An Interface Agent System for CAE*

Akio MIYOSHI**, Genki YAGAWA***, Ryota SHIMIZU****, Toshimichi TAMURA***** and Shoji SASAKI******

With development of CAD/CAE technology and computer hardware, more design engineers are using the CAE software. However, the CAE software is not an easy tool to use because of its versatility and amount of expertise required to understand and evaluate simulation results. An autonomous interface agent that supports Finite Element analysis was proposed and implemented in CAE software. The agent's four major functions include giving overview of analysis procedure, planning capability, error support and proactive help that considers user's situation and past experiences. The system was usability tested by two groups of student testers of two different majors. Results showed agent support was indispensable for novice users. It also showed that first two functions were effective, that great care must be paid in concluding that a user is lost before activating the fourth function and that the way an error case instance is presented to the user should be refined.

Key Words: Computational Mechanics, Finite Element Method, Artificial Intelligence, Expert System, Interface Agent, CAE, User Interface, Human Computer Interaction, Usability Test

1. Introduction

Today, use of computer aided engineering (CAE) software is rapidly expanding in the manufacturing industry. As the number of people who use the CAE system increases, user group has started to include not only analysts that have expertise in the finite element method (FEM) domain but design engineers. For design engineers, this means addition of a burden of learning to use the CAE software on top of other design related work they already have. Historically, a considerable amount of research has been conducted for the purpose of supporting FE analysts11)–16). Most of research on analysts support seemed to be conducted with an assumption that users prepare input data by hand. However, it is no longer applicable to the current situation because complicated three-dimensional geometrical models prepared by CAD systems are divided into mesh automatically or semi-automatically. Therefore, it is important to research for a method that reduces the design engineer's load. On the other hand, artificial intelligence has created the interface agent, a form of intelligent user interface17)–14), and there is a great expectation about its role as a means to improve interaction between a human user and a computer. In order to reduce the time novice CAE users will take to learn to use the CAE software and to perform FE analyses, the authors propose an interface agent that actively supports them. This paper presents the concept of the interface agent, its fundamental functionality and overview of its implementation in a prototype system, overview and results of usability tests and obtained knowledge.
2. The Interface Agent and Its Functionality

Through analysis of use experience of commercial CAE software and interviews with FE analysts, the authors have thought that one of the reasons for the difficulty of CAE software is insufficient dialog between the CAE software and the user. Figure 1 shows an interface agent that works between the CAE software and the user. Basically, the user carries out his/her analysis talking with the CAE software. The agent stays on the spot, observes the dialog and evaluates the situation. If necessary, it asks the user about his/her intent or it gives instruction. The users will react to the agent by giving their intent or read the on line help. This relationship can be equated with that between a new employee and a trainer in a corporate freshman education program in a Japanese company, which is called ‘on the job training’. The authors propose an interface with this kind of concept and set four basic functionalities.

The actual interface agent consists of multiple subagents as shown in Fig. 2. Subagents collaborate with each other to realize the total functionality of the interface agent. The role of each subagent will be described in the paragraph that addresses the corresponding function.

2.1 System overview

Let us give an overview of the system. The system consists of the CAE module and the agent module. The CAE module is further divided into four subsystems, i.e., the solid modeler, preprocessor, FEM solver and the postprocessor as shown in Fig. 3. The preprocessor’s major function is to generate ten-noded tetrahedral mesh automatically and to attach boundary conditions to the mesh. To implement this function, mesh generation program was written by means of the solid modeler’s application programming interface (API) and was registered as a set of commands to the solid modeler. A Windows like graphical user interface (GUI) was also implemented, and it enables users to send any command to the solid modeler through a dialog box. The commands include solid modeling, finite element analysis conditions, mesh generation, analysis execution and results display. The system also has a window for presentation of solid models. Users interact with it to pick geometrical elements when they create and modify solid models and define analysis conditions.

