Weasel: A Mixed-Initiative System to Assist in Military Planning

C. C. Hayes, Capt. A. D. Larson, U. Ravinder

University of Minnesota,
111 Church Street. S. E
Minneapolis, MN 55455
hayes@me.umn.edu

United States Air Force Academy,
2354 Fairchild Drive, Suite 6L101B
USAF Academy CO 80840
adam.larson@usafa.af.mil

NASA Ames Research Center
Mail Stop 262-4
Moffet Field, CA 94035
uravinder@mail.arc.nasa.gov

Abstract

This paper describes a mixed initiative planning system, called Weasel and its evaluation. Weasel was developed to assist military decision makers in the task of enemy course of action generation. The evaluation assesses Weasel's impact on the decision making performance of two potential user groups. When designing Weasel, we aimed to maximize benefits delivered by the software by focusing support functions on key areas in which expert analysts exhibited difficulties. We also aimed to minimize development, training and maintenance costs by designing displays to reflect expert analysts' representations and relying on human problem solving skills where possible. The goals of the evaluation are to 1) assess whether Weasel increases users' problem solving performance, where performance is measured in terms of overall solution quality, 2) identify the most appropriate user group by assessing whether domain intermediates are helped or hindered more than domain experts, and 3) identify possible negative consequences that may occur when Weasel generates a "brittle" solution. The issues explored in Weasel's development and evaluation are common to many mixed initiative systems.

Introduction

This work describes the development and evaluation of Weasel, a mixed-initiative system which assists military planners in exploring possible enemy courses of action (ECOAs). An *enemy course of action* is an arrangement of enemy forces which very abstractly define a very abstract "plan" which may be followed by enemy forces. There is great interest in possible use of decision support tools to assist in military planning; the increased complexity and tempo of modern military operations combined with increased pressure to reduce staff sizes makes it difficult for people to keep up with the demands of operations planning. Mixed initiative systems tend to be more appealing than automated systems in complex, safety critical domains such as this one because of the opportunity to benefit from human judgment. However, before employing mixed initiative systems in decisions with life and death consequences, it is important to first understand both the positive and negative impacts they may have on human problem solving performance.

The design goals behind Weasel were to improve decision making performance in this task for military planners with an intermediate level of experience (i.e. 2 to 5 years training), and possibly for experts (6+ years of training and practice) as well. The evaluation assessed whether use of Weasel changed (improved or decreased) planning performance for two user groups: intermediates and experts, and explored whether there were situations in which use of Weasel might decrement performance.

Mixed-initiative planning and scheduling systems (MIPAS) are computer tools which work jointly with humans to create plans or schedules. MIPAS can be viewed as examples of a larger class of tools called decision support systems (DSSs).

Decision support systems are computer tools which assist human decision makers to make better decisions in any type of task (e.g. medical diagnosis, manufacturing plant layout, product design, etc.) without necessarily making those decisions for them. DSSs may provide support in many ways, for example by providing task-specialized editors, generating whole or partial solutions, or providing solution evaluation and comparison tools. A key philosophical assumption behind DSSs, which is not necessarily shared by all MIPAS, is that the human is the one that should be in control of the decision making process. Weasel is both a DSS and a MIPAS.

High criticality decision making tasks are those in which decisions can result in large costs or catastrophic consequences. Examples of high criticality domains include military planning, search and rescue, and medical diagnosis. They are of interest because they represent areas where problem solving improvements, gained through introduction of DSSs or other means, can yield great value. However, because decisions made in these domains may impact human safety or have political ramifications, it is important to clearly understand how DSS tools impact human decision making before adopting such tools. The possibility of *over-reliance*, i.e. inappropriate trust (Parasuraman, 1997) on a DSS is a major concern in safety critical domains.

