Knowledge-Based Real-Time Control Systems

Årzén, Karl-Erik

Published in:
Proceedings of the SAIS'89 Workshop

1989

Knowledge-Based Real-Time Control Systems

IT4 Feasibility Study

ABB
ASEA BROWN BOVERI

SattControl

TeleLOGIC

Knowledge-Based Real-Time Control Systems

IT4 Feasibility Study

Asea Brown Boveri AB
SattControl AB
TeleLOGIC AB

Department of Automatic Control
Lund Institute of Technology

1988

Persons to contact:

Claes Rytoft
ABB Corporate Research
IDEON Research Park
223 70 Lund
Tel: +46 46 168522

Börje Rosenberg
SattControl AB
Box 62
221 00 Lund
Tel: +46 46 105640

Mats Peterson
TeleLOGIC Research
Baltzarsgatan 20
211 36 Malmö
Tel: +46 40 252109

Karl-Erik Årzén
Department of Automatic Control
Lund Institute of Technology
Box 118
221 00 Lund
Tel: +46 46 108782

Published 1988
Printed in Sweden
Studentlitteratur

Preface

Knowledge-based programming techniques give possibilities to develop automation systems with completely new functions. To do this, methods and principles used in conventional control systems have to be integrated with the new technologies for knowledge processing.

A project aiming at this integration is proposed by Asea Brown Boveri AB, SattControl AB and TeleLOGIC AB in cooperation with The Department of Automatic Control, Lund Institute of Technology within the frame of the swedish IT4 programme. This report is the result of a feasibility study which defines the problem area and proposes a basic system concept for knowledge-based control systems. A separate main project proposal has been written as a parallel and complementary document.

The goals and incentives of the project are

- to define a new concept for real-time control systems where conventional techniques are integrated in a uniform way with new knowledge-based methods and principles,
- to demonstrate critical parts of the concept in two applications,
- to be a pre-competitive project for the participators and a base for final development of products available on the market within 5 - 7 years,
- to combine the forces of the Swedish automation industry into a project that would be difficult for any of the partners to perform individually,
- to strengthen the Swedish competitiveness not only with regard to automation systems but also for the Swedish industry as a whole where control and automation systems are crucial parts both in the production process and in the produced products,
- to rise the competence levels of the participators.

Key problems to be solved in the project are integration of new and traditional technologies, integration of design knowledge and the real-time aspects.

The material which this report is based on has been written by the following members of the project group: Claes Rytoft, Bo Johansson, and Anders Åberg from ABB; Börje Rosenberg from SattControl, Mats Peterson and David Lundberg from TeleLOGIC; and Karl-Erik Årzén, Karl Johan Åström, and Sven Erik Mattsson from the Department of Automatic Control, LTH. The material has been compiled and edited by Karl-Erik Årzén.

1

Introduction

1.1 Background

Current industrial use of computers is essentially restricted to applications which can be described formally. The type of information that can be handled efficiently is restricted to quantitative entities expressed in numeric or alfa-numeric forms. In a typical computer program for, e.g., accounting, material administration, technical calculation or process control the computer only handles repetitive and algorithmic functions such as arithmetics, logics and sequential operations such as sorting or merging.

As a consequence, we have been forced to concentrate the use of computers on well specifyable and repetitive processing of mainly numeric information. This has resulted in a tremendous improvement in the handling of "quantifyable" activities and assets in the industry.

However, the most important and valuable asset of an industrial company, the knowledge capital represented by the expertise of different key-people, has not been a possible target for computer representation and processing until recently. "Knowledge" is mostly qualitative, abstract information which is hard to represent and process in traditional systems.

New programming technologies (e.g., object oriented programming, knowledge-based system techniques, symbolic programming, functional and declarative programming languages, applied artificial intelligence etc) will in the future make it possible to represent and process knowledge.

The visionary and extremely important potential consequences of these new technologies are that the knowledge resources of a company will be subject to the same far-reaching use of computers and increased efficiency as we have experienced in areas suited for conventional computer techniques.

1.2 Process Definition

The topic of this project is real-time, knowledge-based process control. Process control is the sum of the different tasks that interact with a specified process and with the different users of the process. In the context of this paper a process is the controlled flow of matter, energy, or information from generation (source), via transport, storage, distribution, and change to consumption (sink) (DIN, 1985). The flow may be continuous or discrete. The process control system is the system that controls and supervises the operation of the process.

This definition of process includes the process industry, the manufacturing industry, and telecommunication systems. As pointed out by Dhaliwal (1985), processes in these domains have a number of common features:

- The complexity of the systems is such that no single individual or small group of individuals can fully understand them.
- The operations and maintenance manuals may cover several tens of volumes. Maintenance of the documentation is a particular problem. It is difficult to access relevant and correct information speedily.
- Systems are continually changing and evolving since: the process and the operating environment changes; shortcomings in the original specifications come to light; bugs are discovered; and, technology advances.
- The rate of change means that old methods of training and retraining staff are no longer adequate.
- Different users of the systems need markedly different styles of interaction with the system.
- Speedy and accurate correction of faults is required. The hazards of slow or incorrect treatment are:
  - Faults not treated early enough may propagate catastrophically.
  - Dormant faults undetected or left untreated may greatly affect the overall reliability and maintainability of the system.

- The existing control system may itself be prone to failure and thus may mask the true cause of misoperation.

- Wrongly identified faults and consequential repair actions may make matters worse.

- The high reliability of the systems gives problems. Some failures are so rare that it is difficult to ensure that maintenance staff are appropriately predisposed or equipped to handle them

1.3 Incentives for development of knowledge-based control systems

Knowledge handling is of specific importance in production and process control systems. Control systems are carriers of a collected but extracted knowledge about the whole controlled process. A program for how to supervise, control and activate the process is the final, concentrated result of a complex processing of knowledge about

- the controlled process itself,
- the involved components and how they interact,
- the produced products and their properties,
- control theories,
- demands and conditions for service and maintenance etc.

Significant knowledge is accumulated in the process computer and among the operating personnel when a process has been operated for a few years. This knowledge is scattered and neither well represented nor well organized in conventional systems. The reason for this is that todays control systems are not at all suited for processing this type of knowledge. The main weaknesses are the following.

- Incapability to express and implement knowledge-based control functions.

  Control functions based on experience, heuristics, fragmentary knowledge, qualitative knowledge, reasoning in a specific world of conception must today be handled "manually".

- Incapability to represent and communicate the underlying knowledge of different types of control functions.

  Although there is a lot of knowledge involved in the analysis and design phases which precede the final programming of the process control system, only the final algorithmic representation and possibly some verbal explanation of it is processed by the computer. The control system is a black box whose content possibly can be explained by the programmers of the system but often by nobody else. All expertise and underlying reasoning is documented separately, in the worst case communicated verbally to the users who need it in order to improve or extend the process.

- Incapability to visualize a complex process in a user and knowledge oriented way

  Processes, automation, and control systems become more and more complex. This puts increased demands on operators and other users of the control systems without giving them any new supporting tools. Today's systems can in an excellent way visualize the process based on process signals but cannot transfer this to higher, more knowledge-oriented abstraction levels suited for the various users of the system.

Control systems are of great importance to the whole industry and especially to the export industry. This industry is very much oriented towards system-oriented, knowledge intensive and highly automated production processes and products. Effective control systems are of crucial importance both to production and as integrated parts in the products. Knowledge-based control systems will therefore be of great importance and will add a competitive edge to most exported industrial products.

1.4 Development situation

The development and research on knowledge-based control systems are currently very intense with many large international projects. The driving forces are three different groups: user industries, AI companies, and control system suppliers.

Among the user industries, it is primarily the manufacturing industry, the chemical industry including oil refineries, the paper and pulp industry, the power systems industry, and the telecommunication industry that have started activities. As described in Chapter 4, many systems have been developed and a few also fielded. In Sweden, the paper and pulp industry and the power suppliers have started activities.

Small AI consulting companies aimed at the process control market are currently emerging. Initially started because of market needs, the companies help to maintain the interest and also to create new markets by giving courses and seminars. Some of these companies also have expert system products aimed at the process industries. Examples of such companies are Gensym Corp., and Carnegie Group. Also conventional consulting companies such as Framentec, PA Consultants, Cambridge Consultants, and Stone & Webster Engineering Corp. have started activities in the AI – Process control area.

The interest is high among the conventional control system manufacturers. Companies like Honeywell, Foxboro, and Combustion Engineering all have active AI research groups and products on the way. The risk is large that Swedish manufacturers will be left long behind if no serious initiatives are taken now.

Several, large international as well as national research programmes concerning knowledge-based control systems are going on at the moment. The most ambitious effort is found in the European Community's ESPRIT programme.

In analyzing on-going projects, we have found more or less the same approaches in most of them. A knowledge-based system is placed as a front-end to, and separated from, a conventional control system. This leads to systems that are difficult to maintain and install with different operator consoles and man-machine interfaces. In the long run and as a base for knowledge-based, real-time control systems this is not a suitable approach. The special real-time aspects of knowledge-based systems are also mostly neglected.

1.5 The overall goal of the project

The goal of the project is to define and demonstrate a uniform concept for knowledge-based real-time control systems that later on can be used as a base for development of commercial products.

A guidance for the work is a visionary goal of a control system

- that can be programmed in a knowledge and problem oriented way in contrast to today's use of imperative and computer governed programming languages, and

- with which you can have a conversation more or less in the same way as if you had direct contact with the experts on the controlled process, the produced products and the control system.

1.6 Areas of specific importance

As a result of the feasibility study, we have reached the conclusion that the solution to the following problems are of fundamental importance for the development of knowledge-based control systems.

- Integration of conventional and new technologies in one uniform system.
- Integration of design knowledge into the operational control system.
- A practical solution of the real-time problem.

Some of these problem areas are identified to be important also in new ESPRIT II projects. This could facilitate a desirable cooperation.

1.7 Participators

The project is based on a unique constellation of participants consisting of

- Asea Brown Boveri AB and SattControl AB, the two leading supplier of automation systems in Sweden,
- TeleLOGIC, a research company belonging to Televerket with experience from many AI-projects and also involved in international cooperation projects, e. g., within RACE and ESF,
- The Department of Automatic Control at LTH, engaged as a consultant, with an international reputation in research on knowledge-based control systems and with contacts with most of the important research institutions around the world working with knowledge-based control systems.

1.8 Outline of the study

The feasibility study has the following organization.

Chapter 2 describes the handling of knowledge in processes of today. It is pointed out how poorly the different kinds of knowledge is treated today and to what low degree the knowledge is included in the on-line system. This is especially the case for the design knowledge which is the basis for the operation and maintenance of the process. It is suggested that knowledge-based techniques could be a solution to many of the problems.

Chapter 3 contains an general overview of knowledge-based systems with an aim to predict the development situation for the next 5 years. The chapter contains a special treatment of real-time knowledge-based systems. The next generation control systems will of course also be influenced by the technical development within other areas. Therefore, an overview is also given of the development in some of the importants conventional areas.

Chapter 4 contains an overview of proposed and implemented knowledge-based applications in process control. The chapter is organized into a model of process control that composed of tasks, tools, and roles. The chapter contains many examples of different approaches and systems.

Chapter 5 gives an overview of related projects within international and national reserach programmes. The major part of the chapter treats the on-going ESPRIT projects.

Chapter 6 proposes a system concept for the integration of knowledge-based techniques with conventional techniques. The concept is based upon a model that represents the process components and the control system components. Multiple perspectives are proposed to handle different types of knowledge in the model. The central model is surrounded by a set of tools that implements the different control functions in the system, both knowledge-based and conventional, and builds up the different user interfaces needed. The chapter also discusses the real-time aspects of the concept.

Finally, conclusions are given in Chapter 7.

Several visits to companies and universities with related projects have been made during the feasibility study. Travel notes from the visits are included as an appendix. Competence descriptions of the participators and a glossary are also included as appendices.

2

Handling of process knowledge today

This chapter gives an overview of what types of knowledge that exist about a plant, the sources of the knowledge, and how it is used. At the end of the chapter, conclusions are given about the knowledge handling problems in today's systems. AI techniques are suggested as a solution to the problems.

2.1 The life-cycle of a process control system

The life-cycle of a process control system can be split up in to three phases according to Figure 2.1.

- Design Phase

  The main goal here is to create possibilities for the production. This involves design of the process and the control system.

- Operational Phase

  The main goal during this phase is to run the process with maximum profit within given specifications and constraints.

- Maintenance Phase

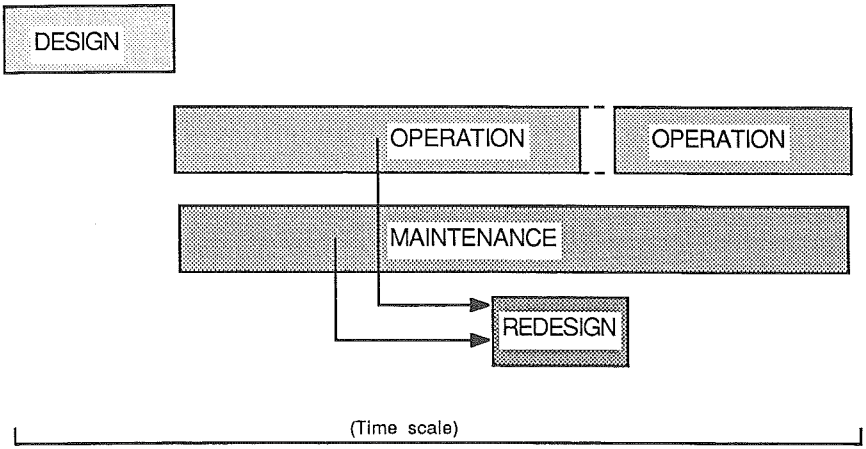

Figure 2.1 The life-cycle of a process control system.

The main goal during this phase is to keep the production facilities running and to modify them according to new available technique and/or feedback from operation.

Although the two last phases run in parallel, there is a clear difference between them. Over the life time of a plant there is feedback loop from operation and maintenance to redesign. It is possible to distinguish between users dealing with design, operation and maintenance. The design personnel are mainly working in an off-line environment and their main task is to provide knowledge that is used to construct the plant. Operation and maintenance are on-line activities that to a large degree depend on design knowledge to succeed.

In the following, we will try to describe the available knowledge and to evaluate which users that depend on what knowledge. We also try to analyze how this knowledge is handled today.

2.2 Knowledge sources and knowledge sinks

2.2.1 Knowledge sources

Most of the available knowledge origins from the design phase. In the design phase there are a lot of different knowledge "sources" or designers involved. They all describe the plant from their view. As an example, there is one mechanical

designer that designs the mechanical parts of the system. There is also an electrical designer that designs and describes the power supply and how the power should be connected to the mechanical devices. Common for these two designers is that they both describe the same object, e.g., a pump, but one describes the electrical view of the pump and the other describes the mechanical view of the pump.

Of course there are knowledge that origin from other phases such as historical and heuristic knowledge that origin from the operational phase, and service information about components that origins from the maintenance phase.

2.2.2 Knowledge sinks

The knowledge sinks are many and differ between the different phases. During design, nearly all designers are dependent of knowledge from other designers. In the operational phase, the operators are the main knowledge sinks, but there are other departments that depend of production information etc. During the maintenance phase, the service staff is the main knowledge sink.

2.2.3 Summary

It is easy to see that there are a lot of knowledge sinks and a lot of knowledge sources and nearly all of them depend on each other. It is also obvious that different sinks need different information about the same components.

Consider a telecommunication system. When a telecommunication line breaks down, the operator who is responsible for the traffic needs to know that the line is down in order to reroute the traffic. The repair personnel also need to know that the line is down but they do also need information about physical location, specifications, connection points etc. A third part is perhaps a statistics department that needs to know that the line has been down and for how long.

2.3 Knowledge about a process

What do we mean by knowledge within the process control area? Knowledge can be described in many different ways.

A distinction is often made between deep knowledge and shallow or heuristic knowledge. Deep knowledge is based on a theoretical model of the function of domain. Heuristic or shallow knowledge on the other hand consists of rules of thumb and of other types of knowledge that are based on experience and difficult to represent with conventional programming techniques.

Another distinction that can be made is between quantitative and qualitative knowledge. Conventional control systems have so far only been concentrated

on quantitative knowledge such as algorithms, procedures, functions, sequential operations, numerical models etc. Qualitative knowledge such as lines of reasoning, qualitative judgements, decision rules, expertize, underlying assumptions, explanations etc., is difficult to represent.

What type of knowledge is available for a process? One attempt to classify the different types of knowledge involved in the life cycle of a plant is given here. For each class of knowledge, examples are given together with a description of on what media the knowledge is stored today.

2.3.1 Functional knowledge

Functional knowledge describes the function of the system and its subsystems during different operating conditions (static, dynamic alert, emergency, start up etc.). Some examples are: What is the goal with the plant?, What is the primary function of a process part, Does it have any secondary function etc. Examples of such documented knowledge are:

Type of documentation:	**Media:**
Goals of operation	Paper
Functional Block diagram	Paper
Mass flow balance	Paper
Flowchart (process & instrumentation)	Paper (CAD)
Functional description	Paper
Design experience	Brain
Common sense physics	Brain
Advanced physics	Simulation programs

2.3.2 Economical knowledge

Economical knowledge describes the basic economical constraints involved. It consists of raw material, product value, quality aspects, maintenance, repair, costs for unplanned stops etc. This type of knowledge often has a different source than the other types, e.g, from the sales department.

Type of documentation:	**Media:**
Order Information	Paper
Economic calculation	Paper
Optimization algorithms	Computer

2.3.3 Physical knowledge

A physical description describes the basic components and how they are connected to each other. Physical knowledge does also include the control system and the program for the control system. All together, the physical knowledge

describes how the process is implemented. Physical knowledge is documented as follows:

Type of documentation:	**Media:**
Layout	Paper (CAD)
Drawings of buildings	Paper
Mechanical drawings	Paper (CAD)
Instrumentation drawings	Paper (CAD)
Electrical connection drawings	Paper (CAD)
Control logic	Computer
Equipment lists	Paper (CAD)
Isometric drawings	Paper (CAD)

2.3.4 Operational Knowledge

Operational knowledge includes information about operational characteristics of the process during, operation, start-up, shut-down and emergency. Operational knowledge is typically documented as follows:

Type of documentation:	**Media:**
Operator instructions	Paper
Equipment manuals	Paper
Operator experience	Brain
Maintenance experience	Brain

2.3.5 Historical Knowledge

Historical knowledge mainly applies to plants in operation. It consists of stored time histories of important process variables. The data is normally organized in a systematic and consistent way. Examples of documentation are:

Type of documentation:	**Media:**
Operational log	Computer
Trend curves	Computer
Feedback information from earlier plants	Paper

2.3.6 Summary

As seen from the examples, most documentation are stored on paper. It is very little that is included in the on-line control system. The documented knowledge is also very shallow. The documents that exist are all based on large amounts of experience and underlying assumption and lines of reasoning. This qualitative knowledge is very seldom included in the documents.

A simple example is a process flow chart. A process flow chart is very difficult to read and understand for those who are not familiar with the process and the control system. The reason for this is that the flowchart only describes things as they are and not gives explanations and motivations for why this is the case. Such qualitative, underlying knowledge is necessary in order to understand the flow chart. In some rare cases, comments are given on the flow chart or references are made to the flow chart from the instruction manuals. It is however not the usual case.

Knowledge is today mainly transferred by paper, through training courses and from man to man. This way of knowledge transfer continuously causes problems. The knowledge is seldom available when it is needed.

Since large amounts of knowledge never is documented, it will not be transfered at all. This knowledge is mainly heuristic, e.g., design knowledge about why a valve is placed before a pump instead of after etc. Lack of this type of knowledge does also create a lot of problems during the control of the plant. The most obvious example is when a new operator is employed. It would be very useful if tools existed that helped in transfering knowledge from experienced operators.

2.4 Knowledge used in the design phase

In the design phase, the main activities are:

- Economic and functional specification
- Process design
- Process equipment design
- Instrumentation and control design

Normally, the design phase involves many people. In most cases, the plant owner is represented by consultants. The main contractor will use several suppliers and consultants.

2.4.1 Economic and Functional Specification

The primary objective when constructing process plants is to make profit from the investment. The main operational factors to be looked upon in a life cycle perspective are:

- The income from the sale of the production
- The costs of raw products

- The operational costs
- The marketing costs

These factors may have a volume aspect as well as a quality aspect connected to them. Simple investment calculations are always made, but this is not enought if you want a detailed risk analysis about volume, quality and sale. Comparison of different production techniques is another factor to be analyzed. Computer simulation of trains running according to different time-tables and job-shop simulation of work-shops are examples of a deep analysis of the production performance.

Even in these early design activities, there is knowledge created. The economical knowledge is primarily used for decisions, but it could be used later on in the design phase and in the operational phase. Today, this kind of information is normally used only once and not reused at all.

2.4.2 Process Design

In the process design activity the main plant specifications are known in terms of:

- Production rates
- Process principles

The design engineers will look on the details of the process flows through the plant and on different operating conditions. As a result there will be functional specifications of the process equipments, interconnection diagrams and specifications on auxiliary systems.

2.4.3 Process Equipment Design

Later on in the design phase, the designers calculate the dimensions and the processing powers of the processing equipments. These activities involve several types of designers like mechanical, electrical and chemical engineers. One difficult task is to coordinate these people and their knowledge needs. The break down of the project in subtasks is necessary, but this is to a certain extent contradictory to the view of the plant as one unit. The output from the process equipment design activity is detailed down to the level needed for the purchasing or the manufacturing of the equipment.

2.4.4 Instrumentation and Control Design

Instrumentation and control systems are needed in the plant in order to coordinate the different parts of the process and to make it possible to control the plant. Some of the subactivities are identification and specifications of:

- The external load and the external resource characteristics
- The operational modes and their state transitions
- The production rates and the quality control strategies
- The strategy of order-bound state transitions
- The batch oriented strategies
- The mixture and the separation control strategies
- The stiffness control strategies of all resources
- The undependent resources to be kept uncoupled
- The balancing strategies of dependent resources

The monitoring, control and operational aspects of the plant are usually not taken into account early enough in the design phase. In particular, this is true, if the process hardware suppliers are different from those who do the control design. Today, functional information is lost to a certain extent and has to be reinvented by the control system designer in the control design activity (and by the process operators every day).

2.4.5 Examples of design

The mechanical designer of a boiler will in his design of the superheaters calculate the heat transfer rates, the type of steel material, the dimensions of the superheaters and their cooling system, the mechanical stress and life time due to temperature changes, the material wear due to particle erosion, the speed of the flue gases, the speed of steam, the state of the steam, the material wear due to water droplets, the overall efficiency variations due to the steam state etc.

Often, the control system designer will not have access to the knowledge of the mechanical design engineer. In the control design of the steam temperature control loop and the gliding pressure control, the control engineer will probably use control system drawings from old installations instead of fresh knowledge from the mechanical designer.

The next one who lacks information is the operator. The hazards, already thought of by the mechanical designer, the operator has to learn about by experience. The life-time effect and the efficiency effect of setpoint and alarm-limit changes of the superheater temperature are often unknown.

The instrumentation technician is responsible for the tuning of the controllers, but he has only a vague idea of the trade off between soft temperature control and superheater heat-chocks at different levels of boiler operation.

In case of an emergency shutdown due to a failure in the valve of the superheater cooling water, the repair personnel have to find the right valve and replace it. Today, this mechanical design information is not available in an integrated and easy way. Drawings are often changed, and not kept up to date.

2.5 Knowledge used during operation

During the operation phase of a plant the main tasks for the operator are:

- Monitoring
- Control
- Planning

All these activities requires a lot of knowledge about the process and the control system.

2.5.1 Monitoring

Monitoring of a process is done continuously. The operator uses the result to optimize the process. To be able to do this optimization, he must have good knowledge about the process and how the process should be run. Some of this knowledge is related to the design phase, and could be found in documents like functional description and operator instructions. Today nearly all of the relevant documentation is only available on paper, with the exception of the flow charts that normally are shown on the screen of the supervisory system. There is no way that the operator could get any information from the control system about the functional knowledge.

On top of all this is the heuristic operator knowledge. An example of a piece of knowledge of this kind is that valve 5 normally gets inoperable after 5 hours of continuous production unless it is flipped every 2 hours. In the systems of today, this type of knowledge is transferred from man to man, or in the best case, it is written down in the operator instruction book and then put in the control

system in the next redesign. Also here there is no way that the process control system could help the operator with this type of information. The operator has to consult the documentation where he seldom finds it.

2.5.2 Control

Control of a process is a delicious task which requires a lot of knowledge. Alarms and new production orders call for control actions. To carry out these actions, the operator needs functional knowledge about the control system and heuristic knowledge about the plant. When an alarm occurs is it up to the operator to identify the alarm and to know how to make the right control actions to correct it or to minimize the damage caused by the fault. If the operator should have any chance to react correctly on the alarms and the trend curves he first of all depends on that the control logic is correct, and secondly that he has or could get knowledge about the correct function of the process.

2.5.3 Planning

When the operator deals with planning tasks he partly depends on information (or knowledge) from the production planning system. This type of knowledge is of a different nature from the above mentioned knowledge, but still it normally appears on paper or perhaps on the screen of another computer system.

To be able to calculate how the process should run the operator needs knowledge about capacities of the plant etc. This knowledge is today not available in the control system. Ideally, the operator should be able to reason with the system about things like capacities etc. Another very interesting function to include in a control system is possibilities for simulation, so that the operator could simulate his actions in advance.

2.6 Knowledge used for maintenance

Maintenance includes among other things the following activities:

- Diagnose
- Repair
- Preventive maintenance
- Documentation of changes
- Spare part handling
- Feedback to design and operation

2.6.1 Diagnose

The diagnose starts with a problem report describing the symptoms of the process disturbance. The purpose of the diagnose is to find the reason why the process does not work satisfactorily. The cause of a problem can be a faulty component, bad raw material, bad design etc.

To find the origin of the symptoms there are mainly two methods of working: heuristically or hierarchically. Normally you start with the first and if not successful, the second method is used.

Heuristic diagnose

In heuristic diagnose, knowledge about faults and symptoms that occur and what causes them is used. The probability of different malfunctions is also taken into account. To do this mapping from symptoms to errors you need a lot of experience from the actual plant or from similar plants. By using heuristic diagnose it is impossible to handle unexpected problems.

The knowledge needed for this type of diagnose is a description of (the most common) symptoms and their causes.

Hierarchical diagnose

Hierarchical diagnose is based on a functional or structural decomposition of the process. The search for the cause of the problem starts from the top. The function of the subparts is investigated. The failing subpart is then the target for the continued search. The search is done recursively until the cause of the problem is found.

To see if a part on a certain level is working correctly, its current function must be compared with the expected one. There must be a number of test points by which the function can be monitored. By comparing the result from the test point with the expected situation, the faulty part can be localized.

To be able to do a hierarchical error search the documentation must support it. The documentation must be hierarchical and function or structural oriented. It is important that the test points and the normal relation between the status of them are explicitly shown.

Difficult situation: The diagnose is much more complex in the case of combinations of more than one fault or intermittent errors. To handle the last one there must be some function to record the course of events.

2.6.2 Repair

Normally it takes some time to replace a faulty component. In the mean time, the production may be kept up by using an alternative production way.

The faulty components is replaced by an equal one. If no suitable component is available, a redesign using available components must be done.

The knowledge needed to find a alternative way for production is indications of the different possibilities to run the process. To be able to replace a component there must be information about suppliers and their identification of the component. For redesign, dimensioning parameters are needed.

2.6.3 Preventive maintenance

The goal of preventive maintenance is to repair a component just before it breaks down or causes production problems.

The problem is to detect errors before they occur. To select the time for the repair you can use a number of strategics:

- Elapsed time
- Elapsed operation time
- Load, i.e., elapsed operation time weighted with a load dependent factor
- Status of components. Estimating the status of a component by using observations and measured values.

The knowledge needed is the duty cycle of the components, the wear of the components as a function of load, how to estimate the condition of a component from observations.

2.6.4 Documentation of changes in plant

The changes done to a plant must be documented in order to keep the plant documentation up to date. It is also important to document replacement or work that has been done, since it is quite common that changes create new problems. This information is valuable for the heuristic diagnose.

Software bookkeeping

A special problem is the book keeping of all the software modules used in a process.

2.6.5 Spare part handling

A problem is to optimize the stock of spare parts. A big stock of spare parts costs capital and a small stock may cause loss in production.

To optimize the number of parts kept in stock you need among others the following information:

- Number of used parts in the plant
- Probability for a fault in a part
- Price
- Cost if the part is not available

Apart from this information, qualitative experiential knowledge is needed. Examples of this could be the reliability of different spare part suppliers, preferences between different suppliers, seasonal fluctuations in the fault propabilities for different parts etc.

2.6.6 Feedback of knowledge

During the maintenance work, a lot of experience is gained. The knowledge should be transferred to the design phase, operation phase, and within the maintenance phase for later use.

Design: The design of new and the revamping of old plants can be improved if the knowledged gained during maintenance can be fed back.

Operation: The operation people can benefit from knowledge on how to avoid problems.

Maintenance: By collecting information in a ordered way during the maintenance work, the heuristic knowledge for diagnose can be extended. This will improve later diagnoses.

2.7 Conclusions

As could be seen from the examples above, large amounts of knowledge is never used during operation and maintenance. What is the reason for this situation? One reason is that much of the knowledge is stored in such a way that it is very difficult or even impossible to use. It could, e.g, be stored in a file somewhere,

perhaps in the plant managers office. Normally, there is no tools available that assist in finding the knowledge needed fast and easy. This is also true even when some of the information is stored in CAD-systems. Furthermore, heuristic knowledge is not stored at all today, mainly because there is no way to store it. In future process control systems, these problems have to be taken care of.

Knowledge-based system (KBS) techniques are focussed on the handling and processing of knowledge. Several knowledge-based systems have been proposed and implemented in process control for different applications. Many of the applications deal with the problems described in this chapter. An extensive overview of different applications is given in Chapter 4.

3

Technical Survey

The possibilities for artificial, machine intelligence have long attracted interest. Since the late 1950s, the area has evolved into a separate, now and then, heavily discussed and questioned, research discipline.

One definition of Artificial Intelligence (AI) is the following taken from Rich (1983)

> "Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better."

AI contains many different research directions. One direction is focussed on the human perception system. This direction contains areas such as robotics, computer vision, speech recognition, and natural language. The goal here is to emulate the human sensory and motory capacities.

Another area concentrates on the human problem solving capacity. This direction includes areas such as knowledge-based systems, automatic theorem proving, and game playing.

The goal of both these directions is to build intelligent machines or computers. The cognitive research directions in AI instead uses AI formalisms as tools to better understand the behavior of the human brain.

Characteristic for AI is that it is an evolving science. What was once considered as belonging to AI is now standard techniques. Examples of this are multiple-user operating systems and windowing systems.

Knowledge-based systems or expert systems is an area within AI that relates to many of the current problems in process control systems. An overview of knowledge-based systems is given in Section 3.1. Object-oriented and rule-based knowledge representation are the two most used techniques for representing knowledge in current knowledge-based systems. An overview of these are given in Sections 3.2 and 3.3. Section 3.4 is devoted to development trends important for knowledge-based control. Real-time knowledge-based systems are treated in Section 3.5. Finally, Section 3.6 discusses development and trends in conventional control systems which will be important for the next generation of control systems.

3.1 Knowledge-based systems – An overview

Knowledge-based systems or expert systems is an area of AI that has grown rapidly during the last few years. The basic definition of the term *expert system* is a program that solves problems within a specific, limited domain that normally would require human expertise. This is a wide and vague definition that also covers many traditional computer programs. A clear definition of an expert system is difficult to state. The situation instead is that an expert system more or less fulfills a number of different characteristics.

The most significant characteristic is the expert problem solving capacity within limited application domains and for problems where conventional programming techniques have not been successful. The reason why conventional techniques do not work is mainly that the problem lacks a clear analytical and/or algorithmical solution or that the existing algorithm is computationally intractable. The expert system instead tries to emulate the problem solving behavior of the human expert. This means that the system tries to represent and execute the expert's knowledge and reasoning strategy.

Another very common characteristic is that the domain knowledge is represented explicitly in an identifiable, separate part of the program. This so called knowledge base is separated from the control strategy or *inference engine* that actually runs the program by operating on the knowledge. A system of this type is referred to as a *knowledge-based system.* The explicit knowledge representation gives an expert system a declarative nature in contrast to traditional procedural programming languages such as Pascal or Fortran where domain knowledge is expressed in the form of program statements. Knowledge representation, i.e. how individual facts are represented and how representations of individual facts are combined to a representation of the complete problem state, is a key issue in expert systems. Rules and object-oriented representation are often used techniques which are discussed in more detail in the following sections. Propositional logic or predicate logic, scripts, and procedures are other techniques.