The agent module was implemented in Java language and tested on Microsoft’s Windows NT 4.0 workstation, Service Pack 4 and 5. The development kit used was JDK1.1.8 and the GUI library was Swing 1.1.1. As an integrated development environment, WebGain’s VisualCafe Expert Edition 4.1 was used.

2.2 Building a mental model

It can be thought that the design engineers who are the target users of the system have little experience in use of CAE software. They can build a good mental model if the interface agent can tell them a correct image of how the CAE software is operated. The good mental model is thought to be effective for a higher work efficiency and control of anxiety in users’ mind. Specifically, the agent has the following functionality: i) it tells a user to read the overview of the general procedure in the Finite Element analy-
sis at the beginning of the session. The procedure is presented as a flow, i) it presents a user results and process of each step in the flow graphically as solid models, mesh and contour plot, which, together with the flow, help users form an overall image of Finite Element analysis in their brain (see Fig. 4), ii) through the total process of analysis, it controls user’s anxiety over the Finite Element analysis by emphasizing important items, de-emphasizing unimportant items and by indicating that a user can go back and forth freely in the procedure flow.

2.3 Planning

In using CAE software, users prepare data and activate the solver when all the data is prepared. Results can be viewed only after solver finishes calculation successfully. Operation of the CAE software is inherently sequential like this. If any one step is missing in this sequence, they can never reach the goal they aim at. Therefore, the nature of user’s work is different from that of other software such as a word processor with which users create an arbitrary object. In Finite Element analysis, planning is thought to be effective as a means of user support. The current system supports user’s planning in the way as shown in Fig. 5. At first, the agent asks users what they want to analyze. Here, a term analysis goal is used to represent what a user want to obtain or do. Users will enter their analysis goal by selecting from available options the agent presents. Then by combining their goal and the knowledge called action models, which include the operation sequence and were prepared in advance for each type of analysis, it generates a plan (how) for this particular session. It shows the next thing to do to users as they go on with their analysis.

The subagents associated with this functionality are the Goal Manager, the Planner and the Progress Manager. The Goal Manager obtains a set of analysis goals from a user, translates them into a simplified language, “Goal description language” and sends them to the Planner. The Planner, in turn, receives the analysis goal from the Goal Manager and generates a plan by means of backward inference applying action models as rules. Figure 6 shows how a plan is generated from goals and action models. The analysis goal is a collection of subgoals. An action model consists of a postcondition, actions and preconditions. A postcondition is actually a subgoal, and if certain preconditions are met, and actions are taken, the goal represented by the postcondition is achieved. The agent searches all the action models for a postcondition that is the same as the subgoal currently processed. If a postcondition is found, preconditions and actions associated with the postcondition are inserted into the analysis goal to replace the current subgoal. This process is recursively repeated until all the elements in the updated analysis goal become simple commands. The final form of the analysis goal is the plan users need. The Progress Manager observes progress of the user and asks the Planner to present what to do next when the user is ready for a next action (See Fig. 7 for a screen shot).

2.4 Error support

Errors do occur in any engineering and scientific software. Sometimes the situation can be solved by the user, sometimes not. If a solution to the trouble can be obtained at all, the proposed agent will store
the set of the error ID and the solution and present them to a user when a same error happens again. Figure 8 shows how the agent supports a user when an error occurs. As soon as a user logs in, the agent starts observing the user's operation continuously. If an error occurs due to a user's operation, the agent stores the history of user's operations for a certain number of steps. It then searches the database for a similar sequence that caused the same error as the current one. If a similar case instance is found, the agent presents solutions associated with the instance to the user. If no similar case instance is found, it keeps observing the user's ongoing troubleshooting effort. If the troubleshooting is found to be successful, the agent removes trials and errors processes from the stored sequence and obtains net sequence that leads to the solution. The net sequence is then registered in the database. Even if a similar case instance is found in the database at the moment an error occurred, the agent continues observation of user’s efforts. During this time, users do not need to bother to provide any information about the recovery process to the agent. Once they find a solution the agent already knows it.

