The Knowledge Engineering Review

Experience using S.1: an expert system for newspaper printing press configuration

M.S. LAN*, R.M. PANOS* and M.S. BALBAN**

* *Rockwell International Science Center, Thousand Oaks, CA 91360, USA*
** *Rockwell International Graphic Systems Division, Chicago, IL 60605, USA*

Abstract

This paper describes some of our experience gathered during the development of an expert system, the press lineup advisor, in which we used the commercially available expert system development tool, S.1.™ Discussion includes: (1) how we used S.1 to develop a system which solves a configuration problem; (2) difficulties we encountered when applying S.1 to this specific reasoning problem; (3) limitations of S.1 from both problem-solving and operational points of view and (4) issues remaining to be solved with respect to generalization of the system.

1 Introduction

In the same issue of this journal, Ward and Sleeman (1988) present a good discussion of the S.1 environment and how its tool set can be utilized by both experienced programmers and novices. Their discussion emphasizes issues related to syntax, semantics, editors, and error messages. Here, we will present some discussion regarding our general system development strategy, the implementation of a limited non-monotonic, hypothetical reasoning problem, and the extensibility of our knowledge base into a broader problem domain than we originally intended.

The press lineup advisor was designed to assist newspaper production operators in determining an appropriate location for each printing plate on a multi-web, double-width newspaper printing press. The composition of a "lineup", or an "imposition", is one of the major tasks that must be performed by a newspaper publisher when preparing its printing press to produce new newspaper products. This task is usually performed by an expert, someone with a considerable amount of experience regarding the press, its operation, and relevant policies of the organization. The essence of this task is to arrange the many image plates for the printing run onto the many locations on the press where images are transferred to paper so as to produce a newspaper product having the correct page/section sequences.

The lineup design process relies heavily on the expert's heuristics. Therefore we believe that a knowledge-based approach is very well-suited to develop a system that can perform this task. Moreover, this approach has been successfully used to solve configuration problems in the past. The best known example is the R1 system (McDermott, 1982) that configures computer systems.

We developed the system in collaboration with lineup experts at the Los Angeles Times. The press lineup advisor therefore contained a knowledge base specific to the Times. It assumed press arrangement at the Times, and encoded their operational policies and knowledge of these experts for designing lineups. The successful implementation of such a prototype system was first reported by Lan, Panos and Balban (1987). Since then, we have devoted our efforts toward increasing the depth and the breadth of the system, increasing its flexibility, and devising methods to make the system become more general and adaptable.

In the following discussion, section 2 presents the problem domain of press lineup including descriptions of printing presses and operational modes. Section 3 discusses the development strategy

S.1 is a trademark of Teknowledge, Inc., Palo Alto, CA, USA

including the knowledge acquisition process, the S.1 expert system development tool, knowledge representation, problem-solving methods and system performance. Section 4 discusses limitations of S.1 from problem-solving and operational points of view. Section 5 discusses issues that we are currently facing to generalize the system further.

2 The problem domain of press lineup

A newspaper printing press is composed of a number of individual printing units usually arranged in a row. Each unit prints onto both sides of a single, continuous sheet of paper (referred to as the "web") which is fed to each unit from a single large roll mounted below. The webs from several units are gathered above the press and directed toward the folder where the collected webs are cut and folded into finished newspaper. Figure 1 illustrates a newspaper production system with two printing units behind the folder and three printing units in front of the folder. In newspaper printing, there are two different modes of operation, i.e. straight run and collect run.

The reasoning process through which an expert will design a lineup is highly dependent on two sets of constraints that will place bounds on the number of lineups that can possibly satisfy a given press run specification. One set of constraints cannot be violated. These hard constraints arise from the design of the press itself. A press has a maximum number of units which are arranged in a fixed pattern relative to the folder, the number of printing units in front of and behind the folder, the number of printing units capable of printing coloured pages, the location of these colour printing units on a press and the colour positions on a given colour printing unit. Paper webs cannot pass through each other and are gathered in a specific fixed order in the folder.