Well-developed explanation facilities are another expert system characteristic. It is necessary that an expert system can explain its reasoning in order to be accepted by the user. It is usually possible to get an explanation for why the system asks a certain question to the user and how a certain conclusion has been reached.

The possibility to reason with uncertainty is another feature that might be present. Knowledge might be uncertain in certain applications and expert systems can then support the representation of, and reasoning with, this uncertainty. This is often implemented in the form of probability measures that reflect the reliability of the knowledge and which are propagated through the reasoning.

Another expert system aspect is the modularity that is provided through the explicit knowledge representation. The knowledge base is built incrementally and can relatively easily be expanded with, e.g., new rules. This makes exploratory programming possible, where prototypes rapidly can be developed and later be used as a part of the final implementation. This is perhaps the main reason why expert systems have the reputation of allowing implementation of very complex systems.

The first step in an expert system application is the knowledge elicitation. Formal techniques for this do not exist. Interviews of human expert is one method used. This is done by *knowledge engineers*. The next important step is the conceptualization where the domain knowledge is structured into basic concepts which represent the important features of the application. The concepts are then transformed into a suitable knowledge representation.

3.1.1 Implementation languages

Most existing expert systems are implemented in a symbolical languages such as Lisp (Winston and Horn 1984; Steele 1984) or Prolog (Clocksin and Mellish 1984) or in some language implemented on top of those.

Lisp

Lisp is the traditional AI programming language and one of the oldest high-level languages still available. The original, pure Lisp was a functional language. The current Lisp dialects combine functional programming with imperative programming. Lisp has a strong emphasize on function applications and recursion. Lisp has a very simple and uniform syntax. The primary data structure is the list. Lisp code is also represented as lists. This makes it easy to write programs that generate other programs.

The Lisp language is basically interactive and interpretative although compiled versions are standard. This makes it easy to programme in an explorative style where program ideas rapidly can be tested. Lisp systems are well-known for

their powerful programming environment with integrated editors, debuggers, and graphics systems.

Prolog

"PROgramming in LOGic" is a logic programming language based on a subset of predicate calculus. It came in the beginning of the 1970s and has mainly been developed in Europe. That is one of the reasons why Prolog is very popular in Europe.

A Prolog program consists of facts and rules which are unified against questions posed by the user. Prolog is very natural for problems that involve search. For general programming tasks, many programmers feel that Prolog is too special. Prolog is often available as an add-on to Lisp systems.

Expert system shells

The separation between the knowledge base and the inference engine has led to the development of *expert system shells* or *frameworks*. A framework is an empty expert system without any domain knowledge. It provides an inference engine and a knowledge representation structure that can be used as a programming tool for implementation of expert systems in different application areas.

3.1.2 Application categories

The applications where expert systems have been used are usually divided into the following groups:

Diagnosis:	to conclude the cause of a given set of symptoms.
Design:	to generate a design according to given requirements.
Planning:	to build up a plan of actions to obtain some given goal.
Monitoring:	to detect unnormal situations.
Interpretation:	to interpret noisy or uncertain signals.

Expert systems have been built for, e.g., medical diagnosis (Shortliffe, 1976), computer hardware fault detection (Milliken *et al*, 1986), interpretation of seismic measurements (Duda *et al*, 1977) and computer system configuration (McDermott, 1980).

Several conditions should be fulfilled in order for an application to be suitable for an expert system solution. The following are examples of such conditions.

- The problem should be limited and traditional solution methods should be unknown or intractable.

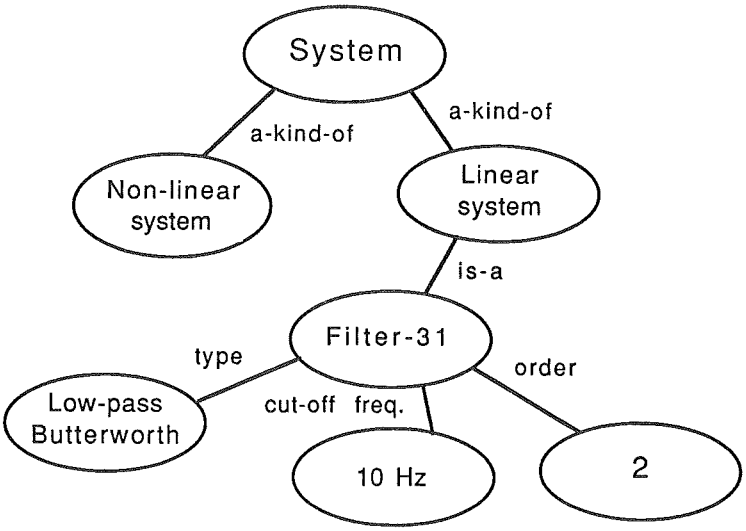

Figure 3.1 Semantic network example

- The problem should be solvable for available experts.
- The goal should be well-defined.
- Domain specific knowledge should dominate over common-sense knowledge.

Most existing system are consultative, i.e., they are meant to be used by a non-expert in a question-answer dialogue to derive a solution to some problem. *Autonomous systems* that solve problems without human assistance are more interesting at least in process control, but much more uncommon. The existing systems are often hybrid systems in the sense that they contain both symbolic expert system components and numerical components. An example of this kind could be a system that interprets disturbed signals with a combination of numerical and symbolical methods for, e.g., speech recognition or signal processing.

3.2 Object-oriented knowledge representation

To represent knowledge as objects with associated attributes is common in existing expert systems. The basis is the *semantic network*, (Quillan, 1966), which represents knowledge as a network of nodes. A node could represent the concept of objects, events, ideas, etc. Associative links represent relations among nodes. An example of a semantic network is shown in Fig. 3.1. A survey is

given in Brachman (1979). Semantic networks represent the combination of a superclass-subclass hierarchy and the description of properties (attribute - value pairs). Another well-known aspect of the network formalism is the instance relation that associates a particular individual with a class of which it is a member. This is often represented with the *is-a* link.

3.2.1 Frame systems

The idea of frame systems, a variation of the semantic networks, was introduced by Minsky (1975). A frame system consists of three different building blocks: frames (sometimes called units), slots and facets. A frame is the equivalent to a node in a semantic network, i.e., it represents concepts of objects, events, ideas, etc. With some abuse of language, frames are often referred to as objects. The slots describe the properties or attributes of a certain frame. In the same way, facets describe the different slots. One of the facets is the actual value of the slot. Others facets could be used to specify which type the slot value may take, default value for the slot or to give an additional description of the slot.

Frames are often divided into two types: those which describe classes and those which describe individual instances. An important concept of semantic networks and frame systems is the inheritance of properties. Inheritance allows class frames which can pass their slots along to subclass frames and to instance frames. Multiple inheritance, i.e., that a frame is a subclass of more than one superclass is common. A simple frame system example is shown in Fig. 3.2.

Procedures can be attached to frames by associating the procedures with slots. These procedures are called *demons* and they provide for so called *access-oriented programming*. Demons are for example used to compute the value of a slot when a reference is made to it and no previous value exists. Another possibility is to have demons that are executed each time a slot is given a new value or each time a frame is created or deleted.

Several frame based knowledge representation languages exist. Some examples are KRL (Bobrow and Winograd, 1977), Units (Nilsson, 1982), PAUL (Hein, 1983), KEE (Intellicorp, 1984), and Epitool from Epitec AB.

3.2.2 Object-oriented programming

At the same time as the frame based knowledge representation languages were developed, a very similar development took place in the area of object-oriented programming, (Stefik and Bobrow, 1986). This area has its historical background in the work on SIMULA (Dahl and Nygaard, 1966) and has its most extreme representative in the programming language SMALLTALK-80 (Goldberg and Robson, 1983). The basic entity of object-oriented programming is the object which has a local state and a behavior. Objects are asked to perform operations by sending appropriate messages to them. Objects have associated procedures

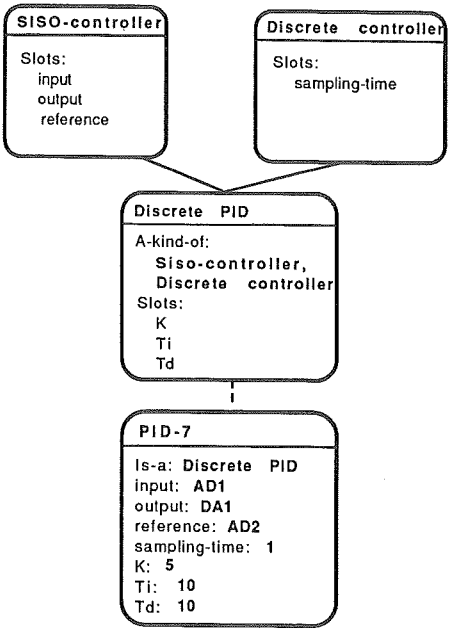

Figure 3.2 Frame system example

called *methods* that respond to the messages. Message passing supports data abstraction and generic algorithms. A protocol, i.e., a set of messages is defined, which specifies the external behavior of the object. The internal implementation of the object can thus easily be changed without affecting the calling programs.

In the same way as in frame systems, objects are divided into classes and instances of classes. The classes builds up a superclass - subclass hierarchy with inheritance. The inheritance is more focused on inheritance of behavior, i.e., of methods, than on inheritance of properties as is the case in the frame systems. The analogue of the slots in frame systems are the variables. Variables are often divided in two types: instance variables that are inherited by the instances and class variables that are attached to a specific class and common to all the instances of this class.

Several different object-oriented add-ons to Lisp exist. Some examples are Flavors (Cannon, 1982), Common-Loops (Bobrow *et al*, 1986) and Object Lisp (Drescher, 1985). The programming language C is undergoing a similar development with the add-ons Objective-C (Cox, 1986) and C++ (Stroustrup, 1985). Although very similar in spirit to frame systems the main purpose of the object-oriented systems is to use the objects for data abstraction in computer programming. The frame systems are instead focussed on using similar facilities for knowledge representation.

3.2.3 Multiple inheritance, composite objects, and multiple perspectives

Inheritance from more than one superclass, i.e., multiple inheritance, is used for two reasons. The first reason is to describe objects that naturally, at the same time, can be viewed as instances of more than one class. Through multiple inheritance, their behavior is described as the combined behavior of all the super classes. The second reason is for structuring purposes. Behavior and attributes that are common to many different classes but which can not be naturally seen as a class that can be instantiated can be grouped together to a *mix-in* class. Mix-in classes are never instantiated and are only used to add a specific behavior to other classes.

Multiple perspectives is used for the case when a single object, at every time, can be seen as the instance of one out of a set of classes. This can be seen as a special case of multiple inheritance where the behavior and attributes from the inherited classes are kept separated in the object instead of being combined. The behavior and attributes of the object is different depending on from which perspective or view the object is looked upon.

The name composite object is used for an object whose attribute values are other objects. Composite objects is one way to achieve hierarchical object structures where a subobject of an object represents a more detailed description of a part of the object.

There is an interesting interplay between multiple inheritance, multiple perspectives, and composite objects. An object with multiple perspectives can, e.g., be described as a composite object where each subobject represent one perspective.

3.3 Rule-based knowledge representation

Rules are the main knowledge representation method used in existing expert systems. This is reflected by the use of the name rule-based system synonymously to expert system. Another name that is used is *production systems* where a production is the equivalent of a rule. The primary reference for production systems is Newell and Simon (1972). Rules often look like

```
if <antecedents> then <consequents>
```

or like

```
if <conditions> then  <actions>
```

The antecedent or condition part of a rule is also called the left hand side (LHS) of the rule. In the same way the consequent or action part is called the right hand side (RHS).

The main parts of a rule-based expert systems are

1. Database or working memory
2. Rulebase
3. Inference engine
4. User interface

The first two parts are commonly referred to as the knowledge base. The database is used to represent facts about the application domain. The data structures in the database vary between different systems. The simplest form is a collection of variables that can take different values. Another quite common data structure is the list. Lists are used in OPS4 (Forgy, 1979) and YAPS (Allen, 1983). Object-oriented data structures are very common. The EMYCIN (van Melle, 1981) class of systems usually use object-attribute-value triplets. Basically the same is used in the OPS5 (Forgy, 1981) system. Other systems use more elaborate frame based systems with inheritance and procedural attachment. Examples of those are KEE (Intellicorp, 1984) and ART (Inference Corp., 1984).

The rulebase contains the rules of the system. It is sometimes partitioned into different rule-groups according to different contexts. The left hand side of the rules typically consists either of patterns that should match the contents of the database or of predicates on the database that should be fulfilled. Most systems allow rules to use pattern matching variables. This makes it possible to write generic rules that can be used by the system for many different facts. The right hand side of the rule either modifies the database in some way or performs some external input or output.

The inference engine applies the rules to the database according to some strategy. The dominating strategies are forward chaining and backward chaining. In some systems these are combined. Since the inference strategy is the real core of the system, these two strategies will be explained in detail later.

The rule-based programming style is especially well suited for certain problems. Suitable problems as well as advantages and disadvantages of rule-based systems are described in, e.g., Brownston *et al* (1985). The power of the rule-based systems is most evident for complex, ill-structured applications that lack efficient algorithmical solutions. The decomposition of the system into a number of relatively loosely coupled rules makes it suitable for problems that are decomposable into subproblems which have no fixed or apparent order. The rule-based ap-

proach supports parallel lines of reasoning as opposed to the primarily sequential execution of conventional languages.

3.3.1 Forward chaining

In a forward chaining system, the left hand sides of the rules are examined to see whether or not they are fulfilled. If so, the modifications to the database in the right hand side of the rules are executed and then the system examines the rules again. Forward chaining systems are sometimes called *data-driven systems.* A survey of forward chaining expert systems is Brownston *et al* (1985). Most forward chaining systems use pattern matching to express when rules are applicable. The LHS of the rules contain patterns that must match the contents of the database for the rule to be fulfilled. An example could look like

```
if (father -x -y)
   (father -y -z)
then
   (add (grand-father -x -z)).
```

In this notation, `-x,-y` and `-z` are matching variables that can match arbitrary symbols. When the same matching variable occurs at more than one place it must match against the same symbol at all occurrences.

The reasoning is performed in what is called a *recognize-act cycle* that has three states. During the match state all rules that are fulfilled are collected into the *conflict set* together with the corresponding database elements. If rules with pattern matching variables are allowed, the same rule can appear in the conflict set several times with different matching database elements. During the select phase, one rule is chosen for execution. If the conflict set contains more than one element, the conflict is resolved according to some *conflict resolution strategy.* During the act state the right hand side of the selected rule is executed.

The conflict resolution strategy is crucial for how the rule execution proceeds. Conflict resolution strategies can be divided into two groups. The first group consists of strategies that order the rules in a predetermined way. Examples of this are strategies that select rules according to the order in which the rules were created or according to a priority associated with each rule. Another example is strategies that favor more complex rules, e.g., rules with many condition elements, before simpler rules. The other group contains strategies where the choice of a particular rule depends on the state of database. An example of this are strategies that select rules on the basis of the recency of the matched database elements. Rules matched by more recent added information are usually favored.

Matching each rule against the contents of the database every recognize-act cycle is time consuming. It can be avoided by saving matching information between

the cycles and by matching only the database elements that are changed at each cycle. This is efficient because typically the database changes very little between the cycles. The most well-known matching algorithm of this type is the RETE algorithm (Forgy, 1982) used in the OPS family. Matching algorithms of this kind usually also allows negated left hand side conditions, i.e., patterns that must not match against the contents of the database.

An effect of the network based rule interpreters is that the recognize-act cycle now has a different ordering. The cycle starts with the select state where a fulfilled rule is selected. During the act state the right hand side of the selected rule is executed. This causes database elements to be added or removed and it is during these database alterations that the actual matching occurs. The network approach to pattern matching affects the addition of new rules to the system in a serious way. Since the rules build up the network used in the matching, this network must exist before database elements are added or removed. This means that a new rule which is added during rule execution will only recognize database elements that has been altered after the rule was entered. In order for the new rule to operate on the total database, the database elements already added to the database have to be refreshed, i.e., removed and added anew.

Another implication of network based systems is the effect they have on the use of predicates in the left hand sides of the rules. The intended use of LHS predicates is to further specialize the rules beyond what is possible through pure pattern matching. An example of this is the trivial rule

```
if (number -n)
 predicate
   (>= -n 1000)
then
   (add (large-number -n))
```

that simply marks a number as being large if it exceeds 1000. Predicates are tested as soon as enough partial matching information is available for the predicate arguments to have values. The reason for this is the wish to prune the network, and thus the search space, as early as possible. This works well as long as the LHS predicates act only on the database contents. It is, however, sometimes desirable to have predicates that act upon information outside the database, e.g., predicates that test if some measured signal exceeds a certain threshold or if a certain global variable takes a given value. In this case, the predicate evaluation may cause problems. The main reason for this is that the network, which in a way contains the state of system, can only be changed by database alterations and not by external events.

3.3.2 Backward chaining

Backward chaining systems are sometimes called goal-directed systems. A backward chainer tries to achieve a goal, or alternatively stated, to verify a hypothesis by trying to prove rules that confirm this hypothesis. A goal could, e.g., be expressed as the need to compute the value of a certain object attribute in the database, and a hypothesis could be expressed, e.g., as a certain value for an object attribute that needs to be verified. If the goal is not immediately available in the database, the backward chainer tries to find the rules with consequents that deduce the goal. A rule is selected and the antecedents of this rule become new goals that must be fulfilled. This causes new rules with these goals in their consequents to be selected and so on. The user is usually asked when a goal cannot be directly proven and no rules are found with the goal in their consequents. If a goal cannot be fulfilled the system backtracks and chooses another rule. The way in which the rules are selected and in which order the subgoals are analyzed determines the search strategy. During depth-first search the first applicable rule is chosen and its first antecedent immediately becomes the new subgoal. In a breath-first scheme all the antecedents of the chosen rule are checked before eventually a subgoal is selected. In a best-first scheme the rule most likely to succeed, e.g., with fewest antecedents, is selected first.

Backward chaining systems are often used for classification problems, see Shortliffe (1976) or Weiss and Kulikowski (1981). In these applications, the number of possible outcomes, i.e., the values of the goal attribute, is typically small. A feature of many backward chaining systems is the possibility to reason with uncertainty. Database elements have associated numerical certainty factors that reflect the amount of belief or disbelief in a certain fact. Rules also have certainty factors denoting to which degree a certain inference can be trusted. The inference engine propagates the certainty factors during the rule execution. The uncertainty feature is often combined with the possibility to let object attributes simultaneously take different values with different amount of certainty.

Backward chaining systems have traditionally well-developed explanation facilities. The explanation facilities are built into the inference engine and the explanations are generated automatically. The two standard types of explanation facilities are the "How?" and the "Why?" questions. When the system has drawn a conclusion the user may ask how the conclusion was reached. This typically results in a trace of the rules that were used in the reasoning. If the system asks the user for additional information, the "Why?" question explains why this information is needed. The user has sometimes the possibility to investigate the outcome of different answers with a "Whatif?" question.

3.4 Research issues and development trends

The "expert system boom" during the last years is more based on improved and commercialized hardware and software than on recent research progress. The main ideas behind current expert systems date from the late seventies. Although expert system solutions have been proposed for a huge amount of applications, it is only for few applications that the research community has reached a consensus on how they should be tackled. Among these are classification problems and configuration problems. This section will point towards some problem areas that remain unsolved and where much research is done.

3.4.1 Dynamic environments

The over-whelming majority of existing expert systems are static. The human user provides the expert system with some amount of initial information which possibly is completed later on during the consultation. In a dynamic environment, the situation is different. The process state varies with time, faults occur that change the behavior of the process, and humans interacts with the process.

The logical background for most existing system are a standard logic, i.e., propositional logic or first order predicate calculus, together with the *modus ponens* rule. This simply says that whenever a fact `A` is known to be true and there is a rule `If A then B`, it is permitted to conclude that `B` is true. Standard logic systems are all monotonic. If the logical statement `A` can be proved from a set of initial axioms, additional axioms or information must not cause the negation of `A` to be provable. If this was the case, the logical system would be inconsistent. Due to the monotonicity property, the belief of the expert system, i.e., the contents of the database, are considered to be true and the system monotonically draws new conclusions from the existing ones. Unfortunately, monotonic systems cannot handle three kinds of situations that often arise in real problem domains: incomplete information, changing situations, and the generation of assumptions during the problem solving process. The monotonicity shows up for example in classification systems where the user rarely has any possibilities to later change some of the information he has provided. A true on-line expert system must provide some way of *non-monotonic reasoning*. In real life we are often faced with the need to draw conclusions based on incomplete or uncertain information. Later, as new information comes in, the basis for the drawn conclusions may turn out to be wrong. The system then must be able to retract these conclusions.

The approaches to non-monotonic reasoning can be divided in two groups. The first group contains solutions where the logic is extended in several ways. The second approach is to include the logical system in a meta-system that handles the non-monotonic issues. Examples of extensions to the logic system are, e.g., the work on circumscription, (McCarthy, 1980), default reasoning (Reiter, 1980), the UNLESS operator (Sandewall, 1972) and the non-monotonic modal logic of

McDermott and Doyle (1980). A compilation of non-standard logics is found in Turner (1984).

Truth maintenance systems (TMS) (e.g., Doyle (1979); de Kleer (1986), Goodwin (1987)) are examples of the meta-system approach. The overall system consists of an ordinary inference system and a TMS that serves as a sort of intelligent database. The task of the TMS system is to determine which data that are to be believed when a new inference has been made and to ensure that the database is consistent. When an inconsistency is detected, it uses dependency-directed backtracking to resolve the inconsistency by altering a minimal set of beliefs.

These theoretical approaches to non-monotonic reasoning have not yet matured. Many problems are unsolved and the techniques have so far only been used in micro-world examples. TMS techniques also require the storage of large amounts of information.

Truth maintenance techniques are becoming available in the larger existing expert system shells. KEE contains an Assumption-based TMS that allows the user to set up dependencies, called justifications, between facts. TMS techniques are in KEE also used to implement multiple worlds. Worlds represent alternative states of knowledge. Worlds are proposed for applications that involve planning, configuring, or testing out different alternatives. They allow the reasoning process to set up hypothetical assumptions which automatically are withdrawn when worlds are deleted. ART contains similar features called multiple viewpoints.

3.4.2 Generic problem solving tasks

The structure and knowledge representation formalism used in many of today's expert system projects are to a large extent governed by available expert system shells. Ideally, the situation should be the opposite. The structure of the domain knowledge should decide which knowledge representation technique to be used.

Chandrasekaran (1986; 1987) and others (Gomez and Chandrasekaran, 1981; Marcus and McDermott, 1987; Clancey, 1985) argue that the abstraction level of current knowledge-based systems is too low. Instead of the level of rules-logic-frames-networks, they advocate the level of generic problem solving tasks. Each generic task uses the knowledge organization and control strategy most natural to the task. Chandrasekaran has looked at the problems of diagnosis, design and planning and identified such generic tasks as *hierarchical classification, abductive assembly, object synthesis using plan selection and refinement,* etc.

The approach is a critique of the uniform and low level architectures of conventional knowledge-based systems which lack expressiveness for higher level tasks and create artifactual control issues which often is mis-interpreted as issues having to do with control at the task level. An example of the latter is the need for different conflict resolution methods. Instead a higher-level, multiform architecture

is proposed. The approach is based on the view that *knowledge representation and use cannot be separated* (Gomez and Chandrasekaran, 1981).

This line of research coincides with the aim for application dependent expert system shells which will be discussed later. It also relates to the current focus on deep model-based knowledge which also will be discussed later. It is likely that task specific tools will play a major role in the future.

3.4.3 Distributed Expert Systems

Distributed artificial intelligence (DAI), (Decker, 1987), is concerned with solving problems by applying both artificial intelligence techniques and multiple problem solving agents. The distributed approach is motivated by the traditional positive aspects of distributed processing systems as well as from the cognitive viewpoint that *"All real systems are distributed"* (Hayes, 1980). DAI approaches reaches from fine grained (connectionist) to coarse grained (distributed expert systems). The connectionist approach will be described in the section on hardware development.

Distributed expert systems have two major dimensions: how control is distributed among the agents and how the agents communicate with one another. The control issue involves the amount of cooperation between agents, how the agents are organized and how coherent problem-solving behavior is obtained among the agents. The communication issue can be delineated into three areas: the communication paradigm, the semantic content of the exchanged information and the protocols used to handle the limited bandwidth.

The two major communication paradigms are shared global memory and message passing. Both have advantages and disadvantages. The most used model for shared memory communication is the *blackboard* model (Nii, 1986a; 1986b).

A blackboard system consists of a shared memory, the blackboard, and a set of logically independent agents or knowledge sources. The knowledge sources operate on and respond to changes on the blackboard. The knowledge sources contain the domain knowledge for a certain part of the problem solving. Knowledge could be expressed either as rules together with an appropriate inference strategy or as ordinary procedures. The choice of which knowledge source that should be executed is determined dynamically depending of the contents of the blackboard. The blackboard is usually organized into hierarchical levels. The relations between the objects are expressed through named links. A control module monitors the changes on the blackboard and decides what action to take, i.e., which knowledge source to activate. Knowledge sources that may be activated are contained in an *agenda*. The blackboard paradigm is a special case of a *oppurtunistic reasoning system*, where the most appropriate reasoning strategy, e.g., forward chaining or backward chaining, is chosen at every time.

In the HASP/SIAP project, (Nii *et al*, 1982), a blackboard system was used for

interpreting sonar signals collected by hydrophone arrays in some area of the ocean. This project is interesting since the system was used autonomously, in real time. The system was however only passively recording incoming information and thus had no feedback element. Blackboard based expert system shells are beginning to emerge. One example is the SOPE system from Advanced Decision Systems. The work by Chandrasekaran on generic problem-solving tasks has also been organized as a blackboard system (Gomez and Chandrasekaran, 1981).

A disadvantage with the shared memory is the bottleneck possibility. A more abstract means of communication is offered by message passing. In the *Actors* formalism, (Hewitt and Kornfeld, 1980; Clinger, 1981), object-like actors with actions and acquaintances reside in parallel with each other.

One of the most studied protocols for distributed problem-solving is Davis and Smith's *contract net* protocol (1983). Agents are organized into classes with general problem-solving goals. *Proposers* propose possible solutions. *Proponents* collect and present evidence in favor of a proposal. *Skeptics* collect and present evidence to disprove a proposal. *Evaluators* examine proposals and balance the work load of the system.

Theoretical work in distributed problem solving is sparse. Rosenschein and Genesereth (1984) have studied communication strategies. Halpern (1986) has written an introduction to the topic of reasoning about knowledge and belief. It is noted that most models are based on *possible worlds*. An agent is said to know a fact if it is true in all worlds that he thinks are possible. Situations are described using some modal propositional logic together with *Kripke structures* which contain a set of worlds, the truth values for the primitive elements in each world, and an equivalence relation for each agent that groups together the possible worlds for that agent.

3.4.4 Planning

The origin of the work on planning in AI is the General Problem Solver (GPS), e.g., (Newell *et al*, 1960). This was the first problem solving program that separated general problem-solving methods from task-specific knowledge. A task was described as a triple of an initial object (state), a goal object (state) and a set of operators. Operators were chosen on the basis of how much the difference between the initial object and the goal object was reduced. No information was assumed to automatically carry through from one state to the succeeding state. This means that the operator was responsible for generating all information in a succeeding state. The representation format of the states and operators were not predetermined and varied from one domain to another.

Much of the work in state-based planning is based on situation calculus, (McCarthy and Hayes, 1969). The representation format is usually first-order predicate calculus or some extension of it. Resolution is used as the problem solving

method. The domain under consideration is assumed to always be in a certain state. A state is described by means of predicates. For example, the fact that an object is at a certain position in certain state can be expressed with the following predicate.

```
at(object1,position6,state7)
```

Events, or actions, are represented as functions that takes a situation, including a state, and returns the resulting state. An axiom that describes that object1 can be pushed from position6 to position7 looks as follows.

```
(∀s) [at(object1,position6,s) ⊃
   at(object1,position7,push(object1,position6,position7,s))]
```

The function push returns the resulting state. An example of a planning system along these lines is described in Green (1969).

A general problem in planning is the problem of which relations that are affected by an action and which are not. This is referred to as "the frame problem", (Hayes, 1973). Frame here means the frame of reference in which a relation is true. In resolution-based planning systems and also GPS, it is necessary to explicitly state all relations that are left unaltered for each and every actions. For example, in the above scenario with objects at different positions it is, e.g., necessary to have axioms that says pushing an object from one position to another doesn't, hopefully, change the position of other objects. Since most actions have local affect, this leads to numerous trivial so called "frame axioms".

The perhaps most well-known planning system is STRIPS (Fikes and Nilsson, 1971). In STRIPS, each operator has associated a set of preconditions, an add list of clauses, and a delete list of clauses. Applying an operator results in the deletion from the model of all the clauses in the delete list and the addition to the model of all the clauses in the add list. All clauses which are not contained in the add or delete lists are assumed to be unaffected by the operator. This is called "the STRIPS assumption". STRIPS is an example of a nonhierarchical planner. This means that the plan developments consists of one level. The individual pieces of the plan are generated one after another starting at the beginning.

Hierarchical planners generates a hierarchy of plans with different degrees of details. The highest degree is an abstraction of the plan and the lowest degree is the full detailed plan. An example of a hierarchical planner is ABSTRIPS (Sacerdoti, 1974). Another example is the NOAH system, (Sacerdoti, 1975). NOAH uses procedural nets to represent plans. The procedural nets incorporate both procedural and declarative knowledge.

Problems with interacting subproblems can occur when a problem has conjunc-

tive goals. The order in which the goals are fulfilled are perhaps not specified, but can be critical for a plan to be found. A different problem arise when the conjunctive goals must be fulfilled simultaneously. The majority of the work in planning concerns sequential planning. Planning of parallel activities is a much more difficult problem. The STRIPS assumption also cause problems for planning problems in dynamic environments. The presumption that the world only is changed by the planning agent's actions makes it difficult to handle externally generated events.

Several planning systems have tried to extend the possible class of planning problem beyond what is allowed in a "STRIPS planner". The SIPE system, (Wilkins, 1984), can handle plans with concurrent actions. A plan consists of partially ordered actions. Actions without ordering are considered to be concurrent. Actions that do not share the same resource can be executed in parallel. The DEVISER system, (Vere, 1983), also allows plans with concurrent actions. External events which are known to occur at a certain future time are allowed. Duration times express how long time actions take. Time windows may be specified for goals, e.g., that a goal should hold between two time points. Deviser models time as nonnegative real numbers where time zero is the time of planning.

An related area is the work on theories of action, i.e., on what constitutes an action. Allen (1984) has developed a temporal logic for reasoning about actions. The driving force behind this works has mainly been problems concerning the meaning of action sentences in natural-language understanding. The temporal logic is based on time intervals rather than time points. The logic is a typed first-order predicate calculus where the types could, e.g., be time intervals, propositions that can hold or not hold during a particular interval, and objects in the domain. Dynamic aspects of the world are captured by the term occurrences. Occurrences are divided into processes, which describes activities not involving a culmination, and events which describes activities that involve a resulting outcome. A similar temporal logic has been developed by McDermott (1982).

The temporal logic of Allen has been extended with two modalities that can be used to support planning problems by Pelavin and Allen (1986). The first modality is the INEV operator. The statement `(INEV i P)` means that at time interval `i`, statement `P` is inevitable, i.e., regardless of what happens after `i`, `P` will be true. Using this operator, the possibility operator, `POS`, can be defined. The second modality is the IFTRIED operator. The statement `(IFTRIED pi P)` means that if plan instance `pi` was to be executed then `P` would be true. The resulting planning system can handle concurrent activities and externally generated events. The frame problem is basically solved through frame axioms. Although this logic system provides a general framework for expressing planning problems it is not obvious how it should be used effectively for practical problems.

A formalism for action structures with partially ordered actions that may occur

in parallel has been developed by Sandewall, (Sandewall and Rönnquist, 1986). In this work actions are described with preconditions, postconditions, and prevail conditions. The pre- and postconditions correspond to the delete and add lists of STRIPS. The prevail conditions describe the conditions of the world that must remain while the action is executed.

A more procedural approach to reasoning about actions and planning is taken in the work by Georgeff, (Georgeff and Lansky, 1986). In this work, actions are described by processes that have both a declarative semantics and an operational semantics. The use of processes is motivated by the fact that much expert knowledge is procedural in nature and thus is better represented procedurally than with action sequences.

3.4.5 Learning

An important aspect of human intelligence is the ability to learn. This is also an important research area in AI. Several learning paradigms exist.

In *inductive learning*, (Quinlan, 1979), a decision tree or a set of production rules is learned from a set of positive and negative examples. The inductive system tries to generalize from the positive examples without including the negative examples. Inductive expert system shells such as SuperExpert and Extran exist on the market. The inductive approach is specially suited for complex classification problems where the classification rules can be difficult to formulate but there exist a wide range of classified examples.

