller, G. 1956. The Magical Number Seven, Plus Or Minus Two: Some Limits On Our Capacity For Processing Information. *The Psychological Review* 63 (March): 81-97.

7

Mittal, Vibhu O., And Cecile L. Paris. 1995. Generating Explanations In Context: The System Perspective. *Expert Systems With Applications* 8 (4): 491-503.

8

Murray, William R. 1988. *Automatic Program Debugging for Intelligent Tutoring Systems*. San Mateo, CA: Morgan Kaufmann.

9

Norman, D. 1983. Some Observations On Mental Models. In *Mental Models*, eds. D. Genter and A. Stevens. (Chapter 1): 7-14. Lawrence Erlbaum Associates.

10

Slotnick, Susan A., and Johanna D. Moore. 1995. Explaining Quantitative Systems To Uninitiated Users. *Expert Systems with Applications* 8 (4): 475-90.

11

Swartout, William R. 1983. XPLAIN: A System For Creating And Explaining Expert Consulting Programs. *Artificial Intelligence* 21: 285-325.

12

Tanner, Michael C. 1995. Task-Based Explanations. *Expert Systems with Applications* 8 (4): 505-12.

13

Wick, Michael R., Pradyumna Dutta, Thomas Wineinger, and Joshua Conner. 1995. Reconstructive Explanation: A Case Study In Integral Calculus *Expert Systems with Applications* 8 (4): 463-73.

Bruce Wooley is an Assistant Professor of Computer Science at Lock Haven University in Pennsylvania. Mr. Wooley is on an educational leave of absence and is currently pursuing a Ph.D. in Computer Science at Mississippi State University.