DOWNLOAD PDF EXPERT SYSTEMS IN PROCESS CONTROL

Chapter 1 : Process Control | Mining | SGS

At the Process Systems Engineering symposium (Love, Expert systems in process control Charge 50kg of Juice-A into Kettle-1 at 1 kg/min Juice A IP V,x SOP AK 19 19 P Fig. 5. Process diagram for example of automatic software generation (Love,).

It would match R1 and assert Mortal Socrates into the knowledge base. Backward chaining is a bit less straight forward. In backward chaining the system looks at possible conclusions and works backward to see if they might be true. So if the system was trying to determine if Mortal Socrates is true it would find R1 and query the knowledge base to see if Man Socrates is true. One of the early innovations of expert systems shells was to integrate inference engines with a user interface. This could be especially powerful with backward chaining. So in this example, it could use R1 to ask the user if Socrates was a Man and then use that new information accordingly. The use of rules to explicitly represent knowledge also enabled explanation abilities. In the simple example above if the system had used R1 to assert that Socrates was Mortal and a user wished to understand why Socrates was mortal they could query the system and the system would look back at the rules which fired to cause the assertion and present those rules to the user as an explanation. In English if the user asked "Why is Socrates Mortal? A significant area for research was the generation of explanations from the knowledge base in natural English rather than simply by showing the more formal but less intuitive rules. These systems record the dependencies in a knowledge-base so that when facts are altered, dependent knowledge can be altered accordingly. For example, if the system learns that Socrates is no longer known to be a man it will revoke the assertion that Socrates is mortal. In this, the knowledge base can be divided up into many possible views, a. This allows the inference engine to explore multiple possibilities in parallel. For example, the system may want to explore the consequences of both assertions, what will be true if Socrates is a Man and what will be true if he is not? One of the first extensions of simply using rules to represent knowledge was also to associate a probability with each rule. So, not to assert that Socrates is mortal, but to assert Socrates may be mortal with some probability value. Simple probabilities were extended in some systems with sophisticated mechanisms for uncertain reasoning and combination of probabilities. With the addition of object classes to the knowledge base, a new type of reasoning was possible. Along with reasoning simply about object values, the system could also reason about object structures. In this simple example, Man can represent an object class and R1 can be redefined as a rule that defines the class of all men. These types of special purpose inference engines are termed classifiers. Although they were not highly used in expert systems, classifiers are very powerful for unstructured volatile domains, and are a key technology for the Internet and the emerging Semantic Web. With an expert system the goal was to specify the rules in a format that was intuitive and easily understood, reviewed, and even edited by domain experts rather than IT experts. The benefits of this explicit knowledge representation were rapid development and ease of maintenance. Ease of maintenance is the most obvious benefit. This was achieved in two ways. First, by removing the need to write conventional code, many of the normal problems that can be caused by even small changes to a system could be avoided with expert systems. Essentially, the logical flow of the program at least at the highest level was simply a given for the system, simply invoke the inference engine. This also was a reason for the second benefit: With an expert system shell it was possible to enter a few rules and have a prototype developed in days rather than the months or year typically associated with complex IT projects. A claim for expert system shells that was often made was that they removed the need for trained programmers and that experts could develop systems themselves. In reality, this was seldom if ever true. While the rules for an expert system were more comprehensible than typical computer code, they still had a formal syntax where a misplaced comma or other character could cause havoc as with any other computer language. Also, as expert systems moved from prototypes in the lab to deployment in the business world, issues of integration and maintenance became far more critical. Inevitably demands to integrate with, and take advantage of, large legacy databases and systems arose. To accomplish this, integration required the same skills as any other type of system. Obtaining the time of domain experts for any software application is always difficult, but for expert systems it was especially

difficult because the experts were by definition highly valued and in constant demand by the organization. As a result of this problem, a great deal of research in the later years of expert systems was focused on tools for knowledge acquisition, to help automate the process of designing, debugging, and maintaining rules defined by experts. However, when looking at the life-cycle of expert systems in actual use, other problems â€" essentially the same problems as those of any other large system â€" seem at least as critical as knowledge acquisition: This provided a powerful development environment, but with the drawback that it was virtually impossible to match the efficiency of the fastest compiled languages such as C. System and database integration were difficult for early expert systems because the tools were mostly in languages and platforms that were neither familiar to nor welcome in most corporate IT environments â€" programming languages such as Lisp and Prolog, and hardware platforms such as Lisp machines and personal computers. As a result, much effort in the later stages of expert system tool development was focused on integrating with legacy environments such as COBOL and large database systems, and on porting to more standard platforms. These issues were resolved mainly by the client-server paradigm shift, as PCs were gradually accepted in the IT environment as a legitimate platform for serious business system development and as affordable minicomputer servers provided the processing power needed for AI applications. The example applications were not in the original Hayes-Roth table, and some of them arose well afterward. Any application that is not footnoted is described in the Hayes-Roth book.