In general, DSSs can have *both* positive and negative impacts on human decision making, possibly improving performance in many situations while degrading it in others. In particular, Smith, McCoy and Layton (1997) describe an experiment exploring a situation in which a DSS, The Flight Planning Testbed (FPT) improved users' average problem solving performance in finding fuel-optimal routes for commercial jets, but also occasionally degraded some users' performance when FPT exhibited *brittle* behavior. Brittle behavior occurs when parameter not modeled by the system impact the solution. Brittle behavior results in generation of inappropriate or inadequate suggestions. Unfortunately, in complex, context dependant domains, it can be difficult to predict when brittle behavior may occur. In FTP's case, brittle solutions were fuel-optimal but unnecessarily risky by human-decision makers' standards. However, the researchers also found that this effect appeared to be mitigated if subjects did their own exploration of the problem before seeing the computer's solution(s). They further hypothesized (but did not test) that additional strategies might also mitigate the impact of system brittleness, such as simultaneous presentation of multiple computer generated solution options, and computer critiquing of human generated options,

All DSSs, simulations, or mathematical models will sometimes exhibit brittleness because they are *necessarily* simplifications of the real world's richness. Therefore their solutions will produce some degree of error which may or may not be predictable. Given that some degree of brittleness is inevitable, it is important to consider how brittle behavior may impact decision makers in many tasks, and if it can be mitigated. One of the goals of this work is to assess whether Layton's findings were generally true for other domains; could one expect the similar results in the domain of ECOA planning? Would brittle solutions generated by Weasel also produce a similar performance decrement? If so, could the effect be mitigated by a similar strategy?

The Task Domain: ECOA Generation

Weasel is part of a trio of tools: CoRaven (Jones at al. 1999), Weasel and Fox (Schlabach, Hayes and Goldberg, 1997), which support a range of problem military planning and intelligence activities, shown as ovals in Figure 1. All steps may be conducted in parallel, and all are repeated many times during the course of a battle. The decision makers who engage in this problem solving cycle include both military operation planners and intelligence analysts. The overall goal of this reasoning cycle is to identify what action(s) the friendly forces should take next, based on continual assessments and re-assessments of the current battlefield and enemy situation. Although the direct output of Weasel is a set of possible (and likely) enemy courses of action, it supports the assessment of friendly courses of action by providing a set of foils. Friendly courses of action are assessed based on their performance against multiple enemy courses of action which might occur.

There is no specific starting or ending point to the cycle in Figure 1. However, before the onset of a battle, intelligence analysts often start at oval 1 (Figure 1) "Plan/Schedule Intelligence Collection." They create a plan for gathering key pieces of information pertaining to the enemy such as the type of unit, likely resources, and location of key elements. This information is gathered by scouts, satellites and surveillance devices (step 2), then it is analyzed (step 3) to produce hypotheses and constraints on enemy resources and location.

Weasel assists analysts in step 4 with the systematic generation of enemy courses of action that are consistent with the intelligence conclusions developed in step 3, and the observed rules of behavior for that enemy. ECOAs, developed jointly by Weasel and the analyst, are passed on to the Fox system, which uses a genetic algorithm and war gaming simulator to generate friendly courses of action (FCOAs) that perform well against those ECOAs. This cycle is continually repeated throughout the battle as the situation changes. Plans for friendly actions must continually be reassessed as the battle unfolds.

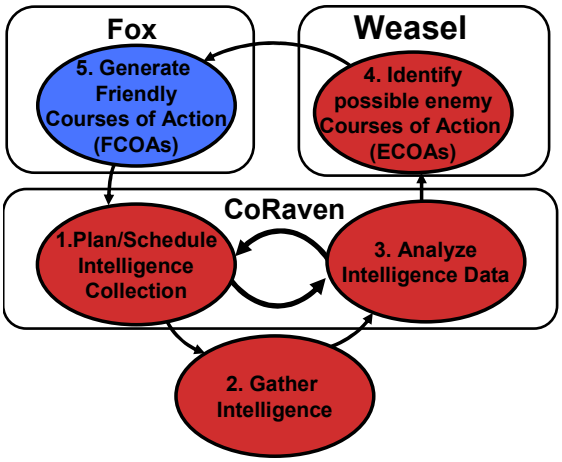

Figure 1: The intelligence collection and planning cycle.

Weasel

Considerations in the Design of Weasel

Intelligence analysts must consider many ECOAs since there are many actions which an enemy might take. Unfortunately, there are an infinite number of possible ECOAs, so even with extensive computational resources one cannot consider them all. Fortunately, most ECOAs are neither useful nor interesting. Analysts typically focus their search on a few *most likely* and a few *most dangerous* (but possibly unlikely) ECOAs.

