DOCUMENT RESUME

ED 388 285 IR 017 434

AUTHOR Pulz, Michael; Lusti, Markus
TITLE Case-Exercises, Diagnosis, and Explanations in a Knowledge Based Tutoring System for Project Planning.
PUB DATE 94
NOTE 7p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programming); *Case Studies; Computer Assisted Instruction; Educational Technology; Evaluation Methods; Foreign Countries; *Intelligent Tutoring Systems; Knowledge Representation; Learning Activities; *Problem Solving

ABSTRACT

PROJECTTUTOR is an intelligent tutoring system that enhances conventional classroom instruction by teaching problem solving in project planning. The domain knowledge covered by the expert module is divided into three functions. Structural analysis, identifies the activities that make up the project, time analysis, computes the earliest and latest start and finish times of each activity, as well slack times and critical paths, and cost analysis computes the minimum project time and the project costs by crashing all the activities. PROJECTTUTOR consists of two subsystems: the authoring component and the tutoring component. The authoring component, which offers natural language text and graphical network representations, is used by the teacher to specify case problems; a case problem consists of a text describing the problem and a framework describing the structure of the problem. The student uses the tutoring component to work on the case exercise. At any time, the student can ask the system to diagnose work, give hints for further solution steps, check whether a solution is correct, explain errors, or produce a (sub-)solution. A qualitative evaluation was carried out, showing that the students appreciated the advantages of the system compared to paper-and-pencil exercises. (Contains 11 references.) (AEF)

Case-Exercises, Diagnosis, and Explanations in a Knowledge Based Tutoring System for Project Planning

MICHAEL PÜLZ, MARKUS LUSTI
Institut für Informatik
Universität Basel, Petersgraben 51, CH-4051 Basel, Switzerland
email puelz@urz.unibas.ch

Abstract: PROJECTTUTOR is an intelligent tutoring system for project planning. As a problem solving monitor, it completes conventional classroom teaching by training problem solving in project planning. The teacher uses the authoring component of the system to specify case problems. A case problem consists of a text describing the problem and a framework describing the structure of the problem. On the basis of an author-defined case text, the authoring component constructs a formal representation (an MPM-network), representing the structure of the problem. The student works with the tutoring component of PROJECTTUTOR. The text of a case problem is presented and the student has to find the structure of the project, build up an MPM-network, and perform time and cost analysis. The system compares the student's solution steps with those generated by the expert module. At any time, PROJECTTUTOR can give diagnosis, hints, and explanations, referring to the domain knowledge of the expert module as well as to the specific case.

Research objectives

The purpose of our research is to build a knowledge based tutoring system to improve *problem solving* in project planning (Pülz & Lusti, 1994). PROJECTTUTOR is able to diagnose student errors and to explain the solutions generated by the expert module. It explicitly uses information in the problem text to enhance explanations. In Operations Research, textbooks usually include case studies. The students solve the exercises using paper and pencil. Eventually they can check their own solutions against the solutions found in the appendix. Textbooks do not give any hints, explanations or diagnoses on the student's work.

PROJECTTUTOR can be used in any environment where project planning is taught. In our opinion, Operations Research is a well suited domain for tutoring systems. The curriculum is stable, structured, and algorithmically tractable (Angelides & Doukidis, 1990). In a study by Belling-Seib, the use of computer technology in the curriculum of Operations Research in Germany, Austria and Switzerland was investigated (Belling-Seib, 1991). It shows that Operations Research is a field where teachware is already used frequently, although most programs are just tools for computing solutions of given models. The purpose of PROJECTTUTOR is to increase the students' ability to develop a formal model from a given case problem and to work on the model until the problem is solved in the intended way.