Another set of constraints arises from considerations of productivity. These soft constraints are usually not violated, but they can be. They involve policy, or guidelines, learned by a particular publisher over long periods of operation. Some of the soft constraints that bound the number of possible lineups involve issues such as maintaining balance between the two sides of the press, directing the smallest sections to the upper formers, minimizing the number of units used, and minimizing the number of "angle bars" used. Most of these soft constraints are guidelines learned by expert operators to minimize undesirable occurrences such as web breaks and to provide a more efficient operation.

No general algorithm is available for configuring a press. The lineup is a diverse function of factors, such as the number of newspaper sections being printed in the particular press run, the number of pages for each section, colour page requirements and, of course, the complement of operational guidelines. There is no unique solution to the lineup design; for a given product specification, there may be (a) *no* possible lineup, (b) only *one* possible lineup, or (c) *more than one*

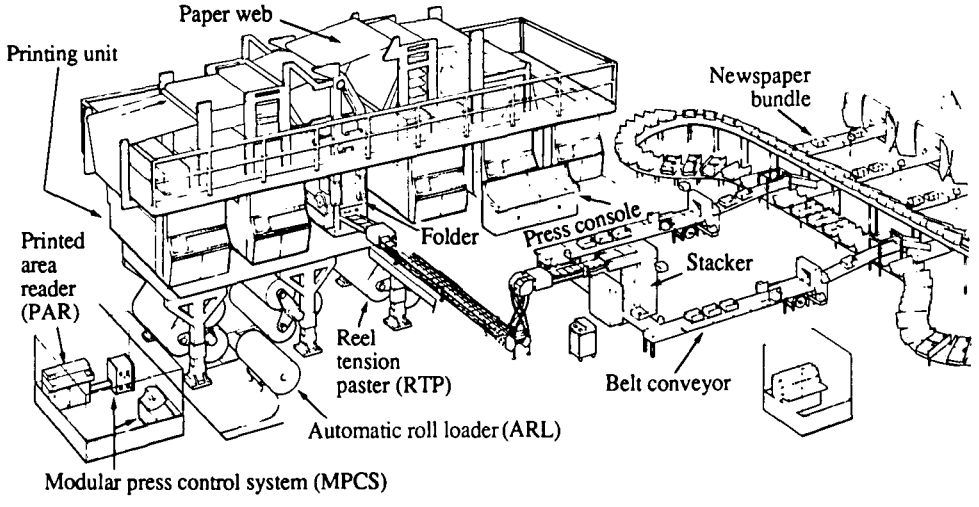

Figure 1 Newspaper production system with two printing units behind the folder and three printing units in front of folder

possible lineup. Therefore, the reasonable and realistic approach to a configuration task of this type is to "construct" an appropriate lineup, instead of, for instance, selecting one from a lookup table.

3 System development strategy

We believe that it was critical to the success of the whole project to carefully choose an appropriate subarea of the whole problem domain around which to build the prototype system. It was important that this subarea contain all of the essential characteristics of the broader problem domain. An appropriate subarea would illustrate the basic problem solving method, could be implemented with available resources within a relatively short period and would also provide a good platform from which to expand the system into the much broader problem domain. For the press lineup composition problem, we determined that this initial system should be capable of devising lineups for printing a newspaper product of up to four sections, having only black/white pages, on a twelve-unit press operating in a straight mode. This description fitted a large fraction of the lineups that were commonly devised in practice.

Our system was built using the S.1 expert system development tool in a XEROX* Interlisp-D environment. Since S.1 provides a sufficiently powerful set of data constructs, control mechanisms, and language vocabulary, we were able to concentrate more on problem-solving issues than on programming issues. Thus in a very short period, we were able to build a small system which covered a limited portion of the problem domain. Then we expanded the system incrementally, checking its problem-solving ability and integrity along the way.

The development of this system more or less followed the typical course of building an expert system as described by Hayes-Roth, Waterman and Lenat (1983). It consisted of three main phases. The first phase involved identifying and conceptualizing the problem. The second phase involved the formalization, implementation, and testing of an appropriate architecture for the system, including constant reformulation of concepts, redesign of representations, and refinement of the implemented system. The third phase was devoted to broadening the breadth of the system's knowledge base. The system has been tested and used at the Los Angeles Times since October, 1986.