When the problem is to learn a plan of action in some domain, the possibility may exist for the learning program to attempt to execute the plans, observing how they fail. Learning from failures of expectation is called *failure-driven learning*.

Discovery is the restricted form of learning in which a learning system acquires knowledge without help from anyone. Two important programs have been built which have proved capable of *learning by exploration* in new domains. AM and Eurisko (Lenat, 1982; 1983) work for example in the domain of 3D VLSI circuits and the design of battle fleets for a space warfare game.

Learning has for long been considered one of the hard problems of AI. Recently, the interest has increased and this interest will probably remain in the future. Neural networks has interesting learning capacities which will be described in the section on hardware development.

3.4.6 Software development

The development of commercial expert system shells goes along two lines: general, hybrid systems and application-specific systems.

Hybrid systems consists of a flexible, integrated development environment that allows a variety of different knowledge representation techniques or program-

ming paradigms. Rule-based systems, frame representation, and object-oriented programming are usually supported. The environment often contains graphical interfaces, editors, browsers, and natural language based interfaces. The systems usually allows for separate development systems and run-time systems. The development is intended to take place on personal workstations and the run-time systems runs on less expensive PCs.

The prime examples are KEE, (Knowledge Engineering Environment) from Intellicorp and ART (Automated Reasoning Tool) from Inference Corporation. These systems also support multiple worlds and truth maintenance. Other examples are Knowledge Craft and the Swedish Epitool. Smaller and less expensive PC-based hybrid systems begin to emerge. Examples are Goldworks and Nexpert Object.

Application-specific systems are aimed at a certain type of application. The Escort system from PA Consultants is a dedicated system for real-time diagnosis. G2 from Gensym Corp. lies somewhere in between. It is general, hybrid system but with special focus on real-time applications.

Integration with conventional software is currently a strong trend. Many expert systems have interfaces to commercial software such as database programs and spreadsheet programs. The reason for the integration trend is the problems that have aroused when expert systems have been fielded. This trend will continue. Conventional software will in the future also include features which today are associated with expert systems.

3.4.7 Hardware development

Many AI applications are extremely processing intensive. To overcome the bottleneck caused by current computers, the research on special AI-machines is extensive. This section will describe both the development of uni-processor workstations and the work on parallel architectures. An important example of parallel AI architectures is the connectivist approach.

In Hwang *et al* (1987), AI computer architectures are classified into three groups: language-based machines, knowledge-based machines, and intelligent interface machines. Language-based machines are tailored for a specific AI language. Knowledge-based machines are tailored to execute a certain knowledge representation technique and intelligent interface machines are tailored to execute a certain AI problem.

Language-based machines for LISP

The nature of LISP poses certain requirements on the underlying computer architecture. The applicative and recursive nature requires an architecture that efficiently supports stack computations and function calling. The use of dynamic data structures makes an automatic storage allocation mechanism with efficient

garbage collection vital. The run-time type checking requires tagged architectures with efficient tag checking.

Lisp machines are personal workstations with special hardware for executing LISP. They have been on the market since the beginning of the 1980s. The most notable systems are the Symbolics 3600 serie and TI Explorer. These are uniprocessor architectures with hardware optimized for LISP and powerful, integrated programming environments. They are, however, also to a large degree research computers which have proved difficult to integrate with existing industrial computer systems.

The development of faster conventional microprocessors such as the Motorola 68020 and the Intel 80386 has made it possible to use conventional workstations such as Sun to execute Lisp almost as efficient as the Lisp machines especially for smaller programs that are not so dependent on efficient garbage collection and paging. The disadvantage is the need to have much memory ($>$ 16 Mbytes) and a reasonably large local secondary storage disc. The programming environment is also still much inferior to what is available on the Lisp machines. With new and faster processors and improved Lisp programming environments, conventional workstations will become well suited for AI applications in the soon future.

Lisp architectures have up to now been built from discrete components. Last year, Texas Instruments released the first Lisp cpu on a single chip. This chip outperforms discrete architectures by a factor of approximately 5 and the Explorer II is currently the fastest Lisp workstation. TI are also collaborating with Apple and they have recently released the μ-Explorer which is a Macintosh II computer with an add-on, chip-based Lisp board. This solution is very prize-performance competitive and will simplify the integration of Lisp architectures with conventional computers. Symbolics have recently also announced their Lisp chip and there are rumours of similar add-on boards to conventional computers also based on this chip.

Within the next few years it will be possible to buy a personal computer or workstation that contains different dedicated add-on boards such as a LISP board, a signal processor board, a graphics processor board and so on. Already now, the 80386 based Hummingboard gives IBM AT powerful Lisp performance.

Language-based machines for other languages

AI computers are also being developed for Prolog-like languages and for pure functional languages. Most of this work is still pure research. Prolog machines usually try to exploit the various kinds of parallelism in Prolog. And-parallelism refers to the simultaneous execution of logically AND-ed clauses. Variables which are common to several AND clauses must be taken care of specially. Or-parallelism refers to the concurrent search for alternative solution paths. This parallelism is non-deterministic and can give multiple solutions. Unification parallelism refers to the parallel matching of clauses in the Prolog database with

the goal clause and to the parallel instantiation of variables to constant variables. The Japanese 5'th Generation programme has led to the construction of the Parallel Inference Machine (PIM) and of Parallel Inference Engine (PIE).

AI machines for functional languages use computational models, such as the dataflow model and the reduction model, which have asynchronous and distributed control. Therefore they can exploit a greater degree of parallelism than von Neumann computers using control-flow. Examples of functional programming machines are ALICE (Darlington and Reeve, 1983) and the FFP reduction machine (Vegdahl, 1987).

Knowledge-based machines

Knowledge-based AI machines are governed by models for representation and manipulation of knowledge. Research is for example performed on machines for forward chaining production systems, e.g. (Stolfo and Miranker, 1986), for object-based systems, (Anderson *et al*, 1987), and for marker propagation in semantic networks, (Fahlman and Hinton, 1983).

Attempts at modeling the brain have inspired massively parallel computers that use thousands of processors, each having limited hardware and a small local storage capability, functioning as extremely reduced instruction set computers (RISC). These are called connectionist architectures and the related research discipline is called connectionism.

Connectionism

Much of the work in AI can be viewed as ways of figuring out the symbols that the mind uses in its reasoning and rules for how the symbols are manipulated. It is assumed that once the symbols and rules are known human intelligence can be modelled. This was stated as a hypothesis by Newell and Simon (1976):

> *The Physical Symbol System Hypothesis.* A physical symbol system has the necessary and sufficient means for general intelligent action.

This assumption is now challenged. AI can match the best human experts on certain narrow technical problems, but it cannot even begin to approach the common sense and sensory abilities of a five-year-old child. In many cases, humans seem to handle information in some form other than the symbolic assertions of traditional AI. The use of massively parallel architectures is now explored in an attempt to get around the limitations of conventional symbol processing. Many of these architectures are connectionist. The system's collection of permanent knowledge is stored as patterns of connections or connection strengths among the processing elements. The knowledge then directly determines how the processing elements interact rather than passively being stored in a CPU. Both

formal, symbolic schemes and analog approaches exist. The analog approaches are often called neural networks due to the possible similarity in function between these networks and the neural network of the human brain.

A connectionist system uses a very large number of small processing elements or units each connected to some other units in the system. The only information stored locally in the units are a few marker bits or a single scalar activity level. Long-term storage is acquired by altering the connection patterns. The simplest way to represent things in a massively parallel network is to use local representations, i.e. each concept is represented by a specific part of the network. There is neurological evidence that the human brain also use distributed representations, i.e., a concept is represented by patterns of activity among several neurons.

The Connection Machine (CM) (Hillis, 1985; Waltz, 1987) is a commercial products that uses local representation. It consists of a conventional workstation together with a network of between 16K and 64K small processor-memory units. All units can communicate with any or all other units. The CM is programmed from the front-end workstation in either extensions to C or Lisp. The CM uses data-level parallelism. Each element of data for a problem is stored and one element is stored per processor. The front-end computer executes a serial program where each step may involve computations in all of the processor units. For example, instead of using loops for computations on array elements all operations can be performed simultaneously. The computing rate of the machine is on the order of 64000 MIPS for single-bit operations and 2000 MIPS for 32-bit adds. One application for the machine is memory-based reasoning systems where large amounts of specific examples are stored instead of general rules. The systems classifies input examples by finding the example most similar among the already stored ones.

Distributed representation can be acquired in a *value-passing system* where each connection has an associated scalar weight. A unit computes its output as a weighted sum of its input which is passed through a nonlinear function. Value-passing units are layered. Each observable feature is represented by an input unit. Each of the possible hypothesis that should be evaluated is represented by an output unit. Units in intervening layers represent higher-order features. The Perceptron (Rosenblatt, 1961) was an early example of a two layer network. For two-layer networks, learning schemes exist which automatically adjusts the weights. For networks with more layers, the learning problem is worse. Back propagation is one technique used which is based on gradient descent.

Typical neural network applications concerns recognizing patterns of various kinds. Examples are speech processing, image recognition and processing of imperfect and inexact knowledge. In process control, smart sensors and control of robotic equipment are some examples.

Intelligent interface machines

This group contains architectures that have been proposed or constructed for e.g. speech-processing, image recognition, and chess-playing.

3.4.8 Summary

Prediction of the future development is difficult. This section has tried to pinpoint some areas where research is being performed that is relevant to using AI system for real-time process control. Theoretical attempts to tackle reason maintenance in dynamic environment is important for true intelligent real-time behavior. Application-specific tools and generic problem-solving tasks are important in order to reach above the rules-frames-object level. Modern control systems are distributed. Therefore the expertise should also be distributed. Learning has many applications in process control. The research oriented hardware development will probably go towards parallel architectures. For commercial use, conventional workstations will provide a good vehicle for AI applications.

3.5 Real-time knowledge-based systems

Real-time expert system applications also exist outside process control. Some areas were the technique have been tried are satellite control, and data analysis; the Pilot's Associate; autonomous vehicles; battle management; aerospace systems; robotics and vision systems; financial advise (for example, a market monitor, adviser, or trader); and medicine (patient monitoring).

Real-time applications contains a set of very complex problems. The following examples are taken mainly from Laffey *et al* (1988).

Nonmonotonicity

Incoming sensor data, as well as inferred facts, do not remain static during the execution of the program. The data are either not durable and decay in validity with time, or they cease to be valid because events have changed the state of the system.

If resources were unlimited, this problem could be solved by sampling at a high enough frequency and by recomputing all inferred facts at every sampling time. This is however not the case. One approached which is used in the PICON and G2 systems which will be described later is to attach currency intervals and time stamps to indicate if a sensor reading is still valid or must be refreshed. The currency intervals and time stamps are also propagated to inferred facts. This technique can only represent that a fact has become unvalid due to passed time. It cannot represent that a fact becomes unvalid due to an occurred event. To do

this Truth Maintenance techniques as described in the previous section must be used. These are however very resource intensive.

Continuous operation

Many real-time systems continue to operate until they are stopped by an operator or by some catastrophic event. Consequently, a real-time system monitor or controller must be capable of continuous operation.

Asynchronous events

A real-time system must be capable of being interrupted to accept input from an unscheduled or asynchronous event. Additionally, events can vary in importance. A real-time system must also be capable of processing input according to importance, even if the processing of less important input must be interrupted or rescheduled.

Scheduling is one way to achieve responsiveness to asynchronous events. In the MUSE system, interruptable knowledge sources are controlled by a scheduler. This also allows concurrent execution of knowledge sources.

Real-time constructs

A real-time expert system must allow a natural expression of real-time constructs such as interrupting the reasoning for a certain time of until a certain event occurs. Furthermore, the reasoning process can often be structured into separate parallel reasoning activities. It is an advantage if they are mirrored in the system.

Interface to external environment

A real-time system needs to gather its data from a set of sensors. Traditionally, expert systems have asked the operator for input.

Uncertain or missing data

Data can lose validity or have questionable validity due to degradation in sensor performance. Thus, a real-time system must be able to recognize and appropriately process data of uncertain or diminished validity.

High performance

For certain applications, high performance is a key issue. This is a bottleneck for many AI systems. In telecom applications, high performance is especially important.

Temporal reasoning

Time is an important variable in real-time systems. Typically, a real-time system needs the ability to reason about past, present, and future events, as well as the sequence in which the events occur.

Reasoning about previous sensor values and inferred facts requires that time histories are stored. Reasoning about future events and explicit reasoning with time is a difficult area. Allen's work on temporal logic for reasoning about actions (1984) is one approach.

Focus of attention

When a significant event occurs, it is important that the real-time system is able to focus its resources on important tasks. This is usually related to a systems possibilities to respond to asynchronous events.

Reasoning under time constraints

Reasoning under time constraints covers the problem where a reasoning system must be able to come up with a solution in time when the solution is needed. Furthermore, the best possible solution within a given deadline is desired. This situation is frequently occurring in process control diagnosis systems. The diagnosis system goal is to detect error symptoms, determine their cause, and propose or carry-out counteractions. The available time before a counteraction must be performed is by large determined by the nature of the error symptoms and the part of the plant where they occur. A time constraint for the solution can be determined by predicting the time before a catastrophic, irreversible situation evolves using the known dynamics of the system. In some situation, direct counter-measures must be taken prior to any diagnosis.

Reasoning under time constraints poses certain demands on the reasoning system. The system must be able to estimate the time needed for its internal reasoning activities and the solutions they might come up with. A measure of goodness on the solutions is also needed. This can be expressed in terms of completeness, precision, and certainty. The reasoning system must also be able to reason at different approximation levels where reasoning at the various levels give different degrees of goodness of the solution and require different amounts of time. Examples of such levels are compiled knowledge versus deep-model knowledge, and deep models at different hierarchical abstraction levels. Reasoning at a high abstraction level may give a quick answer to what part of the plant the symptoms origin from whereas reasoning at a deeper level can pinpoint the exact physical component which have failed.

Reasoning under time constraints is an area were very little has been done. The existing real-time shells have no means to cope with this.

Integration with conventional software

A real-time expert system must typically be integrated with conventional real-time software. The conventional software will performs tasks such as signal processing, feature extraction, and application specific I/O.

Application-specific reasoning strategies

An issue which is often overlooked is how application-specific reasoning strategies, e.g., a fault diagnosis strategy, should operate in a real-time environment. How should information that arrives from the process in the middle of a reasoning chain be handled. This information could perhaps give a better explanation to the current error symptoms. The situation is analogous to how a human can "wait and see" before making up his mind. It is also necessary to consider the potential hazards caused by not acting immediately. In a real-time process environment, the expert system also has the possibility to increase its amount of knowledge about the process through active experiments. This could for instance be used to determine if process components are working properly or not.

3.5.1 Real-time Expert System Shells

This section will contain a overview of existing expert system shells aimed at process control applications. The real-time applications that have been implemented are either written directly in some AI language or implemented in some tool. Control system manufacturers who wish to provide their systems with AI capabilities have two directions to go in: to develop an inhouse system or provide interfaces to third-party expert system shells. The latter approach is taken by The Foxboro Co. who intend to sell their new Intelligent Automation (IA) control system serie together with the shell Personal Consultant Plus.

The available real-time shells are either extensions to conventional tools or dedicated to real-time operation. The only real examples of the latter type are PICON and G2.

G2 and PICON

G2 from Gensym Corp. and PICON from Lisp Machines Inc. (LMI) are both expert system shells aimed at real-time, process industry applications. Gensym Corporation was founded by the group from LMI who previously had developed PICON. Therefore, there is a strong resemblance between G2 and PICON. Both of them are intended to be used on top of a conventional control system and mainly for operator support. Due to the similarities only G2 will be described here.

The G2 knowledge-base: G2 represents the current state-of-the-art in real-time expert system shells. The main parts of G2 are: the knowledge base, a real-time inference engine, the development environment, a simulator, and optional

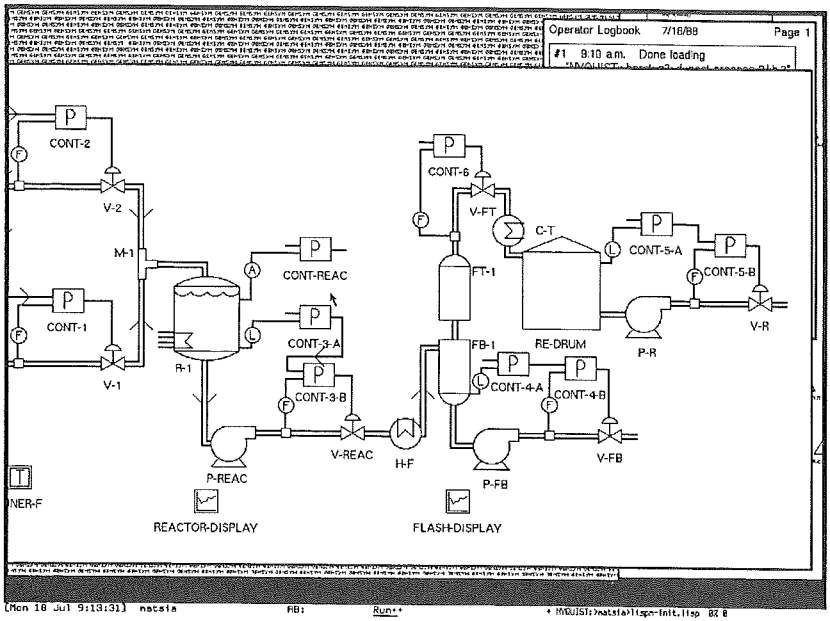

Figure 3.3 G2 process schematic

data servers. The knowledge base consists of three different forms of knowledge: Objects, rules, and dynamic models. Objects are used to represent the different concepts of the application. Attributes describe the properties of a certain object. The attributes may be constants or variables. Object classes with single inheritance are supported. Objects are represented by icons on a schematic as in Figure 3.3. Relations between objects are represented by connection objects. Usually, objects are used to represent the physical components in an application with the connections representing physical connections such as pipes or wires. It is however also possible to have objects that represent abstract concepts and connections that represent general relations between objects. Variables are a special type of objects that represent entities whose values vary over time. Variables have validity intervals indicating the length of time the value remains valid after having received a new value. Other variable attributes determine where the variable receives its value from (e.g., the simulator, a formula, or a data server), and whether a history should be kept for the variable or not. Measurement sensors are represented by variables.

Rules are used to encapsulate an experts knowledge of what to conclude from conditions and how to respond to them. All rules have an antecedent that lists the conditions and a consequent that tells what to conclude and how to respond. The conditions contains references to the objects and their attributes in a natural language influenced syntax. The major type of rule is the *if – then* – rule. G2 supports generic rules that apply to all instances of a certain class.

Dynamic models are used to simulate the values of variables. The models are in the form of first-order difference and differential equations.

The G2 real-time inference engine: The real-time inference engine initiates activity based on the knowledge contained in the knowledge base, simulated values, and values received from sensors or other external sources. Apart from backward and forward chaining, rules can be invoked explicitly in several ways. First, a rule can be scanned regularly. Second, by a focus statement all rules associated with a certain focal object or focal class can be invoked. Third, by an invoke statement all rules belonging to a user defined category, like safety or startup, can be invoked. The possibility to associate rules with several focal objects, focal classes, and categories gives a flexible way of partitioning the rule-base. The scanning of a few vital rules in combination with focusing of attention is meant to represent the way human operators monitor a plant. It is also an important way to reduce the computational burden on the system. Regular scanning of rules and thus updating of information in combination with variables with time-limited validity gives a partial solution to the problem of non-monotonic, time-dependent reasoning. The inference engine automatically sends out request for sensor variables that have become unvalid and waits for new values without halting the system.

G2 has a built-in simulator which can provide simulated values for variables. The simulator is intended to be used both during development for testing the knowledge base, and in parallel during on-line operation. In the latter case, the simulator could be used for estimation of signals that are not measured. The current simulator has however severe limitations. Each first-order differential equation is integrated individually with individual and user-defined step-sizes. This can easily cause problems. The numeric integration algorithm used is a simple Euler method with constant step-size. Further, the simulator interprets the simulation equations which is very time-consuming and thus slows down the system.

The G2 environment: G2 has a nice graphics-based development environment with windows (called workspaces), popup menus, and mouse interaction. Input of rules and other textual information is performed through a structured grammar editor. Facilities for browsing through the knowledge-base exist. The end-user interface is however still severely under-developed. The facilities that exist are primitive graphs, meters, and gauges that can be connected to variables in the application.

The dataservers are the interfaces to either conventional control systems or other signal sources such as e.g. databases. G2 runs on workstations and the communication with the process is done via Ethernet. A future goal of Gensym is to provide standard, off-the-shelf interfaces to the major manufacturers' control systems. In the mean-time, a generic interface exists which the user can modify.

G2 is implemented in Common Lisp and runs currently on Sun, Dec Vaxstation,

TI Explorer, Symbolics, and Mac II. For portability reasons, G2 uses their own window system and object-oriented system. To avoid garbage collection, care is taken for G2 not to generate any garbage. The current G2 system is still a beta release and under development. Many features are lacking. Among those are a proper operator interface, hierarchical objects, program generated objects, procedures, and explanation facilities. Gensym claims that they are working on these issues and that they will be available within a year. The procedure concept of G2 will be based on a Grafcet inspired, graphical, flow-chart representation.

G2 is intended for general real-time applications such as monitoring, diagnosis, and planning. It must, however, be remembered that G2 in itself does not give explicit support for any kind of application. G2 is an expert system shell on the rules-objects-attributes abstraction level. It does not provide any higher-level generic problem solving tasks.

G2 is currently installed at 20 sites with applications in process control, robotics, manufacturing, and simulation prototyping. The PICON system has been sold to around 50 sites with similar applications. One site is SCA in Sundsvall who are working with the system on the
fflstrand pulp plant.

MUSE

MUSE from Cambridge Consultants is a toolkit for the development of medium-scale applications in the area of real-time, embedded systems where the ability to deliver systems on inexpensive hardware is important. The origin of MUSE is a project funded by the Royal Aircraft Establishment, Farnborough, aimed at delivering real-time knowledge-based solutions to embedded avionics systems.

MUSE consists of an integrated package of languages for knowledge representation which all share the same set of database and object structures. The central component of the package is the *PopTalk* language. PopTalk which is implemented in C is derived from the Pop series of languages and has been extended to support object-oriented programming. It is a stack-based language that combines a strong list-processing element with a block-structured syntax. On top of the basic object language, a Frame system is built that includes multiple inheritance and demons.

A major part of MUSE is a set of architectural support facilities that allow a complex application to be split-up into modules. The modules includes knowledge sources and notice boards. A knowledge source contains one or more rule sets and a local storage to hold the data it is reasoning with. A notice board is a special case of knowledge source that only is used for storing data. The knowledge sources allow architectures that range from a single knowledge source containing a single rule set, at one extreme, to a multi-knowledge source system with many notice boards holding data at different levels, i.e. a blackboard architecture at the other extreme. The control of the knowledge sources is handled by the agenda.

The agenda is an ordered list of things to execute, typically but not necessarily knowledge sources. The system allows interrupts among the knowledge sources. When the agenda is empty, a scavenger function spreads notifications of changes around to the knowledge sources. This may then cause knowledge sources to be scheduled to run anew.

MUSE contains two rule-based languages that both operate on the same data structures, namely MUSE objects. The FPS system is a forward chaining production rule system based on a modification of the RETE algorithm. The rules may contain arbitrary PopTalk code both in the condition parts and in the actions. The BCS system is Prolog-type backward chaining rule system that supports depth-first backtracking, unification of logical variables on standard MUSE objects, and flow control via 'cut' and 'fail'.

MUSE is interfaced to the external world through data channels. The physical implementation of the data channels depends on the underlying hardware and software. For example, on a UNIX system the data channels are UNIX sockets. The data channels provide filter functions that only allow through the particular pieces of information that the applications decide are needed. These filters are user-definable and on the lowest level implemented in C.

MUSE is currently running on Sun machines. The system has however the possibility to download applications on simpler board computers for use in an embedded environment.

Personal Consultant Online

Personal Consultant (PC) Online is an add-on package that extends PC Plus from Texas Instruments and gives it certain real-time facilities.

PC Plus is a rule-based shell in the Emycin tradition. The basic way of representing facts is in the form of *parameters*. Parameters correspond to variables that describe different aspects of objects and problems in an application. Parameters are grouped into *frames*, which are used to structure and control the problem solving. The frame concept of PC Plus should not be confused with the use of frames in Frame systems. The rules in PC Plus are standard *if-then*-rules without any pattern matching capability. Rules and parameters may have certainty factors. Backward chaining is the major inferencing strategy but limited forward chaining is available in the form of antecedent rules.

PC Online consists primarily of functions that provide access to data external to the knowledge base for use as parameter values and for frame instantiation control. It adds the ability for a consultation to continue without interrupts for user interaction, alerting an operator only when necessary; and in the form of The PC Online Scheduler and the Knowledge Services includes the examination of time dependencies into PC Plus.

The PC Online Scheduler consists of a schedule list where each entry is a directive to instantiate a specified frame or subframe at a specified time. The entries may not have priorities and the execution of a certain frame can not be interrupted. The Knowledge Services provides the following four types of services: Storage, Trend Analysis, Time Stamps, and Counters. The Storage service permits a more controlled form of storage management than what is provided by the parameters. The Trend Analysis service manages statistical operations on time series of numerical data. The information that can be retrieved are mean value, dispersion, rate of change, past values, and predicted values. The Time stamps service stores time values associated with data used in the consultation. Finally, the Counter service supports simple counting functions.

Nexpert Object

Nexpert Object from Neuron Data Inc. Palo Alto CA is a hybrid system that combines object-oriented and rule-based representation. The object-oriented system supports multiple inheritance and demons. The rule system integrates forward and backward chaining using a single rule format and rule-based modification of the inferencing. The system also supports non-monotonic truth maintenance.

The reason why Nexpert Object can be seen as a real-time system is that it supports interfaces to external programs. The right-hand-side of the rules can call and run external functions or programs. Background processes can launch the inference engine either by suggesting one or more hypothesis to be verified or by volunteering data and thereby start the forward chaining, or modify the system's focus of attention.

The possibility to access external programs is nothing unique to Nexpert Object. Many systems provide similar constructs and therefore, at least in principle, have the possibility to be used for on-line, real-time operation.

Umecorp's Expert Controller

The Expert Controller from Umecorp, Larkspur CA, is a basically rule-based system delivered on a CMOS parallel processing microcomputer. The system is intended for diagnostics, advisory and real-time supervisory and adaptive control. The system is claimed to have a inferencing speed of 8000 rules/sec and to be the fastest expert system product available for many types of real-time automation applications. The system is claimed to contain object-orientation, blackboards, demons, meta-control, backward and forward chaining, hypothetical reasoning, and uncertainty management. The system is compatible with a wide range of conventional control systems. Umecorp's system have been sold to GM.

ONSPEC Superintendent

ONSPEC Superintendent from Heuristics Inc., Sacramento CA, is a rule-based expert system in the ONSPEC product family that also contains control software, I/O interfaces, statistical quality control software, automatic PID controller tuning software, ladder logic, modelling and simulation kit, and historical data analyzer. ONSPEC runs on IBM AT and PS/2 and compatibles. The system consists of rules which are entered interactively in a knowledge base builder that resembles a spreadsheet. The system is modular and each module may contain a rule set, procedural actions, calculation blocks, and menus.

3.5.2 Discussion

Real-time expert system applications contains a set of very complex problems. The solution to these problems influence the architecture and functions of the expert system. This is not always understood. Many projects take conventional expert system shells intended for consultative off-line applications and tries to use them for on-line purposes. That is not the correct approach.

The four systems described constitute examples of real-time expert system shells at various degrees of refinement. Neither of them fulfills all of the demands previously stated on real-time expert systems. The systems that come closest to it and represent the current state-of-the-art are PICON and G2. Not even these systems, however, solve the problem of reasoning under time constraints. In G2 and Picon, the real-time requirements are reflected in the inference engine. Individual rules can have associated scanning intervals and each data item has a validity interval. The MUSE system is an example were conventional, modularized rule-sets are embedded in an environment where the real-time aspects are taken care of by a separate scheduler. A similar approach has been taken by Årzén (1987), where a real-time, blackboard based expert system shell has been implemented. Personal Consultant Online and Nexpert Object finally represent conventional expert system shells which are extended to permit limited real-time behavior.

The implementation languages for these systems are different. G2 and PICON are both written in LISP. The motivation for this is based on development issues. The interactive, symbolical nature of LISP, and the recognized, high productivity of an experienced LISP programmer makes LISP an good implementation language for the software development companies. The user, on the other hand, is not allowed to write any LISP code. One reason for this is the need to avoid garbage collection. Muse is written in PopTalk which in turn is written in C. The Poptalk code is compiled to an intermediary format which is compiled. Personal Consultant exists in versions written both in Scheme, a Lisp dialect, and C. Finally, Nexpert Object is written in C. The wide range of languages used in commercial products indicate that there is no straight-forward answer to which language is best suited for real-time expert system applications.

The products from Umecorp and Heuristics Inc. are difficult to judge. The specifications of the systems are in very broad terms.

3.6 Conventional technology

The previous sections have given a technical overview of knowledge-based systems. One of the goal of this project is the integration of knowledge-based techniques with conventional techniques. Therefore it is important to assess the technical development within conventional systems. As this task is even wider than the previous one, a complete survey is impossible. The topics that are discussed here should rather be considered as indications of areas where the technological development will be of importance for the integration of knowledge-based techniques with conventional control systems.

3.6.1 Process and Control Systems Design

Tools for process and control system design are an important area where much development is being done with conventional techniques. The difference between the development directions is often large for different types of industries. For example, the situation in the telecommunications industry is very different from the process industry.

CAD, CAM, and CIM

An important area with applications mainly in the manufacturing industry and the electronic industry is CAD packages. Examples are systems for mechanical design and electronical design. These systems are undergoing a development from simple drawing tools towards systems integrated with the overall production system. With the trend towards CAM and CIM, the development of integrated design tools with high functionality will increase. The development of, e.g., systems that combine the design of mechanical components with planning of the manufacturing operations that are needed is a necessary component of CIM.

The use of expert systems as front-ends to CAD packages is an area where much work is being done. Examples are CAD packages for electronic systems and VLSI design.

Control system design

Interactive computer packages for identification, analysis, design, and simulation of control systems is a large area where much research is, and has been, done and where commercial products such as Pro-Matlab, Ctrl-C, and Matrix-X exist.

All these systems are interactive, command driven programs for matrix manipulations where the user easily can add new functions to the system. Toolboxes with special functions for identification, control design etc. can be added to Pro-Matlab.

Many simulation packages exist both for continuous and discrete simulation. Simnon from the Department of Automatic Control in Lund is a simulator for non-linear systems on state space forms. ACSL is another system of the same type. Numerous systems exist for discrete event simulation.

Several research programmes are currently running that investigate the next generation of Computer Aided Control Engineering (CACE) programmes. The CACE project is a Swedish national project which is funded by the Swedish National Board for Technical Development (STU) and which is being carried out at the Department of Automatic Control in Lund. The aim of the project is to investigate how new hardware and software technology will influence the next generation of CACE software. Within the project, several feasibility studies have performed concerning such topics as the use of expert systems as intelligent frontends, graphical based man-machine interaction, and symbolic formula manipulation. The project is now concentrating on the development of tools for modelling and simulation. The project is based on a hierarchical, object-oriented model of the process that allows multiple representations and reuse of models. The models are expressed in the form of symbolic equations which will be used to generate efficient simulation code, linearize non-linear systems, generate control code etc. A related research programme is the British Ecstasy project.

The work on CACE tools very seldom has connections to current distributed control systems. The development there is primarily focussed on graphical front-ends for control system configuration. Graphical function block editors are emerging for specifying the control system components and their connections. These system typically allow for function block classes that can be instantiated and connected with mouse interaction in a Macintosh style. Function blocks are mainly used to represent continuous control functions. For binary logic other graphical representations exist. One example is relay ladder diagrams. Ladder diagrams and function blocks are good for combinational logic control where outputs or actions are directly dependent on the states of inputs or conditions. For sequential control problems where the control actions are sequential or time dependent, relay ladder diagram and function blocks can be cumbersome, difficult to design and to troubleshoot.

Grafcet

Grafcet, also known as Sequential Function Chart, is a graphic method for designing and representing sequential logic control applications. Grafcet is originally a French national standard that has been adopted as a IEC standard.

Grafcet charts consist of steps, actions, transitions, and directed links. Individ-

ual sequential steps are represented by numbered squares. The initial step is represented by a square drawn with double lines. Actions associated with the steps are described inside one or more rectangles to the right of each step. Transitions from one step to the next one are represented by short horizontal lines. Conditions associated with transitions are written to the right of them.