Weasel is designed to assist military analysts in thoroughly and systematically considering the *most likely* ECOAs. In this context, the *most likely ECOAs* refers to ECOAs that are consistent with current data and assumptions about most likely enemy position and resources. Terrain constraints and intelligence assumptions provide strong constraints on the search. The likely ECOAs are usually a very small subset of the total possible.

Note that Weasel does not currently assist users with identifying *most dangerous* ECOAs. Analysts must also identify ECOAs that are not necessarily consistent with intelligence assumptions, but which could pose a considerable threat if they were to occur. This is an important task in which analysts would probably welcome assistance. However, outside of brute force exhaustive search and evaluation, we do not currently have an efficient algorithm which could feasibly address this task. Future work may explore approaches to focus dangerous but unlikely ECOAs.

Weasel was developed through cognitive engineering (Smith and Geddes, 2003) and human-centered system development methods. Many design decisions were guided by the objectives to minimize develop, training and maintenance costs, while maximizing benefits to users. In designing a system to meet these objectives, we were very conscious of the fact that ECOA generation is a task at which domain experts already do relatively well, and it is usually performed under time pressure. The majority of computer tools require some overhead to learn and to use, thus, what ever tool we developed had better offer clear benefits to users in areas where they desire assistance. Otherwise there would be little chance they would be willing to take the time required to learn and use them.

With this understanding, we observed analysts performing the ECOA generation in laboratory studies, during which we took "protocol" transcripts (Ericsson and Simon, 1984), and in training exercises such as the Prairie Warrior exercises held in Ft. Leavenworth, KA. Through these observations, we observed that even experts sometimes over looked relevant ECOAs for a variety of reasons. For example, it was often difficult for them to systematically think through all the possible options while simultaneously keeping track of all the current relevant constraints. In other cases, they because fixated on particular assumptions (which were often implicit in their reasoning), forgetting to question them when the context changed.

Based on these findings, we designed Weasel to assist analysts by providing 1) an engine that can systematically enumerate possible ECOAs consistent with a given set of assumptions, and 2) an interface in which they can express and manipulate those constraints. Together these capabilities allow "what-if" scenarios to be described and assessed rapidly. Lastly, we provided an interface displaying hard constraints used by Weasel in order to provide users with insight into (and possibly trust) in the reasoning engine. Making such these constraints observable and explicit may provide an added training benefit to domain novices and intermediates.

Other principles guiding design of Weasel were "computer in the loop" and "minimalist intervention" philosophies. Usually, developers of mixed-initiative technologies view the challenge as bring the "human in the loop." However, for most complex cognitive tasks such as planning, design and medical diagnosis the human *is* in the loop already. Not only are they *in* the loop, humans *are* the loop -- and have been for thousands of years. Thus we feel the challenge should be to bring the "computer in the loop" in a way that is acceptable to humans.

Several design implications follow from a "computer in the loop" philosophy. One is: when in doubt, leave a task and to the human; focus on minimal introduction of computer assistance. A simple computer tool which has been well executed interfaces is more likely to be useful than a more complex one. It will probably also be easier to maintain. We explicitly decided *not* to intervene (at least initially) in tasks such as identifying relevant constraints, or selecting ECOAs for further consideration since these tasks require complex, experience-based judgments which may best be left to humans.

Lastly, Weasel's representations and displays had to fit with analysts' way of thinking about the task. Many of Weasel's representations and displays are based directly on sketches made by analysts on paper or acetate map overlays while doing their work.

Considerations in Weasel's Evaluation

An important part of a human-centered design approach is to evaluate the system early and often. Frequent evaluations provide valuable feedback to system developers as to whether the approach is meeting the design goals so adjustments can be made. There are many properties of mixed-initiative systems that are important to evaluate including ease of use, accuracy of software generated results and overall impact on joint human/computer problem solving performance. The latter is the "bottom line" in many mixed-initiative systems. If users do not derive tangible benefits from the system they won't use it; the computer will be left "out of the loop."

The evaluation aims to address several questions pertaining to users' problem solving performance when using Weasel:

1. Does Weasel actually increase users' average problem solving performance? In this evaluation, performance is measured in terms of overall solution quality,
2. If Weasel does result in a performance change, does it impact performance of domain intermediates more (or less) than domain experts?