In the rest of this section, we describe the realization of the system.

3.a Knowledge acquisition

Our knowledge acquisition activities took somewhat different forms in each of the three phases of the project.

In the preliminary phase, our activity was exploratory. We were searching for the essential character of the problem and the problem solving methods used by human experts. During this preliminary knowledge acquisition effort, our efforts consisted almost entirely of unstructured interviews, since we were still too ignorant to set up appropriately structured interviews or to properly utilize available cases. It was very useful at this stage to have several different people play knowledge engineering roles. We found that having multiple perspectives available was particularly helpful to interpret our experts' information and to deepen our own understanding of the problem domain.

During the second phase of the project, we narrowed the scope of our knowledge engineering activity to less than a two-man rate of effort. The additional perspective provided by more than a single knowledge engineer remained very useful. By this time, we were able to make more use of the large number of cases in the files. The second phase was to produce a prototype system and, our knowledge acquisiton efforts focused onto a particular class of cases.

The last phase of the initial project was devoted to developing the breadth of the knowledge base. Throughout most of this last phase, we found it most efficient to depend primarily upon a single knowledge engineer. By this time we had gained a very good understanding of the problem domain

* XEROX is a trademark of Xerox Corporation

and were able to set up structured interviews, use case files extensively, and construct our own unusual or difficult cases.

During this last phase of the project we depended more heavily on testing to insure the integrity of the knowledge base. The completed prototype was installed at the Los Angeles Times for continuous field evaluation. We found this stage to be very important to the whole development process. Conducting performance evaluations in the field allowed the system to encounter a variety of real world cases, which revealed numerous flaws in the knowledge base and promoted further system refinement. Furthermore, conducting field tests with domain experts not only enabled the development team to obtain feedback, but also allowed the domain experts to gain a better understanding of the expert system approach. Thus, the interviews that followed field tests were quite constructive, and enabled us to make further improvements in the depth and breadth of the domain knowledge in the system.

3.b S.1—An expert system development tool

We chose to develop the press lineup advisor using the S.1 expert system development tool. S.1 provides a knowledge representation scheme, an explicit control scheme, a backward chaining inference engine, an explanation facility, a user interface and an error handling scheme (see S.1 reference manual 1985). This range of power allowed us to concentrate on the problem-solving strategy of the expert system and rapidly build a working system.

Meanwhile, at the time that this project was conceived, only certain commercial expert system development tools were available—small tools such as M.1™ and larger tools such as S.1, KEE™ and ART™. The smaller tools were possibly not powerful enough and of the larger tools, S.1 seemed to require the least learning curve. Although S.1 is based on EMYCIN (van Melle, 1981) and was designed to aid in the development of expert systems that solve diagnostic problems or structured selection problems (Clancey, 1984), it is well suited for developing a consultative type of expert system.

3.c Knowledge representation

Knowledge representation in S.1 takes three principal forms: object-attribute-value (O-A-V) triplets, production rules and procedural control blocks. Declarative knowledge is expressed in the O-A-V form, while the reasoning process can be controlled explicitly through control blocks and implicitly through the backward-chaining inference mechanism. In other words, broad problem-solving strategies can be expressed in control blocks, while judgmental knowledge can be expressed in the form of rules.

In the press lineup advisor, 17 different classes are defined; they can be considered as five different groups for describing press, printing unit, press run, product and imposition, respectively. Classes for data related to the press include PRESS,BACK.UNIT,FRONT.UNIT,PREVIOUS.BACK.UNIT,PREVIOUS.FRONT.UNIT. A printing unit is further divided into four different locations along its cylinders. They are represented by NEAR.LOCATION,NEAR.CENTER.LOCATION,FAR.CENTER.LOCATION and FAR.LOCATION. PRESS.RUN represents a particular press run and the product as a whole. Details of parts of a product are represented by SECTION, SECTION.PART, BUFFER.SECTION, COLOUR.PAGE,COLLECT.COLOUR.PAGE and TABLOID.COLOUR.PAGE. IMPOSITION represents intermediate and final results of a lineup construction process.