The sequential operation of the control system is indicated by the sequential progression of the active steps. Every step is followed by a transition and every transition is followed by a step. Actions associated with each step are only active while the step is active. Provision for parallel, simultaneous sequences is an important feature of Grafcet. Another one is the possibility for hierarchical flow charts in the form of Macro-steps.

Grafcet can be used for many purposes. One is for specification and design. Several commercial systems use Grafcet like languages as a graphical programming language which internally is compiled into PLC code. Grafcet also allows for formal methods for validation and analysis. An example of this is automatic detection of deadlocks. A flowchart representation has also interesting possibilities as an operator interface for monitoring and troubleshooting. Highlighting of active step and transition conditions is one possibility for this (Lloyd, 1985).

Grafcet has been included in Telemecanique's PLC serie. Siemens uses similar ideas in their systems. APT (for Applications Productivity Tool) is part of Texas Instruments's TIstar control system. The APT system provides a Grafcet inspired, graphical language called SFC (Sequential Function Chart) for sequential control problems and a graphical function block language, CFC (Continuous Function Charts) for continuous control problems (Jeffreys, 1987). The APT is also claimed to contain expert system knowledge bases that automatically can generate parts of the control logic. Savoir, Oakland CA, has developed Flexis, a system for design of cell automation control systems that combine Grafcet and object-oriented programming (Morris, 1987). The Flexis system also includes an interface to MAP.

Specification languages

The origin of Grafcet is the Petri net theory (Peterson, 1981) that dates from the beginning of the 1960s. The basic building blocks of Petri nets are places and transitions. The data flow in the net is symbolized by tokens that marks whether a place is active or not. In some networks multiple tokens are allowed in the same step. The distribution of tokens in a network is called a marking.

Petri nets can be used to model parallelism, synchronization, rendez-vous, and resource sharing. Several specializations and generalizations to Petri nets have been developed. State machines allow only one input place and one output place to be connected to each transition. Colored Petri nets allow different colors or classes of tokens. Petri nets with inhibitor allow testing if place is free of tokens. Timed Petri nets associates firing durations to each transition.

Much of the current high interest in network theory among the control community is due to its graphical features. Developments in graphical hardware and software have generated a need for graphical representation languages. Petri nets have long been used in telecommunication systems. In telecommunication systems, the situation is somewhat different than in the process industry with respect to the nature of the automation systems used. The automation systems are based either directly on standard high-level languages such as Ada or on special purpose high-level languages dedicated to telecommunication applications. One such example is PLEX from Ericsson.

The dramatical increase in the number and quality of functions performed by the new computer controlled telecommunication systems has made formal specification tools or FDT's (Formal Description Technique) necessary. General requirements on specification languages include sufficient power of expression, a formal definition, abstraction mechanisms, and means for structuring. Three examples of such specification languages are *SDL, LOTOS,* and *Estelle.* A comparison between SDL, LOTOS and Estelle can be found in (Gelli, 1987). Other specification languages which emphasize abstract data type definition formalisms are CLEAR (Burstall, 1980) and OBJ (Goguen *et al*, 1985).

SDL

SDL (Specification and Description Language) is a language for the specification and description of systems. It has been developed and standardized by CCITT (The International Telegraph and Telephone Consultative Committee). Although originally developed for the telecommunication field, SDL is applicable for general applications involving real time, communicating, interactive systems. It has been designed for the specification and description of the behavior of such systems. It is also intended for the description of the internal structure of the system. SDL covers different levels of abstraction from a broad overview down to detailed design issues. It is not intended as an implementation language. Instead, translation tools to language like Ada have been developed.

The behavior of a system is described by the behavior of the processes in the system. A process is an extended finite state machine that works autonomously and in parallel with other processes. The communication between processes is asynchronous in the form of discrete messages called signals. Processes are contained in blocks that are connected with each other by channels. A hierarchical decomposition of blocks into subblocks and channels is allowed.

SDL provides a formalism for Abstract Data Type definition where an abstract data type is described in terms of its set of values, a set of operations on these values, and a set of axioms defining the operations. The primitive concepts of SDL are defined by an abstract syntax. Based on the abstract syntax, SDL contains one graphical representation, SDL/GR, and one textual phrase representation, SDL/PR.

LOTOS

LOTOS (Language of Temporal Ordering Specification) is a formal description technique that has been developed for the formal specification of open distributed systems, specifically those related to the Open Systems Interconnection (OSI) standards. The basic idea behind LOTOS is that systems can be specified by defining the temporal relations between the interactions that constitute the externally observable behavior of the system. LOTOS is based on process algebraic methods inspired by Milner's work, (1980), on CCS (Calculus of Communicating Systems) and Hoare's work, (1985), on CSP (Communicating Sequential Processes). As SDL, LOTOS contains a formalism for defining abstract data types.

Processes are described by ordering the units of their observable behavior. These units, called events, are atomic instances of synchronized interaction between two or more processes. LOTOS expressions that define the ordering of events are called behavior expressions. Behavior expressions are combined to new behavior expressions by the use of operators. Another feature of LOTOS are transformation rules that transform one behavior expression to others that express the same observable behavior.

Several tools for syntax-directed editing, syntax checking, semantics analysis, graphical interface etc. are being developed for LOTOS.

Estelle

Estelle is built on top of Pascal and therefore very close to an implementation language. The goal of Estelle is to raise Pascal up to a specification language for telecommunications systems. The power of Estelle is its compatibility with later stages of implementation, and the development environment adapted from the Pascal environment.

3.6.2 Control system operation

An important development field concerning control systems operation is information presentation systems. The development of enhanced control algorithms and statistical process control are other important fields.

Information presentation systems

Modern control systems are in general good at presenting information at the signal level. Various presentations, e.g., trend curves or bar graphs, can be chosen and dynamically updated process schematics are standard. A good example of state-of-the-art systems is SattGraph 1000 from SattControl.

SattGraph 1000: Sattgraph 1000 is a modern operator interface built upon computer graphics techniques such as high-resolution displays, windows, menus, animation, and mouse interaction. A hierarchical window concept combined with

information zooming allow the operator to move around in the process schematic at various levels of detail. A Macintosh style graphical editor is used for creating graphical objects and drawing process diagrams. The coordinates and colors of graphical objects can be connected to process variables and thus easily provide for animation.

Graphics software and hardware: The development in computer graphics is currently very fast. Graphics standards is an important issue. There is today only one generally accepted graphics standard: GKS (Enderle *et al*, 1984; Hopgood *et al*, 1983) adopted by ISO. An extended and improved standard, PHIGS (SIS, 1986; Shuey *et al*, 1986), supported strongly by IBM is coming. GKS is rather low-level and today's implementations are unacceptably slow. PHIGS has high-level features such as hierarchical structures for graphics.

An important part of a graphics system is the window system. Here the standard situation is very difficult. Several systems exist on the market, e.g., Macintosh, SunViews, X-Windows etc. It seems as if X-Windows will become the de facto standard.

The development in graphics hardware is even more difficult to judge. The development goes towards very high resolution display monitors combined with special purpose graphics processors. One of the leading manufacturers of graphics workstations is Silicon Graphics. Their IRIS 4D/70 superworkstation combines a RISC processor with a VLSI based graphics subsystem that handles object rotation, translation and scaling, six-plane clipping, and scaling to screen coordinates at a rate of 149.000 3D coordinates per second. This is displayed on 1280 x 1024 high resolution display. Texas Instr. and DEC are other manufacturers of graphics processors.

Statistical Process Control

Much attention is today being paid to what is called Statistical Process Control (SPC) and the integration of this into distributed control systems. SPC stands for the use of statistical analysis tools and techniques such as display of mean values and standard deviations, correlation methods, spectral analysis, hypothesis testing etc. The main use of the technique is to increase the operator understanding of the process and dependencies among its variables.

Adaptive controllers

Improvements of conventional controllers and specially adaptive controllers are continuing. Most manufacturers have some kind of adaptive controller in their product range either as a stand-alone controller or as a part of a distributed control system. The most advanced standard adaptive controllers of today probably Firstloop from First Control Systems AB and Novatune from ABB. Novatune

uses a minimum variance control strategy. Firstloop is based on pole-placement and can handle varying time delays.

3.6.3 Other relevant techniques

This section contains an overview of other hardware and software fields where the development will be important for the next generation of control systems.

Hypertext

The most general knowledge representation is by natural language sentences or text. From an expert system approach, text is far too less formalized to be useful. For other purposes, such as information storage and retrieval, efficient representation of textual information is an important issue.

The term 'hypertext' has been used quite loosely during the past 20 years for many different collections of features (Conklin, 1987). Another name for the topic is *nonlinear text.* The information stored in a file system hierarchy is linear. In many cases this organization is sufficient but for more and more applications a linear organization is inadequate. Techniques that extend the traditional notion of "flat" text files by allowing more complex organizations of the material have emerged. Mechanisms are being devised that allow direct machine-supported links from one textual chunk to another; new interactive interface techniques allow the user to directly interact with these chunks and to establish new relationships between them. With the new videodisc technology the concept *hypermedia* has been defined in which the chunks linked together can include graphics, time series signals, digitized speech, audio recordings, pictures, and animation.

Several implementations of Hypertext ideas have been done. The NoteCards system from Xerox Parc is one well-known implementation (Halasz *et al*, 1987). The HyperCard system from Apple is now being delivered together with Macintosh II. Symbolics' Document Examiner is another example. Here, the entire documentation of the Symbolics 3600 Lispmachines is stored on-line and easily accessible to the user.

Hypertext ideas are probably also very useful in a process control context. They could provide the basis for knowledge transfer between the different users of the system. Provided with efficient browsing tools and means for the user to enter different kinds of information in a structured way, the technique could fulfill many of the desired goals.

Micro-processors

New generations of micro-processors are developed at an ever increasing speed. The competition between Intel, Motorola, National and similar companies is high and results in new processors such as the Intel 80486 and the Motorola 68040. The

high speed and large virtual memory of these systems make them a reasonable alternative also for AI applications.

In a longer perspective the RISC (Reduced Instruction Set Computer) processors are emerging. An early example of this is the IBM RT. Sun has developed the SPARC architecture for its Sun-4 series of workstations. This architecture is now being adopted by other manufacturers as well. The SPARC architecture has been designed to support the C programming language and UNIX, numerical applications, and AI and expert system applications using Lisp and Prolog.

Computer Memory

Knowledge-based programming techniques are ill-reputed for requiring much primary and secondary storage. With technology progress this is becoming less and less of a problem. Primary memory 4 Mb DRAM chips are already here and 16 Mb chips are expected within a few years. For secondary storage, the optical disk technology is emerging. Erasable magneto-optic disks are expected that allow the storage of 600 Mbytes on a 5-1/4 inch disk. This is about six times as much per square inch as the most advanced Winchester disks.

Database systems

Database techniques are of growing importance in modern control systems and will be an essential part of a future knowledge-based control system. Relational databases begin to include object-oriented representation. They are also extended to allow on-line processing. SQL is emerging as a standard for database interfaces. Special purpose database architectures are also developed.

4

Knowledge Based System Applications in Process Control

This chapter gives an overview of knowledge-based applications in process control. Many applications have been proposed and implemented. In Section 4.1, a model of the tasks, tools, and roles involved in process control is presented. The rest of the chapter identifies different knowledge-based tasks and tools. Definitions and presentations of the problems are given within each task together with an identification of possible knowledge-based tools. Examples and approaches from implemented projects are also given for each task.

4.1 A Model of the Tasks, Tools and Roles in Process Control

One way to model a process control domain is to look at it as a collection of tasks, tools and roles, interacting with each other. The lowest level contains tasks to be performed in order to assure proper function of the controlled process. To handle these tasks in an efficient way, different tools are built. On the highest level are roles interacting with the tools trying to solve the tasks. Some roles could be identified as single persons, but other roles could be carried out by a group of persons or a whole organization. A single person could also have different roles depending on what he is doing with the process. An operator could e.g. sometimes work with maintenance. He is then acting in a different role than the operator role and perhaps working with other tools. Tools could

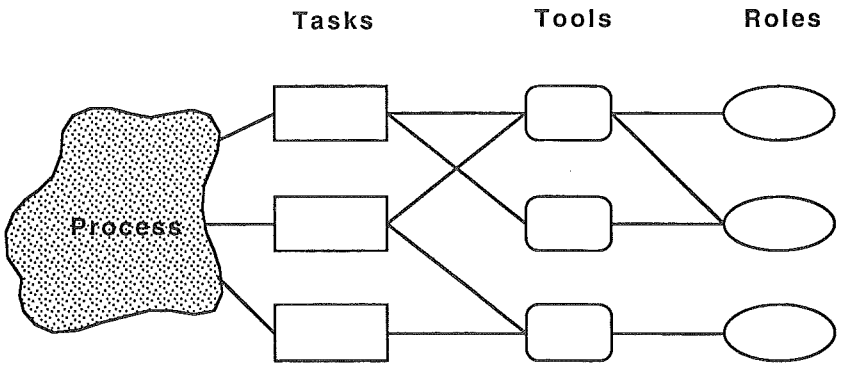

Figure 4.1 The tasks, tools, and roles model

either be autonomous or assist the different user roles. The same tool may be used to fulfill different tasks and to assist different user roles. Figure 4.1 shows the decomposition into tasks, tools, and roles

The interface between the roles and the tools in Figure 4.1 is an area not to forget. The man-machine interface is much more than just the graphics presented to the users. The interface must reflect the task the user is working with and make it possible for the user to focus on the problems rather than the data he is working with. One may also think of knowledge-based components in this interface. The system must be able to adapt to different users with different skill and experience. This could mean extensive help to some and short recommendations to other users. A man-machine interface capable of this must contain some sort of user modelling, where the behaviour of different users is taken into account. Knowledge based systems is one way to implement such intelligent man-machine interfaces.

The different tasks within process control, discussed in this chapter are divided into the following groups.

- Design
  - Process and control system design
- Operation
  - Monitoring
  - Control

  - Planning

- Maintenance
  - Preventive maintenance
  - Repair

This decomposition matches the three steps in the life-cycle of a system defined in Chapter 2 and shown in Figure 2.1.

Complexity criteria

Implemented knowledge-based systems within the different tasks can also be described according to their scope and nature. This description is also a measure of the complexity of the tasks. The following three criteria are used.

Single loop — plant-wide

Off-line operation — on-line operation

Operator assistance — closed loop operation

The cases in each pair above should be regarded as extreme cases and an actual system usually falls somewhere in between. The first criterion states if the system is intended for the global, plant-wide level or if it is used on the single loop level. The second criterion states if the system is used in off-line mode, e.g., in an analysis or design phase, or if it is used for on-line operation. The last criterion tells whether the system is used mainly for information and advice presentation to the operator or if itself actually carries out the control actions. This criterion is only applicable for on-line systems. The technically most difficult applications are those that are plant-wide, on-line, and used in closed loop.

4.2 Design

Tasks within this area are slightly different from other tasks. They are not connected directly to the operational environment. Knowledge about the configuration of the process and different functions implemented in the system are generated by the different design groups. This knowledge is used by many users and tools working with the process after delivery. One way to look at the design tasks is to picture them as knowledge sources, distributing knowledge to users and tools throughout the process.

4.2.1 Process and control design

Problem description

As described in Chapter 2, design knowledge is not used as efficient as possible in the process control systems of today. Knowledge is often stored only as separate documents and files which are difficult to access for the persons in need of the knowledge.

Design knowledge and documentation support tools

The use of knowledge based systems for capturing design knowledge and supporting the system documentation are important in order to store and distribute the design knowledge to other users groups. These tools could make it possible for the designer to produce the system documentation in a formalized way. A knowledge based system supporting designers must be able to store and handle a variety of both quantitative and qualitative process knowledge such as e.g., design decisions, drawings of the process, and descriptions of process function. All this knowledge could be represented using several knowledge representation methods like semantic networks, scripts and heuristics. Transformations and refinement of this knowledge may be necessary before presenting it to, e.g., operators and maintenance staff. Powerful browsing and retrieval functions must be included in the tools.

These tools are not limited to the design phase. They will be used during all phases of a process and by many different user groups.

Design assistant tools

Design assistant tools are important both in process design and control system design. The goal is to capture and encode expert designers' knowledge and experience and make it available to other designers. The system could automatically generate parts of a design from specifications. The systems could also be used to give critique on design suggestion provided by the designer.

A special case of control design is control system configuration. Here, a design tool could assist in the configuration and installation of the control system.

Approaches and examples

The majority of the work that has been done concerns knowledge-based design assistants. A knowledge-based system for process design in the case of factory design is described in Fisher (1986).

For control design at the plant-wide level, expert systems have been proposed for selecting appropriate input-output pairings given process configuration and control specifications (Niida and Umeda, 1986).

The University of Maryland in collaboration with du Pont has developed DI-CODE, an expert system for distillation control design, (Birky *et al*, 1988). DI-CODE is implemented in the KES shell, primarily using backward-chaining rules. The underlying knowledge model is based on the Goal tree – Success tree concept borrowed from Modarres, which will be described later. The design knowledge is extracted from P. Buckley at du Pont. The system uses a standard question and answer dialogue with graphical presentation of the final control system.

The Foxboro Co. has a similar system for distillation control design based on the experience of Foxboro's chief application consultant, Greg Shinskey (1986). Foxboro also has the Batch Reactor Consultant for design of different types of batch reactors (Blickley, 1987). The system consists of a conventional question and answer type of expert system together with graphical presentation and simulation facilities. Both Foxboro's systems are used for customer support on a consultative basis.

Expert systems for control design also apply to the design of single loop controllers. The Tunex system is an internal du Pont system for parameter tuning of PID controllers for various types of loops. The system is implemented in Personal Consultant Plus.

Another use of expert systems is as an intelligent interface to existing computer aided control design packages. The aim is to capture knowledge about different control design techniques such as SISO lead/lag compensation (James *et al*, 1985), multivariable design using linear quadratic theory (Birdwell *et al*, 1985), multivariable frequency domain design (Pang and McFarlane, 1987), state feedback and estimation (Trankle *et al*, 1986). The degree of automatization in design varies between the systems. In Larsson and Persson (1987), the expert system plays the role of an intelligent help system (IHS) for system identification. The help system spies on the user in order to find out his goal and gives guidance about which commands that are relevant to reach this goal. Scripts are used to represent meaningful command sequences.

Configuration of computer systems is a well-defined expert system application with examples such as DEC's R1-XCON system. Honeywell Inc's Customer Service Division has developed a configuration system called the Intelligent Software Configurator to assist field personnel in installing DPS 6 computers more rapidly. The Foxboro company has similar plans for its IA serie of control systems.

4.3 Operation

The second phase in the system life-cycle, shown in figure 2.1, is operation. This phase includes monitoring, control and planning.

4.3.1 Monitoring

Problem Description

To utilize the process in an optimal way, it is necessary to overview all process data available and come to fast decisions on which control actions to take. These decisions are based on real-time data collected by the process control system. Process alarms and failure situations must be handled fast to keep the quality at an acceptable level. In order to optimize the quality, a global view of the process and centralized decision making are crucial factors. To perform the right control actions, the control engineer is forced to search through and analyse a multitude of information. These tasks are becoming more and more difficult. Processes have increased in complexity. For example, future oil platform control rooms are being planned that will make available up to 20.000 signals for just two or three operators (Sachs *et al*, 1986). Processes are also dependent on automatic control systems to a higher degree than before. Much effort has been put on gathering and displaying process information, while very little has been done to aid operators in better understanding the overall process state.

Operators, engineers and others responsible for process operation must be able to focus on problem solving rather than searching through a large amount of process data in order to find the causes of alarm situations or quality losses.

A knowledge based system with knowledge of the process (e.g. configuration, optimization strategies and heuristics), could handle failure and alarm situations and determine the nature of the problems. It could also detect changing patterns in process data and produce early warnings, avoid serious failures and loss of quality. The system could either invoke automatic control actions or give suggestions to the operators.

Intelligent monitoring tools

An intelligent monitoring tool is a possible knowledge based system, dealing with real-time data. This tool could support the user choosing the right information to get a good overview of the process status. The user could tell what he wants to know or what he wants to focus on and a knowledge based tool could collect relevant information to the operator.

These monitoring systems must contain knowledge of how the process is designed and knowledge of how to optimize the quality. The users of these tools are the operators responsible for safe operation of the process.

Intelligent alarm handling tools

Alarm handling is an area in process control where knowledge based systems have been applied with good results. The alarms generated in a process which are presented to the operators are often on a low descriptive level, e.g., of the type *too low* or *too high*, and only give rudimentary information about the problem. A primary alarm is often followed by secondary alarms which obscure the cause of the original alarm. Therefore, the alarms must be analysed before the real problem can be revealed. The task of analysing the causes of alarms requires knowledge about configuration and function of the process.

In the telecommunication domain, serious cable failures could cause many alarms to be sent to the operator. A situation like this where the operator is working under time pressure is especially difficult to manage. A knowledge based system could help the operator in finding the cause of the alarms and give suggestions how to avoid too lengthy service losses. The system could, e.g., give suggestions on alternative routing possibilities and how to manage different faults.

There are several examples of knowledge-based systems for alarm handling. Two different approaches exist. One approach is to use low level alarms from the process and analyse them to find the cause of the problem. The other approach uses measurements from the process and produces descriptive alarms based on facts about how the process behaves in different situations. This approach makes it possible to produce early warnings of evolving faults. A typical alarm system contains rules which describe causal relations between measurements or low level alarms and their possible causes, and causal relations between different alarms.

Condition monitoring tools

A natural extension to a monitoring system is the ability to detect error situations before they occur. This type of prediction is done by the operator by looking at trend curves. A condition monitoring tool could be seen as an on-line diagnosis system, monitoring the process in order to reveal deviations from normal operation. Typical applications are finding worn-out components, changes in external conditions, etc.

Diagnosis tools

Diagnostic tools could be used to tackle problems like intermittent failures and failures not causing alarms. The systems could be used as an on-line support tool for trouble shooting. It must contain basic design knowledge about the process and heuristic knowledge from skilled operators and maintenance technicians. A diagnosis system is more focussed on interaction with the user than an alarm handling system. The knowledge and experience of both the user and the knowledge based system are combined to be able to deal with difficult problems.

Memory-based monitoring and diagnosis tools

Modern control systems are strong on measuring process variables and storing large amounts of data cheaply and efficiently. It is however difficult for the process engineers to make use of this vast amount of raw data. Many multivariable relationships cannot directly be seen by visually inspecting the raw data. System cultivation or hypothesis feedback modelling (Moore, 1986) is an emerging research area that makes use of reduced precision computation and concepts from algebra to extract process attributes relevant to hypotheses of system operation, and inference algorithms that can produce tests on process attributes to conclude that a specific hypothesis is correct.

As data storage and retrieval techniques are improved, the possibility to make further use of the control systems historical databases increases. One possibility could be to store normal and unnormal operating conditions that have occurred and to make use of them to guide the operators in prediction of the system state, recognizing error situations that have occurred before, and to give suggestions on control actions that may improve the situation.

Approaches and examples

Monitoring, diagnosis, and alarm handling are perhaps the most well-known expert system applications in process control. A large amount of applications exist of which several have been fielded.

The expert system is usually used on top of a conventional control system. The applications are typically plant-wide and the expert system is used more or less on-line. Usually the expert system is used as an operator assistant.

First generation diagnosis systems

Diagnosis systems may operate from different principles. The first generation of diagnosis systems are based on the operators heuristic knowledge about faults that usually occur and what causes them. This knowledge is usually represented as rules. Another name for this kind of knowledge is compiled knowledge where compiled refers to the compilation of past experiences into rules.

One motivation for the approach is the widely recognized observation that product quality and plant operation varies between different operator shifts. Certain operators are more skilled in running the process. By capturing their knowledge, spreading it out, and providing it on a 24-hour basis, productivity may increase.

The drawback with the approach is that the knowledge is process specific and easily becomes out of date when the process is updated. It is also impossible to

detect unanticipated faults for which the operators have no solutions. An advantage which the approach is that is based on compiled knowledge and therefore more efficient than deep, generic methods.

Second generation diagnosis systems

The second generation of diagnosis systems are based on *deep-level* or *first-principles* knowledge. Deep-level knowledge stands for knowledge that is generic and model-based. The knowledge can be represented in various ways. Possible models are causal models, local fault models, etc. The first generation diagnosis systems try to represent heuristic knowledge of the faults that might occur in the plant and their causes. Deep knowledge systems instead model the correct, intended behavior of the plant. During diagnosis, measurements from the plants are compared against the plant model. Mismatches between the model and the measurements are indications to faults.

The advantage with this approach is that the diagnosis system is generic and that it, at least in theory, can detect all possible faults in a process. The drawback is that evaluation of deep-model knowledge is slow compared to using compiled knowledge.

Deep diagnosis systems are probably the largest research area in process control oriented expert systems. Unfortunately, there is no universally agreed upon formalism for how knowledge should be represented.

Two different approaches exist. One is based on a functional decomposition of the process and the other is based on a structural decomposition. A structural decomposition divides the plant into subsystems on the basis of spatial relationships or based on subsystems that can naturally be decoupled. A functional decomposition divides the plants into subsystems depending on functionality. Since a single equipment unit can have more than one functional task it can be assigned to more than one subsystem. Often a sharp distinction cannot be made between functional and structural decompositions. The functional approach is, however, claimed to be more efficient.

The deep knowledge approach and the heuristic approach complement each other. The experiential, heuristic knowledge cannot be excluded. It is however unwise to depend solely on heuristic knowledge.

MODEX2: MODEX2 is a system developed by Venkatasubramanian (Venkatasubramanian and Rich, 1987) at Columbia University. The system integrates compiled knowledge with deep-level knowledge in a two-tier architecture. The deep-level knowledge used falls into four categories: constraints derived from material and energy balances, confluence equations representing the qualitative influence of one variable on another variable, a library of local fault models of different process units, and causal models of process units. The decomposition

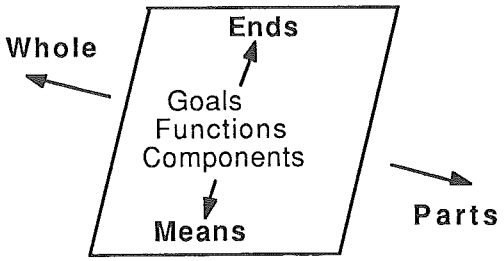

Figure 4.2 MFM abstraction relations

used is mainly structural. When diagnosing a process upset, the system starts by looking for heuristics concerning this symptom. If none is found, the system starts a deep level diagnosis. Whenever a new signal symptom is evaluated, the system start by looking for heuristics. The heuristic knowledge can be viewed as shortcuts through the search space that speeds up execution. Work is also performed investigating automated learning of heuristics (Rich and Venkatasubramanian, 1987). MODEX2 is implemented with a combination of object-oriented programming and rules.

Multilevel flow models: The Multilevel Flow Modelling (MFM) technique developed by Lind (1983; 1987) aims at a functional description of complex processes with diagnosis as one application. Different types of abstraction are used in MFM. Systems are described in terms of tree different types of entities: goals, functions, and physical components. Each entity represents a particular view of the system, i.e., a system can be described by the goals it should achieve, in terms of the functions provided, and by its physical components. Two basic types of relations exist among entities: the part–whole relation and the means-ends relation. The structure is described in Figure 4.2.

A basic feature of MFM is a set of primitive, icon-based, function concepts related to the representation of flow of material, energy, and information. Examples of these are the storage function, the balance function, the transport function, the barrier function, the source function, and the sink function. MFM also contains graphical representations for goals and control functions, for relations among functions, and for networks of functions. The MFM technique has been used by Soren T. Lyngso for diagnosis of power plants.

Goal trees – Success trees: The Goal tree–Success tree (GTST) method for modelling deep knowledge has been developed by Modarres (Chung and Modarres, 1987; Kim and Modarres, 1987; Modarres and Cadman, 1986). The GTST model is a top-down model. The objective or goal of the process is hierarchically partitioned into subgoals which then are further partitioned. The partitioning is continued until the description of the goals cannot be made without referring to

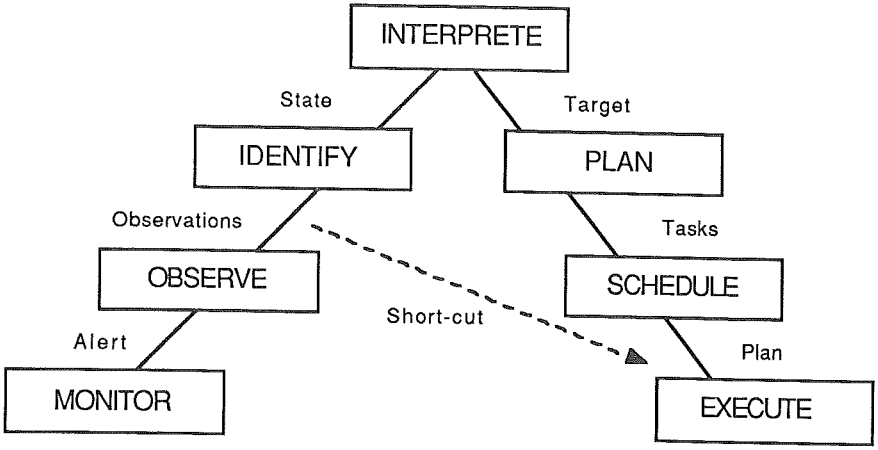

Figure 4.3 Rasmussen's decision model

plant hardware. This consists the goal tree. Looking upwards from any subgoal towards the top goal defines *why* the specific subgoal must be satisfied. Looking downwards from from any goal towards the bottom of the tree defines *how* the goal is satisfied. The success tree is a logical model of the plant hardware from which success paths can be determined. A success path shows various components whose proper operation guarantees the successful operation of the system. The GTST model have been used in a pilot system called CFWAVA (Expert System for Condensate and Feedwater Availability Operation) for a lab-scale pressurized water reactor.

Cognitive Operator Models

A large interdisciplinary research area concentrates on the role of the human operator in complex information processing systems. Inspired by cognitive psychology, attempts are made to model the behavior of human operators in situations with high cognitive burdens. Much concern is put on the nature of the mental models that human operator are claimed to have of the physical process that they are controlling. Much of the work in this field is based on empirical studies. A generally accepted model or even taxonomy does not exist yet.

Rasmussen (1986) has developed the decision model of Figure 4.3 that describes the different tasks involved in systems control. In the figure, rectangles represent decision tasks which require information processing for their solution whereas the words in small letters indicate states of knowledge. The model describes the sequence of tasks to be performed in a rational diagnosis and planning approach. In reality, many operators make a shortcut from a set of observations to an action.

This can be dangerous in supervision of complex processes.

Rasmussen (Vicente and Rasmussen, 1987) also describes the activities associated with three levels of cognitive control: the skill-based level, the rule-based level, and the knowledge-based level. Skill-based activity refers to behavior that is readily available as well-learned, automatic reactions that can be executed without requiring attention. Rule-based activity refers to behavior where the person knows what to do, either from memory or from an external description. Knowledge-based activity refers to behavior where the person does not know what to do or where a suitable description cannot be found. He therefore has to use his general knowledge to structure the situation and develop a suitable strategy.

An issue that is coupled to the cognitive approach is the design of the user interface. The process complexity apparent to the operator depends on the technology and the structure of the user interface. Several projects study the user-interface aspects. A major part of the ESPRIT I project GRADIENT described in Chapter 5 concerns the user-interface. The IGE (Intelligent Graphical Editor) module aims at providing the designer of graphical presentations with adequate decision support throughout the design process. The project also concerns expert system techniques for information selection and information presentation in different situations.

Qualitative simulation and analysis

Prediction and avoidance of future error situations sooner or later lead to simulation. Simulation also requires process models. A large practical problem is the fact that reliable quantitative models are difficult to extract and to keep up to date. This is one of the motivations for qualitative simulation and analysis.

The recent interest in qualitative methods has its roots in the work in Naive Physics (Bobrow, 1984). Naive Physics aims at describing and analysing the behavior of physical systems in the qualitative terms that humans use. De Kleer and Brown (1984) have developed a qualitative physics that associates qualitative constraints with the components and connections that make up a system. Forbus (1984) has developed a qualitative process theory that derives a constraint model from the set of active processes identified in a physical situation. Kuipers has developed the QSIM algorithm (1986) that predicts the possible qualitative behaviors of a system. Qualitative abstractions of differential equations, so called constraint models, form the basic representation.

A constraint model consists of symbols representing physical parameters and constraints of how the parameters interact. Some constraints specify mathematical relationships such as derivate, sum, and product whereas others assert qualitative functional relationships between parameters. Examples of such relationships are monotonically increasing or decreasing relations. Parameter values are expressed

qualitatively in terms of their order relations with a set of landmark values which always include zero and $\pm\infty$. A minimal set of landmarks gives the qualitative values positive and negative.

The outcome of a qualitative simulation is a graph of possible future states of the system. This is called an envisionment. The possible behaviors of the system are paths through the graph starting at the initial state.