Chapter 2 : Expert system - Wikipedia

An expert system is a computer software technology developed from artificial intelligence research. It may be used for intelligent manufacturing process control and, when properly designed, has the capability to imitate human behavior. An expert system's value is to assist a human in executing.

MET is also able to perform complex statistical analysis, create dynamic models and efficiently perform time-consuming expert tasks in real time. The MET platform can perform these decision-making processes by interfacing with any control system or third party software. Whether it is capturing the best operating practices from experts or using third party software as part of a control strategy, the MET platform is an extremely robust inference engine. The two components of an expert system are the software package and the integration of the solution within your operation. The software provides the foundation and the tools for constructing an effective, robust and reliable expert system. It directly impacts on development time and maintenance. The greatest impact on the success of implementation is the integration within the plant and the culture of the operation. In-depth interviews and on-site commissioning ensure that your entire team drives the development and tuning of the expert system. When we leave your site, the team will be familiar and comfortable with the system and your operators will anticipate the changes invoked by the expert system. Full acceptance and utilization translates into improved performance for your circuit. The SGS advanced systems group can help you take advantage of the proven benefits of advanced control technology to improve productivity. To increase the level of use and acceptance of your expert system, a user-friendly product is needed that minimizes the learning time required to provide a powerful, open and extendible platform for the site administrators. The product must provide tools to the site administrators for developing, maintaining and rapidly integrating expert systems. The product we have developed is known as the MET Toolkit. It combines numerous features that enable maintenance and development. The MET Toolkit provides powerful functionality for the site administrator, but ultimately provides significant value to our coders as well. Accelerate the development and troubleshooting of new logic. Facilitate the gathering of specific user-defined information to assess the performance of process unit operations. SGS expert systems are always on duty, optimizing your output on every shift. They provide a best practice repository that stays with the system, even when the staff and management change. MET explains the control decisions, helping operators gain a deeper understanding of the process while on the job, empowering the operations group to critique the system while providing a powerful training and reinforcement tool for the less experienced team members. Partner with SGS today to gain a competitive advantage through our advanced process control.

Chapter 3 : FLSmidth - Advanced process control

A General Expert System Architecture for Process Control 7. More on supervisory expert control 8. An example of supervisory expert control Expert Control Systems.

Operator impact Rule-based comparisons Applying fuzzy logic to control the reactor using only the three existing process measurementsâ€”output flow, composition, and temperatureâ€”imposes a severe performance limit on the system. Without a mathematical derivative capability in the rule syntax the system can react to the current values of the measurements, but not to how fast they are changing. However, derivative action is very helpful in controlling variables that respond with a dominant capacity lag. For this application, product temperature is a lag dominant variable. For this reason, the design needs a fourth controlled variableâ€”incremental temperature change. This allows the design to include logic that reacts more strongly when the temperature is changing than when it is steady. The setpoint for this variable is 0, meaning that the system should seek to keep temperature constant at its absolute setpoint. Reactor controls use three subsets for the controlled variables, which are applied to the measurement error to accommodate a variable set -point. The controlled variable error subsets are: The design uses five subsets for the output variables, to accommodate the number of combinations of the 4 controlled variables. The subsets that describe the manipulated variable changes are: The design also includes logic to decouple product flow and composition. For example, if both product composition and product flow are high, then the logic should force a decrease in both ingredient flows. Table 1 shows the rules for changes in the flows of ingredients A and B to control product flow and composition. In this context, positive error means measurement is higher than setpoint. Matrix intersections define the logic of the rules. There are a total of nine combinations. The rule for each combination forces two control actions. Table 2 shows a similar rule set for controlling product temperature and its incremental change. For example, when temperature error is positive PS , but temperature change is negative NS , then steam flow should not change ZE. Again, there are nine combinations, but each has only one control action. Since steam flow does not affect either product flow or composition, this part of the logic does not require any decoupling responses to change the ingredient flows. Fuzzy logic control performance The trend graphic shows the response of the fuzzy controls to the same changes in production rate and product composition as were used for the previous control applications. Certainly, this fuzzy logic application provides adequate control. Table 3 shows that the performance of this fuzzy logic controller is worse than either form of regulatory control, with one exception. The fuzzy logic provided superior control of product composition during a production rate change, showing an index of 0. This is 5 times better than advanced regulatory control. This is because the requirement for maintaining composition is so straightforward. The two ingredient flow loops have identical dynamics, and the reactor is a pure delay. As long as the logic changes the ingredient flows simultaneously and in the proper proportion, product composition will remain constant. By every other measure, performance of this fuzzy control system is relatively poor. Since the logical design lacks any counterpart to feedforward for temperature control, the temperature performance index for production rate changes is 1. Because the design does not include a composition change variable, the logic could not provide any equivalent to a derivative function for composition control. Consequently, the index for composition setpoint changes was worse than for basic regulatory control, 2. Since the temperature control logic is essentially the same as simple feedback control and the temperature change variable allows a derivative-like response, the temperature index for composition setpoint changes is close to that for basic regulatory control, and much worse than advanced regulatory control. A much simpler fuzzy logic controller could have been applied. If the product flow and composition control logic had not included decoupling actions, the overall solution would have been functionally equivalent to single PID loops without derivative function, and the performance would have been worse in all aspects. Likewise, a more complicated fuzzy logic controller could have been applied. Feedforward logic could have been included in the temperature control rule set by adding variables for changes in the ingredient flows. But with the addition of every new variable into the design, the number of combinations and rules increases exponentially. Further, there is no way in fuzzy logic to provide