The subagents that are involved in this functionality are the User Situation Manager, the Casebase manager and the Help Agent. The User Situation Manager accesses the User State and obtains sequences of user's operations. It also detects an error and notifies it to the Casebase Manager. The Casebase Manager detects user's success in troubleshooting and extracts net operation sequence for error recovery. Moreover, it has a capability to register and look for a case instance in the casebase. The Help Agent presents past solutions to error recovery based on the request from the Casebase Manager.

2.5 User model, situation model and proactive help system

Sometimes users cannot proceed for themselves even if they use the agent's functionality described above. However, even in that case, they do not necessarily know that they are faced with a problem. They do not always actively read the on-line help for a solution. Even if they do try a search in the on-line help, they do not always find appropriate keywords to retrieve information they need. The proposed interface agent is capable of providing information necessary to users just when they need it.

As illustrated in Fig.9, the agent monitors users' behavior by watching mouse movement, mouse clicks and keyboard inputs while they manipulate the CAE software. If the users' operation is delayed for a certain period of time or if they keep browsing menu items, it concludes that they are in trouble. In a new small window, the agent shows them a prioritized list of help information items based on the situation at that time, users' past operation history and general users' trend. The rating of each information item's
necessity is determined in the following manner. First of all, each item is evaluated from the viewpoint of necessity at the moment (called Class 1). Say, if a dialog box for creating a new geometrical model is open, information for creating a new model is likely to be useful more than any other information. Secondly, the current user's past knowledge with this software is evaluated (called Class 2). For example, if a user had read information for setting displacement constraints a few times, the user will probably no longer need it. Thirdly, information items are evaluated based on the necessity in general (called Class 3). For example, if a knowledge item for setting parameters for designating node distributions was read many times by other users, the knowledge is likely to be useful for the current user too. The summation is taken over all the three classes to calculate the necessity of each help information item,

\[ P = \sum_i \Delta P_i \]

where \( P \) is the total point for an item, and \( \Delta P_i \) is a contribution from the Class \( C_i \). This way of delivering information is called 'push' or 'push technology' where the system sends information to a user without the user requesting it. The push agent hides itself if the user does nothing to the window for a certain period of time. This is to avoid the user's becoming indifferent to the information source by keep showing it and to make it more noticeable when the information is refreshed.

The subagents associated with this functionality are the Idle Checker, the User Situation Manager and the Help Agent. The Idle Checker determines if the user's operation is suspended from the user behavior data the User Situation Manager provides. The User Situation Manager handles the user behavior data, determines the user level and rates the help information items. It asks the Help Agent to present help items to the user. Figure 10 shows a screen shot where agent's proactive help is provided.

3. Usability Test\(^{130}\)

Five usability tests were performed as system development advanced. Here, the forth test (called Test A) and the fifth test (called Test B) are reported.

3.1 The objective of the test

Usability test was chosen as the form of evaluation of the system. The objective was to find the system's problems and the validation of the agent functionality.

3.2 The Procedure of the test

Test A was a day and a half long and was conducted with twenty-six testers in April 1999. Twenty-four of them were either senior students or master's course students with mechanical engineering major. Remaining two were faculty members. All of them knew the word Finite Element Method. Twenty-three percent of them had an experience of FE analysis, and thirty-one percent had taken an FEM course, and fifteen percent had made an FEM program and twelve percent had used a 3D CAD or 3D computer graphics software before.

Twenty-six testers were divided into nine groups with an average of three members and tests were performed with a three-hour time limit. Three parallel sessions were conducted three times. The machine power for the three PCs was Pentium II 300 MHz with 256 MB memory, Pentium 133 MHz with 128 MB memory and Pentium 166 MHz with 128 MB memory respectively. One of the authors attended the test as an observer. Basically, testers were supposed to perform an analysis task only with the agent's support. However, if the test observer thought that the session would overrun the allowed time, he gave them a hint. During the session, testers' behavior was recorded on video. The software also logged their commands. The observer watched their behavior and wrote down what he noticed. If he thought necessary, he used contextual inquiry. While the testers were waiting for the system's response, they answered a questionnaire.