A large problem with qualitative simulation is the difficulty to decide which behavior that actually will take place. Systems with feedback will in general give multiple behaviors. The technique also mainly treats the stationary case. Current research tries to use quantitative, measured information to resolve the ambiguities.

Industrial projects

Many industrial projects have been implemented within monitoring and diagnosis. A short overview of some of the most interesting will be given.

Westinghouse Corp. have developed GEN AID, a diagnostic expert system for on-line generator diagnostic service (Osborne *et al*, 1985). GEN AID is running in Westinghouse's diagnostic center in Orlando, FL and is connected to eight Westinghouse generators at different power plants. The system monitors 110 sensors at each generator and can identify 350 error conditions using over 8.500 rules. GEN AID is developed using a domain-specific tool called PDS (Procedural Diagnostic System) developed by Westinghouse and Carnegie-Mellon University. Westinghouse plans to field TURBINE AID for turbine diagnosis and CHEM AID for steam-chemistry diagnosis in May, 1988.

The Alarm Filtering System (AFS) developed at Idaho National Engineering Laboratory uses functional relationships between alarms to filter and rank alarms allowing the operator to focus on the most important alarms (Corsberg, 1987). A problem in many systems is nuisance alarms that stem from harmless process states. Conventional systems usually have the possibility to assign priorities to alarms. This ranking is however static and cannot adjust to different operating modes. The basic problem is to determine an alarm's importance relative to the current state of the process. AFS uses knowledge about generic alarm relationships and knowledge specific to a certain alarm or process state. Examples of generic alarm relations are *direct precursors* that express that there is a causal relation between two alarms and *required actions* that express how, given an active alarm, the operators expect some system response or alarm within a specified amount of time. AFS is implemented using object-oriented programming.

Honeywell has developed COOKER, a real-time process monitoring and operator advisory system for batch manufacturing processes (Allard and Kaemmerer, 1987). COOKER is implemented on a Symbolics and connected to a Honeywell

TDC 2000 Process Control system and to a programmable logic controller via an IBM AT. COOKER provides the operator with continuous identification of the current phase of the process, gives assistance in avoiding or recovering from undesirable process conditions, gives advise on recovery actions to take, indicates the degree of urgency of the actions, notifies the operator when the process condition is back to normal, and provides explanations of its rationale. Honeywell sees three components as necessary for real-time batch process monitoring: phase tracking, condition expectation monitoring, and event expectation monitoring. During each phase in the batch the operators have expectations about conditions that should hold over the process variables and expectations of events which should occur within some time period. In an initial approach, a Problem/Cause Tree approach was used represented mainly as rules in KEE. Several problems became apparent. The rules tended to depend very much on the methodology for handling information and problem solving used which caused much rewriting of rules. The second approach used a Goal/Subgoal approach somewhat similar to the Goal tree – Success tree approach by Modarres. By modelling the desired, correct behavior of the plant instead of a set of problems and causes much was gained. The single representation could be used both for monitoring and diagnostics. The system is implemented using object-oriented programming.

The FALCON (Fault Analysis Consultant) system has been developed by The Foxboro Co., du Pont, and University of Delaware for fault detection and analysis at an adipic acid reactor at du Pont (Lamb *et al* 1986). The system is a combination of a quantitative approach to diagnosis based on first principles, and heuristics. All this knowledge is implemented as rules.

The YES/MVS system is an experimental expert system for the operation of large IBM mainframe computers (Milliken *et al*, 1986). Several versions of the system have been developed and evaluated with good results. The basic architecture is a RETE-based forward chaining production system with real-time extensions. An offspring of this project is the YES/L1 expert system shell (Cruise *et al*, 1987) that has further been developed to the IBM product Knowledge Tool that combines rule-based programming with PL/I.

STOCHASM performs diagnosis for the lubrication oil subsystem in a a gas turbine propulsion unit on a US NAVY ship. Campbell Soup Co. uses an expert system for diagnosing soup cooker problems. The ESCORT system for process diagnosis from PA (described in the travel notes) is based on causal relationships between process variables. A large installation of the system will be done at British Petroleum during 1988.

In the telecom area, SHOOTX (Koseki, 1987) is an example of a diagnosis system finding failures in a packet switched network, where both design and heuristic knowledge is used. In the LES (Laffey, 1986) system, a switching network is represented in a frame structure to facilitate the reasoning process. Toast is an example of an intelligent on-line assistant working in a power network (Sarosh,

1986).

4.3.2 Control

Problem Description

Knowledge-based systems can also be used for active control of a process. Traditionally, control systems are working with control of single signals in a process, keeping the values at given set-points. Knowledge based control system could work with more complex control tasks, like optimizing production or quality. Faults and alarm situations may also be corrected automatically.

A knowledge based control system may give the opportunity to work on a higher abstraction level when optimizing a process. Goals of operation could be described in more general terms and the system itself could translate those goals to basic control actions. A system like this must contain knowledge from a variety of domains, e.g. control knowledge, process models, and heuristics.

An example in telecommunications would be a tool capable of performing intelligent routing in a network. The tool must have knowledge about network configuration and the traffic situation in order to make decisions on alternative routing tables. It could also be able to simulate various decisions to make optimal decisions. A control system like this could automatically avoid traffic congestion and keep the service quality at a high level.

Manual control assistance tool

A first step towards closed loop control is to assist the operator in the manual control. A manual control assistance tool could use simulation to give "What if" support, i.e. to show what will happen in the process if the operator makes a certain control action or changes a certain parameter.

Quality control tool

Quality control is often mentioned in connection with knowledge-based systems. The aim here is to close the outermost control loop that determines the quality of the end product. This loop is often characterized by control objectives that are difficult to measure directly and thus exposed to the operators subjective judgements. The knowledge used is the operators heuristics combined with theoretical knowledge.

Plant-wide tuning tool

Plant-wide tuning is one possible application of knowledge-based systems. The goal here is to, on a plant-wide basis, monitor and adjust the operation of the control loops of the plant.

Direct expert control tools

In this tool the knowledge-based system is used directly for closed loop control. Several different approaches exist. They have all only been applied to smaller parts of a plant.

Approaches and Examples

The British RESCU project has focussed on the quality control problem. The target plant was an ICI chemical batch process subject to unmeasurable control variables, variations in feedstock, a wide range of specifications of the different end products, and unreliable instrumentation. The knowledge-based system was based on the standard procedures for running the plant, the operators expertise, and theoretical, chemistry knowledge and consisted of about 600 rules.

The company Artificial Intelligence Technology has performed a feasibility study on knowledge-based plant-wide tuning on the account of Combustion Engineering. The study has so far resulted in a demonstrator that can handle the tuning of PID controllers, both in single and cascaded loops. The method that is used is based on periodically introduced step changes in the control signal and measurements of the reaction curves.

Fuzzy Control

Fuzzy control is one approach to direct expert control. The area has existed since the beginning of the 1970s, far before the current expert system boom. In fuzzy control, expert system techniques are used to mimic the human operators' manual control strategy. It is expressed as qualitative, linguistic rules for how the control signal should be chosen in different situations. These rules replace the conventional controllers. The underlying logic is based on the fuzzy logic of Zadeh (1965). The intended applications are control of complex processes for which either appropriate models do not exist or are inadequate, but where the human operators can manually control the process satisfactorially. The fuzzy control approach can be used both for direct control and for set-point control of conventional controllers.

The fuzzy control approach has been successful in certain applications. One example is control of cement kilns. The British cement company Blue Circle claim to have spent 100 man years trying to model and control cement kilns by conventional techniques without success. Fuzzy controllers is today a standard approach for control of cement kilns. The Danish cement company F.L. Schmidt

are developing and selling their own fuzzy controller, FCL. Sira Ltd in the U.K. also have a system called Linkman.

The claimed advantage with the fuzzy control approach is mainly that it gives a very good operator understanding. This makes it possible to in a intuitive way control very tight coupled multi-variable systems, something which is difficult with conventional techniques. Fuzzy controllers are also inherently non-linear which is a desired property in many cases. It has been shown that a single loop fuzzy controller which only uses the output and the rate of change of the output is equivalent to a multi-level relay controller. Further it has been shown that, as the quantification levels are made smaller, the controller approaches a conventional PI-controller. A second feature of fuzzy controllers is the possibility to let the controller generate a control output only if the control error is large enough. This "do-nothing-if-not-needed" effect is not achieved by linear controllers.

The fuzzy control approach is based on shallow knowledge where the experience of the operators is compiled into rules. The drawback with all shallow approaches is that they have a limited justification and a narrow range. In the work by Leitch (1987), it is argued that the standard decomposition into shallow and deep knowledge is not sufficient. Rule-based control approaches can very well be based on deep knowledge and vice versa. Instead he proposes a division into empirical knowledge and ontological knowledge. The empirical knowledge is based on experiential knowledge and may consist of both shallow empirical models and deep empirical models. The ontological knowledge is based on theoretical knowledge which is analytic and derived from first principles. Artifact (Francis and Leitch, 1985) is an example of a controller based on a deep empirical model and implemented in Prolog.

Expert Control

Expert control is another approach to direct expert control. Expert control seeks to extend the range of conventional control algorithms by encoding general control knowledge and heuristics regarding auto-tuning and adaptation in a supervisory expert system. The motivation for this is the observation that many control loops are badly tuned or run in manual mode. This situation decreases the production quality and increases the cognitive burden of the process operators. The reasons for the poor control are many. One is that the control loop is badly tuned from the beginning. Another is that the operating conditions have changed since the initialization of the controller. The conventional solution to the problem is to use some kind of adaptive controller. Although adaptive controllers are beginning to emerge in industrial control practice they do have problems. One is that they have many parameters which must be set explicitly. Examples are model orders and prediction horizon. Such information can be difficult to obtain and operators typically lack the intuitive understanding they have with conventional PID controllers.

The idea behind expert control is to provide the controller with enough general control knowledge for it to be able to itself automatically extract the process knowledge needed. The controller consists of an "intelligent" combination of a knowledge-based system and a set of control, identification, and supervision algorithms.

In the work by Årzen (1986; 1987; Åström *et al*, 1986) relay autotuning is used to build up process knowledge. In the work by Porter and Jones (Porter *et al*, 1987) a step response method is used.

The Exact controller from Foxboro (Bristol, 1977; Kraus and Myron, 1984) is marketed as an expert controller. The Exact is a performance adaptive PID controller based on pattern recognition. The controller monitors the control error searching for signs of load disturbances. The ratio of the peaks of the error signal at two consecutive peaks and the time period between the peaks are used to adjust the PID parameters. The system contains a reasonable amount of expertize about PID settings and can in that respect be considered as an expert system. The implementation does however bear no signs of expert systems. Performance adaptive or identification-free approaches to adaptive control have also been reported by others.

Learning Control

The learning control area has its roots in the beginning of the 1960s when the interest and optimism were high in artificial intelligence and cybernetics. Much research concerned system modelled after the human brain. Systems such as the Percpetron (Rosenblatt, 1961) and ADALINE (Widrow, 1962) belong here. These ideas were also applied to control problems. Other names for this area are intelligent control and self-organizing control.

In learning control systems, the controller should be able to estimate unknown information during its operation and determine an optimal control action from the estimated information. Different learning schemes have been proposed. Pattern classification techniques, sometimes with adaptable decision thresholds, are one class. Other examples are bayesian estimation and stochastic approximation. Much of the work in learning control systems could just as well fall into the adaptive control area. Examples of work in the area are Fu (1970, 1971) and Saridis (1977, 1983).

An inverted pendulum on a moving cart has been one of the benchmark control problems were learning ideas have been tried out. The Boxes system (Michie and Chambers, 1968) used a technique were the state space — the position and velocity of the cart and of the pendulum – was quantized into $5 * 5 * 3 * 3 = 225$ different compartments or boxes. During the learning phase, the system updates the decision frequencies in the boxes for the move left and the move right decision depending on the length of time the pendulum stayed upright and the number

of times the box was addressed during the run. Eventually, the system learned to balance the pendulum for up to 25 minutes at a stretch. The same problem have since then been studied by Anderson (1986) using a two-layer connectionist network and by Connell and Utgoff (1987) who in their system CART do not depend on a predefined quantization of the continuous state space.

An interesting aspect of learning control is the possibility to, by manual control, learn the system how to control a plant. This may provide a new functionality of the control system that is similar to the teach-in methods used in robotics.

4.3.3 Planning

Problem description

There are two aspects of planning in process control. The long term planning involves extensions and reconfigurations of the process and the process control system. There is also the short time planning of tasks that must be performed in the near future to assure proper process and system function and to optimize the process. Planning is based on a thorough investigation and evaluation of process performance.

Analytical tools based on numerical techniques such as dynamic programming and linear programming exist for planning tasks in many areas. These tools are often not used to the intended extent. The reason for this is that the mathematical modelling assumptions that the tools require do not match the actual planning situation. Heuristic knowledge and qualitative preferences cannot be included in the mathematical tools and thus make them less useful. Knowledge-based planning can be seen as an alternative approach and as a complement to analytical tools.

Production planning tools

Production planning is a task based on knowledge from several domains. A tool capable of supporting planning in process industry or telecommunication networks must be able to overview all data and knowledge needed to optimize production or services. These kind of tools could have a various degree of user interaction. Some times the user could propose production levels, routing tables or other crucial process parameters and the planning system then present the consequences of these actions. Another possibility is that the system itself both proposes how to control the process and presents the expected results to the user.

Knowledge based systems have an important role in these kind of tools since many planning decisions are based on process knowledge. To make a good production plan it may be necessary to incorporate knowledge from the following domains:

- Physical knowledge
- Knowledge of process configuration
- Knowledge of planned actions like stops, repair and maintenance work, work schedules, etc.
- Experience of how the process usually reacts in similar situations
- Statistical knowledge about uncertainties in process parameters and measurements

This enumeration of essential knowledge stresses that knowledge based planning systems must be integrated with knowledge bases containing all this useful knowledge. A production planning system must contain simulation facilities in order to foresee process behaviour. The users of this kind of tools is process operators and process engineers, working with daily operation of the process.

Resource planning tools

Resource allocation of, e.g., staff or equipment, is another planning task. Scheduling requires good overview of resources and knowledge about priority and cost of the tasks to be performed and equipment to use. Some of this knowledge could be stored in the process control system. Expert systems could be used to support scheduling of staff and equipment making use of the resources as efficiently as possible. Many knowledge based system are currently used to solve scheduling tasks. These systems need a representation of the objects concerned, e.g. persons in the staff, and knowledge of how to optimize the usage of these objects. They must also be able to reason with time.

The users of this kind of tool are the administration and planning groups and the users of various kind of equipment.

Process reconfiguration planning tools

This type of planning results in process redesign or reconfiguration. The planning requires a multitude of knowledge. Experiences of the process behavior during the past must be collected and treated. This knowledge is built on process statistics, events of overload and failures, etc. All the data must be evaluated and an overall picture of the situation in the process should be produced. This task could be supported by an expert system, containing knowledge of how statistics should be treated and knowledge of how the process is built and what services it could offer. In the telecommunication domain, knowledge-based systems could be used

to analyse network and traffic data and make conclusions about, e.g., performance and quality of service. This reasoning with statistics could be used for planning purposes, to produce reports on network performance and quality of service and to support the designers when making reconfigurations of the network. These reports could include descriptions of, e.g., overload and failure causes.

The users of these tools are working with areas of process operation or planning and design of extensions and modifications of the process. Other user groups are economical staff, quality analysers and process managers.

Approaches and examples

Planning sequences of actions and the time scheduling of these actions is a problem that is found in most industries. In the discrete manufacturing industry, the problem is frequent in the case of job shop or factory scheduling applications (Smith *et al*, 1986). The process industry have similar applications specially for batch processes but also for continuous processes during upstart, shutdown, and production changes.

The factory scheduling problem concerns the allocation over time of a finite set of resources to specific manufacturing operations such that the orders for parts received by the factory are produced in a timely and cost-effective fashion. In more detail the problem consists of the determination of an appropriate sequence of operations, or process routing, for each order and the assignment of required resources and time intervals to the operations. In order to do this the scheduler must be able to predict shop behavior through the generation of schedules that accurately reflect the full detail of the environment and the stated objectives of the organization. The scheduler must also be able to reactively revise and maintain the schedule in response to changing shop conditions.

The job shop scheduling problem is governed by a set of constraints. These constraints can be grouped into scheduling restrictions that must be met and sheduling preferences that provide a basis for selection among possible choices. The scheduling restrictions includes causal restrictions that constrain the ordering of the operations, physical constraints, i.e., the operating conditions of a certain machine, and resource unavailability. The sheduling preferences include organizational goals, i.e., meeting due dates, maximizing resource utilization etc., and operational preferences reflecting the heuristic knowledge present in a given scheduling situation.

The ISIS series of job shop scheduling systems (Fox *et al*, 1982; Fox, 1983) from CMU were based on an order-based decomposition strategy. The ISIS-2 system was demonstrated in a Westinghouse Turbine Components Plant in 1984. Based on the experiences of this system the OPIS system has been developed (Ow and Smith, 1986).

Power generation and distribution systems is an large area that early recognized

many potential expert system applications and where much work has been done on planning (Sakaguchi *et al*, 1987).

Load flow planning and load forecasting have both traditionally been solved by analytical methods. Expertise is however needed to select appropriate software tools, set initial data, analyze output etc. The system ADAPOS (Fujiwara *et al*, 1985) has succeeded in representing this kind of knowledge and is used on a daily basis for load flow planning. In load forecasting, the analytical models used are complex and result in a heavy computational burden. Knowledge based load forecasting has been suggested as an alternative to analytical methods by Rahman *et al* (1987).

The unit commitment problem, where generating units must be scheduled over some time period taking into account power system requirements, is a task where mathematical programming techniques traditionally have been used. It has however been observed that the output from these programs often are unacceptable to the operators. Expert systems for unit commitment are under development (Mokhtari *et al*, 1987).

4.4 Maintenance

Maintenance and process repair may affect the function and quality throughout the whole process. To handle this situation it is important that the process control system can distribute information about which services and functions that will be affected and how long this will last. Putting together all the necessary information in order to give an overall view of the situation is a task based on knowledge of similar situations and knowledge of process topology. A knowledge based tool to support this task could also be used to plan future maintenance to affect the process as little as possible. Knowledge-based systems may also be used by the maintenance and repair staff to get an overview of what work has been done before on the process and how this will affect the present work.

Many of the tools mentioned under the operation phase can be used for maintenance support as well.

4.4.1 Preventive Maintenance

Problem description

A problem with planned maintenance is that it disturbs the operation of several user groups. A knowledge based system with knowledge of the maintenance need and history of maintenance carried out, could use this knowledge to control and plan the maintenance work that must be done. The system must have an overall view of the process and access to information of how interferences in the process

will influence the process function. This information enables a knowledge based system to give suggestions of how to do the preventive maintenance in the most effective way. Another important aspect is that the system also could distribute the information of actual on-going maintenance work to users which are affected.

Approaches and examples

Network maintenance scheduling is an important issue in power systems. The operators have to make outage schedules that takes into account the service reliability and the effect on the overall network. This task requires expert knowledge and experience and consequently expert systems have been suggested (Uenishi *et al*, 1987).

4.4.2 Repair

Problem description

Repair personnel have the same need for diagnosis and trouble shooting tools as the operations personnel. A distinction is that the diagnosis tools more often are of the traditional off-line type. Knowledge-based systems may also be used for handling failure reports and feeding this information back to the process designer. Another possible use is for assistance in the selection of correct spare parts.

4.5 Conclusions

As shown in this chapter, many knowledge-based tasks and tools exist in process control. The enumeration that has been done it still not complete. An area that not has been mentioned is training. Knowledge-based training simulators that use both quantitative and qualitative models is an area where research is being performed. Still more examples exist.

In many areas, substantial progress has been made with many implemented systems. The majority of the systems are, however, only looking upon a single task. An attempt to an overall solution that encompasses different tasks is lacking.

The knowledge needed for the different tasks is of different nature and with different representation forms. Therefore, a general knowledge-based control system must allow for representation of several different types of knowledge.

The important issue for a control system manufacturer is not the tools themselves. What is important is the requirements that the different applications put on the knowledge representation formalisms and the programming paradigms of the underlying general control system. These requirements must be met by the control system.

Much of the current research in both diagnosis and planning is based on deep knowledge about the process. This requires that the knowledge-based control system has the proper means for representing component models and models of connections among components. The deep knowledge based strategies are usually implemented with object-oriented programming techniques rather than rule-based techniques.

It is however also very important to capture experiential knowledge of different kinds in the system. This knowledge is usually represented as rules. A combination of object-oriented and rule-based represented is therefore a minimal requirement for a knowledge-based control system.

5

International and national research programmes

Several, large international as well as national research programmes concerning knowledge-based control systems are going on at the moment. The most ambitious effort is the European Community's ESPRIT programme.

5.1 ESPRIT I

The ESPRIT I is a large research programme on information technology. The programme started in 1984-85 and will last until 1989. At least five of the current projects within ESPRIT I concern knowledge-based control in one way or another. The projects are very large, typically 70 man years and 10 MECU (70 MSEK) per project. The response we have had from Danish and British companies involved in ESPRIT projects is that the projects are very important for the companies and that the projects would have been impossible to carry out without the international cooperation and support.

5.1.1 GRADIENT

GRADIENT (Graphics and Knowledge-based Dialogue for Dynamic Systems or shortly Graphical Dialogue Environment) is an ESPRIT I project started in 1985. The consortium consists of CRI, ABB, University of Kassel, and University of Strathclyde with CRI as the prime contractor. The goal of the project is to develop an operator environment that will ease the work of the operator and

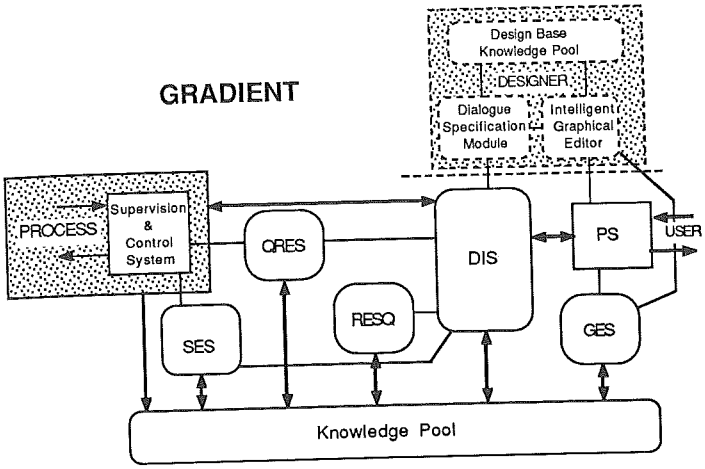

Figure 5.1 The GRADIENT system architecture

increase his cognitive capacity. The operator should be able to concentrate on problem solving rather than having to search among enormous amounts of raw data. The subgoals of GRADIENT are:

- Designing an intelligent alarm system, supporting the operator in alarm analysis
- Reducing the human factor in process control
- Developing concepts and tools for storing process knowledge and experience in the control system
- Giving the operator a more natural way to communicate with the control system.

The GRADIENT system architecture is shown in Figure 5.1. GRADIENT consists of five parts.

- An expert system for failure detection, alarm analysis, and diagnosis (QRES, Quick Response Expert System).
- An expert system for identifying and correcting inconsistencies in operator actions (RESQ, Operator Response Expert System).

- An expert system for the manipulation of the presented pictures according to the process state (GES, Graphical Expert System).

- A (partly knowledge-based) dialogue module, which supervise the dialogue between the user, the process and the rest of the expert systems (DIS, Dialogue System). One part of the dialogue system is an Intelligent Graphics Editor (IGE).

- A support expert system (SES).

Several different knowledge bases are accessible for the different expert system components. Examples are a system design KB and a failure state KB.

QRES is a rule based expert system for process diagnosis. It may contain generic rules reasoning with objects on the class level. QRES is controlled by a scheduler controlling the following modules: Input handler, Data handler, Listener, Analyser, and Output handler. All modules are implemented in one single process. QRES has some facilities to handle real time aspects in the reasoning. Time is used by the Listener when testing for faults in the application. The scheduler can handle non-monotonous reasoning in some situations.

The development of SES has just started. The intention is to develop a system that contains procedural support, predictions of failure consequences, and state-based diagnosis.

RESQ's task is to monitor the operator's activities and check the steps he is taking to solve a problem. RESQ is divided into three parts: a plan recognizer, an error handler, and a plan evaluator. The plan recognizer uses a plan library to match the operator's actions against. The plan library contains all possible plans that the operator could use in different situations. When an actual plan has been found, the operator actions are checked against this plan. The system will be able to give suggestion on short cuts and warn the operator when he diverges from the plan. RESQ is not yet implemented.

All parts of GRADIENT are implemented in KEE. The reason for this is the need for a common rapid prototyping environment. It has been observed that KEE is far from sufficient with respect to execution speed and real-time facilities.

The project will be demonstrated on two different processes. One packet switching data network and one power plant. A pre-prototype (MARGRET, Multifunctional All-purpose Gradient Experimental Test Environment) has been developed for the power plant demonstrator. The entire project will end in 1989.

5.1.2 KRITIC

The KRITIC project is an ESPRIT I project that has stretched from 1985 to 1987. KRITIC stands for "Knowledge Representation and Inference Techniques in Industrial Control". The consortium consists of British Telecom, Framentec, Krupp Atlas Elektronik, and Queen Mary College with Krupp as main contractor. The goal of KRITIC is to develop environments for constructing expert systems which would be capable of coping with high levels of complexity for applications in demanding industrial environments.

Two demonstrators, the first for the control and diagnosis of advanced Telecom switching systems, the second for the control of power distribution networks have been completed. A basic set of tools have also been developed. These tools include: a frame based knowledge representation language, AVALON; a rule-based expert system shell, MIKIC; two graphical description languages, G-MOD and V-GRAPH; a blackboard system, BBF; a planning system with dependency-directed backtracking, CELL-PLAN; and a high-level environment for explicitly specifying control flow, CELL-TISSUE.

MIKIC is an object-oriented, forward and backward chaining, rule based system. MIKIC integrates a rule interpreter with the Flavors object oriented language system so that rules are invoked by message passing in the same way as procedural methods. MIKIC rulesets may form the knowledge sources associated with the blackboard system BBF. A truth maintenance system has also been integrated with MIKIC.

A layered blackboard approach is used in the diagnostic system where each layer correspond to an step in the diagnostic problem solving strategy. The architecture of the system is shown in Figure 5.2. The diagnostic system is described in more detail in Williamson *et al* (1987).

5.1.3 QUIC

QUIC Toolkit is a short name for the ESPRIT I project "KBS Toolkit for Industrial Applications" which started in 1986 and will last four years. The consortium consists of the three domain companies, Aerospatialle, Ansaldo and F.L.Smidth, two software companies, CAP Sogeti and Framentec, and two research organizations, CISE (prime contractor) and Heriot-Watt University.

The goals of the project are:

- to define a general architecture for KBS tools in industrial applications,
- to classify the different knowledge-based tasks involved,
- to provide a set of tools for different tasks,

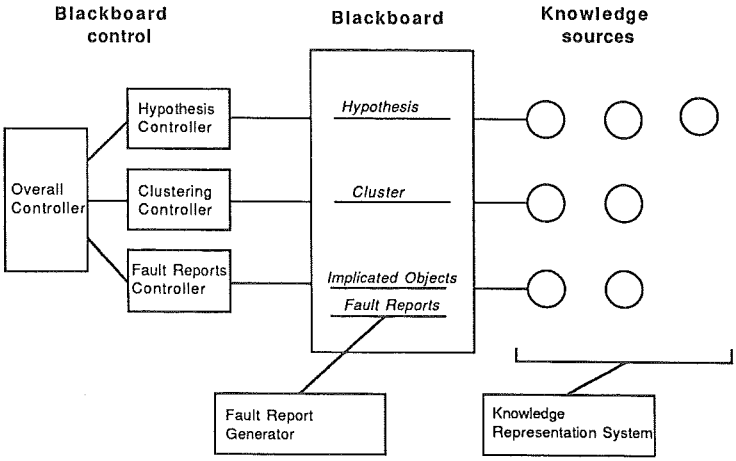

Figure 5.2 Overall architecture

- to integrate empiric and ontologic knowledge (concepts defined by Leitch at Heriot-Watt University),
- to validate the ideas on industrial demonstrators,

The project has classified the tasks involved into:

- Interpretation — the transformation of observed data into the adopted state representation
- Execution — the transformation of decisions into data suitable for action
- Prediction — the generation of future states from known or assumed current states
- Identification — the determination of (unknown) past states from known or assumed known current state
- Decision — the decision making process whereby the known or assumed complete state is used to construct conclusions.

Different systems such as diagnostic systems, control systems, simulation systems, and monitoring systems are then constructed by combining tools for the individual tasks. Tools for different tasks can also be either empirical, i.e., based on experiential knowledge, or ontological, i.e., based on theoretical knowledge.

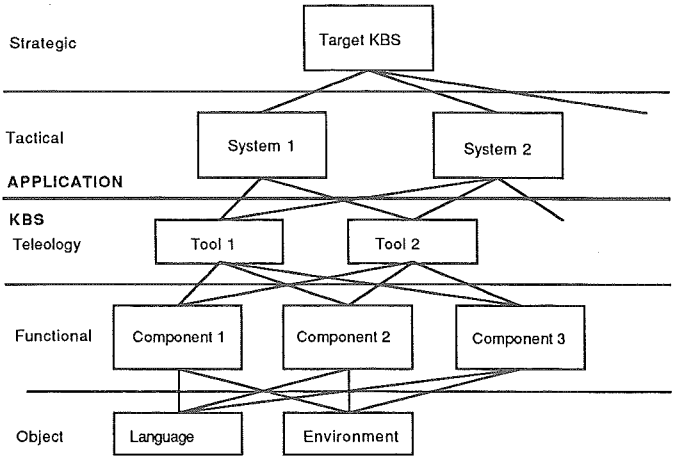

Figure 5.3 QUIC Toolkit architecture

The toolkit architecture used in QUIC is shown in Figure 5.3. The object level consists of the programming languages and environments used. These are Common Lisp and KEE. The functional level consists of the components of the different tools. Several components have been defined and partly implemented. One component is a component based language that includes a constraint propagator, a causal net generator, a truth maintenance system and a qualitative predictive engine. Another component is a fuzzy rule-based language with a focusing mechanism. A third component is an event graph language based on Petri net theory. The tools that have been defined are an ontological identification tool consisting mainly of the component based language, an empirical decision tool based on the rule-based component and a qualitative prediction tool.

The systems on the tactical level which have been defined are one ontological and one empirical diagnosis system, one fuzzy and one sequential control system and one ontological simulation system. The systems will be demonstrated on three industrial applications. These are a power plant, a satellite control application and a cement plant.

5.1.4 Other related ESPRIT I projects

Several other ESPRIT I projects have connections to knowledge-based control systems. The EUROHELP project is looking specially on the role of the operator in industrial systems and is aimed towards intelligent operator support systems. One of the members in this project is CRI.

The goal of the "Expert System Builder" project is to develop general, application independent expert system tools. An approach similar to KEE has been taken

with a strong emphasis on good object-oriented modelling facilities. The project has resulted in the STELLA environment with the expert system tools ODIN and TOR. This system has mainly been developed by Sören T. Lyngsö of Denmark. Within this project, the expert system tools have been used to implement a diagnostic expert system based on the Multilevel Flow Model concepts of Morten Lind. This have been applied to a simulation of a small part of the Risö power plant and will be applied in full-scale to the Svanemöllen power plant outside Copenhagen.

Within the telecommunication area two ESPRIT I project have focussed on specification languages. In the PANGLOSS project, LOTOS has been used for the design of a general-purpose internetting gateway. Within the SEDOS project, several LOTOS tools have been developed.

5.2 ESPRIT II

The second phase of the ESPRIT programme will start during 1988. At this stage (May 1988), it is not clear which proposals that have been accepted. In the guidelines for ESPRIT II, however, several research topics concern the use of expert systems for industrial applications. One topic is Real-time Knowledge-based Systems. A large consortium consisting among others of Thompson, GEC, and Marconi has supplied a 200 man years project proposal for this topic. According to our information, it is likely that it will be approved. Another topic is Cooperating Expert Systems with process control applications. Here, the consortium behind the KRITIC project has made a new proposal. A third area is Temporal Reasoning. This topic is likely to go to a consortium including Ferranti.

5.3 The RACE programme

The objectives of RACE (R&D in Advanced Communication in Europe) is to develop the broadband communication techniques needed to build an European IBCN (Integrated Broadband Communication Network). RACE is divided into 44 projects within broadband technology.The projects address a variety of subjects from picture coding in HDTV to methods and tools in broadband software development. The total budget of RACE is approximately 370 MECU which means more than 4000 man-years. There are some project in RACE that address tasks in real time control. These projects will develop a network management system to handle IBC networks. AIP (Advanced Information Processing) techniques, e.g. KBS techniques, are evaluated in these projects. This is a short description of the most relevant RACE projects in this context:

ADVANCE: Evaluation of AIP techniques in network and customer administration system for IBCN. The project is putting the effort on non real time

applications. TeleLOGIC is a member in the ADVANCE project.

