

Design Issues of a Knowledge-based Welding Advisor

Stephen D. Kleban

Sandia National Laboratories
Albuquerque, NM 87185
sdkleba@sandia.gov

Abstract

Expert system implementation can take numerous forms ranging from traditional declarative rule-based systems with if-then syntax to imperative programming languages that capture expertise in procedural code. The artificial intelligence community generally thinks of expert systems as rules or rule-bases and an inference engine to process the knowledge. The welding advisor developed at Sandia National Labs and described in this paper, deviates from this by codifying expertise using object representation and methods. Objects allow computer scientists to model the world as humans perceive it giving us a very natural way to encode expert knowledge. The design of the welding advisor, which generates and evaluates solutions, will be compared and contrasted to a traditional rule-based system.

Background

Welding in the US is a \$50B economy which generates upwards of \$7B of waste per year due to rework and scrap. Many of the problems associated with welding arise due to the high degree of "trial and error" that typically accompanies the development of weld schedules, weld joint designs, and material selection. SmartWeld (Mitchiner et al. 1996), which includes the welding advisor, developed at Sandia National Laboratories (SNL), is an attempt to decrease the amount of waste due to failed welds by providing expertise to both novice and expert welders as a check or first order recommendation. Different weld processes and joint designs have different properties, such as depth of metal penetration, heat input, residual stress, distortion, sensitivity to loading and sensitivity to corrosive environments. Choosing the weld process and joint geometry is the core problem.

Given weld requirements, the advisor suggests a welding process, a joint geometry, and a welding schedule. The advisor currently houses knowledge about ten welding processes and forty different joint geometries. It accesses a database of proven historical weld schedules, and selects a schedule based on a similarity measure, comprised of material type, joint thickness, joint geometry, and welding process. Once a weld schedule is selected, the ultimate

goal of this system is to translate this schedule into NC format to run the actual welding machinery.

Problem Solving Structure

The intent of a welding advisor is to advise, not dictate. With this in mind, it is very important not to recommend a single solution but a set of solutions where the user can be actively involved in making the final selection. A generate and test approach fulfills these requirements by first intelligently constructing a set of plausible weld process and joint geometry combinations and then evaluating each scenario according to a set of criteria. The final step is to rank the scenarios and present the results to the user for final selection.

This system can be implemented many different ways, including a rule-based solution that can be either goal or data driven. Rules are a declarative expression of expertise that are especially useful for problems that are inherently search based like planning or design (Luger & Stubblefield 1993). Control is implicit via the inference engine and the knowledge engineer needs only to represent the domain expertise in rules. Rules can be used for many other types of problems, but may not be the best approach or even a poor choice when trying to force fit a problem into a rule-based format. For example, a problem that has a great deal of structure and sequence has to be modeled in rules with control rules, priority, or an agenda. Even the famous MYCIN expert system (Buchanan & Shortliff 1984), a diagnosis system for meningitis, was criticized by the doctors because the interactive dialog appeared random and seem to not follow any line of reasoning.

Implementing the welding advisor in a purely rule-based manner is acceptable but awkward given the inherently sequential nature of the problem. There is significant structure and sequence in the problem that would be wasted if everything was coded in rules and given to an inference engine for search. Structure and sequence could be implemented in rules but that would result in an inefficient and unnatural representation.

An acceptable hybrid solution would have been to capture the structure and sequence in an object-oriented

framework and use small rule sets to encode discrete chunks of knowledge. This hybrid approach exploits the strength of both the rule-based and object-oriented approaches. The rule-based approach offers a natural way to declare knowledge and perform search through the rule space but is clumsy for its data expressiveness and sharing. The object-oriented paradigm is almost just the opposite with excellent data modeling through encapsulation and data sharing through messaging. Welding knowledge could be represented in rules with implicit control with the object-oriented framework providing the global explicit control.