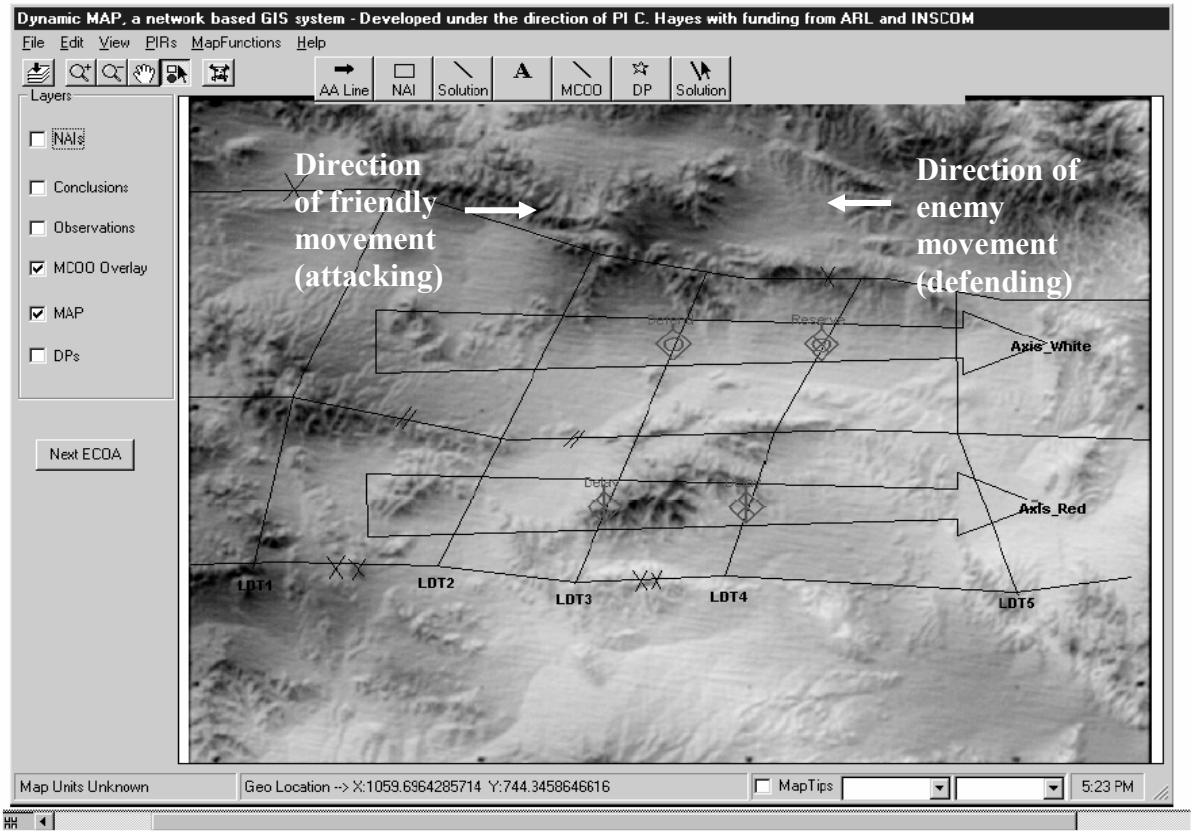

Figure 2: An enemy course of action (ECOA) in the context of the terrain.

3. Are there negative consequences that may occur when Weasel generates a "brittle" solution?
4. Can negative impacts of brittle solutions be reduced by presenting computer solutions *after* the human has generated some of their own solutions?

From a development standpoint, these questions will help us to assess whether our basic approach is reasonable, what users groups should be considered as "customers," ways in which the system may sometimes hurt performance, and possible strategies to avoid pitfalls.

ECOA Representations

An example of an ECOA is shown in Figures 2 and 3. Figure 2 shows an ECOA on the terrain; friendly forces are attacking the enemy (but only enemy forces are shown. In this context, an ECOA is an assignment of battlefield locations and fighting roles to enemy units. Battlefield locations are specified in terms of intersections on a grid formed by markers on the map called avenues of approach and lines of defensible terrain. *Avenues of approach* (AAs) are shown in Figure 2 as large horizontal arrows; they represent corridors between mountains and other obstacles through which troops can move. The direction of the AA arrows indicates the direction of attack (and the friendly movement). *Lines of defensible terrain* (LDTs) are shown in Figure 2 as thin vertical lines which are placed across narrow parts of the AAs; the represent areas where defenses tend to be setup and where fire fights occur. The diamonds placed at the intersections of AAs and LDTs represent specific enemy units.

