Experiments with an Expert System Interface

Larsson, Jan Eric; Persson, Per

1987

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT–3196)/1–11/(1987)

Experiments with an Expert System Interface

Jan Eric Larsson & Per Persson

Department of Automatic Control
Lund Institute of Technology
September 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Final report

Date of issue
September 1987

Document Number
CODEN: LUTFD2/(TFRT-3196)/1-11/(1987)

Author(s)
Jan Eric Larsson
Per Persson

Supervisor

Sponsoring organisation
The National Swedish Board for Technical Development, (STU), contract no. 85-3042

Title and subtitle
Experiments with an Expert System Interface

Abstract

This is the final report of the project "Experiments with an Expert System Interface." It describes an expert system interface for system identification, when using the interactive identification program Idpac. The interface works as an intelligent help system, using the command spy strategy. It contains a multitude of help system ideas. The knowledge representation is taken care of with scripts and rules. Scripts are used to describe the procedural part of the knowledge in the interface. Production rules are used to represent diagnostic knowledge. Conclusions of the project are given.

Key words
Expert Systems, Help Systems, Intelligent Front-Ends, Man-Machine Interfaces, Scripts, System Identification

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

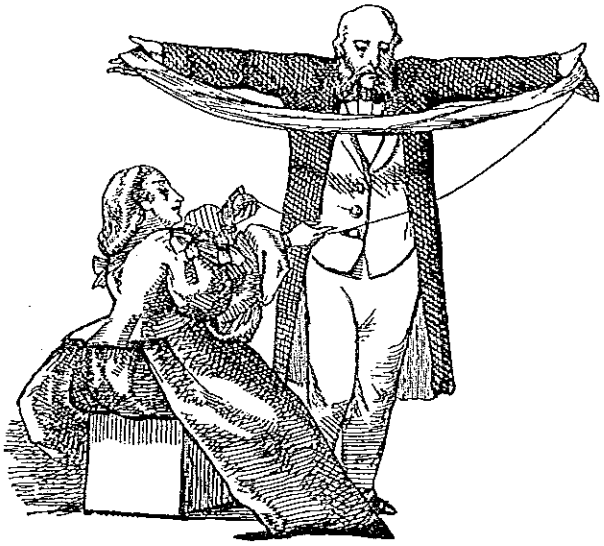
Language
English

Number of pages
11

Recipient's notes

Security classification

Experiments with an Expert System Interface



`(ihs)`—Help in every situation

Experiments with an Expert System Interface

Final Report to STU
Contract number 85–3042

Jan Eric Larsson
Per Persson

Department of Automatic Control
Lund Institute of Technology
September 1987

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

Published 1987
Printed in Sweden

Contents

Introduction

Knowledge-based computer programs, expert systems, are rapidly being developed and will probably be quite common in the near future. When the techniques used in these programs grow more reliable and become better known, it will be obvious for a CAD program to use them. This project has been aimed at using knowledge-based programming techniques to build an intelligent help system, and building a small knowledge database for system identification. The target program used is Idpac, an interactive, command-driven program package for system identification.

The project is part of the Computer-Aided Control Engineering project, running at the Department of Automatic Control at Lund Institute of Technology. An intelligent help system like the one described here would be an essential part of any modern man-machine communication. But in addition to this, the project has provided additional input to the CACE project. It has given valuable knowledge about the design and use of knowledge-based programs and programming tools. Several demands on the design of high-level problem-solving languages have been stated. Finally, the project has shown the need for designing open and modular tools instead of closed program packages like Idpac.

Description of the Problem

The project has been centered around the solution of the following problems.

An inexperienced user often has a general idea of what he wants to do, but does not know exactly how to do it. He will need guidance. This is not taken care of by an ordinary "list all available commands" help system. Therefore a *goal related* help facility should be available.

Sometimes a user does not wish to have any help, or the help system has no help to give. In such cases help should not be forced upon the user. For this reason the help system should be totally non-invasive, i.e., only come into action on the user's request.

Idpac uses a flexible and powerful command dialog and several users have already learnt to use it. Therefore this communication should be kept when the expert system is added, instead of using a question and answer dialog. To do this a flexible and easy-to-use front-end must be developed and interfaced to an expert system framework.

A large part of the knowledge of an expert interface for running Idpac concerns sequences. Sequences may be represented using production rules to implement a state machine, but this will become cumbersome when the number of states grow. Therefore we have introduced the concept of *scripts*, as a data structure for describing sequences.

The help system needs not only procedural knowledge about Idpac, but also knowledge of system identification in general. This is required for it to be able to diagnose problems, estimate the validity of results and propose further tests. This function can be taken care of by an ordinary production rule system.

Part of the practical difficulties of running Idpac is keeping track of details, such as remembering file names, what operations have been performed on what data, values of parameters, and so on. The help system should aid the user in this and keep the information about details in a database.

One important task for the help system is to support retrieval of knowledge from the knowledge database to the user. Several facilities for this must be provided. The system should help the user with interpreting results and suggest what to do next. It should prompt for parameters in commands and give thorough explanations of them in the process. There is also an on-line dictionary to explain the vocabularies of system identification and Idpac.