the equivalent of dynamic compensation. Changing the ingredient flows does not immediately affect product temperature. For proper compensation, the logic would have to be capable of delayed actions, which would require creating timers and signal queues. The solution would have to be much more complex. The fuzzy logic controller provided superior control of product composition during a production rate change. Expert systems Fuzzy logic is a well-defined and mature technology. Its success depends on the quality of the logic implemented in the rule set s. In contrast to fuzzy logic, there is no precise definition for expert system technology. Human beings are smarter than any computer system. They can integrate a wide range of dynamic and steady-state information that may not be available to a control system into control action decisions. Furthermore, not all experts are created equal. Not all experts can clearly explain what they know. They will often disagree, or have varying degrees of correct understanding. Because there is no precise definition of the technology of an expert system, it is impossible to quantify what its performance can or will be. Operator impact A rule-based system uses a set of concepts and tools that will probably be unfamiliar to process operators. Depending on the effort put into its human interface, such a system will appear more or less as a black box. Much depends on whether the rule-based system is used in closed-loop control, or functions in an off-line advisory mode. They will have little or no understanding of how it works, how to adjust its behavior, or what to do if they disagree with its decisions, other than simply ignoring or disabling it. This makes it even more important to involve the operators in the development of the system. Final assessment The objective of a rule-based system is to achieve control through a set of rules that imitates the analysis and decision-making process of an experienced human operator. But there is very little that is standard about how humans think and make decisions. For this reason, rule-based systems are very individual and unique solutions with a variety of complexity and scope. This is an advantage and a disadvantage. Rule-based systems can be more flexible and creative than other control technologies. They provide a convenient way to introduce non-mathematical considerations into a control solution, and can include inputs that are difficult or impossible for other technologies to consider. Rule-based control systems can be the best solution for mixed systems where control logic must take several kinds of conditions into account, and any system is better than fully manual control. But this flexibility becomes a liability when it is not needed. Rule-based systems are a poor replacement for even simple PID loops, and they can quickly become too complicated when more complex control structures are needed. There is no simple way to deal either with interactions among process variables or process dynamics, because the logic of the rules generally evolves from steady state responses. As the number of rules increase, either in the original design or through modification over time, there is a strong potential for introducing unrecognized conflicts into the overall logic. Until they are debugged, this can easily degrade production operations. Such a conflict may not be immediately evident and its consequence can appear unexpectedly. Further, a fuzzy logic or expert system applied for closed-loop control still has to be tuned, and it is just as vulnerable to the problems created by varying process gains as any other system. A system under rule-based control can still oscillate, and there are no widely accepted procedures for tuning these systems. Often, the designer is the only one who understands the rule parameters well enough to tune them. Unless the rule structure is fairly simple, their performance is likely to degrade when the designer is no longer available to maintain the system. Generally speaking, the performance of rule-based systems for typical process control problems is not as good as mathematical algorithms, which are more standard, efficient, and powerful. The next installment in this series will introduce the concepts of model predictive control. Lew Gordon is a principal application engineer at Invensys; www.