Figure 3 shows an ECOA sketch, which is an abstracted version of the ECOA in Figure 2. All the details of the terrain have been abstracted except for the AAs and LDTs. The labels, "Def" "Del" and "R" represent the roles of the various units: defense, delay and reserve, respectively.

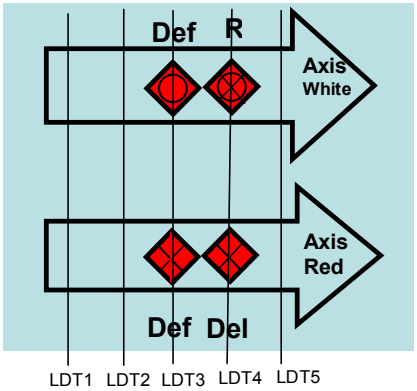

Figure 3: An enemy course of action (ECOA) sketch.

ECOAs can be thought of as the first step in a plan for future enemy actions. Alternatively, one can think of each ECOA as a specific layout of the enemy's chess pieces (i.e. units). However, the board is only partially observable, so many possible board layouts must be considered based on the little you can directly observe or indirectly guess.

System Description

There are several steps by which Weasel generates the most likely enemy COAs, as shown in Figure 4. Each of these steps will be described below.

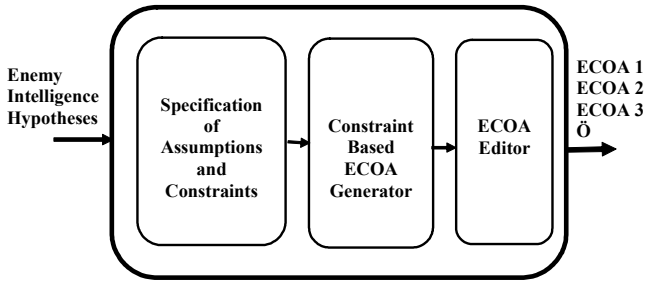

Figure 4: A System diagram of Weasel's components.

Weasel uses several types of intelligence hypotheses as inputs to constrain enemy position. These hypotheses may be generated by the analyst from intelligence reports, or in this case, by a decision support tool called CoRaven. CoRaven uses a belief network to compute the probabilities of various hypotheses from intelligence reports (in the form of SALUTE messages), and then it visually displays its conclusions. Figure 5 shows the display of one type of intelligence hypothesis: depth of the enemy defense. The depth of defense indicates how far west the enemy has penetrated from their starting point, which in this example is near LDT 5. Thus, there are 4 hypotheses under current consideration which are: the enemy has penetrated as far as LDT1, LDT2, LDT3 or LDT4. The color of each LDT indicates the probability of each hypothesis, where black is less than a 5 % probability. As the probability increases, the LDT becomes brighter (whiter).

CoRaven computes these probabilities based on current intelligence reports. Each report is shown as a small symbol on the map in Figure 5. The black symbols indicate places where intelligence observations have been made, and nothing of interest was seen, while the gray (or red in the color version) symbols indicate observations of enemy activity. As new observations are reported, the intensities (i.e. probabilities) of the LDTs shift. In this