A typical rule shown in figure 2 is used here to illustrate the rule construct in S.1. This rule states that the system can conclude that the attributes, *First,Colour.Printability.Check.For.Large* (a Boolean attribute) and *Large.Colour.Imposition.Technique* (a text attribute), which are defined on

M.1 is a trademark of Teknowledge, Inc., Palo Alto, CA, USA
KEE is a trademark of IntelliCorp, Mountain View, CA, USA
ART is a trademark of Inference Corp., Los Angeles, CA, USA

```
DEFINE RULE    LARGE.COLOUR.PRINTABLE.010
::APPLIED.TO     i:imposition
::PREMISE        already.existing(s1:section, s2:section |
                 section.number[s1] = off.side.front.section[i] and
                 section.number[s2] = main.side.front.section[i] and
                 off.side.front.has.colour[i] and
                 main.side.front.has.colour[i] and
                 for.all.existing(cp1:colour.page | part.of[s1,cp1])
                 page.number[cp1] = 1 or page.number[cp1] = last.page[s1] and
                 for.all.existing(cp2:colour.page | part.of[s2,cp2])
                 page.number[cp2] = 1 or page.number[cp2] = last.page[s2])
::CONCLUSION   begin
                 first.colour.printability.check.for.large[i];
                 large.colour.imposition.technique[i] = simple.scheme;
               end
::CATEGORIES     {large.colour.printable.rules}
```

Figure 2 A typical rule for coloured page requirement. Within it i, s1 and s2, cp1 and cp2 are instance variables for the class instances imposition, section and colour.page, respectively

the object IMPOSITION, are equal to "true" and to the value "simple.scheme," respectively, if it can establish within the working memory that two existing sections assigned to the *Off.Side.Front.Section* and *Main.Side.Front.Section* press locations of the IMPOSITION both have colour pages and that the page number of those coloured pages is either page number one or the last page of the respective section.

A part of a control block shown in figure 3 is used here to illustrate the control block construct in S.1. It states that the system will repeatedly invoke a second control block, *Generate.Imposition.For.Large.Colour*, to design a hypothetical lineup for a product with interior coloured pages and then invoke a third control block, *Test.Imposition*, to check whether or not it

```
/ ********** GENERATE AND TEST **********/

  repeat
      if already.existing(i:imposition | current.imposition(i))
          then
              begin
                  if already.existing(bu:back.unit, fu:front.unit | current.back.unit(bu)
                                                                   and current.front.unit(fu))
                      then
                          invoke generate.imposition.for.large.colour(pr,bu,fu,i,p);
                  invoke test.imposition(i);
                  if not printability.for.large.sections.based.on.lineup[i]
                      then
                          begin
                              invoke seek.alternative(i);
                              if evolution.technique[i]~ = none
                                  then
                                      invoke setup.for.new.imposition(i,p);
                          end;
  until already.existing(i:imposition | current.imposition(i) and
                              (printability.for.large.sections.based.on.lineup[i] or
                              evolution.technique[i] = none))

/ ********** END **********/
```

Figure 3 Portion of the control block in S.1 for the generate-and-test method. Invoke statements cause other control blocks to be executed

meets the requirements. This process will be repeated until either a correct lineup is designed or no technique can be found in the control block *Seek.Alternative* to derive a correct lineup from the current hypothetical lineup.

In addition to the main knowledge base implemented in S.1, several external functions were written in Lisp to display the generated lineup graphically. Functions which cannot be implemented in S.1 were also written in Lisp—for example, to allow the user to use the computer mouse to make modifications on a lineup devised by the system.

3.d Problem-solving method

The problem-solving strategy built into the press lineup advisor closely resembles that used by human experts. It follows an overall strategy of sequential refinement. A given problem is first examined at a coarse level of detail; then details are added in stages until either a conflict occurs or a complete lineup is formed. In more specific terms, the system will first consider the number and relative sizes of the edition's sections and logically place them at coarse locations on the press (e.g. offside, behind the folder), then consider the disposition of individual pages within sections, and only lastly consider the disposition of interior coloured pages.