The curriculum

Figue 1 describes the educational goals of PROJECTTUTOR. As prerequisites the students need the basics of graph theory and project planning. The domain knowledge covered by the expert module is devided into three parts. The *structural analysis* identifies the activities that make up the project. Each activity is associated with an average duration and optionally minimum duration and crashing costs. Next, the connections between the activities are identified and drawn using directed arcs connecting pairs of activity nodes. There are different types of connections (start-start, start-end, end-start, end-end), depending on whether a connection refers to the start or finish times of the associated activities. MPM-graphs usually use the start-start connection type. Connections can either be positive, in which case they are labelled with a positive number, saying that an activity

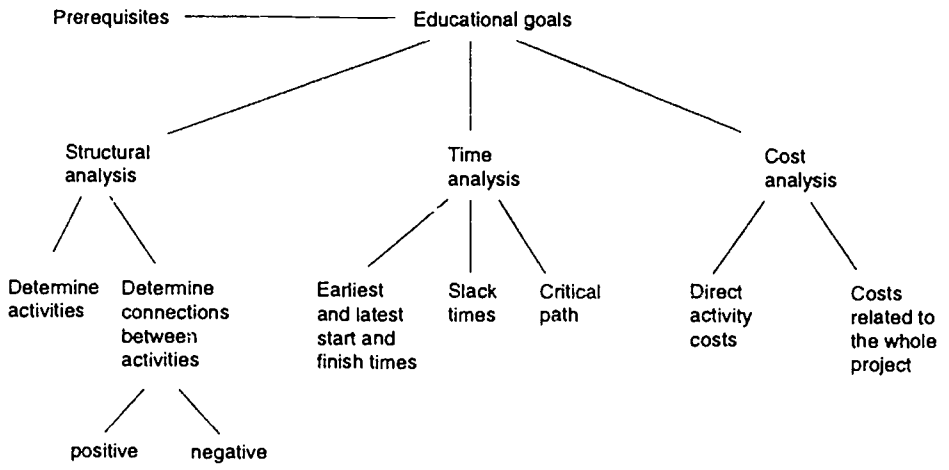

Figure 1: Hierarchy of educational goals

can start *no earlier than* a certain amount of time after the start or the end of the preceding activity, or they can be negative when labelled with a negative number. A negative number indicates that the *latest start* of an activity is a certain amount of time after the start or the end of the preceding activity.

Time analysis computes the earliest and latest start and finish times of each activity, their slack (float) times, and the critical path(s). *Cost analysis* computes the minimum project costs by using the slack times of the non-critical activities or determines the minimum project time and the project costs by crashing all the activities. Different types of costs, such as direct activity costs, costs associated with the whole project (both of which are variable costs), and costs that are not relevant for the crashing of activities (fix costs), have to be identified.

The system

PROJECTTUTOR consists of two subsystems, the *authoring* and the *tutoring component* (Lusti, 1992). The intention of the authoring component is to make the system more flexible. The teachers can specify their own case-exercises. The authoring component is the knowledge acquisition tool of the system. In PROJECTTUTOR only case knowledge can be acquired. The domain knowledge of the expert module is compiled and therefore static. The tutoring component is used by the students to work on the author defined case-exercise.

Knowledge acquisition and internal representation (authoring component)

The authoring component allows to edit the case base. A case-exercise is first described by an unstructured text. The teacher then overlays the text by marking relevant pieces of text with different colours. Similar ideas of analysing the information contained in prescriptive texts by using NLP methods have been described before (e. g. Moulin & Rousseau, 1990). Our system does not use NLP methods. This is because the author is free to type any text he/she wants. In particular, no keywords have to be used. Analysing such texts automatically is still very difficult. Instead, the author has to explicitly tell the system which piece of text contains which information. In the following example, the bold piece of text indicates an activity: "The **planning of the gardens** can start no earlier than 10 days after ..."

As an example, figure 2 shows a problem text in the window on the left side of the screen, including a marked text piece denoting a negative connection. Each marked piece of text has one of the following meanings: activity, average duration, minimal duration, crashing costs, connection, time requirements of a connection, variable project costs, or fixed project costs.