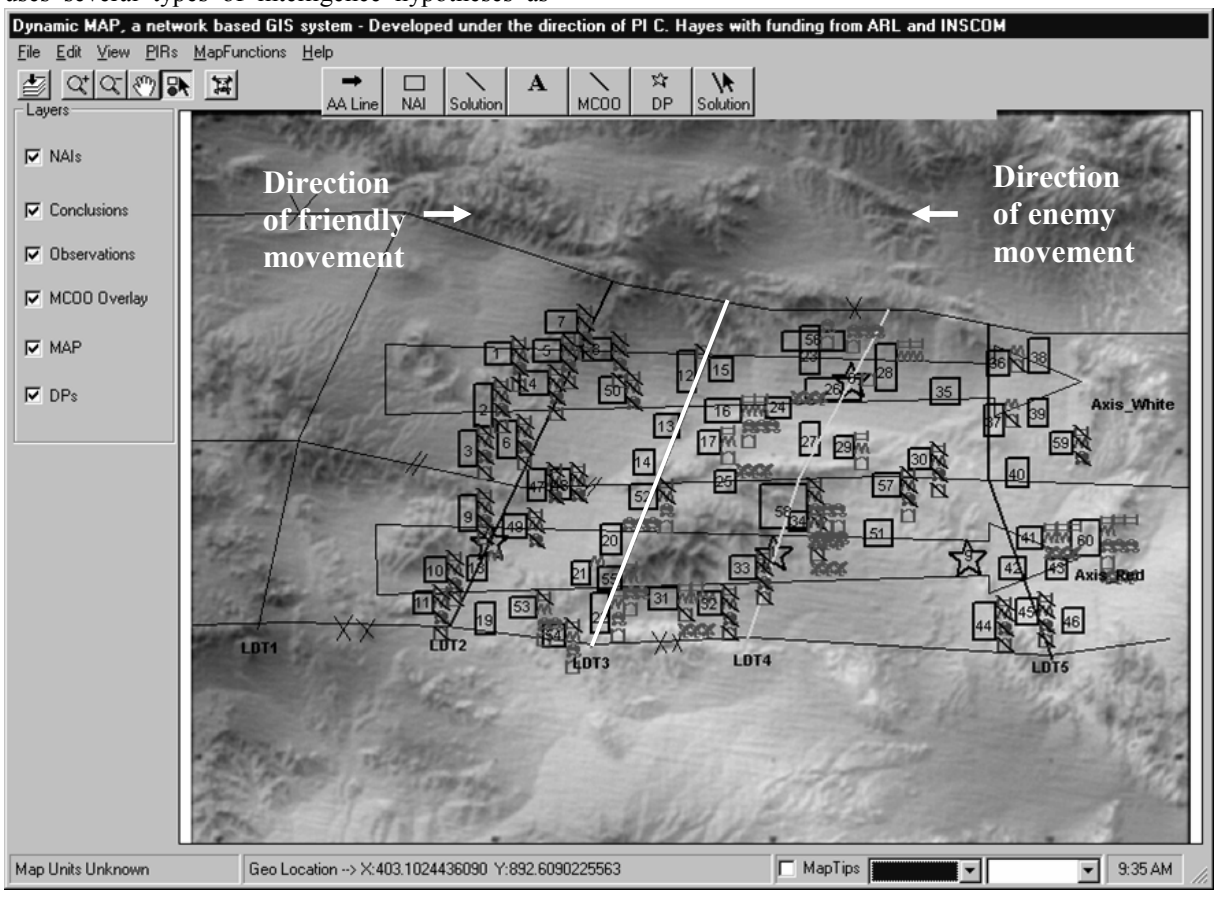

Figure 5: Partial results from Co-Raven's intelligence analysis.