Our realization of the problem-solving process generally fitted well into the S.1 development tool. The explicit control blocks served to implement the broad, sequential refinement strategy. Two problem-solving methods were used in the system: a rule-based backward reasoning method and a generate-and-test method. Most of the system, which seeks a solution for the detailed utilization of individual press units, was implemented in a rule-based backward reasoning method. The generate-and-test scheme was needed to solve the placement of colour pages. Figure 3 illustrates a portion of the generate-and-test scheme.

Figure 4 shows a diagram of the lineup generation procedure. After the system obtains the

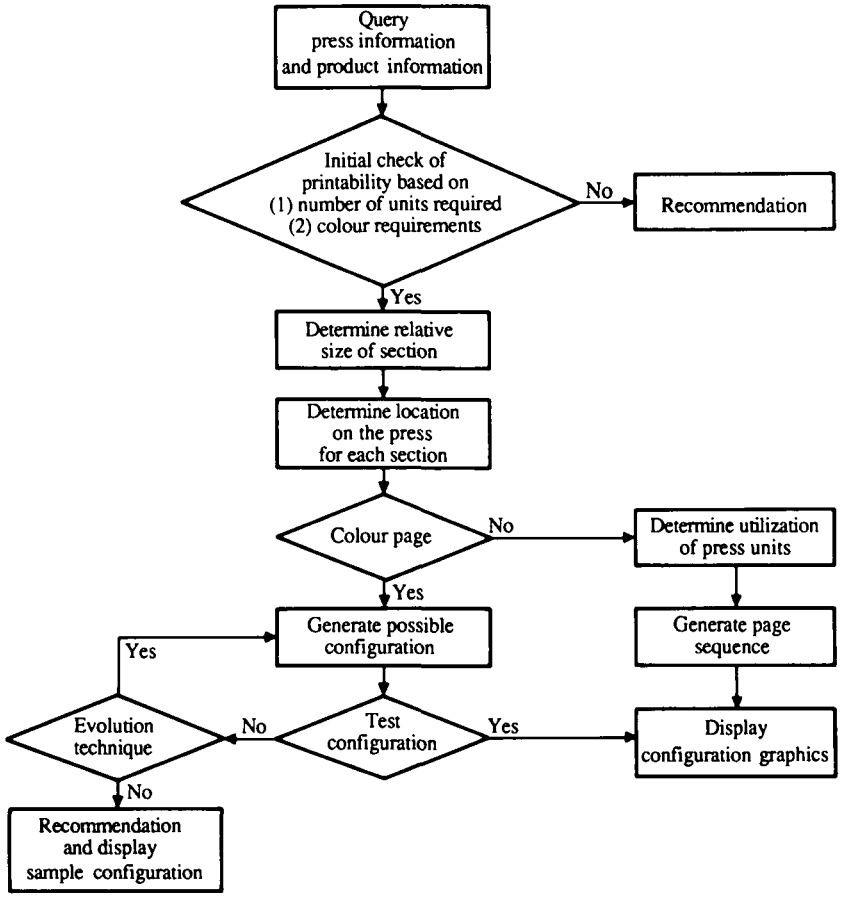

Figure 4 Diagram of the lineup generation procedure

necessary press and product information, it will first determine the coarse placement of sections on the press, for example, mainside or offside, and in front of or behind the folder. Depending upon the product, the system will use a "simple" scheme to design a lineup for black/white newspapers or invoke the "generate-and-test" scheme for newspapers with coloured pages. Within the block labelled "Generate Possible Configuration" of figure 4, the determination of press unit utilization and the generation of page sequences are also performed.