Marked text pieces are immediatly transformed to an internal representation reflecting the project structure (see figure 3). The representation is a list of text pieces that provides for the insertion of attributes like "activity", "duration", or "connection". Moving text preserves the attributes. Deleting text is only possible under *integrity constraints*. If, for example, the only piece of text about a certain activity is deleted, the system warns the author. He/She can then specify another piece of text or remove the activity from the network.

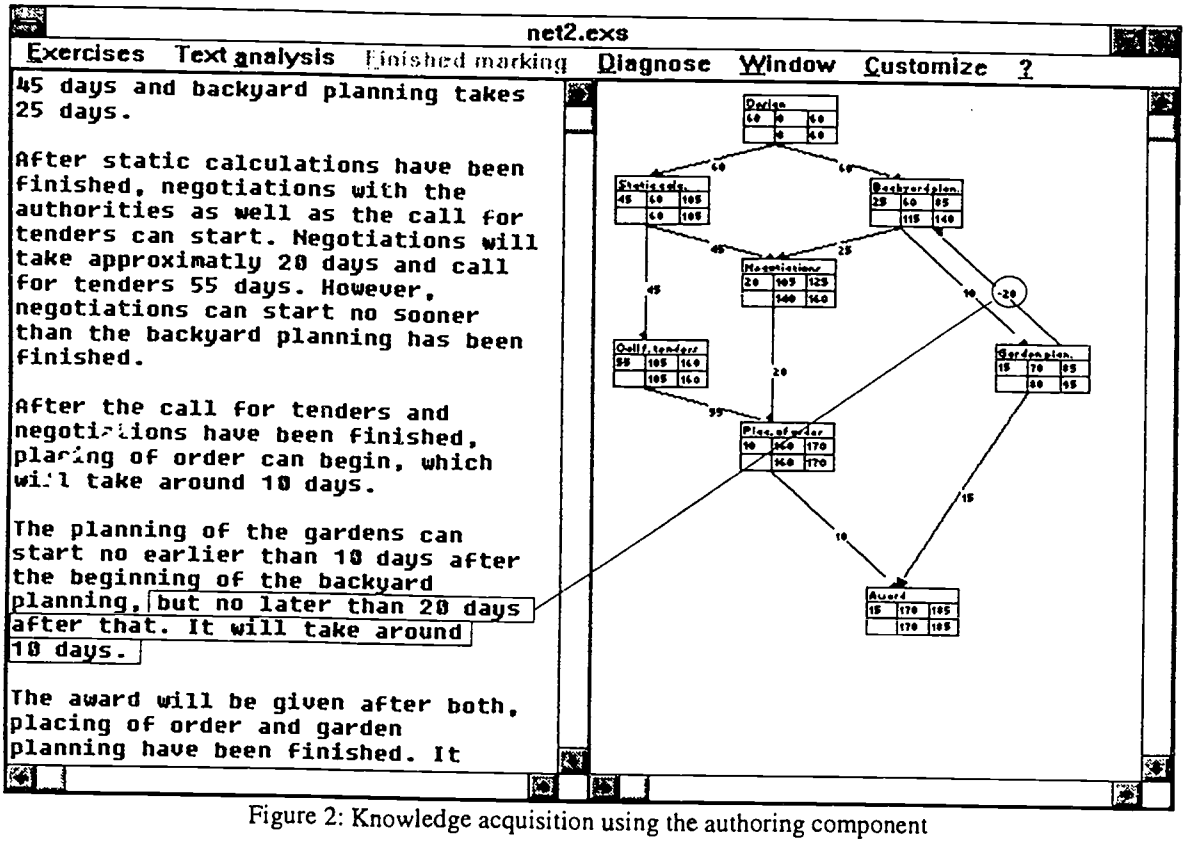

Figure 2: Knowledge acquisition using the authoring component

After adjusting the internal datastructure, the system automatically updates the *formal representation* in the network window on the right side of the screen by generating a corresponding activity-object (a node), a connection between a pair of objects (an arc), or time and cost data (see figure 2). The author can see the effects of his/her text analysis on the graph. Furthermore, the system detects inconsistencies within the author's definitions. The internal datastructure is stored in the case base and is used by the explanation facility of the tutoring component.