Chapter 4 : Advanced Systems | Mining | SGS

The Application of Expert Systems to Process Control Bevan P.F. Wu An expert system is a computer software technology developed from artificial intelliÂ- gence research.

It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine. Of course, the term intelligence covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of those. But most progress to date in AI has been made in the area of problem solving -- concepts and methods for building programs that reason about problems rather than calculate a solution. AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called knowledge-based or expert systems. Often, the term expert systems is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often than not, the two terms, expert systems ES and knowledge-based systems KBS , are used synonymously. Taken together, they represent the most widespread type of AI application. The area of human intellectual endeavor to be captured in an expert system is called the task domain. Task refers to some goal-oriented, problem-solving activity. Domain refers to the area within which the task is being performed. Typical tasks are diagnosis, planning, scheduling, configuration and design. An example of a task domain is aircraft crew scheduling, discussed in Chapter 2. Building an expert system is known as knowledge engineering and its practitioners are called knowledge engineers. The knowledge engineer must make sure that the computer has all the knowledge needed to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as symbol patterns in the memory of the computer -- that is, he or she must choose a knowledge representation. He must also ensure that the computer can use the knowledge efficiently by selecting from a handful of reasoning methods. The practice of knowledge engineering is described later. We first describe the components of expert systems. The knowledge base of expert systems contains both factual and heuristic knowledge. Factual knowledge is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field. Heuristic knowledge is the less rigorous, more experiential, more judgmental knowledge of performance. In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the "art of good guessing. One widely used representation is the production rule, or simply rule. The IF part lists a set of conditions in some logical combination. The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving action taken. Expert systems whose knowledge is represented in rule form are called rule-based systems. Another widely used representation, called the unit also known as frame, schema, or list structure is based upon a more passive view of knowledge. The unit is an assemblage of associated symbolic knowledge about an entity to be represented. Typically, a unit consists of a list of properties of the entity and associated values for those properties. Since every task domain consists of many entities that stand in various relations, the properties can also be used to specify relations, and the values of these properties are the names of other units that are linked according to the relations. One unit can also represent knowledge that is a "special case" of another unit, or some units can be "parts of" another unit. The problem-solving model, or paradigm, organizes and controls the steps taken to solve the problem. If the chaining starts from a set of conditions and moves toward some conclusion, the method is called forward chaining. If the conclusion is known for example, a goal to be achieved but the path to that conclusion is not known, then reasoning backwards is called for, and the method is backward chaining. These problem-solving methods are built into program modules called inference engines or inference procedures that manipulate and use knowledge in the knowledge base to form a line of reasoning. The knowledge base an expert uses is what he learned at school, from colleagues, and from years of

experience. Presumably the more experience he has, the larger his store of knowledge. Knowledge allows him to interpret the information in his databases to advantage in diagnosis, design, and analysis. Though an expert system consists primarily of a knowledge base and an inference engine, a couple of other features are worth mentioning: Knowledge is almost always incomplete and uncertain. To deal with uncertain knowledge, a rule may have associated with it a confidence factor or a weight. The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called reasoning with uncertainty. An important subclass of methods for reasoning with uncertainty is called "fuzzy logic," and the systems that use them are known as "fuzzy systems. When an answer to a problem is questionable, we tend to want to know the rationale. If the rationale seems plausible, we tend to believe the answer. So it is with expert systems. Most expert systems have the ability to answer questions of the form: The most important ingredient in any expert system is knowledge. The power of expert systems resides in the specific, high-quality knowledge they contain about task domains. AI researchers will continue to explore and add to the current repertoire of knowledge representation and reasoning methods. But in knowledge resides the power. Because of the importance of knowledge in expert systems and because the current knowledge acquisition method is slow and tedious, much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure. Knowledge engineering is the art of designing and building expert systems, and knowledge engineers are its practitioners. Weinberg said of programming in The Psychology of Programming: Knowledge engineering is the same, perhaps more so. We stated earlier that knowledge engineering is an applied part of the science of artificial intelligence which, in turn, is a part of computer science. Theoretically, then, a knowledge engineer is a computer scientist who knows how to design and implement programs that incorporate artificial intelligence techniques. The nature of knowledge engineering is changing, however, and a new breed of knowledge engineers is emerging. Today there are two ways to build an expert system. They can be built from scratch, or built using a piece of development software known as a "tool" or a "shell. Though different styles and methods of knowledge engineering exist, the basic approach is the same: The engineer then translates the knowledge into a computer-usable language, and designs an inference engine, a reasoning structure, that uses the knowledge appropriately. He also determines how to integrate the use of uncertain knowledge in the reasoning process, and what kinds of explanation would be useful to the end user. Next, the inference engine and facilities for representing knowledge and for explaining are programmed, and the domain knowledge is entered into the program piece by piece. It may be that the inference engine is not just right; the form of knowledge representation is awkward for the kind of knowledge needed for the task; and the expert might decide the pieces of knowledge are wrong. All these are discovered and modified as the expert system gradually gains competence. The discovery and cumulation of techniques of machine reasoning and knowledge representation is generally the work of artificial intelligence research. The discovery and cumulation of knowledge of a task domain is the province of domain experts. Domain knowledge consists of both formal, textbook knowledge, and experiential knowledge -- the expertise of the experts. Tools, Shells, and Skeletons Compared to the wide variation in domain knowledge, only a small number of AI methods are known that are useful in expert systems. That is, currently there are only a handful of ways in which to represent knowledge, or to make inferences, or to generate explanations. Thus, systems can be built that contain these useful methods without any domain-specific knowledge. Such systems are known as skeletal systems, shells, or simply AI tools. Building expert systems by using shells offers significant advantages. A system can be built to perform a unique task by entering into a shell all the necessary knowledge about a task domain. The inference engine that applies the knowledge to the task at hand is built into the shell. If the program is not very complicated and if an expert has had some training in the use of a shell, the expert can enter the knowledge himself. Many commercial shells are available today, ranging in size from shells on PCs, to shells on workstations, to shells on large mainframe computers. They range in price from hundreds to tens of thousands of dollars, and range in complexity from simple, forward-chained, rule-based systems requiring two days of training to those so complex that only highly trained knowledge engineers can use them to advantage. They range from general-purpose shells to shells custom-tailored to a class of tasks, such as financial planning or real-time process control. Knowledge acquisition refers to the task