To generate a lineup, the system will first determine which portion of the press should be devoted to which section of the desired newspaper product. This allocation of press resources is based on the total number of sections, their order in the final product, and their relative sizes. Next, the system will determine the best utilization of individual printing units. *Utilization* is one of the important attributes defined on the class objects FRONT.UNIT and BACK.UNIT. *Utilization* can take on the values: "unused," "full," "main half," "off half," "main three quarters," "off three quarters," "full for 3 colour," "main three quarters for 3 colour," "off three quarters for 3 colour," "mainside up offside down," or "offside up mainside down." Once the coarse section locations have been determined and the *utilization* attribute values have been determined for each printing unit, the basic lineup task has been accomplished.

3.e System performance

Depending upon the complexity of product requirements, the system will take from less than one minute to about three minutes to design a lineup. Figure 5 shows a typical lineup for a four-section

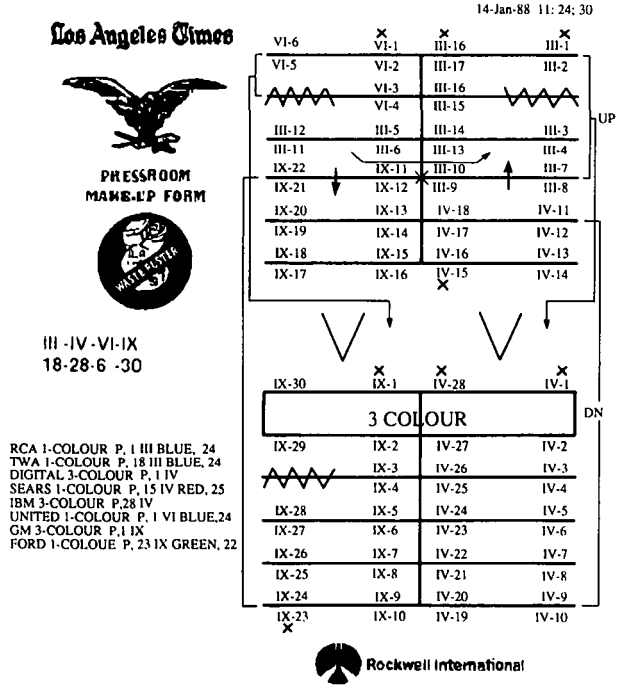

Figure 5 A lineup for a four-section newspaper with eight coloured pages printed on a twelve-unit double-width press in a straight mode. Number of pages for each section is 18, 28, 6, and 30, respectively. Each horizontal line represents one printing unit. "V" symbols at the center of the press represents formers; only two lower formers are shown in this diagram. Roman numerals and letter represent section names, while Arabic numerals represent page numbers. Arrow symbols across the center of the press represent angle bars; arrows point in the direction of barred webs. The ripple mark indicates that the marked portion of that printing unit is not used

newspaper with coloured pages printed on a twelve-unit double-width press in a straight mode.

4 Difficult reasoning issues

Although we have successfully implemented a working system in S.1, we did encounter some difficulties. Difficulties arose when we tried to add an ability to reason hypothetically. The need to reason hypothetically arises in the situation where we must construct a possible solution to a lineup problem from a failed attempt.

S.1 has been developed for monotonic reasoning activities. The generate-and-test method that we have implemented is really more of an "evolve-and-test" approach and therefore involves a basically non-monotonic reasoning process; in other words, it involves changing and negating portions of the existing dynamic data base.

In the specific context of our lineup composition activity, this hypothetical reasoning activity then requires that the system reconsider the use of each printing unit in a previous lineup. This action further requires that the system change the value assigned to some of the attributes that have already been determined.

S.1 presented us with several difficulties as we implemented this "evolve-and-test" process. Our principal difficulty arose from the fact that S.1 does not permit a change in the value of attributes, once determined. Less troublesome, but related difficulties arose from other S.1 characteristics, such as the inability to disable objects, once created, or to reset instance variables.

These characteristics of S.1 caused difficulty when we tried to maintain hypothetical presses. Since attribute values are unchangeable, we were forced to create whole new presses, which in turn, required that we take great care in accounting for the proliferating objects in the knowledge base. Our solution to these problems involved creating an object accounting method that carefully used the value of the instance variable and, separately, used values for attributes such as, *Unit.Name*. Control blocks were then necessary to copy over old attribute values that were unaffected by the immediate hypothetical considerations. We also had to add new attributes, such as *Unit.Forgotten*, to act as tags for the state of the system's experience with an object.