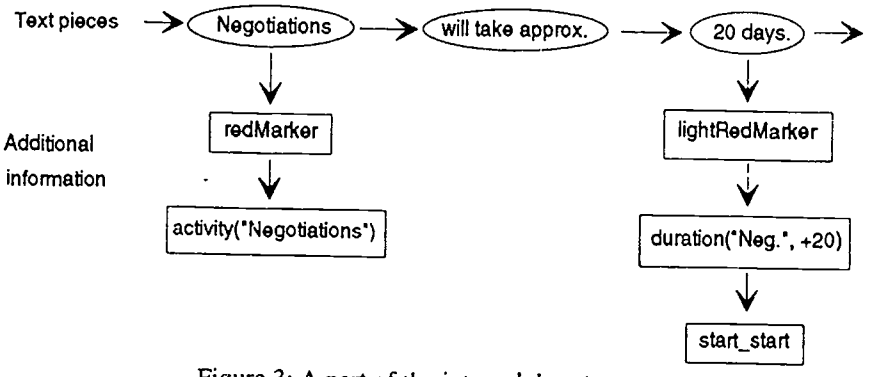

Figure 3: A part of the internal datastructure

Tutoring component

When the students work on case-exercises, PROJECTTUTOR does not explicitly lead them through a chain of tutoring steps, but they have to decide by themselves which step to take next. However, there is an implicit tutoring process since the students have to follow a certain sequence of tasks in order to arrive at the correct solution. At any time the students can ask the system to: diagnose the work the students have done so far, give

hints for further solution steps, check whether a (sub-)solution is correct, explain why a (sub-)solution is not correct, or produce a (sub-)solution.

PROJECTTUTOR tries to behave like a human tutor who is available while students work on a problem. The students can ask the tutor and the tutor will give comments, hints, or explanations. The tutor has to decide which answer to give. In the current version, no explicit pedagogical decisions are made. Instead, every question that the students ask generates a local reaction of the system, consisting of a how-explanation trace (a tree) and references to relevant information in the problem text. By navigating through the tree an explanation dialog is carried out and links to the exercise text are shown.

Diagnosis means checking for discrepancies between the student's and the system's solution. This can be done at any time and consists of determining the current state of the student's work and checking the student's current (sub-)solution against the one generated by the expert module. If an error is detected, the system finds out whether the error occured because of incorrect network construction or incorrect transformation of the network.

Incorrect network construction can be due to misinterpretation or overlooking of information contained in the problem text. Students might forget to draw an activity node or a connection between two activities. Or they draw a connection in the wrong direction or between the wrong nodes. The system points out the misinterpreted or overlooked information in the problem text, allowing the student to improve the network.

Errors in transforming the network can be caused by different kinds of mistakes. There are different cognitive levels involved. An error might simply be due to a wrong arithmetic calculation, which is on a low cognitive level and can be easily recognised by the student. The system can simply give a hint pointing out the error.

At a higher level, project-planning algorithms to be performed on the network might be carried out incorrectly. Those algorithms are, for example, computing the earliest and latest start and finish times, the slack times, and the critical path within the time analysis. Here it is necessary to trace back the operations performed by the student. PROJECTTUTOR can then mention the erroneous step. The student might either recognize his/her mistake and repeat the relevant steps or, if there is a problem in understanding the algorithm, the student can ask the system to demonstrate the algorithm step by step. If necessary, the student can invoke the hypertext help facility to look up textbook knowledge explaining the algorithm and domain terms.

If the system detects that the student made the mistake because of overlooking or misinterpreting information in the exercise text, it refers to the corresponding text pieces.