The purely object-oriented approach taken by the welding advisor is a result of evolutionary prototyping. We started implementing the generate and test control structure using Kappa™, an object-oriented development environment from Intellicorp, and continued in this environment codifying the welding expertise in methods. Although a hybrid solution is perhaps preferable, the object-oriented approach provided an invaluable solution to the difficult and time-consuming process of knowledge engineering and domain understanding.

Rule-Based Approach

A rule-based approach could solve this problem by taking weld requirements as inputs to start the rule chaining and resulting in a ordered set of solutions. A high level rule to start the process would be:

```

if GeneratePlausibleScenarios((IN)WeldRequirements,
(OUT)Scenarios) and
  EvaluateEachScenario((IN)Scenarios,
(OUT)EvaluatedScenarios) and
  RankAndScoreScenarios((IN)EvaluatedScenarios,
(OUT)RankedAndOrderedScenarios) and
  DisplayResults((IN)RankedAndOrderedScenarios)
then
  Done.

```

In this statement we see the sequence coded in a rule. Each clause in turn would start the chaining process in a corresponding rule set. The first two clauses would have rules that encode actual welding expertise. The last two have nothing to do with search or knowledge representation and would be part of an awkward representation as mentioned above. It also has a large amount of complex knowledge that is handled nicely in a rule-base with opportunistic search. The unwieldy part is in specifying the control structures needed to satisfy the clause, EvaluateEachScenario(), because that is where the wealth of welding knowledge resides and requires orchestration, sequence, and control to a fair degree. For example, a

geometry with a backing bar requires additional analysis that could be implicitly invoked by pattern matching, but in methods, the actual testing for a backing bar is performed and the appropriate messages are sent.

Object-Oriented Approach

Encapsulation, polymorphism, and message passing are the aspects of object-orientation used most in the welding advisor. Objects are a very natural and powerful way to model the world (Rossen, 1990) and reasoning processes of human experts. When initially interviewing welding experts, the domain understanding proceeded very quickly because of the natural expressiveness of objects. It very quickly became clear what the high level objects were (JointGeometry, WeldProcess, Material, etc.) and what, at least on a simple level, were the relationships and interactions between these objects. For example, welding processes have preferred materials and joint geometries have preferred welding processes. This knowledge engineering could proceed independently of the design of the generate and test engine because the knowledge is completely separate.

The only unanswered questions were what model to use for the computer assistant (best answer, set of answers, ranked and scored set of answers with what-if capability, etc.) and how to generate the solution (purely generative, candidate creation and evaluation, etc.). The goal was to build a system that was highly interactive where designers and engineers could examine in great detail the results of the advisor and perform extensive what-if analysis. A generate and test approach was selected because it typically generates a number of plausible scenarios offering an exploratory environment where the user can examine a wealth of detailed information as to why a scenario either passed or failed.

The generate and test approach is palatable for the new user in that it will not absolutely dictate solutions but rather offer advice. As AI researchers are well aware, it is important not to threaten the users but to give them a tool that assists in their work. That is clearly the case with the welding advisor because welding is so complex it is naïve to imagine the welding advisor replacing welding engineers. The advisor could help train entry-level engineers, but will probably be most useful in its completeness of analysis of weld designs by reminding experts of some details that may be otherwise overlooked.

The control of flow is shown in Figure 1. Up front criteria for welding are determined and implemented as tests in the advisor. A designer or engineer presents requirements for a given weld and the evaluation object take those requirements, the tests, and any other inputs needed from databases, user queries, or deduced

The Functionality of the Advisor

knowledge to produce a ranked and scored ordered set of possible weld designs. The wealth of welding knowledge resides in the tests: they use the joint geometry knowledge and weld process knowledge to evaluate a scenario against the given requirements. For what-if analysis, the user is provided with the capability to change the importance or weighting of any of the criteria.