AIM: Standards for maintenance in IBCN.

NEMESYS: The role of AIP techniques for traffic and quality in service management for the IBCN.

5.4 The EUREKA programme

Another large European research programme is EUREKA. Main participants are the EC and EFTA. It was initiated in 1985 and has a general scientific profile. Standards are important and market orientation is stressed. The program has currently about 70 projects under way. Funding comes from the individual participants and amounts today to a total of about 3000 MECU. Two projects with a bearing on the one under consideration is EAST (European Advanced Software Technology) and ESF (Eureka Software Factory). TeleLOGIC is a participant of ESF which is aimed at industrializing the software engineering process.

5.5 ALVEY

The British ALVEY programme also contains related projects. The most relevant is the RESCU project which started in 1984 and was completed in 1986.

5.5.1 RESCU

The RESCU project was the first Alvey Community Club and consisted of 23 industrial companies and three universities. The prime contractor was Systems Designers. The objectives of the club was to promote awareness and to validate the new KBS technology.

The application in RESCU was quality control in a chemical batch process. The actual plant chosen was a batch reactor from ICI. The process could produce several different chemicals and the outermost quality control loop was distinguished by unmeasurable variables, large variations in feedstock, a wide range of specifications on the different end products, and unreliable instrumentation.

The objectives of the system was to give advice to maintain product quality closer to the desired specification than achieved by the operators themselves. This was done by recommending recipes and by predicting product quality. Other goals was to reduce the numbers of adjustments and to minimize the use of expensive feedstock. The process was well instrumented and equipped with a Fox Control system.

The knowledge-based system was implemented in a Poplog environment on a 4 Mb, VAX 11/730. The requirements of the system was the ability to reason with time and over time, the ability to reason with uncertainty, and the possibility to reason with different sources of knowledge. Two different types of knowledge sources were used. Procedural or active knowledge sources contained procedural knowledge of how things should be done. These knowledge sources included knowledge about process tracking, recipe recommendations, batch quality assessments, reconciliation of estimates, and updating of system parameters. Declarative or passive knowledge sources contained knowledge about plant models, product specifications, relations between variables etc. Both active and passive knowledge sources were implemented in terms of rules. The whole system contained 730 rules. The end performance achieved was better than standard ICI procedures but slightly worse than expert operators.

5.5.2 COGSYS

The work in the RESCU project has been continued in the COGSYS project. COGSYS is also an industrial club with Systems Designers as the main contractor. The goal of COGSYS is to develop a real-time expert system shell for process control. Object-oriented features will be added to the rule-based RESCU system. ABB is one of the members of the industrial club.

5.6 Other programmes

There exist also other research programmes related to knowledge-based control systems. One such example is the KUSIN/NORDFORSK programme were companies and universities in the Nordic countries are allowed to join. A large project within this programme is the ROTA project aimed at diagnostic systems for rotating machinery. Swedish collaborators in this project are ABB STAL, ASE Europe, and FFV Aerotech.

5.6.1 DUP

DUP (Development of user-friendly operation systems for the process industry) is a Swedish research programme funded by the Swedish National Board for technical Development (STU). This programme is focussed on the end-users of modern control systems. DUP is an interdisciplinary programme composed of Computer Science, Automatic Control, Process Technology, Cognitive Psychology, and Environmental Science. Important areas within DUP are knowledge-based system for design support and for operator support. Three key industrial areas have been selected for DUP. These are the chemical industry, the paper and pulp industry, and the food manufacturing industry.

The main difference between this project and DUP is that this project is driven by the control system suppliers and not primarily by the user industries.

5.6.2 MDA

MDA (Människa – Datateknik – Arbetsliv) is another Swedish national programme that looks upon the user situation in complex information systems. MDA has a very strong direction towards cognitive psychology and environmental studies.

5.6.3 CACE

The Computer Aided Control Engineering (CACE) project is a Swedish national project which is funded by the Swedish National Board for Technical Development (STU) and which is being carried out at the Department of Automatic Control in Lund. The aim of the project is to investigate how new hardware and software technology will influence the next generation of CACE software. The project concentrates on the development of tools for modelling and simulation. The project is based on a hierarchical, object-oriented model of the process that allows multiple representations and reuse of models. The models are expressed in the form of symbolic equations which will be used to generate efficient simulation code, linearize non-linear systems, generate control code etc.

6

A Suggested Concept

This chapter proposes a basic system concept for knowledge-based control system. The kernel of the concept is a model of the process and the control system. Different knowledge tools operate upon this model. A set of demands on the concept is given. Before the actual concept is described, knowledge representation in general and in process control is discussed.

6.1 Overall demands on a knowledge-based control system

Tools and structures for knowledge representation, as described in the previous chapters, are the new and important issue in a concept for knowledge-based control systems. However, this extended functionality is less useful if not implemented in a way that secures that the overall demands on a control system are met. For the entire and complete concept it is of fundamental importance that the following features are carefully considered.

- Integration of conventional and new technologies in one uniform system.

  Future control system applications will require as much procedural functionality as today's systems. Efficient representation and execution of control algorithms, sequence control and logic control still will be the main demand. The new is that this main part of the system in a uniform way can be combined with knowledge-oriented functionality. In the new concept the procedural part has priority and must be handled as effective as in today's systems.

- Real-time support

  The ability to secure real time aspects is crucial for control systems. Introduction of new techniques and conceptions as reasoning, object-oriented programming, uncertainty, processing of qualitative data etc will also introduce new very complex real-time problems. It is a necessity to find practical solutions to these problems.

- Incrementality

  A production or process plant is a live system which is continuously changed and updated. In a knowledge-based control system with its many types of representation it will be still more difficult to decide how to change and to survey consequences of a change. The new concept must support the user with a step-wise, evolutionary way to extend and improve the control of the process.

- Ortogonality

  In traditional control systems an introduced fact normally is a numeric or alfanumeric piece of information used in a straightforward algorithmic way. In a knowledge-based system the same introduced or created fact or conception can be used both in many different types of representation (procedures, rules, objects) and for many purposes. To handle maintenance and up-dating of the system it is a necessity to have a system where each specific piece of information or fragment of knowledge is stored in only one place independent of where and how it is used.

- Transparency

  All control and supervisory functions in a control system are based on underlying knowledge. Expertize and underlying reasoning must normally be documented separately and in the worst case it can not be documented at all. New technologies will in a better way support explanation facilities where underlying knowledge can be attached to control functions and made direct available in the control system. As such improved transparency is of extreme importance it has to be given a high priority when developing a new system concept.

6.2 Knowledge representation in general

In the fields of computer science and artificial intelligence a variety of knowledge representation formalisms are employed. Some examples are object oriented representations, logical representations, procedures and rules. For a further discussion the reader is referred to Chapter 3. In general, knowledge representation formalisms exhibit the same power of expression in the sense that anything expressed in any one formalism can be equally well expressed in any other formalism. However the knowledge representations formalisms vary greatly in the amount of work needed to design, maintain or process (execute) a model.

An example is that object oriented formalisms in general are vastly superior to procedural models in the design phase, but, as a rule, they are more tedious to process. The modelling and maintenance advantages together with increased hardware capacity have however led to a greatly increased popularity for object models.

It is important to note that, in general, any knowledge representation formalism can be employed to model a mass of knowledge. Decisions on what to use are design decisions depending on cost-capability trade-offs.

The same argument holds for the computer realization of the formalisms described above. Consider as an example, an object oriented representation of some knowledge. It can be realized in a variety of ways. Some examples are as a relational database, as a hierarchical database, as a network database, as lisp lists, as a set of prolog predicates etc. The choice of realization is, as the choice of knowledge representation formalism, a design decision depending on cost-capability trade-offs.

6.3 Knowledge representation in process control

Three dimensions can in a simple model be regarded as spanning the knowledge in a process control system. These dimensions are type, depth and structural organization of knowledge.

6.3.1 Type of knowledge

Type of knowledge encompasses knowledge type either from an application point of view, then encompassing the functional knowledge, physical knowledge, historical knowledge, economical knowledge and operational knowledge and so on defined in Chapter 2, or from an representation point of view, then encompassing the various representational formalisms like logic, procedures or object-oriented formalisms.

Physical knowledge could be modelled in an object knowledge base. The

attributes of the objects describe properties of the components and relations between components. Both general knowledge about the control system and about process components, and special knowledge about a specific plant and control system installation can be modelled in this way. This is done by using both class objects and instance objects. In this way a specific installation can be seen as an instantiation of general class objects. The activity of designing the process control system can in a simplified model be regarded as designing the object structure representing the system.

Functional knowledge can be coupled to the physical knowledge through the object structure. An interesting approach in this context is Multilevel Flow Models.

Economical knowledge is weakly coupled to the the physical knowledge base. This knowledge can for example be modelled as an order database together with a knowledge base containing economic rules. Optimization algorithms can be stored in a procedural knowledge base together with governing rules. This knowledge could be used by knowledge-based production planning tools.

Operational knowledge is tightly coupled to the physical knowledge base. Knowledge of this type can be represented as rules. These rules in general refer to objects or sets of objects in the object structure describing the system or the installation. If possible, rules should be stored in objects. This will increase the structure and processing efficiency of the knowledge base.

Historical knowledge containing stored time histories of important process variables can be stored in a conventional data base with references to and from the object data base.

Pictures and text information from the knowledge bases described above can be stored in picture and text data bases with references between the object base and relevant parts of these data bases. Internal linkage in these data bases could be accomplished through, e. g., a hyper-media approach. An example is that a drawing or picture of some object e. g. a fan or operator console, is stored in a pictorial data base, but is accessed through the object and regarded as a property or view of that object.

6.3.2 Depth of knowledge

The knowledge depth dimension describes the expressional power of some amount of knowledge. A model of, e.g., an object could be designed in a variety of ways. For any object with some degree of complexity a complete model will be very complex and, hence, very hard to manage and process. This problem could to an extent be diminished through several models of the same object but with varying complexity. A model should then have the properties of being simple enough to solve the problem under hand in a reasonable amount of time and have completeness enough to give a sufficiently correct answer.

6.3.3 Structural organization of knowledge

The structural organization of knowledge describes the structuring primitives for some amount of knowledge. For the special case of process control systems, important structural concepts are the service views and the organizational views.

Service views or functional views

The service views or the functional views of the knowledge are the views related to the activities, functions or services that the process control system is expected to supply to the organizations employing it. There are a variety of functional views, depending on the design of and the services demanded of the process control system. As a rule, however, the functional views reflect the knowledge base structure as described earlier, i. e., they consist of physical views, functional views , operational views, economical views etc.

Organizational views or users views

The organizational views or user views are related to the different organizations or user groups that interact with the process control system.

There are four user views groups:

- The design views. Show the system as seen from the system design organization.
- The process view. Shows the system as seen from the process.
- The operations views. Show the system as seen from the process operations organization.
- The maintenance views. Show the system as seen from the maintenance organization.

Different users of the process control system needs different information and different ways of interacting with the system. A process operator and and instrumentation technician has totally different information needs. The process operator may want to see a dynamically updated process diagram together with trend curves of important signals and an alarm logbook. He also needs presentations of condensed information about the overall process status, advice on which control actions to take, explanations to process upsets etc. The instrumentation technician on the other hand needs displays of wiring diagrams, electrical connection drawings and geographical layouts. He needs to report to the system that a certain component is out of order or has been exchanged. He also need

trouble-shooting assistance and advice about how the repair is performed. The latter may include pictures of the components which, e.g., show how a component should be unscrewed etc.

The different views can be seen as transformations of the basic functional views into interfaces suitable to different user groups.

The design view: This view is what the system design organization sees of the system. This view is very important in that all the other views are designed, at least in part, in this view.

A very simple and fundamental model describing the overall activity of building a plant could regard it as a three-phase activity.

1. The pre-design phase: Design of the system components.

2. The design phase: Composition the components into a system.

3. The realization phase: Instantiation and/or transformation of the system into an installation or plant.

The first phase, i.e. the one preceding the actual system design phase is normally also considered as system design. It is however performed by other persons. This phase includes design of the individual plant components. This is often by independent equipment supplier companies. The phase also includes the design of the general system components in the control system. This task is done once and for all when the general control system is developed by the control system manufacturer and can only with difficulty be changed later.

The second phase, the phase of system design can in this simple model be regarded as selecting objects from an object knowledge base (a product structure, which is delivered from the pre-design phase). These objects are the system components, or building blocks, for the designer. Examples of such objects are telephone exchanges, computers, servers and trunk lines. Examples from process control are pumps, valves, heat exchangers and pipes. Examples from the instrumentation and control system are sensors, actuators, and controllers. These objects are then composed into a system. This is performed through adapting the components for the system task under hand, e.g. by programming and other types of modifications. These components are then connected into an object structure that describes the process control system.

The phase succeeding the system design is the realization of the system, i. e., installation or building of the plant. This entails instantiating or modifying the object structure describing the process control system into a plant or installation.

The process view: This view is the model of the knowledge base as seen in the interface between the process and the system. It is especially complex to design since real time demands on this model or view in general are very high. The state-of-the-art approach is to design this view manually or to use very powerful processing capabilities to directly execute the knowledge in the "central" knowledge base.

The operations view: This view is what personnel engaged in plant operations sees of the system. In general real time demands are placed upon this view.

The maintenance view: This view is what the maintenance group sees of the plant. In general, real time demands are not so harsh as they are on the operations view.

All these four major views can then be further subdivided into subviews that are specially suited for certain user groups.

6.4 A Basic System Concept

The basic concept consists of a model on which a set of "intelligent tools" operate. The tools provide the different user views of the process and performs the knowledge-based functions of the system.

6.4.1 Hierarchy levels

The system concept is based on an object model. The basic entities which are represented as objects in this model are the physical components in the plant, i.e., pumps, valves, heat-exchangers, robots, manufacturing machines, queues, multiplexers etc. and the control functions, i.e., sensors, controllers, PLC code, etc. Attributes in the objects are used to represent the properties of the components. The connections or links between the basic objects represent physical connections such as electrical connections or flow connections.

Hierarchy levels are needed in order to represent the process at different degrees of resolution. The highest level in the hierarchy represent the whole plant. The lowest level represent the basic entities. With hierarchy levels, individual objects may be grouped together into one object at a higher level. For example, a physical component together with sensors, actuators, and controllers can be modelled as one object at a higher level with the individual objects as submodels. PLC sequences that involve the combined control of several components are represented on a hierarchical level where all the components are included. One may also think of a situation where connections among objects are hierarchical. For example, on a high level in the hierarchy, several interconnected objects can be seen as one high level connections with connections and objects as submodels.

The possibility to decompose objects into class objects and instance objects facilitates maintenance. In complex plants, a component may appear several times. For example, a chemical plant may have several PID controllers or a telecommunication network consists of several nodes. Maintenance is simplified if all appearances share a common description. Class objects are used to represent general knowledge about the process and the control system. The hierarchical structure must exist also among the class objects.

The basic object model is necessary in order to have a structured way of representing and storing various types of knowledge. With hierarchy levels, knowledge and information can be stored at the most appropriate level.

6.4.2 Multiple perspectives

Up to this point the described model only contains physical knowledge, i.e., knowledge of the physical properties of the components of the plant and of their connections. The hierarchical abstraction mechanism has only been used to group together geographically and logically related components. This leads to a tree structure where the root corresponds to the whole plant and the leaves correspond to the basic components.

This simple structure is not enough. As shown earlier, several views of the process are needed. These views are either user views or functional views. As described earlier, different users or organizations need different information. Functional views reflect the knowledge base structure as described earlier, i.e., physical knowledge, functional knowledge, operational knowledge, economical knowledge etc. These different kinds of knowledge must also be represented.

A possibility to represent the different functional views is to use objects with multiple perspectives or views. An example of a part of the model is shown in Figure 6.1. Some examples of perspectives needed are the physical perspective, the functional perspective, and the behavioral perspective.

The physical perspective contains the physical parameters of the component. It also contains the included submodels and their connections.

The behavioral perspective describes the internal behavior of the component. For a physical component this may be a differential equation description of how inputs and outputs relate to each other. It may also be rules that describe causal relationships between inputs and outputs. For a PID-controller or some other control block, the behavorial perspective simple describes the operation of the control block, e.g., in the terms of a formula or as control rules. For a PLC sequence, the behavorial perspective consists of the actual boolean equations. The behavorial perspective of an object composed of submodels can be generated from the behavorial perspectives of the submodels.

The functional perspective describes the function of the different components.

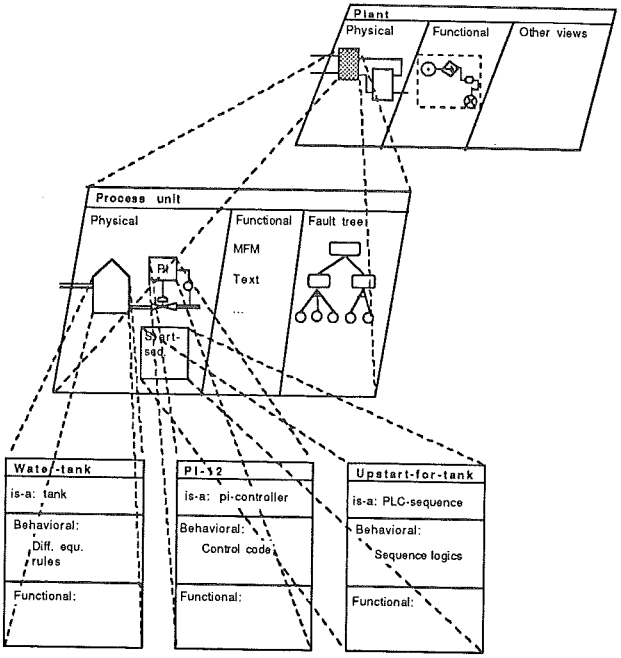

Figure 6.1 Hierarchical, multi-perspective object model

Some examples are: "What is the primary function of a process part?", "Does it have any secondary function?", "What is the control objective of certain controller?". On higher levels in hierarchy the functional view could for example be described in terms of Multilevel Flow Model (MFM) models.

The combination of the physical, structural decomposition and a functional decomposition usually destroys the tree structure of the hierarchy and turns it into a directed, non-cyclic graph or even into a network. The reason for this is that a single component usually has more than one purpose seen from a functional perspective. This is typically the case in process industry where several energy and material flows take place at the same time and involving the same components. An example could be an heat-exchanger that is involved in both a material flow loop and an energy flow loop. This leads to situations where the same basic component is contained in more than one functional higher order level.

The model may also contain other perspectives. Some examples are an economical perspective and an operational perspective. These perspective are typically only used at higher hierarchical levels.

It is also desirable to be able to describe perspectives at various depths of knowledge. This is typically the case for the behavioral perspective in the model of physical components. The most detailed description could probably include a

complete quantitative model based on differential equations. It is however also very natural to have models where only the dominating dynamics are included. In an even less detailed description static relations or qualitative, causal relations could be used. Another possibility to describe behavior would be in terms of rules. If several depth levels are used, it may be desirable to not necessarily generate the behavior of a models from the submodels. Instead another less detailed description of the behavior could be used on the higher level. This could be used for reasoning at different levels of abstraction.

The basic model described is based on object orientation. This does, however, not exclude other knowledge representation techniques such as rules or procedures. These are instead used in the different perspectives at the different hierarchical levels. Rules could be used to describe behavior both in loose terms for a physical component and in the form of fuzzy rules for fuzzy controller. Heuristic knowledge built up during operation by the operators could be stored in an operational perspective on the hierarchical level most appropriate. Procedures are available in terms of PLC sequences and control algorithms.

6.4.3 The "central" model

The proposed basic architecture is influenced by the belief that *"knowledge representation and use cannot be separated"*. This leads to different perspectives where the involved knowledge is represented in ways that simplifies the intended use of it. This is contradictory to the desire to have a single knowledge representation where each piece of knowledge is stored in one place. Ideally, it would be desirable to automatically transform knowledge between different perspectives. If this is possible is an open question. In the proposed architecture, knowledge involving the same object is at least stored close to each other.

Although a central model is advocated, the model is not physically implemented in one place. On the contrary, the need to distribute the computational burden to to many processors is even higher in knowledge-based control systems. The basic philosophy is, however, that the model should be seen as one entity during design, maintenance and operation.

6.5 Knowledge tools

The object model is the base of the concept. A set of tools operate on this model as shown in Figure 6.2. One purpose of the tools is to provide the different user views of the process. The operator interface should perhaps consist of a dynamically updated process diagram together with curves of different signals. He also needs presentations of condensed information about the overall process status, advice on which control actions to take, explanations to process upsets etc. This is provided by a tool that operate on the basic model and build up

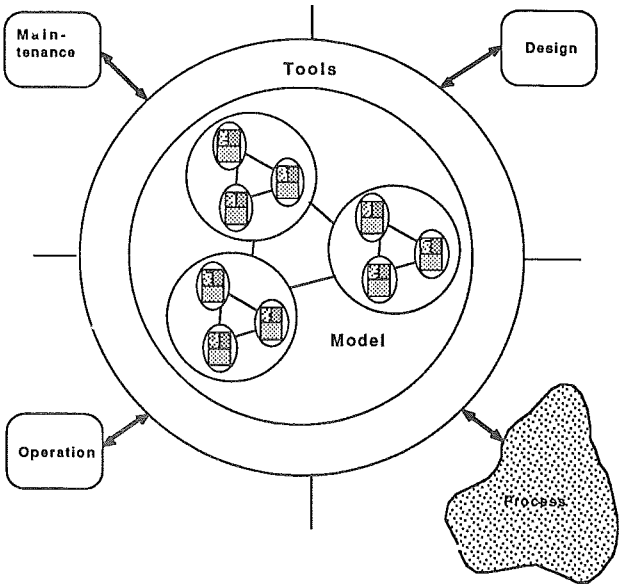

Figure 6.2 System concept

the user view from this. Another example is the control system maintenance group that instead needs information about such things as in which physical the different control blocks are implemented, how they are interconnected, what the internal signal addresses are etc. They also need access to diagnostic tools for trouble-shooting and repair assistance. This is provided by a special tool that extracts the desired information from the central model.

One user view of this hierarchical model would naturally be in the form of interconnected icons representing the different objects. Good possibilities to move around in the object space is very important. Graphical techniques for mouse interaction and windowing are natural to use. If MFM techniques are used to model functional perspectives, more than one graphical language are needed.

A second purpose of the tools is to implemented the different tasks that are needed. One such tool is the control block and PLC-code interpreter. This tool resembles what is available in conventional system today. For efficiency reasons, this tool probably uses an internal, compiled representation of the control parts of the model during execution.

There are many examples of other tools. One may think of a design support tool that helps the designer in building the the basic model. This may include more or less automatic generation of parts of the model.

Another tool is an monitoring and diagnosis tool. This could for instance make

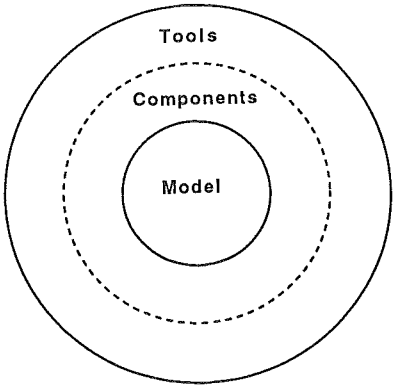

Figure 6.3 Basic concept with model, components, and tools.

use of the functional perspectives of the model in combination with the empirical rules stored in the operational perspectives. Still other tools are planning tools, simulation tools etc.

The tools mainly operate on the knowledge in the basic model. They do however themselves contain specific knowledge about their specific task. For instance, the diagnosis tools needs general knowledge of diagnosis and the operator view tool may need knowledge about the human cognitive behavior.

The internal implementation of the tools may be different. For some tools, conventional techniques are most appropriate for other tools knowledge-based techniques are best suited.

The tools are not independent of each other. The tools that provide the user views uses the tools that implemented the different knowledge tasks. Several tools may contain similar subpart. The subparts can be seen as components out of which tools are constructed. This give the structure of Figure 6.3. Examples of components are inference engines, truth maintenance systems, simulators etc.

The blackboard analogy

The proposed concept can also be seen as a blackboard system. The central, hierarchical model with its different perspectives constitutes the blackboard. The knowledge tools can either be seen as individual knowledge sources or as composed internally out of a set of knowledge sources.

6.5.1 Real-time aspects

The real-time aspects of the system are very important. The use of a separate tool for the execution of the conventional control code ensures that the conventional control performance not is slowed down.

The special real-time aspects of knowledge-based systems are most pronounced for the real-time knowledge-based tools such as, e.g., an intelligent monitoring and diagnosis tool.

The parameters and variables in the central model can be decomposed into those that remain constant and those whose value changes with time. In order to reason over time, histories must be stored of changing process variables.

In order to maintain an up-to-date and consistent picture of the dynamic environment, several techniques are possible. Since the knowledge-based part is integrated with a conventional control system, there is no principal objective against keeping the information up-to-date through fast sampling. In a purely, knowledge-based system this would cause communication bottlenecks due to limited communication rates at the interface to the control system.

If fast sampling is not possible, the use of validity intervals and time stamps to indicate if a sensor value is still valid is one possibility. With this technique, facts inferred by the system are marked with a currency interval that tells how long they are valid. This techniques is computationally relatively inexpensive. The technique can however only represent that a fact becomes unvalid due to elapsed time. It can not represent that a fact becomes unvalid due to an occurred event.

To do this, a Truth Maintenance System must be used, which records the dependencies between inferred facts and sensor values. As long as the sensor data is kept up-to-date, the TMS will ensure that the knowledge base is consistent. This method is however very resource intensive.

An important feature of the knowledge-based tools is that they can handle asynchronous events in a timely manner. This involves interrupting the current reasoning activity and concentrate on more important matters. Focusing of attention is related to this. If the real-time knowledge-based tools are implemented as a set of interruptable knowledge sources controlled by a real-time scheduler this can be achieved.

6.6 Conclusions

A system concept has been proposed. This is by far not complete. Most of the details remain to be specified. The concept is impossible to evaluate before it is tried on a real application. This is the goal for the main project within the IT4 frame.

Hardware and software requirements have not been given. This is also task for the main project.

7

Conclusions

Today's control systems are very good at handling quantitative knowledge. This is expressed as procedures, control logics, and sequential logics. Control systems are however very poor at representing the underlying, functional knowledge which always is present. This knowledge is mainly qualitative.

Of the knowledge preceeding involved in the analysis and design phase preceding the final programming of the control system, it is only the final algorithmic representation and possibly some verbal explanations of it that can be put in the computer. This results in control systems that are "black boxes" which often cannot be understood by anyone else than the original programmer. All expertise and underlying reasoning must be documented separately or communicated verbally to the end users. Much knowledge is not transferred at all.

The situation is cumbersome since it complicates improvements and extensions to the process. The process operation and maintenance are also highly dependent on a correct understanding of the function of the controlled process to, e.g., correctly find the causes of faults. The problems are amplified in the current trend towards highly automated processes. This leads to increased requirements on the process operators. In the same time, the operators are not given corresponding assisting tools with increased functionality.

Many of the problems are caused by the fact that the underlying design knowledge is not included in the operational control system. To do this requires that qualitative, heuristic knowledge can be represented and utilized.

Knowledge-based systems is an area within AI that focuses on the representation and utilization of qualitative, expert knowledge. Many knowledge-based applications have been proposed and implemented in process control. So far, all

knowledge-based applications have been implemented on top of and separated from the conventional control system. In the long run this is not the correct approach. Problems arise during the integration of the knowledge-based system with conventional computer systems. Multiple operator consoles are needed with different man-machine interfaces. Redundant information is stored in both systems. An integration is therefore needed between knowledge-based techniques and conventional control systems.

Furthermore, a future knowledge-based control system must contain the necessary knowledge representation formalisms and programming paradigms that are needed by the knowledge-based application. These must be identified, introduced, and combined with conventional methods. Most of the projects that have been performed only look upon one specific application. A general knowledge-based control system must contain means of expression that are suited for a multitude of applications.

The feasibility study has identified knowledge-based tasks and tools within process control and different approaches for solving them. The approaches have been studied with respect to the requirements they put on the underlying knowledge representation. A common feature among many applications is the need for a model of the process components, including the control system, and their interactions. An object-oriented model is the usual way to achieve this. A second common feature is the need to represent heuristic, experiential knowledge. This is usually done in terms of rules. Therefore, a combination of object-orientation and rules is a minimal, identifiable requirement on knowledge-based control systems.

Process control contains many real-time expert system applications. As such they contain several difficult problems. In many projects this is not fully understood. Instead, conventional expert systems intended for consultative, off-line operation are used in real-time. This is not the correct way to solve the problem. The real-time demands have a strong influence on knowledge representation and knowledge-based system architecture that must be taken care of from the start.

There is currently a very high interest in knowledge-based applications within process control. Commercial products are beginning to emerge. To succeed the technique must however be integrated with conventional control systems. The integration can only be performed through initiatives from the control system manufacturers. The feasibility study has shown that the technical prerequisites for an integrated knowledge-based control system exist.

A basic system concept has been outlined. The kernel of this concept is an object-oriented, multi-perspective model of the process components and the control system. The model allows several hierarchical levels. The multiple perspectives are used to represent the different types of knowledge about an object that are needed. Although the model is basically object-oriented, it allows the inclusion of other types of knowledge such as rules and procedures.

The model is surrounded by a set of tools. The tools have two functions. They build up the users interfaces that different user groups need. The interfaces can be seen as transformations of the underlying model. They also implement the different functions of the system. The functions are either knowledge-based or use conventional techniques.

A

Travel Notes

A.1 Visit to Denmark, 15/2 – 16/2

During February 15 – 16/2, a visit was made to Denmark. The places visited were the Servolaboratium at DTH, Soren T. Lyngsö, and F.L. Schmidt. The group consisted of Claes Rytoft and Kent Bladh from ASEA, Börje Rosenberg from SattControl, Mats Petterson from Telelogic and Karl Johan Åström, Karl-Erik Årzén and Per Persson from the Department of Automatic Control.

A.1.1 The Servolaboratory, DTH — 15/2

Professor Morten Lind gave a seminar about MFM (Multilevel Flow Modelling), which is based on his earlier work at Risö National Laboratory, where he worked under professor Jens Rasmussen. MFM is described in Chapter 4. Morten Lind did also present other research projects at the department.

SIP is a joint project between the Servolaboratory, University of Copenhagen, and School for Architects in the Academy of Arts. The first phase of the project is described by Lind (1987). The report contains a description of how a process operator's working-tasks have changed and what demands these changes put on the system description and the system presentation. The paper also contains suggestions of a new abstraction model and of a new graphical symbol language. The language is based on MFM and the so called "time glass" metaphor.

A.1.2 Sören T. Lyngsö - 15/2

Sören T. Lyngsö (STL) is a supplier of equipment and systems in the areas of telecommunication, industrial automation and ship automation. We met Gunni Fredrikssen (General Manager of the AI Division), Finn Jensen (Technical Coordinator of AI activities in STL) and Kjell Larsen.

As a spin-off from participation in ESPRIT-projects (ESPRIT- project 96, The Expert System Builder, and an ESPRIT-project on how to use expert systems on ships) STL has developed two AI-products for process control; ODIN which is a tool for structured knowledge programming in process control applications and TOR which is a run-time environment for the programs developed by ODIN.

The systems are based on UNIX, Common Lisp with FLAVORS and X- Windows. ODIN currently runs on AI-workstations. Macintosch II with TI's Lisp processor is contemplated as a future run-time system for TOR.

ODIN is a general tool for development of large expert systems in the same spirit as KEE. It especially supports the knowledge engineering and modelling. It has well-developed facilities for object-oriented modelling. To facilitate modular and well structured systems it has different layers adapted to the needs of different experts involved in building the system. ODIN has been used for an in-house diagnosis application based on the Multilevel Flow Model techniques developed by Morten Lind described in Chapter 4. The application used a simulation of a part of the Risö power plant. A demonstration was given of this.

The current price for ODIN was DKK 200.000:- and for TOR DKK 5.000:-.

A.1.3 F.L. Schmidt – 16/2

F.L Schmidt is one of the world's largest suppliers of cement plants and machinery. The group visited Mogen Levin at the Research Lab. The reason for visisting F.L Schmidt is their work in fuzzy control. Control of cement kilns is the major application where fuzzy control has been successful. The work begun in the beginning in the 1970s with the first installations between 1972 and 1975. Since the beginning of the 1980s, F.L. Schmidt has delivered fuzzy controller for cement kilns on a large scale.