The *explanation component* comprises explanations generated by the domain knowledge of the expert module and the knowledge about the meaning of the text stored in the internal representation of the problem text. As many references as possible are made to the problem text. If the student makes a mistake and asks for a diagnosis, the system responds by highlighting a relevant piece of text in the case text and displaying additional information.

In the following, a dialog between a student and the system is outlined. Computer output is written in italics and comments are put in brackets. (In fact, the dialogue is menu driven):

Why is the latest start time of the garden planning not 155 days?
(The student asks for a diagnosis and the system highlights the occurrence of a mistake in the network.)
Because there is a mishap in the structural analysis.
(The system realized that the mistake is caused by an error in an earlier part of the educational goals, which is the structural analysis; see figure 1.)
Why?
(The student asks for a more precise explanation.)
The connections in your network are not correct.
(The system moves to the next level in the explanation tree, but still keeps the explanation somewhat general. The explanation resembles a hint.)
Why?
(The student asks for further explanations.)
*Look at the following information in the problem text: "... The planning of the gardens can start no earlier than 10 days after the beginning of the backyard planning, **but no later than 20 days after that...**".*
(The system highlights the corresponding piece of information in the exercise text which was overlooked by the student. Here the explanation trace is explicitly connected to the problem text.)
How do I specify such a relation in the network?

(The student does not know how to incorporate this information in the network and asks the system for further help.)
You should use a negative connection.
(The system reached the lowest level in the explanation tree.)
What is a negative connection?
(The student does not know that term and asks for more help.)
A negative connection is...
(The system starts the hypertext help facility.)

We believe that *explanation traces* generated by the domain knowledge of the expert module should be *accompanied by references to the problem text.* Other research comes to similar conclusions (Reiser, Ranney, Lovett & Kimberg,, 1989). Studies on mathematical text problem solving found that unstated or hidden relations in the problem texts are a major source of errors (Nathan, 1993). A system should also be able to present different representations of the problem. PROJECTTUTOR provides two views of a case problem. One is the unstructured exercise text and the other is the formal network representation.

Communication module

The communication module of PROJECTTUTOR is crucial for the acceptance of both its authoring and tutoring component. The nodes and arcs can be moved around by dragging and dropping. At any time, the students can ask questions. *Direct manipulation* lets the students (as well as the teacher) see the effects of their work immediatly (LeBlanc, 1993). If the students specify a link between two activities, this link is drawn immediatly in the network. The same is true after an author has marked a piece of text indicating a link between activities.

Matching the student's work with the system's work

As described earlier, the curriculum is devided into different parts (see figure 1). There is a certain order in these curriculum parts. For example, the student has to correctly finish the structural analysis before carrying on with the time analysis. However, the system allows the student to work on the time analysis, even if the structural analysis has not been finished correctly. The idea here is to let the student find out about the error in the earlier phase (*explanation by contradiction*; Rätz, 1993; Rätz & Lusti, 1992). The student then has to go back to the structural analysis and correct it. Since the time analysis is based on the structural analysis, the work that has already been done in the time analysis is lost. The student has to redo work that he/she might have already proved to know. In those cases, PROJECTTUTOR can do this work for the student. This is in order to keep the student concentrated on crucial problems and to automize minor ones.

When working on project planning exercises using paper and pencil, the students often have to recompute a whole network after they realized that the structure of the network is not correct. This is tiresome and demotivating.

PROJECTTUTOR keeps track of the student's work in a simple way. Once a subtask has been completed correctly, e.g. computing the slack times, it is marked as known by the student. If the same subtask has to be performed again at a later time, PROJECTTUTOR offers to do it for the student.

PROJECTTUTOR also compares the student's work with the one done by the expert module (see figure 4). A matching algorithm is implemented that detects missing, wrong, and incorrect items in the student's work. Based on this information, diagnosis and explanations are built up.

Discussion and further refinements

We described PROJECTTUTOR, a monitor for training problem solving in project planning. The curriculum and the levels of competence were mentioned. We discussed the different moduls of the system, stressing the knowledge acquisition part, the tutoring component, and the communication module.