\begin{table}
\centering
\begin{tabular}{|l|c|c|}
\hline
Item & Test A & Test B \\
\hline
Tester's major (year) & Mechanical Eng. (senior, master) & Quantum Eng. and Systems Science (junior) \\
\hline
In volunteer & yes & ne \\
\hline
Number of times (tests) & 9(90) & 10(29) \\
\hline
Allowed time (hrs) & 2 & 3 \\
\hline
Has conducted FE analysis & 3% & 7% \\
\hline
Has taken FEM course & 3% & 17% \\
\hline
Has written FEM programs & 3% & 17% \\
\hline
\end{tabular}
\caption{Comparison of testers in two tests}
\end{table}
Test B was conducted continually for eight days starting late May to early June in 1999. Testers were all junior students who took a course of computational mechanics at the department of Quantum engineering and Systems Science. Table 1 shows the comparison of testers in the two tests. As addressed in the table, difference in the major and in the year where the students were resulted in a significant difference in knowledge and experience in FE analysis and 3D CAD and a motive for participation in the test.

In this test, twenty-nine testers were grouped into sixteen teams with one to four members. The allowed time was three hours. One or two teams tested the system at a time. The machine power for the two PCs was Pentium II 300 MHz with 256 MB memory and Pentium 133 MHz with 128 MB memory. The way the observer was involved in the test was the same as in Test A.

Since the system was still in the development phase, minor improvements were made to the system after Test A was finished. Particularly, changes were made so that the small agent window that appears for pushing help information is closed automatically if the user does not access the window for some period. Therefore, comparison between two tests will no be made in a strict sense here.

3.3 The task of the test

A common analysis task was used. A bicycle crank arm was selected as the object of analysis. The task was to create a 3D shape of the part as indicated by a drawing, to perform stress analysis with a given material and design load and to check if the computed margin of safety as given in Eq. (1) is positive. If time allowed, testers were requested to make efforts to design a lighter part.

\[
MS = \frac{S_u}{FS \times S_{\text{max}}} - 1
\]  

where \(MS\) is the margin of safety, \(S_u\) is the ultimate stress of the given material, \(FS\) is the factor of safety, and \(S_{\text{max}}\) is the maximum induced stress in the part when a design load was applied, respectively.

3.4 The evaluation method

The questionnaire was prepared based on Ref. (19). The first part of the questionnaire consists of questions about background knowledge and experience about the FEM and the CAD. It also asks about the tester’s general skill level in computer manipulation and preferences. The second part requests the tester to evaluate agent’s each function, the interface agent as a whole and each step in the operation flow and asked for comments. Except for the first set of questions, testers were asked to rate the agent on a seven-point scale where four is average. Testers were requested to answer the questionnaire while they wait for the system’s response so that they would not forget the details of their situation. The total number of the questions was set to be less than one hundred. The average score of all the testers was calculated for each function to evaluate its validity.

Observations during the test were sorted out to find the system’s problem. They were matched against corresponding items in the electronic log and their mistakes were corrected and missing information was added. Efforts were made to identify the reasons for the testers’ behavior. On the other hand, for the team for which session was recorded in video, testers’ behavior was intensely followed while the session was played back on the display. System problems detected in this work were classified and their frequency was counted.

4. Test Results

In Test A, a sixty-seven percent of the nine teams arrived at the final step of displaying analysis results within the allowed time. A twenty-two percent went further and halfway through a weight reduction study. An eleven percent finished an analysis for a modified shape but no team arrived at displaying results for the second analysis. Advice was given to testers an average of 4.9 times before a reasonable result was obtained or if no result was obtained at all, before the allowed time was over.

A forty-four percent of the sixteen teams in Test B succeeded in obtaining and displaying a reasonable result. A twenty-five percent was halfway through a weight reduction study when the allowed time was over, and a nineteen percent could obtain and display a refined result. Average number of times advice was given to a team was 5.6.