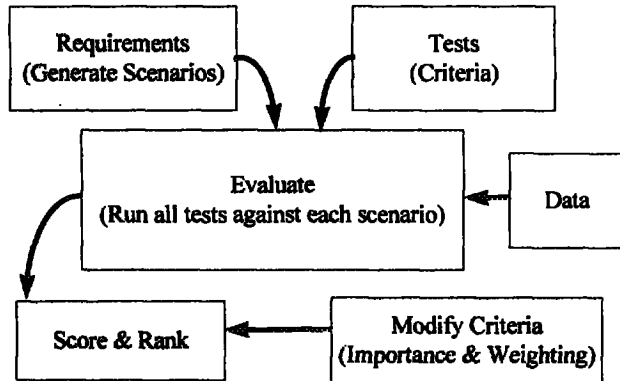


Figure 1. The flow of control of the generate and test approach in the welding advisor.

Another advantage of the object-oriented approach is reuse. We abstracted the generate and test methodology into a generic advisor. The welding advisor specialized certain objects and methods to capture the details of the welding domain, but the structure of the generate and test approach was completely inherited. Another project at SNL will use this basic approach and specialize the generic advisor for advice on near-net shape determination (Rivera, 1996) as shown in Figure 2.

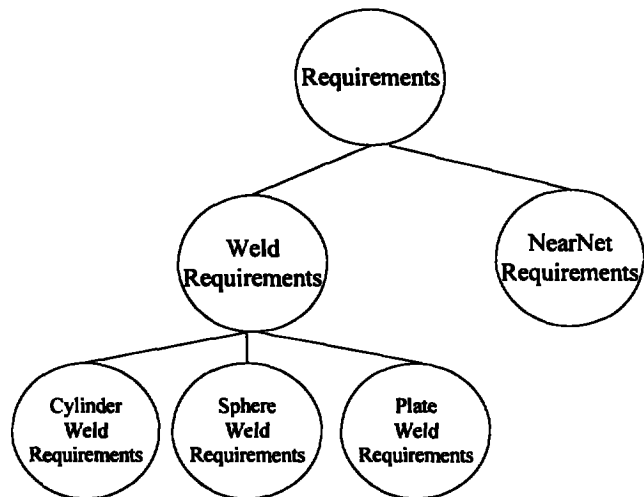


Figure 2. Object hierarchy specializing the requirements object of the generate and test approach, where lower level objects add specific data and methods.

Once the requirements for the weld are specified the user typically spends most of their time manipulating the table of results shown in Figure 3. The rows in the table represent the plausible welding scenarios. They were generated using heuristics for combining welding process and joint geometry configurations that address the design requirements of the weld.

The fourth column is the Score awarded each scenario, based on its performance. The performance is assessed by examining the criteria in the form of tests. Each ranking criterion has an "importance" and a "weight" associated with it. "Required" criteria are hard constraints, and "desired criteria" are soft constraints. If a user wishes to omit one of the normal criteria from the evaluation function, the "don't care" importance for that criteria is chosen.

By default, all criteria are set to "required" importance, as opposed to "desirable" or "don't care" and each has a weight of 50. Both the importance and weight of each criterion can be changed as shown in Figure 4 by the user (lower right corner button in the window, Figure 3) to develop an understanding of the sensitivity of the weld scenario rankings to the importance and weighting of each criterion. This allows the user to rapidly perform trade-off assessments. Other "what-if" analysis is available by changing performance requirements by either changing specification of design or answers to the advisor's queries.

The first column of Figure 3, the scenario number, has a color coded background that indicates if the scenario passed all hard or soft constraints (criteria). Global constraints are part of the performance requirements. If a scenario satisfied all constraints (required and desired) then it is awarded a green color. If one or more desired constraints are violated then the scenario is awarded a yellow color and if a required constraint is violated it is red. The scenarios are sorted and displayed in descending order. The columns to the right of the table depict the individual scores of the tests performed on each scenario.