PROJECTTUTOR includes an authoring component which offers two consistent representations to the teacher: the natural language text and the graphical network representation. The tutoring component provides a powerful explanation facility (incorporating the problem text and the explanation traces generated by the expert module).

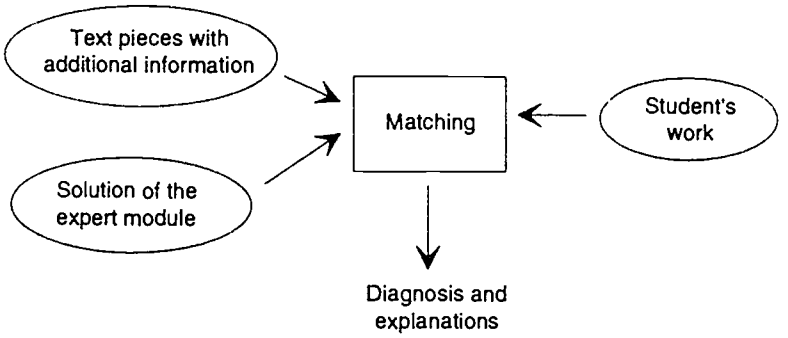

Figure 4: Matching the student's and the system's work

A qualitative evaluation was carried out, showing that the students appreciated the advantages of the system compared to pure paper-and-pencil exercises. However, further improvements have been pointed out in several directions. An extension of the curriculum to include CPM- and PERT-techniques, capacity-, milestone-, and trend-analysis would improve the usefulness. Case problems could be extended to vary during the exercise, e. g. the system could tell the student that a certain activity is delayed and the student has to react. The system would then become more of a simulation tool. Student modelling could be extended and pedagogical knowledge could be added so the system can adapt better to the individual student.

PROJECTTUTOR is written in PDC Prolog and runs under Windows 3.1.

References

Angelides, M. C., Doukidis, G. I., Is There a Place in OR for Intelligent Tutoring Systems?, *Journal of the Operational Research Society* 41 (6) 491-503, 1990

Belling-Seib, K., Educational OR software/Computer assisted OR education, *Methods of Operations Research* 64, 645-647, 1991

LeBlanc, M. D., From natural language to mathematical representations: A model of 'Mathematical Reading', in NWAN93, 126-144

Lusti, M., *Intelligente Tutorielle Systeme: Einführung in wissensbasierte Lernsysteme*, Bd. 15.4 des Handbuchs der Informatik, München 1992, Oldenbourg

Moulin, B., Rousseau, D., Structuration d'une base de connaissances a partir du contenu de textes prescriptifs, in *Tenth International Workshop Expert Systems & their Applications, Specialized Conference on Natural Language Processing & ITS Applications*, Avignon, May 28-June 1st, 1990

Nathan, M. J., A simple learning environment improves Mathematical reasoning, in NWAN93, 162-186

Nwana, H. S., *Mathematical intelligent learning environments*, Oxford 1993, intellect

Pülz, M., Lusti, M., *Ein wissensbasiertes Lernsystem zur Projektplanung*, WWZ-Discussion Paper, Nr. 9402, Universität Basel, Basel 1994

Rätz, T., *Erklärungen in wissensbasierten Lernsystemen am Beispiel eines Tutors zur Normalisierung von Datenbanken*, Frankfurt 1993, Lang

Rätz, T., Lusti, M., Explanation Strategies: Realization in a Tutor for Database Normalization, in Brézillon, P. (ed.), *Proceedings of the European Conference on AI, Workshop W15: Improving the Use of Knowledge-Based Systems with Explanations*, Institut Blaise Pascal, Université de Paris, Rapport 92/91, 1992 47-56

Reiser, B. J., Ranney, M., Lovett, M. C., Kimberg, D. Y., Facilitating Students' Reasoning with Causal Explanations and Visual Representations, in BIER89, 228-235