The fuzzy logic controller of F.L.Schmidt is delivered as an option to FDL-SDR (Supervision, Dialogue, and Reporting System). FDL-SDR is a computer-based control module for applications in the cement and related industries. An ordinary kiln controller consists of about 50 fuzzy rules.

F.L.Schmidt is a partner of the ESPRIT I, QUIC project. An overview of the QUIC project was given. This project is described in Chapter 5. The main motivation for F.L.Schmidt to participate in the project is that it gives them resources to develop the next generation of the fuzzy controller that would have been difficult to get from other sources. A demonstration was given of the fuzzy

controller in the QUIC toolkit. A demonstration was also given of a training simulator for a cement process.

A.2 Visit to USA, 22/2 – 4/3

During February 22 to March 4, Karl-Erik Årzén, Sven Erik Mattson, and Bernt Nilsson from the Department of Automatic Control visited USA. The main purpose of the visit was to attend a one-week course on G2 at Gensym Corp. in Cambridge, MA. The second week was spent visiting several companies and universities with related projects.

A.2.1 G2 course at Gensym Corp., 22/2 – 26/2

The G2 course was held by Mark Allen, responsible for training courses and documentation on G2. During the course, several of the employees at Gensym participated and gave lectures. The course was attended by eight persons, Stephen Woo and Ron Pho from Exxon in Canada, Mohan Bhalodia from Exxon in U.S.A, Chuck Noren from Martin Marietta, Per-Olav Opdahl from Computas Expert Systems in Norway, and the three of us from Automatic Control. The course gave a good feeling for the possibilities of G2 with many hands-on exercises on Lisp machines. A general description of G2 is found in Chapter 3.

The first day consisted of a general introduction to G2 and a detailed description of the object-oriented features of G2. The second day, the G2 simulator and the different ways of referring to objects and attributes were described. The rule-based part of G2 was covered during the third day and day four was devoted to developing a G2 application. Finally during day 5, Michael Levin, the senior programmer behind G2 gave information about the forthcoming developments of G2. The next year will be devoted to creating a operator interface, supply possibilities for transient objects and relations, add Grafcet inspired procedures, improve the simulator, and add hierarchical objects. Whether Gensym will manage to carry out all these improvements or not is difficult to judge. It is however clear that even if they only succeed partly, G2 will be a very powerful system.

One experience of the course is the powerful capacities of the object-oriented graphical system. In most examples and demonstrations, the graphical objects are only used to represented physical components in the process and the connections are used to represented physical interconnections among components. The concept is however far more powerful than so. The graphical objects may represent general abstract as well as concrete concepts and the connections may represent general relations among objects. In this way it is possible to combine process schematics with graphical, functional descriptions of the process. Examples were given where icon objects were used to represent activities in a process planning setting.

An interesting experience during the course was the change in attitude from the industrial participants. During the first day, they were rather dubious towards G2 and whether it was relevant for meeting industrial demands. However, as the course went on they slowly changed their attitude and during the last day they were more or less ready to sign contracts. An important criticism against G2, however, was the total lack of user interface. The interface issues towards existing control systems were also discussed frequently.

A final experience from the course is that it is very difficult to judge a system like this before a larger application has been made.

A.2.2 Dep. of Chemical Engineering, MIT — 26/2

Bernt Nilsson visited Professor George Stephanopoulos, MIT, Cambridge, MA. Prof. Stefanopoulos is well known for is work in knowledge-based systems applied to chemical processes. He and his group are currently working on a project called Process Design Kit. It is an environment for computer aided design of chemical processes.

A.2.3 Foxboro Company — 29/2

Karl-Erik Årzén, Sven Erik Mattsson, and Bernt Nilsson visited the Foxboro Company, Foxboro, MA. They met Edgar Bristol, Dick Shirley, Bill Pappe, and Peter Hansen. First, a presentation of Foxboro's new control system generation, the *IA (Intelligent Automation) Series* was given. This system is based on IBM PC compatibles and may be integrated with general IBM PC software. As an example, Foxboro will sell IA systems together with the Personal Consultant Online expert system shell.

Foxboro started with research on expert systems in 1982. Foxboro participated (together with Du Pont and the University of Delaware) in the Falcon project. A demonstration was given of a alarm handling system implemented in Loops that automatically, from the process schematics, generated causal models. Later, the Foxboro Exact controller was discussed. Sven Erik Mattsson and Karl-Erik Årzén gave a seminar where they presented the department, the CACE project, Hibliz and the work on real-time expert systems.

A.2.4 Artificial Intelligence Technologies — 1/3

Karl-Erik Årzén visited AI Technologies, Hawthorne, New York and met their director Michael Stock, George Yates, and Joe Cernada. AI Technologies is a small company in the area of industrial expert system applications. Their basic product is a toolkit that provided interfaces between DEC's Common Lisp and several VMS services such as DCL, GKS, RDB etc. AI Tech. also has a background in relational databases.

AI Tech. is a customer-driven company and some of their buzzwords are Integrated AI, Cooperating Expert Systems, and Distributed AI. The main reason for the visit was the system TunePro that AI Tech. has developed as a demonstrator for Combustion Engineering. TunePro is a prototype of an on-line, expert system based, plant-wide PID tuner that can handle single as well as cascaded control loops. The method is based on introducing step changes in the set points and measuring the reaction curves. The system was demonstrated.

A demonstration was also given of a "Intelligent Simulator" that simulated a discrete event model of a potato chip fryer. The system was implemented on a Vaxstation in Common Lisp using Mercury. Mercury is an internal AI Tech. expert system shell that combines object-oriented modelling, rules, and strong links to relational databases via SQL. The intelligence in the system came from the use of object-oriented programming. The system was extremely slow.

A.2.5 IBM, Thomas J. Watson Research Center — 2/3

Karl-Erik Årzén visited Keith Milliken and Alan Finkel at IBM Thomas J. Watson Research Center, Hawthorne, New York. This group is using RETE-based production systems for monitoring of computer mainframes. The YES/MVS system was an early OPS5-based version of the system were certain real-time primitives were added to OPS5. From this, the language YES/L1 has been developed where production systems are integrated with the PL/I language. This has recently been an IBM product under the name KnowledgeTool. Several versions of the system have been installed in an in-house computer center with good results and good responses from the involved operators.

A.2.6 Dept. of Chemical Engineering, Columbia Univ. — 1/3

Bernt Nilsson visited Professor V. Venkatasubramanian at Columbia University, New York. He presented his work on deep-model based fault diagnosis systems. A demostration of the prototype system MODEX was shown.

A.2.7 Du Pont — 2/3

Bernt Nilsson met Bjorn Tyreus at Du Pont, Newark, Delaware. Tyreus work on object-oriented modelling of chemical processes was discussed. He also has experience of several expert system applications at Du Pont.

A.2.8 Carnegie-Mellon University, 1/3 – 2/3

Sven Erik Mattsson visited Carnegie-Mellon University, Pittsburgh, PA. He met Art Westerberg at the Chemical Department. They are developing a general modelling language called ASCEND with many similarities to the CACE project in Lund. He also met Ingemar Hulthage at the Robotics Institute, which is an

independent research institute at CMU with a number of projects in the expert system area. Hulthage had developed an expert system for the design of Al-Li alloys sponsored by Alcoa.

A.2.9 Systems Research Center, 3/3 – 4/3

Karl-Erik Årzén, Sven Erik Mattsson and Bernt Nilsson visited the Systems Research Center (SRC) at the University of Maryland. SRC is a NSF funded national research center. The two days contained a very extensive programme. Some of the persons that they met were Professors Odd Asbjornsen and Thomas McAvoy at the Chemical Department. An expert system for distillation column design, DICODE, was demonstrated. John Baras the director of SRC gave a talk on some of their reserach programmes. These included expert systems for signal processing, control of large flexible structures, adaptive routing in communication networks, and production scheduling for manufacturing. A common factor in the project was the intention to use the μ-Explorer.

Mohammed Modarres at the Chemical Department gave a talk on his work in deep model-based fault diagnosis using the goal-tree/success-tree concept. A demonstration system for a reactor has been implemented that can handle 180 signals in real-time. The system is implemented on a Vaxstation and an IBM AT. Kevin Balon and Larry Lebow described their work in expert control.

A.3 Visit to the UK, 25/4 – 28/4

Claes Rytoft and Anders Åberg from ABB, Börje Rosenberg and Lars Pernebo from SattControl, David Lundberg from Telelogic, and Karl-Erik Årzén and Sven Erik Mattsson from The Department of Automatic Control visited the U.K. April 25 – 28.

A.3.1 Intelligent Automation Lab, Heriot-Watt Univ., 25/4 – 26/4

On April 25'th, Karl-Erik Årzén and Sven Erik Mattson visited Roy Leitch at the Intelligent Automation Lab, Heriot-Watt University, Edinburgh, Scotland. Dr. Leitch has been active in knowledge-based control since 1982. He is currently involved in a lot of ESPRIT activity. The day was devoted to discussion on future collaboration between Heriot-Watt and the Department of Automatic Control. Matters that were discussed were a possible joint ESPRIT Basic Research Action proposal on Reasoning under time-constraints and a collaboration between the CACE project and new ESPRIT II project concerning Training simulators based on qualitative reasoning. Sven Erik Mattsson and Karl-Erik Årzén gave seminars on the CACE project and expert control.

On April 26'th, the rest of the group joined. Dr. Leitch gave a presentation of his involvement in the RESCU project and the ESPRIT I, QUIC project. These projects are described in Chapter 5.

A.3.2 The AI Department, University of Edinburgh – 26/4

The AI Department has groups in the fields of Theorem proving, Natural language, CAD-design and Robotics+Vision. We were visiting the Robotics laboratory under the leadership of Dr Chris Malcolm.

Dr Malcolm showed us an experimental system for automatic assembling using a robot and automatic planning.

There were two main inputs to the system. One was the description of the final product in terms of the external shape. In the demonstrated cases the shapes were a cube and a sofa. The position of the final product was also included.

The second input was the description of the different parts and their approximate positions. The parts consisted of smal cubes glued together in different shapes. They were glued together in an inexact way in order to get non-ideal and uncertain shapes of the parts. It was possible to combine the parts in hundreds of ways to get the final shape.

The description of the environment to the assembly task was not regarded as a normal input. The environment consisted of the robot, the ground floor (a table surface), a big cube used as a fixture and a stick used as a robot tool.

The first activity for the system was to generate a plan of all robot actions in order to get the assembled final product out of the initial state. After that, the robot performed the assembly using the derived action plan. No precise fixture was used, the big cube had an inexact position. This cube was used in cases where there was a need for the robot to take a new grip on a part.

The planner was hierarchical with refinements of the plan on lower levels. The representation of the objects in the planner was precise without any uncertainties. Thus the generated final plan would be ideal in nature. Uncertainties in dimensions, positions etc were taken care of in the assembling stage using "feed-back" ways of thinking. Dr Malcolm said, this was the key to their success.

The top level in the planner did only concern with the question of how to combine the parts to the final product. This level was only dependent on information of the parts and the product. No manufacturing equipment information was included. The next level included the need for the robot to have enought space for the gripping tool in the final assembling stage. The third level looked upon the gravitational stability of the parts in their different positions and orientations. This was needed as holding fixtures were excluded. The parts had to rest stable in the positions they had been put into.

Other levels dealt with orientational planning of the parts and regripping as a mean to extend the number of orientations of the parts. A primitive optimization of plans existed. Direct movements of parts from initial to final position were prefered over movements involving regripping.

The planner was programmed in PROLOG. The user interface was primitive but understandable.

During the assembly phase the robot pushed the parts to known positions by using the stick tool and a special pattern of movements. The final adjustments were made by the gripping tool. Dr Malcolm said, this was used as a cheap solution instead of tactile sensors and vision. Vision will be included later this year.

We were quite impressed by the demonstration. The assembly task seemed to be quite advanced. The CAD-design AI group seemed to be interested in incorporating above ideas in their integrated CAD system. The Turing institute (Glasgow) did also show interest in the demonstration.

A.3.3 Applied Institute for Artificial Intelligence (AIAI) – 26/4

The AI activities at The Edinburgh University are divided into four departments:

- Department of AI, dealing with natural language, robotics, vision, theorem proving, etc.
- AIAI (Artificial Intelligence Application Institute) with 30 persons divided into three groups. A knowledge based planning group, a knowledge representation and expert system group and a programming system group. AIAI is the commercial part of the AI Department.
- Centre for Speech Technique Research (CSTR).
- Cognitive science and computer science.

In these four departments there are approximately 120 employees.

The AIAI

We visited Ken Curry working in the planning group at AIAI. This group is mainly working with knowledge based techniques for nonlinear planning.

The idea behind AIAI is to spread AI technology to the industry. To do this, AIAI have collected and developed various AI tools that people from industry can come and learn how to use. AIAI supports the industry with training and education on these tools. The institute is completely financed by its customers.

The planning group is mainly working in the area of non-linear planning. The domains they have experience of are primarily in the non real-time area. Their opinion is that there is very little done in knowledge based planning in real time environments. They are trying to develop knowledge based system able to combine classical planning with scheduling and simulation. Research areas are e.g. net theory and knowledge representation using graphs. These methods are used to represent the actions and achievements of operators at work.

Their overall goal for the research in knowledge based planning is to include the whole life cycle of process control. All knowledge should be contained in a common representation.

Using results from the NONLIN system for non-linear planning they have developed a tool called OPLAN (Open PLANning architecture). This tool is written in POP and supports the use of a blackboard architecture. The tool contains heuristic knowledge about planning and uses domain independent pruning techniques. They hope to use this tool in real applications, one possibility is a proposed ESPRIT II project with GEC, ICL and CRI concerning scheduling.

A.3.4 PA Computers and Telecommunications - 27/4

PA is an international management and technology-oriented consulting group that is represented in 22 countries and which has 2400 employees. PA Computers and Telecommunications is a division of PA with a staff of 350 people. We met R A E Sargeant and Paul Sachs.

The company has developed a real-time expert system called ESCORT and claims to be "the leading real-time expert system supplier in the world".

The development started with a demonstration system in 1985. It was followed by a feasibility study 1986 especially in the areas of off/on shore, oil & gas and food industry. 1987 the specification of the system was completed and 1988 a full implementation was expected. The first real system, a very big oil& gas application for BP, was to be delivered in June 1988.

ESCORT is a real time expert system which "reduces cognitive overload on operators" in control rooms. It can enable crisis avoidance action to be taken before alarm systems respond. It is limited to this type of applications. New applications, eg for planning, require development of specific new modules. No such development of new applications has been done.

ESCORT is not a control system. It only handles the reasoning process in a real-time control system. To do this it needs two computers, one for the AI-computation and one for communication with a database and communication with a process computer for the traditional part of a process control system. Connected to the AI-computer is one operators interface and one engineers interface.

ESCORT is dedicated to monitoring and diagnosis. It is based on causal relationships that are defined for the components in the plants. A causal network is created which is compared against measurements from the plant. Mismatches are indications on failures. ESCORT has possibilities to store historical information. This is done by incrementally saving the causal networks for different situations.

A demonstration was given of the prototype system. It was connected to a real-time simulation of a part of a plant. The operator interacted with ESCORT through a touch-sensitive screen. The output from ESCORT was a priority list of problems in the plant that required the operators attention. The operator could ask for advice and explanations on the problems.

Present ESCORT system requires a Symbolic 3600 as AI-computer and a uVax as communication computer.

Loops was used in the first prototype. In the current version, KEE is used for the object-oriented parts of the program but only for compiling knowledge and not in run-time. The engineering interface comprises 80% of the code. All procedural knowledge has to be processed by the process computer. Garbage collection is avoided by internal memory management.

10 man years have been spent developing ESCORT.

Typical current cost of an ESCORT system: Hardware £ 140.000 + Software – AI/MMI £ 250.000 + Application software and management £ 250.000.

A.3.5 Cambridge Consultants Limited — 28/4

David Lundberg, Sven Erik Mattsson, Lars Pernebo and Anders Åberg visited Cambridge Consultants Limited, Cambridge and met Jeremy N Clare (Consultant, Instrumentation and Systems Division), Roberto Zanconato (Artificial Intelligence) and Richard Stobart (Control Engineering).

Cambridge Consultants Limited was set up in 1960. The company's aim is to help clients improve their business performance through the effective application of technology. CCL now employs more than 200 people, and is a subsidiary of the international consultancy group Arthur D. Little. Its clients range from governments and multinationals to new venture-backed companies, and are situated all over the world. 60% of the work results in manufactured hardware and software. Examples of applications are cutting of silicon wafers, wrapping of chocolate bars, measurement of thickness of sea ices from a helicopter and a vehicle for inspection of road surfaces.

MUSE

Our motive to visit CCL was to learn about MUSE, which is a toolkit for the development of real-time applications of artificial intelligence. By real-time they

mean that it should respond to external events within the timescale of the application; it should be run-time efficient, support interrupt handle, allow prioritizing of events, have an interface to outside world. A detailed description of MUSE is found in Chapter 3.

MUSE contains a language system, a development package and an example delivery system. The development system contains a structure editor, run-time browser, run-time debugger and a static knowledge base checker.

MUSE is available on Sun workstations and will be ported to VAX/VMS. The current commercial network site price is GBP 25000 and a single licence is available for GBP 15000. The university prices are GBP 8000 and GBP 5000.

Applications of MUSE

The existing applications of MUSE are in the area of alarm handling and condition monitoring. They had one system for analysing alarms from a helicopter engine. It could explain the reason for the pilot and advice him whether he could neglect the alarm and continue the flight or if he should land as fast as possible or when convenient. They also had systems for on-line condition monitoring of diesel engines. It had to cope with variations in fuels and could predict maintenance. You want to predict maintenance so you can do it when the ship is in a harbour and avoid break downs at sea.

The very first application, which actually started the development was a military situation assessment problem. It tries to assess the situation when you have two ships and a number of hostile aircrafts approaching them. The goal is to come up with a defence strategy; at which aircrafts should the ships fire?

A more control oriented application was selection of control modes for position control of a hoover craft with two types of thrusters. It assessed the environment: the direction and speed of wind, tide and waves. It evaluated the performance of the system: trends and saturating controllers.

They also considered the possibilities to combine rules and numerical procedures like Kalman filters for conditioning monitoring and estimation of system state and system parameters.

A.3.6 SIRA Ltd — 28/4

Claes Rytoft, Börje Rosenberg, and Karl-Erik Årzén visited SIRA Ltd in Chislehurst, Kent. SIRA is a non-profit company that mainly is dealing with technology services and research and development. The company was founded 1918 by the UK-government. Today the company have 250 employees and the chairman of the Sira group is Dr. John Alvey.

SIRA:s R & D department consists of an industrial part and a Military & Space department. The Industrial department has three groups of which one is the KBS

(Knowledge Based System) group that we visited. The KBS-group is headed by Dr. J C Taunton. The group has been working with KBS-technique since 1984 and one of the main activities have been to organize a expert systems club. Between 1984-87 this club had about 60 companies as members, must of them UK-based. Within the club there have been different sub-committees dealing with specific tasks, such as:

Rule-based KBS and fuzzy control: Most of the work here was done by SIRA and British Petroleum (BP). SIRA has developed their own fuzzy controller named LINKMAN, that is sold commercially mainly to the cement industry.

Troubleshooting: The work in this group was mainly done by SIRA and British steel. The project name is SAVOIR.

Alarm management: This group mainly concentrated on power generation and their main tool was PICON.

Process scheduling and planning: This group has made prototype implementations for the food, glass and oil business using KEE.

Since the beginning of 1988 SIRA has restructured their Expert system club which today has about 30 members, whereof 2 or 3 is non-UK companies. The new club has started one activity aimed for process management. The project runs for about 9 months and the main tool is KEE on a Texas Explorer. The goal for the project is to try to show that a KBS-system could handle more than one task at the time, and to show that a KBS-system does not have to be rule based. Their priority of the tasks is diagnose, maintenance, optimization and control.

The important issues in knowledge-based control according to their views are reasoning about time and knowledge representation.

The ideas at SIRA are very close to ours. They were interested in maintaining contacts with us during the project.

A.4 Visit to CRI, Denmark, 4/5

Claes Rytoft, Börje Rosenberg, David Lundberg, and Karl-Erik Årzén visited CRI (Computer Resources International). They met AAge Jonasen, Michael Jepsen, Bent Madsen, Ole Ravntoft, and Brian Wheeldon. CRI have approximately 260 employees. The company is divided into the following six departments.

- AIP (Advanced Information Processing) department
- Software Services
- Software Tools
- Manufacture Automation (PROCOS)
- Space
- Defence & Community

The AIP department, that we visited, is participating in several EC project. Among these are three ESPRIT projects, GRADIENT, Eurohelp and ESCORT and one RACE project called ADVANCE. Our hosts at CRI where all working in the GRADIENT project. A description of GRADIENT is found in Chapter 5.

CRI uses the GRADIENT prototype to control a packet switching data network. The network is simulated in a SUN computer. The network is designed using KTAS datapak network PAXNET as an example. The prototype is implemented on two TI Explorer computers. One is hosting the network description and one the QRES alarm handling system. When the prototype is completed, the SES and RESQ expert systems should run on another TI Explorer. As long as Strathclyde is not ready with the graphics, the demo system has to use KEEPICTURES in KEE for the user interface. This makes the prototype so slow that it is quite impossible to use. The response time for changing one single KEEPICTURES window on the TI Explorer is about 45 seconds Just to call a pop-up menu takes about 15 seconds.

A quite interesting concept is used in the GRADIENT project. Several expert systems are solving different tasks in process control. These expert systems exchange data and knowledge over communication links. Deep knowledge is used in the process description and rule-based expert system can use this process model in the reasoning process.

Future directions

Design knowledge stored in the process models should come directly from the designer. An important trend will be to develop tools for the designer, making it possible to automatically extract the process knowledge when designing or re-designing the process. The fault diagnosis will probably always need unstructured expert knowledge, added to the expert systems in the traditional way.

B

Competence Profile

B.1 Asea-Brown Boveri AB — ABB

ABB is one of the worlds largest electro-technical companies. The Swedish part of ABB delivers electronic equipment and automation systems to a yearly revenue of about 4.000 MSEK of which 70% comes from export. To be a leading supplier of automation systems is a prerequisite for delivering complete plants which is large part of ABB's business. An important part of ABB's R & D is therefore focussed on this area.

ABB belongs to the technological forefront in many areas. Some examples are robotics, technique for power system monitoring, and automation systems. ABB is represented in the IDEON Research Park in Lund since 1983. In Lund, the work is concentrated on R & D in man-machine systems, control and optimization of distant heating networks, and advanced sensor electronics. ABB in Västerås has a Corporate Research Group in AI. Similar groups exist is Heidelberg and Baden. ABB is one of the members of the ESPRIT I project GRADIENT and also a member of the COGSYS industrial club.

B.2 SattControl AB

SattControl is among the world leaders in process control and automation. The company has a broad competence within information technology in general.

SattControl belongs to the technological forefront in man-machine communication systems and develops process control languages, MAP communication systems, database techniques, work cell computers for CIM and FMS applications, real-time programming, and knowledge-based systems. SattControl AB has about 1.100 employees and belongs since one year to the Alfa-Laval group.

The R & D group is located in Lund and consists of about 100 persons. The marketing is done both by wholly-owned market companies, whereof 10 is outside Sweden, and by Alfa-Laval's world-wide organization.

B.3 Televerket and Telelogic

The Swedish Telecommunications Administration (Televerket) is a government-owned enterprise and public utility with about 50.000 employees. Televerket plays a dominant role in the telecommunications community based on its large investments, industrial organisation, and extensive R & D on new services and products within telecommunication systems. A large part of the R & D on programming methodology is placed at the subsidiary company TeleLogic.

TeleLogic develops and markets technique, products, and services for system development based for instance on SDL and Ada. The company also perform consulting within the communication area. TeleLogic has a partly-owned marketing company in Bruxelles and cooperates with TeleSoft, San Diego on ADA. TeleLogic is established at 7 places in Sweden and has in all about 250 employees. The Malmö branch of TeleLogic is focused on tools and methods for SDL, system development methodology, and base techniques for AI and knowledge-based systems. It consists of about 80 persons.

B.4 Department of Automatic Control, Lund

The department of Automatic Control at Lund Institute of Technology has about 30 persons. The department is headed by Professor Karl Johan Åström. There are six persons at the professorial level. Twenty-nine DSc degrees have been awarded, and more than 370 students have made MS theses at the department. The group has a wide experience in adaptive control, computer aided control engineering, robotics, real-time programming, real-time knowledge-based systems and control applications. The department has good computing resources in the form of VAX, Sun, Silicon Graphics, Symbolics, Mac II and a good control library. The department has access to G2 and KEE. The department also has excellent relations with industry.

Dr. Karl-Erik Årzén has recently finished his Ph D thesis "Realization of Expert System Based Feedback Control". In this work a real-time, blackboard based

expert system shell has been developed and implemented (Årzén, 1986a; 1986b; 1987; Åström and Anton, 1984; Åström *et al*, 1986).

Dr. Sven Erik Mattsson is the project leader for the Computer Aided Control Engineering (CACE) project. The project is aimed at developing an environment for modelling and simulation. The concepts which are used are hierarchical models with model classes, multiple realizations, and symbolic equations for knowledge representation (Mattsson, 1988a; 1988b; Elmqvist and Mattsson, 1986). A prototype of this system is currently being implemented in KEE.

The department have also a project where an expert system is used as an intelligent front-end to an interactive package for systems identification (Larsson and Persson 1986; 1987). This project has led to two licentiate theses.

C

Glossary

Active value

Frame attribute to which a procedure, or demon, is attached. The procedure is executed when the attribute is changed or referred to. The programming style is called access-oriented.

Agenda

A prioritized list of waiting activities. Used in blackboard systems to schedule knowledge sources.

Antecedent

The IF-part of a production rule. Other names are premise and condition.

Application-specific system

Knowledge-based system framework aimed at a specific type of applications.

Artificial Intelligence

A subfield of computer science, which according to one definition is the study of how to make computers do things at which, at the moment, people are better.

Attribute

A property of an object. Also called slot.

Autonomous system

A system that solves problems without human intervention. Autonomously guided vehicles and sonar interception systems are some examples.

Backtracking

A search strategy that makes guesses at several stages in the search procedure. If a guess gives an unsatisfactory result, the system backtracks and makes a new guess.

	Could be dependency-directed or chronological. Chronological backtracking always backtracks to the latest guess made.
Backward chaining	An inference method where the system starts with what it wants to prove and then tries to find the necessary facts in the data base or as the conclusion of a rule. Also known as goal-directed search. Contrast with forward chaining.
Blackboard	A data base used by several independent knowledge sources to exchange information about the problem solving and to store the problem solving state.
Blackboard architecture	An expert system architecture in which several independent knowledge sources each examine a common data base, called a blackboard. An agenda-based control system continually examines all of the possible pending actions and chooses the one to try next.
Causal model	A model of a physical object that expresses the causal relations among the involved signals, or events.
Certainty factor	A number that measures the certainty, credibility or confidence a fact or rule has.
Circumscription	A technique for formalizing certain notions in nonmonotonic reasoning. It corresponds to minimizing the number of objects having certain properties. In effect, one is considering conjectures that for certain properties P, an object x does not have P unless it is required to do so. Developed by McCarthy.
Class	A group of objects that share the same attributes and behavior. Organized into an inheritance lattice.
Cognitive science	An interdisciplinary research area concerning the principles by which intelligent entities interact with their environment. Includes topics from psychology, computer science, physiology, philosophy, engineering etc.
Compiled knowledge	Knowledge that has been structured and compiled into a form that is efficient for executing. The heuristic knowledge of, e.g., a human process operator can, e.g., be compiled into a production rule format.

Composite object — An object whose attribute values are other objects.

Concept — Used as a designation for real or abstract objects and terms, that are represented in a knowledge-based system.

Conflict resolution — The technique of resolving the problem of multiple matches in a rule-based system.

Connectionism — A highly parallel computational paradigm. Pieces of information are represented by very simple computing elements that communicate by exchanging simple messages. Complex computations are carried out by virtue of massively parallel interconnection networks of these elements. Self-learning abilities.

Control (of a KBS) — The method used by the inference engine to regulate the order in which reasoning occurs. Backward chaining, forward chaining, and blackboard agendas are all examples of control methods.

Data-driven — An approach to problem solving that starts from current or initial information and employs forward chaining.

Declarative knowledge — A description of *what is*. Contrast with procedural knowledge, which is a description of *how to*.

Declarative programming — An organizational technique for computer programs. The wanted result and preconditions are stated instead of a step-by-step description of how to solve the task. Contrast with procedural programming.

Deep knowledge — Knowledge of basic theories, first principles, axioms, and facts about a problem domain. Often in the form of a model of the behavior of the problem domain. The model could be expressed as, e.g., a causal model.

Default reasoning — Patterns of inference that permit drawing conclusions suggested but not entailed by their premises. Especially important for systems that must reason from incomplete information.

Demon — A procedure that is attached to a frame attribute. The procedure is executed when

	the attribute is changes or referred to. The programming style is called access-oriented.
Dependency-directed backtracking	A problem solving technique for evading contradictions. Is invoked when the problem solver discovers that its current state is inconsistent. The goal is, in a single operation, to change the problem solver's state to one that contains neither the contradiction just uncovered nor any contradictions encountered previously. This is achieved by consulting records of the inferences the problem solver has performed and records of previous contradictions.
Empirical knowledge	Knowledge based on empirical or experiential observations of a process.
Experiential knowledge	Knowledge based on empirical or experiential observations of a process.
Expert control	Seeks to extend the range of conventional control algorithms by encoding general control knowledge and heuristics in a supervisory expert system.
Expert system	A computer program that uses expert knowledge to attain high levels of performance in a narrow problem area. The results are compatible with those of a human expert. Knowledge-based system is sometimes used as a synonym.
Expert system framework	A computer environment that provides different tools for implementing expert systems. One or many knowledge representation techniques are supported. Also expert system shell.
Expert system shell	A computer environment that provides different tools for implementing expert systems. One or many knowledge representation techniques are supported. Also expert system framework.
Explorative programming	Programming without any given specifications. Requires powerful and flexible programming environments and programming languages that support rapid testing and prototyping.
Facet	Describes an attribute in a frame system. One facet is the value of the attribute. Other

	facets may give additional documentation, specify allowed data types of the attribute value, or record justifications for the attribute.
First generation diagnosis system	Knowledge-based diagnosis system based on empirical knowledge, usually represented as rules, of how fault symptoms and causes relate.
Forward chaining	A problem solving technique where hypotheses are verifies by starting with known facts and trying to make deductions from these. The same as data driven. Contrasts with backward chaining.
Frame	A knowledge representation scheme based on the idea of a frame of reference. A frame consists of slots or attributes that describe the features of the frame. The slots are further described by facets.
Frame axioms	Axioms that describes which facts that are changed by an event and which that are not. Only possible in closed worlds.
Frame problem	A problem in temporal reasoning. In order to reason about the effect of an event on the world, it is necessary to assert not only what effects it has on the worlds but also what effects it does not have.
Functional views	Knowledge views related to the activities, functions or services that the process control system is expected to supply to the organizations employing it. Also service views.
Fuzzy controller	A controller based on rules of how the control variable be selected based linguistically quantized values of the measured variables. Uses fuzzy logic to describe the quantized values.
Fuzzy logic	A logical theory where the truth values 'true' and 'false' have been replaced with more approximate values like 'not very true', 'not likely' and 'very unlikely'.
Garbage collection	A background activity where free memory are reclaimed to the programming system. Important in languages with dynamic memory allocation, e.g., Lisp.

Goal-directed system An inference method where the system starts with what it wants to prove and then tries to find the necessary facts in the data base or as the conclusion of a rule. Also known as backward chaining. Contrast with data-driven system.

Heuristic A rule or some other piece of knowledge that is based on experience or observation: a rule of thumb.

Hierarchical planning Top-down planning that allows macro actions which themselves may be plans.

Hybrid system An expert system shell that allows a variety of different knowledge representation techniques.

Hypertext A technique that extends the traditional notion of "flat text" files by allowing more complex organizations of the material. Mechanisms that allow direct machine-supported links from one textual chunk to another and new interactive interface techniques allow the user directly interact with these chunks and to establish new relationships between them. Hypermedia also includes non-textual informations such as images, time series signals, audio recordings etc.

Induction system A system that can deduce rules from a material consisting of many examples from the problem area.

Inference The process of drawing conclusions from premises.

Inheritance A process where new objects in a hierarchical structure can get new attributes deduced from more general objects in the structure.

Inference engine The part of a knowledge based system that contains the general problem-solving knowledge.

Instance An object that describes a unique member of some object class.

Instantiation The process where a new individual of a certain type is created.