of endowing expert systems with knowledge, a task currently performed by knowledge engineers. The power of an expert system lies in its store of knowledge about the task domain -- the more knowledge a system is given, the more competent it becomes. Bricks and Mortar The fundamental working hypothesis of AI is that intelligent behavior can be precisely described as symbol manipulation and can be modeled with the symbol processing capabilities of the computer. In the late s, special programming languages were invented that facilitate symbol manipulation. In the early s another AI programming language was invented in France. Here is an inference rule: A variety of logic-based programming languages have since arisen, and the term prolog has become generic.

DOWNLOAD PDF EXPERT SYSTEMS IN PROCESS CONTROL

Chapter 5 : Advanced Process Control: Fuzzy Logic and Expert Systems | Control Engineering

We use cookies to give you the best possible experience on ResearchGate. Read our.

Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge. The data is collection of facts. The information is organized as data and facts about the task domain. Data, information, and past experience combined together are termed as knowledge. Components of Knowledge Base The knowledge base of an ES is a store of both, factual and heuristic knowledge. Knowledge representation It is the method used to organize and formalize the knowledge in the knowledge base. Knowledge Acquisition The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledge base. The knowledge base is formed by readings from various experts, scholars, and the Knowledge Engineers. The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills. He acquires information from subject expert by recording, interviewing, and observing him at work, etc. The knowledge engineer also monitors the development of the ES. Inference Engine Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution. In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution. Adds new knowledge into the knowledge base if required. Resolves rules conflict when multiple rules are applicable to a particular case. It considers all the facts and rules, and sorts them before concluding to a solution. This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence. It explains how the ES has arrived at a particular recommendation. Verbal narrations in natural language. Listing of rule numbers displayed on the screen. The user interface makes it easy to trace the credibility of the deductions. It should make efficient use of user input. Expert Systems Limitations No technology can offer easy and complete solution. Large systems are costly, require significant development time, and computer resources.

Chapter 6 : Expert System ES IIIâ„¢ | Glass Service

*Expert Systems in Process Control [F.L. Jovic] on calendrierdelascience.com *FREE* shipping on qualifying offers. Advances in artificial intelligence, smart process transmitters and positioners allied with the use of computers in process control has led to an increase in application of expert systems.*

Chapter 7 : Chp 1: Expert Systems And Artificial Intelligence

First attempts at using expert systems for process control involved taking a "snap-shot" of plant data and using a static expert system paradigm to perform inference.

Chapter 8 : Artificial Intelligence Expert Systems

Our authors and editors. We are a community of more than , authors and editors from 3, institutions spanning countries, including Nobel Prize winners and some of the world's most-cited researchers.