The detailed analysis of each scenario is available by pressing the "Explanation" button at the lower portion of the window in Figure 3. The explanation includes details about each of the tests performed explaining how a particular scenario performed against the weld requirements. For example, a scenario's explanation might read, "For a Low Voltage Electron Beam Process, a smooth underbead requirement, square joint geometry, we recommend a joint thickness between 0.05mm and 3.0mm, not 8.0mm as you have specified." Also provided in the explanation is any anecdotal information not pertaining to tests that may be interesting to the user, such as, "Since we have two sided access, we can satisfy the smooth underbead requirement by removing the backing bar." or "Weld only from one side."

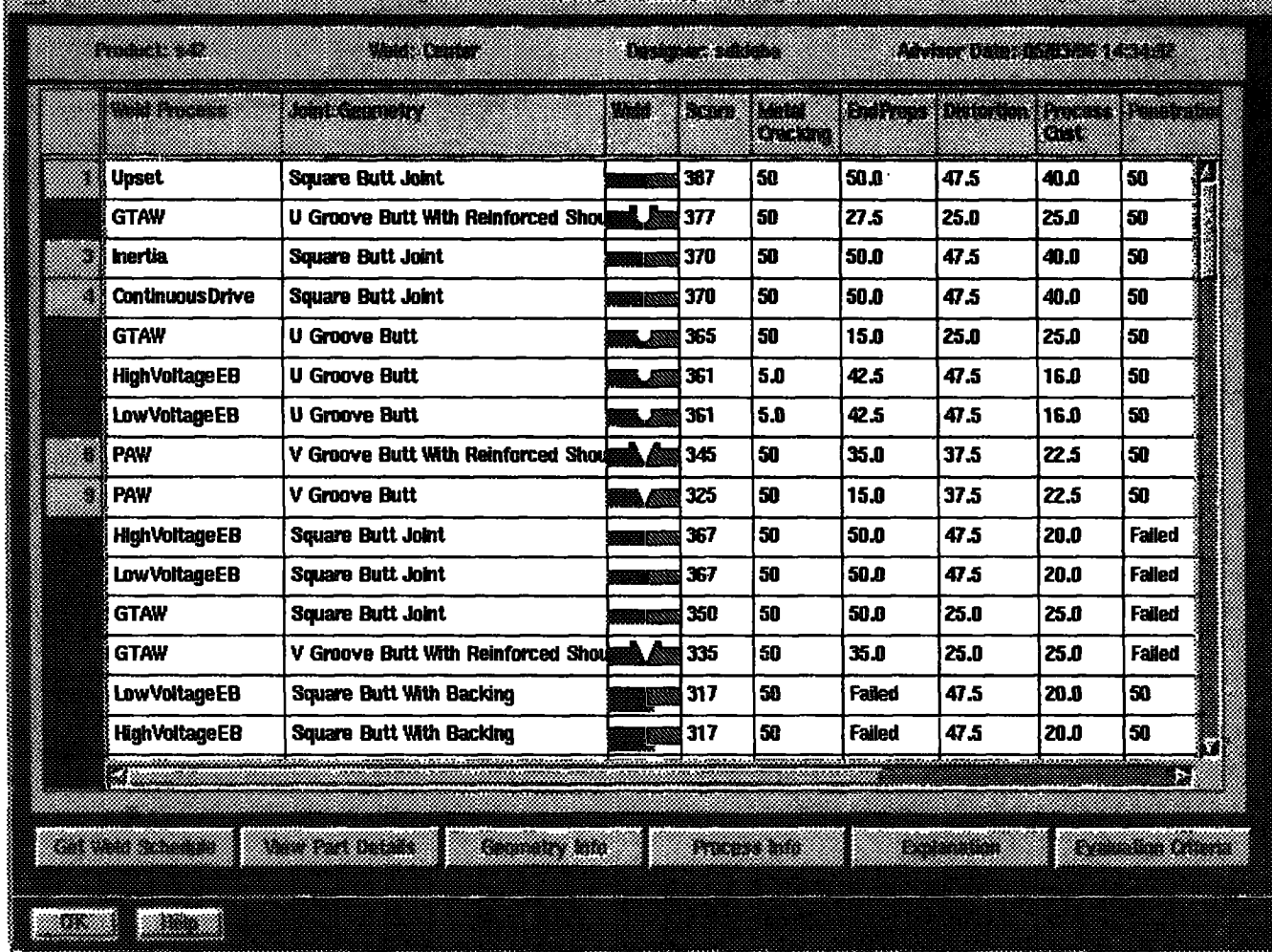


Figure 3. The table of results as output by the advisor.