4.1 Results of the questionnaire

Table 2 shows the summarized results of the questionnaire for the two tests. The capability of describing the overview of the FE analysis and planning received an affirmative evaluation for both of the

<table>
<thead>
<tr>
<th>Table 2 Results of the questionnaire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Overview of FE analysis</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Planning support</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Proactive help</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Error support</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
tests. Evaluation of proactive help for Test A was a little better than average. Test B, however, gave a little poor evaluation except for the understandability. Error support received a poor evaluation for both of the tests and it is thought that there is a problem to be solved.

The difference in ratings between the two tests is thought to be due to the difference in the knowledge and experience of FE analysis that the testers had. The more knowledge and experience testers have, the more easily they understand the system. Therefore, the higher ratings.

4.2 Observations and discussion

The most remarkable point the observer noticed during Test A was that testers could not possibly reach a solution within the allowed period of time if there was not an interface agent to support them. The observer felt so because groups that read the agent's guidance more carefully proceeded faster than those who did not pay much attention to the agent's messages. Actually, a tester made a similar comment while he was answering the questionnaire. However, the system was still incomplete from many points of view. Most of the testers spent more time for 3D geometry modeling than for setting analysis conditions. But help information explaining the concepts of specific geometry manipulations was insufficient. For example, current modeler does not allow placing a primitive geometry at an arbitrary location at the time of generation. One must generate a primitive at the origin and then translate it to some location. Current help information does not refer to the fact and some of the testers kept trying to do it with one command in vain.

In Test B, agent's support was also indispensable. Proactive help capability received lower ratings than Test A because of the following inconveniences: i) the agent closed its push window because there was no testers' action on it for some period of time and concluded that testers did not need it at that time although they were still reading help information; ii) push agent appears more often than being comfortable to testers; iii) the agent disturbed them while they are entering analysis conditions or while they are converting load values by calculators. At this time, testers' response was often emotional and in a few cases, the system's unsatisfactory behavior repeated and caused testers to respond more emotionally.

The problem of agent appearing too often could be solved if the criteria for deciding whether the users have difficulty is changed from the period of time during which users do nothing to the software to some event that is strongly associated with user's difficulty. It is because doing nothing does not necessarily mean users have a trouble. The authors think that one could similarly solve the other problem in which push agent disappears too early by using events for the criteria for closing the agent's window.

At the time of error notification by the interface agent, users could read past cases only when they click a button labeled "past error information". However, almost no users took the trouble to read it. It is thought that there was not sufficient explanation about the services that the agent provides. It could also be inferred that users build a mental model that says error messages from computer systems are generally difficult to understand. In the message box for error notification as many as three exclamation marks were placed and the background color was vivid yellow. This might have intimidated users. If errors are less emphasized and the benefit of reading provided information is emphasized, more users are expected to read the help information.

5. Concluding Remarks

The authors proposed to integrate a CAE system with an interface agent that has a capability to support design engineers in performing Finite Element analyses. A prototype system was implemented on a Windows PC and was evaluated by a series of usability tests. As a result, it was known that testers were able to construct a simple geometrical model and perform a linear elastic stress analysis with a support from the interface agent and some advice from the test observer within three hours. The following shows the obtained knowledge:

i) In a test where testers were all volunteers and a thirty percent of them had taken a course of the FEM, the interface agent was evaluated as a little better than average except for the error support functionality.

ii) In a test where all testers were non-volunteers and a seventeen percent of them had taken a course of the FEM, the agent was evaluated as a little better than average for the functionality of presenting overview of the FE analysis procedure and planning functionality. Push help functionality received a little worse than average rating and the evaluation of the error support functionality was poor.

iii) Reason for the poor rating was studied and modified agent specifications were proposed.

Acknowledgements

The authors would like to thank former Professor Dr. N. Maeda of the Department of Quantum Engineering and Systems Science, the University of Tokyo and Associate Professor Dr. T. Tsuji of Department of Mechanical Engineering, Shizuoka
University for their cooperation in execution of usability tests. Cooperation of Ricoh Company, Ltd. in this work is also acknowledged.

References