The Research Programme

The project originally started with a masters project, Larsson [1984], Larsson and Åström [1985 a]. In this, an overview of the problem area was given and several demands on a solution given. A small system based on a standard expert system shell using backward-chaining was also written, and a small knowledge database developed.

The masters project provided valuable inspiration for the STU project. First, a small prototype was developed. It contained the basic ideas of a database built with both scripts and rules, and a non-invasive command spy. This prototype has been described in Larsson and Persson [1986 b]. Experiments with this program showed that the concepts were good, and thus provided the base for the next phase.

After the testing of the prototype, a full fledged system was implemented. This effort formed the main part of the project. This implementation has been described in Larsson and Persson [1987 a]. The result is a system useful for demonstrations and testing by selected users. It has also been used in the undergraduate course "Process Identification," given in the fall of 1987. However, to make the system into a robust product would still be another major effort.

In the final stage, a knowledge database for a realistic example was developed. This enables the system to handle a session of parameter estimation with the maximum likelihood method. A technical report describing the knowledge database was written, Larsson and Persson [1987 c], in addition to the user's manual, Larsson and Persson [1987 b].

Description of the Implemented System

The implementation of the expert interface has formed a crucial part of the project. It contains all essential parts, as a command parser, script matcher, and production rule system. In addition to this there are several utilities that must be present if the program is to work as a realistic example. These are a query module, a file system, and interfaces to the user and Idpac. There are some limitations, though. The interface currently handles only a subset of all the Idpac commands, and there are some irregularities in the syntax of some Idpac commands that are not fully supported.

The methods used include an extensive use of formal language definitions, compiler techniques, standard expert system shell techniques, and object oriented programming. Each Idpac command is described in a formal grammar, which is used by the parser. This makes it very easy to enter a new command or macro. The script language definition is closely reflected in the design of the script matcher, and the script language can be extended with some programming effort.

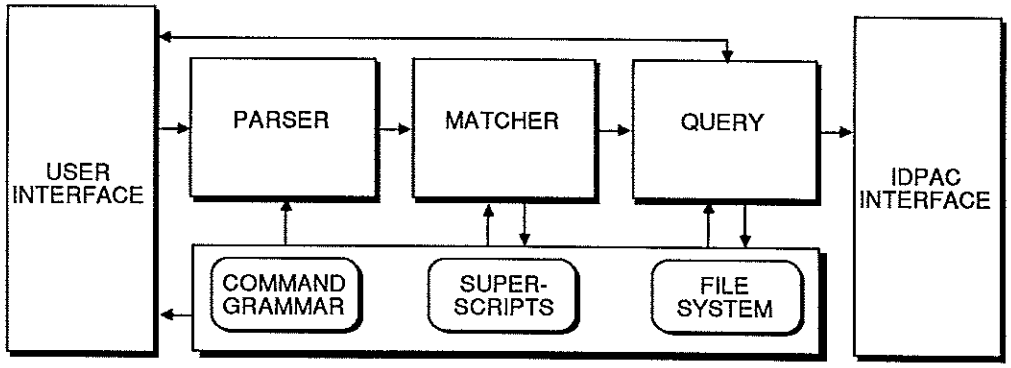
The project has implied the design of a new programming language, the script language. The design of this language and the experimentation with it, i.e., writing scripts and rules has meant a large effort. At the later stage of the project, the building of a realistic knowledge database for maximum likelihood estimation has been another large effort.

The expert interface is made up from several parts. Most of the parts work on a common database.

The user interface reads a command from the user and transforms it into a Lisp list. It provides all the input and output functions used by the other parts of the interface. In this way, all of the system's dependence on terminal types, graphics, etc., is collected in one place.

The command parser checks the commands for syntactical correctness and supplies defaults in the same way that the parser of Idpac does. In this process it transforms the commands into a more convenient form. The parser accepts commands with arguments left

out, as the other routines will fill information in, by defaulting from scripts or asking the user.



Layout of the system.

The script matcher incrementally keeps track of the script data structures and updates them according to the incoming commands. The commands are transformed, and files may be defaulted with the help of knowledge from the scripts.

Each script object inherits a YAPS database. When a command has been successfully matched against a script and the script is updated, facts may be put in its database. This takes care of all information that is not directly available in the scripts, e.g., the results of different commands, etc.

The system allows any number of different scripts to be followed in parallel. The superscripts are used to accomplish this. Each superscript contains the current state of a session with one or several scripts. One of the superscripts is currently active and the others, if any, are waiting in a suspended state. When a command does not match any script in the current superscript, the system tries to find a new current superscript by testing the suspended superscripts and also a superscript in the initial state.

The query module goes through the command description and tries to fill in the remaining unknown entries by asking the user about them. In this way the user may give only the command name, and then he will be prompted for all the arguments left out. The query module also sends messages to the file system about created and deleted files.

The file system keeps track of all the files created and used during an Idpac session. It does this by storing data about the files in a directed graph structure. This enables the file system to show the ancestors or descendants of a file, i.e., the files used in the creation of and the files created with the use of a specific file.