A typical fusion weld presented to the advisor will result in 50 - 80 weld scenarios. Each scenario will have a detailed explanation of its performance. In addition, the user can perform "what-if" analysis numerous ways. The user can spend a great deal of time in analysis by browsing the 50 - 80 explanations which all may change after each iteration of "what-if" analysis providing an excellent environment for training.

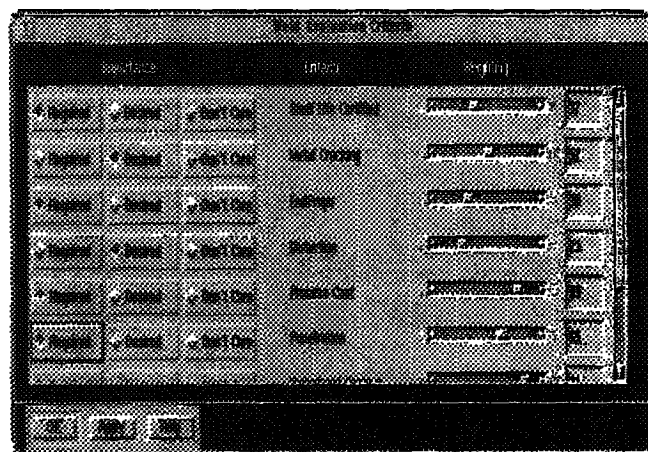


Figure 4. Window to change importance or weighting of evaluation criteria.

Stand-alone Architecture

Although the welding advisor is intimately connected to the SmartWeld environment, it can function as a stand alone advisor in light of the recent work on the new distributed agent architecture (Mitchiner et al. 1996). In addition, the advisor functions as black box within a WWW browser, with the weld requirements as input. The rank, ordered list of welding scenarios is the output which is displayed in the WWW browser.

Future Work

From: Proceedings of the AI and Manufacturing Research Planning Workshop. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

The welding advisor today has the capability to train new welding engineers and to guide, analyze, check, and verify the weld designs of more experienced personnel. Refining the knowledge in the advisor is a never ending task. In fact, the "smarter" it gets, the more users, and with more users come more input. In addition to refinement, new tests and performance criteria are being added. Work has begun to add knowledge about material preparation and purchase requirements. This summer we will be exploring both generative and adaptive weld schedule generation based on historical data and knowledge.

Acknowledgments

The author acknowledges the patience and enthusiasm of Ken Hicken and Jerry Knorovsky, the welding experts at Sandia, initial prototype and design of the advisor by Dave Messink of Intellicorp, and objects verses rules discussions with Bill Stubblefield and Dave Messink. Also, Kim Mahin and John Mitchiner are acknowledged for the support and development of the SmartWeld program. This work was performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract numbers DE-AC04-76DP00789 and DE-AC04-94AL85000.

References

Luger, G., and Stubblefield, B., 1993. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Redwood City, CA: Benjamin/Cummings.

Mitchiner, J., Kleban, S., Hess, B., Mahin, K., Messink, D., 1996. SmartWeld: A Knowledge-based Approach to Welding. From these proceedings.

Buchanan, B. and Shortliff, E., eds. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Hueristic Programming Project*. Reading, MA: Addison-Wesley.

Rossen, M. B., and Sherman R. Alpert, 1990. The Cognitive Consequences of Object-Oriented Design. *Human-Computer Interaction*, Volume 5, 345-379.

Rivera, J., Stubblefield, W, and Ames, A, 1996. From these proceedings.