From an operational point of view, the unchangeable characteristic of determined attribute values can also result in being somewhat of an inconvenience to the user. During a consultation, the user supplies input data. Any mistake or need for change in the input data will require restarting the consultation. For the press lineup advisor, we have considered the need to develop an input buffer to provide more convenient input behaviour for the user.

5 Issues about generalization of the system

Since the initial prototype development, the system has been expanded through several evolutionary stages. The initial objective of this development project was to demonstrate the applicability of expert system technology to the newspaper printing industry. This objective was achieved so rapidly and so successfully that interest from a variety of newspaper publishers drove us to evaluate the possibility of generalizing the system to accomodate requirements that result from differences in press arrangement and in operational policies among various publishers.

Like other expert systems, the press lineup advisor is domain-specific. It was originally developed based on the printing presses of the Los Angeles Times and their operational policies. At the same time, there are many similarities with respect to performing the lineup task. Since the knowledge-based approach itself tends to promote easier code modification, it is possible to consider customizing the existing system for different publishers. Nevertheless, we do not believe that such customization is a good approach since difficulty will arise from maintaining and supporting different knowledge bases for different publishers. Also we can forsee that in the future, even for a particular publisher, it is very likely that the system will require modifications in order to incorporate new press arrangements and policy changes.

At this point in our system generalization efforts, we believe that there are two other options to explore; a top layer of system configuration code, set up to provide a developer with easily modified, site specific attribute values, or a layer of code that would instead, focus on the underlying causes of the variety of site specific policies and process constraints. The first solution seems to offer a practical solution to our generalization objective that we believe could be developed within the expressive power available in S.1. Within the context of the lineup domain, the second option amounts to adding a deep reasoning capability to the problem-solving in the system and would represent the most general solution to the lineup problem. However, the extra development effort may not provide an increase in generality commensurate with the increase in development efforts.

As a matter of fact, the questions about the generalization and the adaptability of an expert system apply to other similar types of systems as well. This consideration certainly deserves further research, and we are currently investigating these issues. It is clear that the development of the press lineup advisor has entered into a new stage that is beyond what has been described by Hayes-Roth, Waterman and Lenat, (1983) regarding the typical process of building expert systems.

6 Summary

We have described an expert system development project in which we used a commercially available tool to construct a solution to a configuration type of problem. Our development process largely followed a course typical of other documented expert system development projects.

The S.1 development tool performed well through most of the project and we have described how we encoded several problem-solving methods in S.1. We also briefly discussed some aspects of our system problem-solving behaviour where extraordinary care was necessary.

Finally, we mentioned our concerns regarding the technical difficulty that we expect to encounter as we try to extend the depth and the breadth of the knowledge base to cover a much broader problem domain that we had originally intended.

Acknowledgements

The authors gratefully acknowledge the assistance of the Los Angeles Times. This work was supported by the Graphic Systems Division of Rockwell International.

References

Clancey, W J, 1984. "Classification problem solving" Proceedings of the National Conference on Artificial Intelligence, pp. 49–55.

Hayes-Roth, F, Waterman, D A and Lenat, D B, (Eds.) 1983. *Building Expert Systems*, Reading, Massachusetts: Addison-Wesley.

Lan, M S, Panos, R M and Balban, M S, 1987. "A knowledge-based approach to printing press configuration" Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications, pp. 38–43.

McDermott, J, 1982. "R1: A rule-based configurer of computer" *Journal of Artificial Intelligence* **19**(1), pp. 39–88.

S.1 Reference Manual, (1985). Palo Alto, CA: Teknowledge, Inc.

van Melle, W, 1981. *System Aids in Constructing Consultation Programs*, Ann Arbor, MI:UMI Research Press.

Ward, R D and Sleeman, D, 1988. "Learning to use the S.1 knowledge engineering tool" *The Knowledge Engineering Review* **2**(4).