The database contains the command grammar used by the parser, the scripts and rules used by the script matcher, the file tree of the file system, and state variables for keeping track of the user state, internal tracing, and so on.

The Idpac interface handles the communication with Idpac. It transforms the commands delivered by the query module into text strings which are read by Idpac. The expert interface and Idpac reside in two different VMS processes. The Idpac interface sends the processed commands to Idpac via a VMS mailbox. In this way no changes had to be done to the Idpac program itself. The routines for interprocess communication are written in C.

The system is written in Franz Lisp, Foderaro and Sklower [1981], extended with Flavors, Allen *et al* [1984], and YAPS, Allen [1983]. It consists of about 6000 lines of code and runs under VMS, Digital [1984], and Eunice, Kashtan [1982], on a VAX 11/780.

Conclusions

The general idea of building a help system based on expert system techniques and including it in a CAD program has been presented and a solution given. This solution includes the development of *scripts*, a data type for describing sequences, and an implementation of an expert interface. A small knowledge database for system identification has also been developed. As far as we know, the implemented system is unique in its use of a non-invasive strategy based on a knowledge database containing both scripts and rules.

The project has given a valuable input to the CACE project, and an intelligent help system is an essential part of a modern CACE program.

Also, the project has given experience and insight in the use of knowledge-based programming tools, the demands of high-level problem-solving languages, and the design of CACE tool-boxes.

In particular, the following conclusions seem important. First, a conclusion of the project is that it is indeed possible to use an expert system and still retain a command style dialog. A second conclusion is that not all knowledge in a database for system identification using Idpac need be implemented with production rules. Scripts are a better way to represent sequences, particularly in problems where both methods and goals are well known. A good rule is to use as much as possible of the structure of the problem in the solution. The use of scripts supported by rules in a forward chaining strategy will probably reduce the overall size of the knowledge databases considerably. Finally, our experiences of running the system clearly shows that an intelligent help system is clearly useful for all, except maybe for expert users. It seems particularly well suited for casual users of Idpac, and therefore become unused to it in the intervals between uses.

In addition, some demands on the tools used in a project of this kind may be stated. It is important that user communication follow a consistent formal grammar. In Idpac each command has its own peculiarities, a thing which has caused much trouble. Also, there is a strong need for software hooks, e.g., to be used when an error occurs in a program called by another. Without them, it is very difficult to connect different programs.

List of Publications

LARSSON, J. E. (1984): *An Expert System Interface for Idpac*, Master thesis, TFRT–5310, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and K. J. ÅSTRÖM (1985 a): "An Expert System Interface for Idpac," *Proceedings of the 2nd IEEE Control Systems Society Symposium on Computer-Aided Control System Design*, Santa Barbara, California.

LARSSON, J. E. and K. J. ÅSTRÖM (1985 b): "An Expert Interface for Idpac—Paper Presented at Santa Barbara '85," Technical report, TFRT–7308, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and P. PERSSON (1986 a): "Ett expertsystemsnitt för Idpac, (An Expert System Interface for Idpac)," *Proceedings of the SAIS '86 Workshop*, The Swedish AI Society's Annual Workshop, Linköping, April 24–25, 1986.

LARSSON, J. E. and P. PERSSON (1986 b): "Knowledge Representation by Scripts in an Expert Interface," *Proceedings of the 1986 American Control Conference*, Seattle, Washington.

LARSSON, J. E. and P. PERSSON (1986 c): "Knowledge Representation by Scripts in an Expert Interface—Paper Presented in Seattle 1986," Technical report, TFRT–7332, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and P. PERSSON (1987 a): *An Expert Interface for Idpac*, Licentiate thesis, TFRT–3184, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and P. PERSSON (1987 b): "The (ihs) Reference Manual," Technical report, TFRT–7341, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and P. PERSSON (1987 c): "A Knowledge Database for System Identification," Technical report, TFRT–7342, Department of Automatic Control, Lund Institute of Technology, Lund.

LARSSON, J. E. and P. PERSSON (1987 d): "Ett intelligent gränssnitt för systemidentifiering, (An Intelligent Interface for System Identification)," *Proceedings of the SAIS '87 Workshop*, The Swedish AI Society's Annual Workshop, Uppsala, May 18–19, 1987.

Other References

ALLEN, E. M. (1983): "YAPS: Yet Another Production System," Technical report, TR–1146, Department of Computer Science, University of Maryland, Baltimore County, Maryland.

ALLEN, E. M., R. H. TRIGG and R. J. WOOD (1984): "The Maryland Artificial Intelligence Group Franz Lisp Environment," Technical report, TR–1226, Department of Computer Science, University of Maryland, Baltimore County, Maryland.

DIGITAL EQUIPMENT CORPORATION (1984): *Introduction to VAX/VMS System Routines*, VAX/VMS Version 4.0, Digital Equipment Corporation, Maynard, Massachusetts.

FODERARO, J. K. and K. L. SKLOWER (1981): *The Franz Lisp Manual*, University of California, Berkely, California.

KASHTAN, D. L. (1982): "EUNICE: A system for porting UNIX programs to VAX/VMS," Artificial Intelligence Center, SRI International, Menlo Park, California.