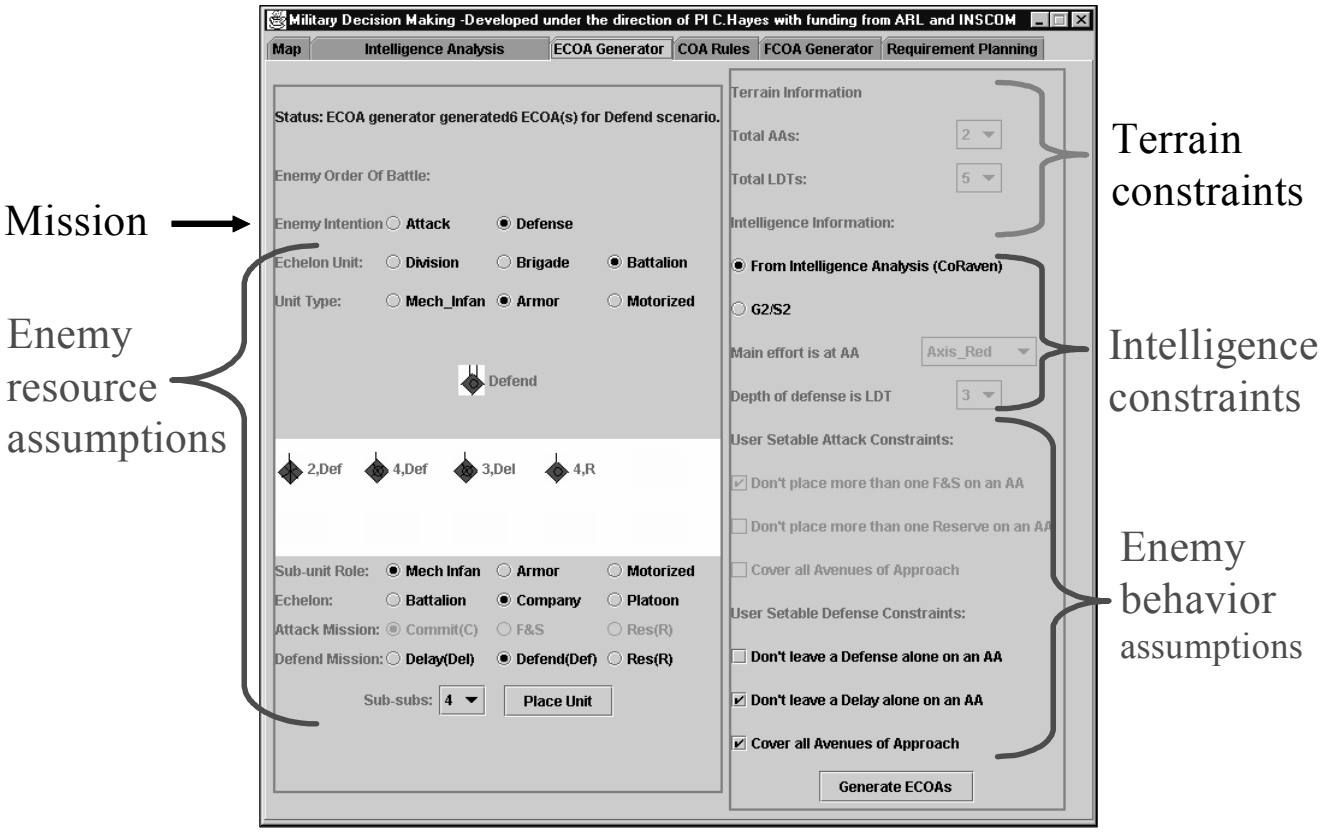

Figure 6: Weasel's Interface for Specifying Enemy Constraints and Assumptions

example, the depth of enemy defense is most likely at LDT3, indicated by LDT3's light color. LDT4 is a close second. The analyst can choose how many of these hypotheses to consider. In our example we will focus on the assumption that the enemy has penetrated to LDT3.

Additional intelligence hypotheses (not shown) address the question "Where is the main defense?" In this example, reports cumulatively indicate that the main defense is most probably in the southern AA, Axis Red. This is also given as a constraint to Weasel.

Specification of Assumption and Constraints. Further information which the analyst must specify is: the enemy mission: attack or defend; the size and composition of the enemy forces (battalion, company, platoon, etc) and assumed rules of enemy behavior. Figure 6 shows that, for our example, the analyst has specified the enemy mission as "defense." The enemy unit under consideration is an armor battalion. The analyst further assumes that the battalion is composed of the sub-units shown as red diamonds in the lower left of Figure 6.

Three "soft" rules of enemy behavior are shown in the lower right of Figure 3: "Do not leave a Defense Unit alone in an avenue of approach (AA)," "Do not leave a delay unit alone on an AA," and "Cover all avenues of approach" (with defending units). These are rules which the enemy may or may not follow when planning their COAs. The user can state his or her assumptions about whether or not the enemy will follow these rules by checking (or not checking) the boxes next to the rules. Checking a box turns that rule on. Un-checking it turns the rule off. In the example in Figure 6, the user has chosen assume that the enemy might leave a defense unit alone on an AA, but will not leave a delay unit alone, and will cover all AAs. Weasel's planner uses the checked rules as constraints when constructing enemy COAs.