Job shop scheduling The job shop scheduling or factory scheduling problem concerns the allocation over time

	of a finite set of resources to specific manufacturing operations such that the orders for parts are received by the factory are produced in a timely and cost-effective fashion.
Knowledge	Information that is used to behave in an intelligent way.
Knowledge acquisition	The process of acquiring, structuring and organizing the knowledge of a particular domain. Also knowledge elicitation.
Knowledge base	The part of a knowledge based system that contains the knowledge.
Knowledge-based system – KBS	A computer system that contains knowledge and is able to reason with that knowledge to reach solutions. Sometimes a synonym to expert system. See expert system.
Knowledge elicitation	The process of acquiring, structuring and organizing the knowledge of a particular domain. Also knowledge acquisition.
Knowledge engineer	The person who designs and builds the expert system. This person should have experience of artificial intelligence methods.
Knowledge representation	Formalisms used to represent knowledge. By using these formalisms, it is possible to handle and manipulate the knowledge. Typical formalisms are semantic networks, frames and predicate logic.
Knowledge source	Knowledge module in a blackboard system.
Knowledge tools	Provide the different user views and perform the different tasks in the process control system. Terminology used in in the basic system concept in Chapter 6.
Learning control	A combination of AI techniques and control theory that utilizes various learning schemes for control purposes. Also intelligent control and self-organizing control.
Lisp machine	Workstation with dedicated hardware for Lisp execution. Has very powerful programming environment.
Local fault model	A model that relates faults in a physical component with possible causes internal to the component.

Mental model The human operator's apprehension of how the process behaves.

Message passing Communication method between objects in a object-oriented system. Supports data abstraction and generic algorithms.

Method Procedure associated with an object that responds to a certain message.

Modal logic A logic system that includes the notions of necessity and possibility. A proposition is necessarily true if it could not be the case that it was false. A proposition is possible if it is not necessary that it be false

Modus ponens An inferencing rule which says that whenever a fact **A** is known to be true and there is an implication $\mathbf{A} \Rightarrow \mathbf{B}$, it is permitted to conclude that **B** is true.

Monotonic reasoning A logical system where axioms that have been stated and conclusion that have been drawn are not allowed to change during the reasoning process. The set of beliefs is monotonically increasing. The case for standard logic systems.

Multilevel flow models – MFM A technique for describing physical systems that emphasizes functional relations among the involved components. Systems are described in terms of goals, functions, and components. Two main abstraction relations exist: the part–whole relation and the means–ends relation. Developed by M. Lindh.

Multiple inheritance An inheritance mechanism where a class may have more than one superclass. Contrasts with single inheritance.

Multiple perspectives Used in some object-oriented system for the case when a single object, at every time, can be seen as the instance of one out of a set of classes. It can be seen as a special kind of multiple inheritance where the behavior and attributes from the inherited classes are kept separated in the object instead of being combined together.

Multiple worlds Represents alternative states of knowledge in which different assumptions have been made. They allow the problem solver to set up hypothetical assumptions which automatically

	are withdrawn when worlds are deleted. Also multiple viewpoints and hypothetical worlds.
Neural network	Connectionist system modelled after the neuron structure of the human brain.
Nonmonotonic reasoning	A knowledge based system allowing new information that can make old deductions become false. This is very important when information changes during execution.
Object-attribute-value triplets	A way of storing knowledge in an object-oriented system.
Object-oriented programming	A style of programming based on directly representing physical objects and abstract concepts in the machine. The basic entity is the object which has a local state and a behavior. Objects are asked to perform operations by sending messages to them. Examples of programming languages are SIMULA and SMALLTALK.
Ontological knowledge	Knowledge based on theoretical knowledge which is analytic and derived from first principles.
Opportunistic reasoning system	A reasoning system that changes inferencing strategy depending on the problem solving state.
Organizational views	Knowledge views related to the different organizations or user groups that interact with the process control system. Also user views.
Predicate logic	A classical logic which is based on the use of predicates to express relations among objects. The formal basis for Prolog. Also predicate calculus or first order logic.
Premise	The IF-part of a production rule. Sometimes it is called 'antecedent'. The THEN-part is called 'conclusion'.
Procedural	A technique for organization of programs, by using a step by step description of how to solve a problem. The opposite word is declarative.
Procedural knowledge	A description of *how to*. Contrast with declarative knowledge, which is a description of *what is*.

Process The controlled flow of matter, energy, or information from generation (source), via transport, storage, distribution, and change to consumption (sink). The flow may be discrete or continuous.

Process control The different tasks that interact with a specified process and with the different users of the process.

Process control system The system that controls and supervises the the operation of a process.

Production system A rule-based system where rules or productions are matched against the contents of a working memory and executed by forward chaining inferencing.

Production rule A type of rule in a knowledge based system, usually expressed as an IF-THEN statement.

Propositional logic A classical logic which is based on propositions without any internal semantics.

Qualitative models Models of a process in the terms that a human uses when describing and analysing the process. Used for simulation and analysis. Based on the ideas in Naive Physics.

Qualitative knowledge Knowledge about and based on matters that cannot be measured quantitatively.

Quantitative knowledge Knowledge about and based on matters that can be numerically measured. An differential equation model is one example.

Recognize-act cycle The execution cycle of a forward chaining inference engine. Fulfilled rules are collected during the match phase. During the select phase one rule is chosen for execution. During the act phase the right hand side of the rule is executed.

Rule A formal way to specifying a fact, directive or strategy. The most common way to represent it is by the IF-THEN construction.

Rule based system A program organized as a set of rules.

Script A knowledge structure containing a stereotype sequence of actions.

Second generation diagnosis system Knowledge-based diagnosis system based on deep-level or first principles knowledge about the problem domain.

Semantic network — A knowledge representation method based on a structure of nodes that represent objects and named arcs between the nodes that define attributes and relations.

Service views — Knowledge views related to the activities, functions or services that the process control system is expected to supply to the organizations employing it. Also functional views.

Single inheritance — An inheritance mechanism where a class may have only one superclass. Contrasts with multiple inheritance.

Situation calculus — A temporal formalism where a situation is a snapshot of the universe at a given moment. Actions are the means of transforming one situation to another.

STRIPS assumption — States that the only facts that are changed by a plan operator is the ones which are explicitly named in the add and delete list of the operator. A way to handle the frame problem in closed worlds.

Superclass–subclass hierarchy — A directed graph that describes the relations among object classes. A subclass inherits behavior and attributes from its superclasses.

Tasks-tools-roles model — A model of process control. It consists of the tasks to be performed in order to assure proper function of the controlled process, the tools that do this, and the roles which interact with the tools. The roles represent single users or user groups.

Temporal logic — A logic system that includes time intervals or time instants and the truth relations over time.

Time constrained reasoning — The situation where a reasoning system must be able to come up with the best solution before a certain deadline.

Truth maintenance system — A system that revises sets of beliefs when new information is found to contradict old information. Inconsistencies in the set of beliefs are resolved by using dependency-directed backtracking to alter the minimal set of beliefs which is responsible for the contradiction.

User views — Knowledge views related to the different organizations or user groups that interact with

	the process control system. Also organizational views.
Validity interval	A time interval that tells how long the associated fact is valid.
Working memory	The fact database in a production system.

References

ALLARD, J.R. and W.F. KAEMMERER (1987): "The goal/subgoal knowledge representation for real-time process monitoring," *Proceedings IJCAI-87*, Milano, Italy, pp. 394–398.

ALLEN, E.M. (1983): "YAPS: Yet another production system," TR-1146, Department of Computer Science, University of Maryland.

ALLEN, J.F. (1984): "Towards a general theory of action and time," *Artificial Intelligence*, **23**, 123–154.

ANDERSON, C.W. (1986): "Learning and problem solving with multilayer connectionist systems," Ph.D. Dissertation, COINS Technical report 86-50, University of Massachusetts, Amherst, MA.

ANDERSON, J.M., W.S. COATES, A.L. DAVIS, R.W. HON, I.N. ROBINSON, S.V. ROBINSON and K.S. STEVENS (1987): "The architecture of FAIM-1," *IEEE computer*, **20**, 1, 55–65.

ÅRZÉN, K-E. (1986): "Use of expert systems in closed loop feedback control," *Proc. of American Control Conference*, Seattle, WA.

ÅRZÉN, K-E. (1987): "Realization of expert system based feedback control," Ph.D. thesis CODEN: LUTFD2/TFRT-1029, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ÅSTRÖM, K.J. and J.J. ANTON (1984): "Expert control," *Proc. 9'th IFAC World Congress*, Budapest, Hungary.

ÅSTRÖM, K.J., J.J. ANTON and K.-E. ÅRZÉN (1986): "Expert control," *Automatica*, **22**, 3, 277–286.

BIRKY, G.J., T.J. MCAVOY and M. MODARRES (1988): "An expert system for distillation control design," *Computers and Chemical Engineering*, To appear.

BIRDWELL, J.D., J.R.B. COCKETT, R. HELLER, R.W. ROCHELLE, A.J. LAUB, M. ATHANS and L. HATFIELD (1985): "Expert systems techniques in a computer based control system analysis and design environment," *Proc. 3rd IFAC/IFIP Int. Symp. on Computer Aided Design in Control and Engineering Systems*, Lyngby, Denmark.

BLICKLEY, G.J. (1987): "Designing control systems with an expert system," *Control Engineering*, **September**, 112–113.

BOBROW, D.G. (1984): in Bobrow (Ed.): *Qualitative Reasoning about Physical Systems*, Reprinted from the journal of Artificial Intelligence, volume 24.

BOBROW, D.G. and T. WINOGRAD (1977): "An overview of KRL, a knowledge representation language," *Cognitive Science*, **1**, 1, 3–46.

BOBROW, D.G., K. KAHN, G. KICZALES, L. MASINTER, M. STEFIK and F. ZDYBEL (1985): "COMMONLOOPS: Merging Common Lisp and object-oriented programming," Intelligent Systems Lab. Series ISL-85-8, Xerox Palo Alto Research Center, Palo Alto, California.

BRACHMAN, R.J. (1979): "On the epistemological status of semantic networks," in N.V. Findler (Ed.): *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York.

BRISTOL, E.H. (1977): "Pattern recognition: An alternative to parameter identification in adaptive control," *Automatica*, **13**, 197–202.

BROWNSTON, L., R. FARRELL, E. KANT and N. MARTIN (1985): *An Introduction to Rule-based Programming*, Addison-Wesley, Reading, MA.

BURSTALL, R.M. (1980): "The semantics of CLEAR, a specification language," *Proc. of the 1979 Copenhagen Winter School on Abstract Software Specification,*, LNCS 86, Springer-Verlag.

CANNON, H.I. (1982): "Flavors: A non-hierarchical approach to object-oriented programming,," unpublished paper.

CHANDRASEKARAN, B. (1986): "Generic tasks in knowledge-based reasoning: High level building blocks for expert system design," *IEEE Expert*, **1**, 3, 23–30.

CHANDRASEKARAN, B. (1987): "Towards a functional architecture for intelligence based on generic information processing tasks," *Proc. IJCAI-87*, pp. 1183–1192.

CHUNG, D.T. and M. MODARRES (1987): "GOTRES: An expert system for fault detection and analysis," Department of Chemical and Nuclear Engineering, The University of Maryland, College Park, Maryland.

CLANCEY, W.J. (1985): "Heuristic classification," *Artificial Intelligence*, **27**, 3, 289–350.

CLINGER, W. (1981): "Foundations of actor semantics," Tech. rep. 633, MIT Artificial Intelligence Lab., Cambridge, MA.

CLOCKSIN, W.F. and C.S. MELLISH (1984): *Programming in Prolog*, Springer-Verlag, Berlin.

CONKLIN, J. (1987): "Hypertext: An introduction and survey," *Computer*, **September**, 17–41.

CONNELL, M.E. and P.E. UTGOFF (1987): "Learning to control a dynamic physical system," *Proc. AAAI-87*, pp. 456–460.

CORSBERG, D. (1987): "Alarm filtering: Practical control room upgrade using expert system concepts," *InTech*, **April**, 39–42.

COX, B.J. (1986): *Object Oriented Programming - An Evolutionary Approach*, Addison-Wesley, Reading, MA.

CRUISE, A., R. ENNIS, A. FINKEL, J. ELLERSTEIN, D. KLEIN, D. LOEB, M. MASULLO, K. MILLIKEN, H. VAN WOERKOM and N. WAITE (1987): "YES/L1: Integrating rule-based, procedural, and real-time programming for industrial applications," *Proceedings of the Third Conference on Artificial Intelligence Applications*, IEEE Computer Society, Washington, D.C..

DAHL, O.J. and K. NYGAARD (1966): "SIMULA–an algol-based simulation language," *Communications of the ACM*, **9**, 671–678.

DARLINGTON, J. and M. REEVE (1983): "ALICE and the parallel evaluation of logic programs," Tech. rep., Imperial College of Science and Technology, London, England.

DAVIS, R. and R.G. SMITH (1983): "Negotiation as a metaphor for distributed problem solving," *Artificial Intelligence*, **20**, 63–109.

DE KLEER, J. (1986): "An assumption-based TMS," *Artificial Intelligence*, **28**, 127–162.

DE KLEER, J. and J. S. BROWN (1984): "Qualitative reasoning about physical systems," *Artificial Intelligence*, **24**.

DECKER, K.S. (1987): "Distributed problem solving," *IEEE Transaction on Systems, Man and Cybernetics*, **SMC-17**, 5, 729–740.

DHALIWAL, D.S. (1985): "The use of AI in maintaining and operating complex engineering system,".

DIN (1985): no. 19222, March 1985.

DOYLE, J. (1979): "A truth maintenance system," *Artificial Intelligence*, **20**, 231–272.

DRESCHER, G.L. (1985): "The ObjectLisp user manual (preliminary)," LMI Corp., Cambridge, MA.

DUDA, R.O., P.E. HART and R. REBOH (1977): "A rule-based consultation system for mineral exploration," *Proc. of the Lawrence Symposium on Systems and Decision*, UC Berkeley, California, pp. 306–309.

ELMQVIST, H. and S.E. MATTSSON (1986): "A Simulator for Dynamical Systems Using Graphics and Equations for Modelling," *Proceedings of the IEEE Control Systems Society Third Symposium on Computer-Aided Control Systems Design (CACSD)*, Arlington, Virginia, September 24–26, 1986, pp. 134–139, Accepted for publication in Control Systems Magazine.

ENDERLE, G., K. KANSY and G. PFAFF (1984): *Computer Graphics Programming (GKS—The Graphics Standard)*, Springer-Verlag.

FAHLMAN, S.E. and G.E. HINTON (1983): "Massively parallel architectures for AI: NETL, THISTLE and BOLTZMANN machines," *Proc. of National Conference on Artificial Intelligence AAAI-83*, pp. 109–113.

FIKES, R.E. and N.J. NILSSON (1971): "STRIPS: A new approach to the application of theorem proving in problem solving," *Artificial Intelligence*, **2**, 189–208.

FISHER, E.L. (1986): "An AI-based methodology for factory design," *AI Magazine*, **7**, 4, 72–85.

FORBUS, K.D. (1984): "Qualitative process theory," *Artificial Intelligence*, **24**, 85–168.

FORGY, C.L. (1979): "OPS4 User's manual," Technical report CMU-CS-79-132, Department of Computer Science, Carnegie-Mellon University.

FORGY, C.L. (1981): "OPS5 User's manual," Technical report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University.

FORGY, C.L. (1982): "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, **19**, 1, 17–37.

FOX, M.S. (1983): "Constraint-directed search: A case study of job shop scheduling," Ph D thesis, Carnegie-Mellon University.

FOX, M.S., B.P. ALLEN and G.A. STROHM (1982): "Job shop scheduling: An investigation in constraint-directed reasoning," *Proc. Second National Conference on Artificial Intelligence*, pp. 155–158.

FRANCIS, J.C. and R.R. LEITCH (1985): "Artifact: A real-time shell for intelligent feedback control," in M.A. Bramer (Ed.): *Research and Developments in Expert Systems*, Cambridge University Press, UK.

FU, K-S. (1970): "Learning control systems – review and outlook," *IEEE Transactions on Automatic Control*, **15**, 210–221.

FU, K-S. (1971): "Learning control systems and intelligent control systems: An

intersection of artificial intelligence and automatic control," *IEEE Transactions on Automatic Control*, **16**, 70–73.

FUJIWARA, R. *et al* (1985): "An intelligent load flow engine for power system planning," *Proc. of 1985 Power Industry Computer Application Conference*, pp. 236–241.

GELLI, P. (1987): "Evaluation and comparison of three specification languages: SDL, LOTOS and ESTELLE," in R. Saracco and P.A.J. Tilanus (Eds.): *SDL'87: State of the Art and Future Trends*, Elsevier Science Publications (North-Holland), pp. 211–231.

GEORGEFF, M.P. and A.L. LANSKY (1986): "Procedural knowledge," *Proceedings of the IEEE*, **74**, 10, 1383–1398.

GOGUEN, J.A., J.P. JOUANNAUD, J.P. MESEGUER and K. FUTATZUGI (1985): "Principles of OBJ2," *Proc. of 12th Symposium on Principles of Programming Languages.*

GOLDBERG, A. and D. ROBSON (1983): *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, MA.

GOMEZ, F. and B. CHANDRASEKARAN (1981): "Knowledge organization and distribution for medical diagnosis," *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-11**, 1, 34–42.

GOODWIN, J.W. (1987): "A theory and system for non-monotonic reasoning," Ph.D. thesis, Linköping Studies in Science and Technology, Dissertation no. 165, Linköping Institute of Technology, Sweden.

GREEN, C. (1969): "Application of theorem proving to problem solving," *Proc. IJCAI*, Washington DC, pp. 219–239.

HALASZ, F.G., T.P. MORAN and T.H. TRIGG (1987): "Notecards in a nutshell," *Proc. of the ACM Conference on Human Factors in Computing Systems*, Toronto, Canada.

HALPERN, J.Y. (1986): *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference*, Kaufmann, Los Altos, CA.

HAYES, P. (1973): "The frame problem and related problems in artificial intelligence," in A. Elithorn and D. Jones (Eds.): *Artificial and Human Thinking*, Jossey-Bass Inc..

HAYES-ROTH, F. (1980): "Towards a framework for distributed AI," *SIGART Newsletter*, **October**, 51–52.

HEIN, U. (1983): "PAUL–The kernel of a representation and reasoning system designed for knowledge engineering tasks," AILAB Working paper No 16, Linköping University, Sweden.

HEWITT, C. and B. KORNFELD (1980): "Message passing semantics," *SIGART Newsletter*, **October**.

HILLIS, D. (1985): *The Connection Machine*, The MIT Press, Cambridge, MA.

HOARE, C.A.R. (1985): *Communicating Sequential Processes*, Prentice-Hall Intl..

HOPGOOD, F.R.A., D.A. DUCE, J.R. GALLOP and D.C. SUTCLIFFE (1983): *Introduction to the Graphical Kernel Standard (GKS)*, Academic Press.

HWANG, K., J. GHOSH and R. CHOWKWANYUN (1987): "Computer architectures for artificial intelligence," *IEEE Computer*, **20**, 1, 19–27.

INFERENCE CORP. (1984): *ART – User Manual.*

INTELLICORP (1984): *Knowledge Engineering Environment (KEE) – User Manual*, Menlo Park, CA.

JAMES, J.R., D.K. FREDERICK and J.H. TAYLOR (1985): "The use of expert-system programming techniques for the design of lead-lag compensators," *IEE Conference, Control '85*, Cambridge, England.

JEFFREYS, S. (1987): "Software simplifies batch control design," *Control Engineering*, **September**, 107–109.

KIM, I.S. and M. MODARRES (1987): "Application of goal tree-success tree models as the knowledge-base of operator advisory systems," *Nuclear Engineering and Design*, **104**, 67–81.

KOSEKI, Y., S. WADA, T. NISHIDA and H. MORI (1987): "SHOOTX: A multiple knowledge based diagnosis expert system for NEAX61 ESS," *ISS Proceedings.*

KRAUS, T.W. and T.J. MYRON (1984): "Self-tuning PID controller uses pattern recognition approach," *Control Engineering*, June, 106–111.

KUIPERS, B.J. (1986): "Qualitative simulation," *Artificial Intelligence*, **29**, 289–338.

LAFFEY T. J. and T.A. NGUYEN (1986): "Reasoning about fault diagnosis with LES," *IEEE Expert*, **Spring**.

LAFFEY, T.J., P.A. COX, J.L. SCHMIDT, S.M. KAO, and J. Y READ (1988): "Real-time knowledge-based systems," *AI Magazine*, **9**, 1, 27–45.

LAMB, D.E., P. DHURJATI and D.L. CHESTER (1986): "Development of an expert system for fault identification in a commercial scale chemical process," *Proc. Sixth Int. Workshop on Expert Systems, 2*, pp. 1371–1382.

LARSSON, J.E. and P. PERSSON (1987): "An expert system interface for IDPAC," Technical report TFRT-3184, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

LEITCH, R.R. (1987): "Modelling of complex dynamic systems," *IEE Proceedings Part D*, **134**, 4, 245–250.

LENAT, D.B. (1982): "AM: Discovery in mathematics as heuristic search," in R. Davis and D.B. Lenat (Eds.): *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, New York.

LENAT, D.B. (1983): "EURISKO: A program that learns new heuristics and domain concepts," *Artificial Intelligence*, **21**, 1,2, 61–98.

LINDH, M (1983): "A systems Modelling Framework for the design of integrated Process Control Systems," Risö-M-2409..

LIND, M (1987): "Systembeskrivelse og presentation i processkontroll," Report from the SIP project.

LLOYD, M. (1985): "Graphical function chart programming for programmable controllers," *Control Engineering*, **October**, 73–76.

MARCUS, S and J. MCDERMOTT (1987): "SALT: A knowledge acquisition tool for purpose-and-revise systems," Technical report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa.

MATTSSON, S.E. (1988a): "On Model Structuring Concepts," To be presented at the 4th IFAC Symposium on Computer-Aided Design in Control Systems (CADCS), August 23–25 1988, P.R. China.

MATTSSON, S.E. (1988b): "On Modelling and Differential/Algebraic Systems," *Simulation*, Accepted for publication.

MCCARTHY, J. (1980): "Circumscription–A form of non-monotonic reasoning," *Artificial Intelligence*, **13**, 27–39.

MCCARTHY, J. and P.J. HAYES (1969): "Some philosophical problems from the standpoint of artificial intelligence," in B. Meltzer and D. Mitchie (Eds.): *Machine Intelligence*, American Elsevier, New York.

MCDERMOTT, J. (1980): "R1: A rule-based configurer of computer systems," Technical report 80–119, Carnegie-Mellon Department of Computer Science.

MCDERMOTT, D. (1982): "A temporal logic for reasoning about processes and plans," *Cognitive Science*, **6**, 2.

MCDERMOTT, D. and J. DOYLE (1980): "Non-monotonic logic I," *Artificial Intelligence*, **13**, 41–72.

MICHIE, D, and R.A. CHAMBERS (1968): "BOXES: an experiment in adaptive control," in E. Dale and D. Michie (Eds.): *Machine Intelligence 2*, Oliver and Boyd, Edinburgh, pp. 137–152.

MILLIKEN, K.R., A.V. CRUISE, R.L. ENNIS, A.J. FINKEL, J.L. HELLERSTEIN, D.J. LOEB, D.A. KLEIN, M.J. MASULLO, H.M. VAN WOERKORN and

N.B. WAITE (1986): "YES/MVS and the automation of operations for large computer complexes," *IBM Systems Journal*, **25**, 2, 159–180.

MILNER, R. (1980): *A Calculus of Communicating Systems*, LNCS 92, Springer Verlag, Berlin.

MINSKY, M. (1975): "A framework for representing knowledge," in P.H. Winston (Ed.): *The Psychology of Computer Vision*, McGraw-Hill, New York.

MODARRES, M. and T. CADMAN (1986): "A method of alarm system analysis in process plants with the aid of an expert computer system," *Comput. Chem. Eng.*, **10**, 557–565.

MOKHTARI, S. *et al* (1987): "A unit commitment expert system," *Proc. of the Power Industry Computer Application Conference.*

MOORE, B.C. (1986): "Hypothesis feedback models for multivariable systems," Technical Report, Dow Chemical Co., Plaquemine, Lousiana, submitted to the IEEE Trans. on Automatic Control.

MORRIS, H.M. (1987): "Design station uses AI for factory cell control," *Control Engineering*, **September**.

NEWELL, A. and H.A. SIMON (1972): *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

NEWELL, A. and H.A. SIMON (1976): "Computer science as empirical inquiry: Symbols and search," *Communications of the ACM*, **19**, 3.

NEWELL, A., J.C. SHAW and H.A. SIMON (1960): "Report of a general problem-solving program for a computer," *Proc. of an International Conference on Information Processing*, UNESCO, Paris, France, pp. 256–264.

NII, H.P. (1986a): "Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures," *AI Magazine*, **7**, 2, 38–53.

NII, H.P. (1986b): "Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective," *AI Magazine*, **7**, 3, 82–106.

NII, H.P., E.A. FEIGENBAUM, J.J. ANTON and A.J. ROCKMORE (1982): "Signal-to-symbol transformation: HASP/SIAP case study," *AI Magazine*, **3**, 2, 23–35.

NIIDA, K. and T. UMEDA (1986): "Process control system synthesis by an expert system," in M. Morari and T.J. McAvoy (Eds.): *Chemical Process Control – CPCIII*, CACHE, Elsevier, Amsterdam.

NILSSON, N.J. (1982): *Principles of Artificial Intelligence*, Springer-Verlag, Berlin.

OSBORNE, R.L., A.J. GONZALEZ and C.A. WEEKS (1986): "First years experience with on-line generator diagnostics," *American Power Conference*, Chicago, IL.

OW, P.S. and S.F. SMITH (1986): "Toward an opportunistic scheduling system," *Proc. Nineteenth Hawaiian International Conference on System Sciences*, pp. 345–353.

PANG, G.K.H. and A.G.J. MCFARLANE (1987): *An Expert System Approach to Computer-Aided Design of Multivariable Systems*, Springer Verlag, Berlin.

PELAVIN, R. and J.F. ALLEN (1986): "A formal logic of plans in temporally rich domains," *Proceedings of the IEEE*, **74**, 10, 1364–1382.

PETERSON, J.L. (1981): *Petri net theory and the modelling of Systems*, Prentice-Hall.

PORTER, B., A.H. JONES, and C.B. MCKEOWN (1987): "Real-time expert tuners for PI controllers," *IEE Proceedings Part D*, **134**, 4, 260–263.

QUILLIAN, M.R. (1966): "Semantic memory," Report AFCRL-66-189, Bolt, Beranek & Newman, Cambridge, MA.

QUINLAN, J.R. (1979): "Induction over large databases," Report HPP-79-14, Stanford University, Stanford, CA.

RAHMAN, S. *et al* (1987): "An expert system based algorithm for short term load forecast," *Proc. of IEEE PES Winter Power Meeting.*

RASMUSSEN, J. (1986): *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, North-Holland, New York.

REITER, R. (1980): "A logic for default reasoning," *Artificial Intelligence*, **13**, 81–132.

RICH, E. (1983): *Artificial Intelligence*, McGraw-Hill, New York.

RICH, S.H. and V. VENKATASUBRAMANIAN (1987): "Failure-driven learning in expert systems for process fault diagnosis," *Proc. of AIChE Annual Meeting*, New York.

ROSENBLATT, F. (1961): *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington DC.

ROSENSCHEIN, J.S. and M.R. GENESERETH (1984): "Communication and cooperation," Tech. rep. HPP-84-5, Stanford Heuristic Programming Project, Stanford, CA.

SACERDOTI, E.D. (1974): "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence*, **5**, 115–135.

SACERDOTI, E.D. (1975): "A structure for plans and behavior," Tech. note 109, AI Center, SRI International Inc., Menlo Park, CA.

SACHS, P.A., A.M. PATERSON and M.H.M. TURNER (1986): "Escort – an expert system for complex operations in real time," *Expert Systems*, **3**, 1, 22–29.

SAKAGUSHI, T., H. TANAKA, K. UENISHI, T. GOTOH and Y. SEKINE (1987): "Prospects of expert system in power system operation," *Proc. of 9th Power Systems Computation Conference*, Cascais, Portugal.

SANDEWALL, E. (1972): "An approach to the frame problem, and its implementation," in B. Meltzer and D. Michie (Eds.): *Machine Intelligence*, Wiley, New York, pp. 195–204.

SANDEWALL, E. and R. RÖNNQUIST (1986): "A representation of action structures," *Proc. of the 5th National Conf. on Artificial Intelligence*, AAAI, Philadelphia.

SARIDIS, G.N. (1977): *Self-Organizing Control of Stochastic Systems*, Marcel Dekker, Inc., New York.

SARIDIS, G.N. (1983): "Intelligent robotic control," *IEEE Transactions on Automatic Control*, **28**, 5, 547–557.

SAROSH N. T. and L.V. LEAO (1986): "Toast: The power system operator assistant," *IEEE Computer*, **July**.

SCHINSKEY, F.G. (1986): "An expert system for the design of distillation controls," in M. Morari and T.J. McAvoy (Eds.): *Chemical Process Control – CPCIII*, CACHE, Elsevier, Amsterdam.

SHORTLIFFE, E.H. (1976): *Computer Based Medical Consultations: MYCIN*, Elsevier, New York.

SHUEY, D., D. BAILEY and T.P. MORRISSEY (1986): "PHIGS: A Standard, Dynamic, Interactive Graphics Interface," *IEEE Computer Graphics and Applications*, Vol. 6, No. 8, August 1986, pp. 50–57.

SIS (1985): "Datorgrafik—PHIGS, Programmers Hierarchical Interactive Graphics Standard," Technical report no. 306, SIS—Standardiseringskommisionen i Sverige.

SMITH, S.F., M.S. FOX and P.S. OW (1986): "Construction and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems," *AI Magazine*, **7**, 4, 45–61.

STEELE JR., G.L. (1984): *Common Lisp*, Digital Press, Digital Equipment Corp..

STEFIK, M. and D.G. BOBROW (1986): "Object-oriented programming: Themes and variations," *AI Magazine*, **6**, 4, 40–62.

STOLFO, S.J. and D.P. MIRANKER (1986): "The DADO production system machine," *Journal of Parallel and Distributed Computing*, **3**, 2, 269–296.

STROUSTRUP, B. (1986): *The C++ Programming Language*, Addison-Wesley, Reading, MA.

TRANKLE, T.L., P. SHEU and U.H. RABIN (1986): "Expert system architecture for control systems design," *Proc. ACC*, Seattle, WA, pp. 1163–1169.

TURNER, R. (1984): *Logics for Artificial Intelligence*, Ellis Horwood Limited, Chichester, England.

UENISHI, K. *et al* (1987): "Maintenance scheduling of electric power systems," *Proc. of the IEEE/UNIPEDE Workshop on Expert Systems.*

VAN MELLE, W., A.C. SCOTT, J.S. BENNETT and M. PEAIRS (1981): "The EMYCIN manual," Technical report HPP-81-16, Computer Science Department, Stanford University, California.

VEGDAHL, S.R. (1987): "Architectures that support functional programming languages," in V.M. Milutinovic (Ed.): *Advanced Topics in Computer Architecture*, Elsevier Science, New York.

VENKATASUBRAMANIAN, V. and S. H. RICH (1987): "Integrating heuristic and deep-level knowledge in expert systems for process fault diagnosis," *Proc. of NSF-AAAI Workshop on Artificial Intelligence in Process Engineering*, Columbia University, New York, NY.

VERE, S.A. (1983): "Planning in time: Windows and durations for activities and goals," *IEEE Transactions on Pattern analysis amd Machine intelligence*, **5**, 3, 246–267.

VICENTE, K.J. and J. RASMUSSEN (1987): "The cognitive architecture of decision support systems for industrial process control," *Proc. of First European Meeting on Cognitive Science Approaches to Process Control*, Marcoussis, France, pp. 1–16.

WALTZ, D.L. (1987): "Applications of the connection machine," *IEEE Computer*, **20**, 1, 85–97.

WEISS, S.M. and C. A. KULIKOWSKI (1981): "Expert consultation systems: The EXPERT and CASNET projects," *Machine Intelligence*, Infotech State of the Art Report, Pergamon Infotech Ltd., Maidenhead Berks, U.K..

WIDROW, B. (1962): "Generalization and information storage in networks of Adaline neurons," in M.C. Yovits, G.T. Jacobi and G.D. Goldstein (Eds.): *Self-Organizing Systems 1962*, Spartan Books, Washington DC.

WILKINS, D. (1984): "Domain independent planning: Representation and plan generation," *Artificial Intelligence*, **22**, 269–301.

WILLIAMSON, G.I., J.W. BUTLER, S.G. KING and J. BIGHAM (1987): "Using KBS in telecommunications 2," KRITIC Technical Memo Mo. 12.

WINSTON, P.H. and B.K.P. HORN (1984): *Lisp*, Addison-Wesley, Reading, MA.

ZADEH, L.A. (1965): "Fuzzy sets," *Inform. Control*, **8**, 338–353.