Additionally, there are "hard" rules of behavior which the enemy will (almost) always follow. These rules are shown in Figure 7. For example, "The leading unit in an AA must be a delay or defense unit." The rules are divided into two sets which apply respectively to enemy offensive (attack), and defensive maneuvers. The rules in Figure 7 are treated as hard constraints because they represent either

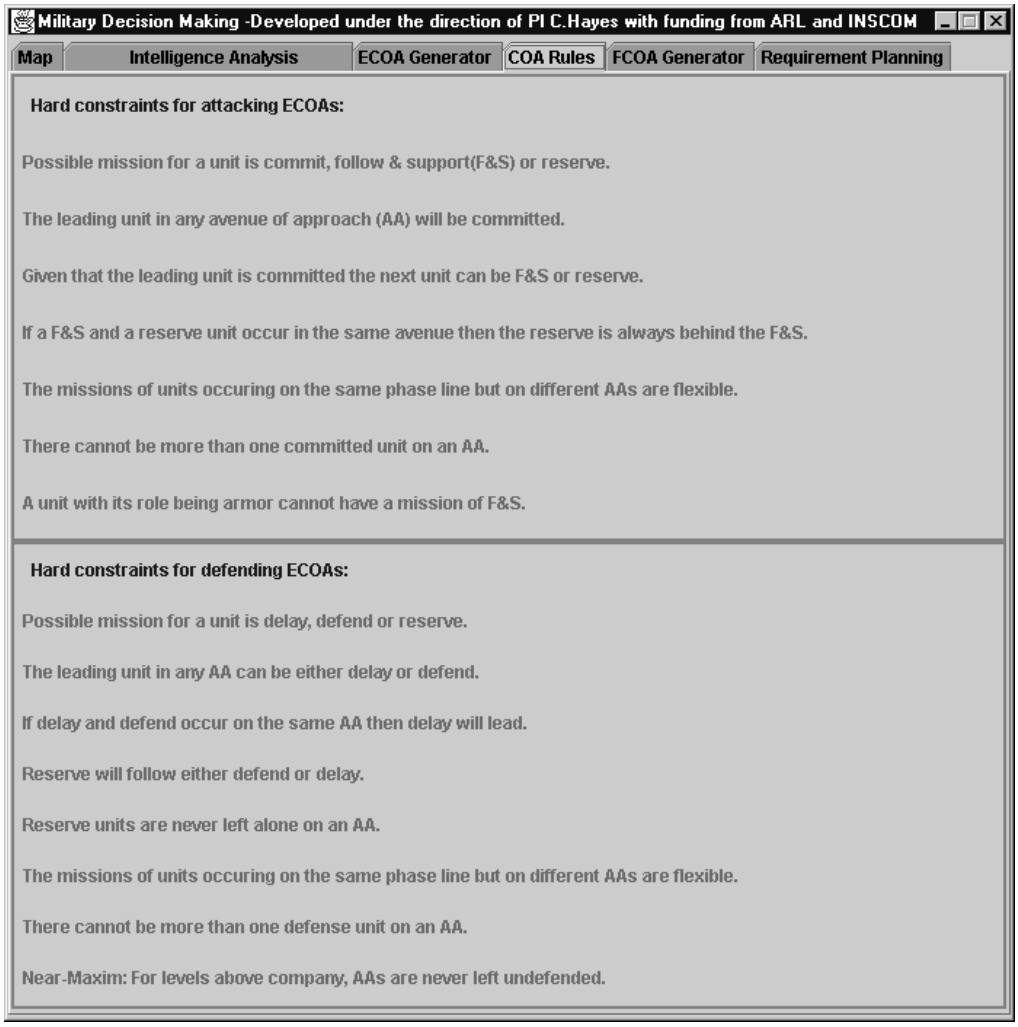

Figure 7: Users can view Weasel fixed constraints

definitions which are relatively fixed, or they represent maneuvers that cannot be easily modified without endangering the unit or requiring lengthy preparation on the enemy's part (i.e. training and field exercises). Since few constraints (rules) in any domain can be said to be truly fixed, future work will examine whether to make some of the rules which are currently hard constraints into user settable constraints. Some of these considerations include determining *who* should be allowed to make changes to relatively hard constraints (e.g. domain intermediates, experts or only special system maintainers?) and weighing the utility of adding flexibility (which may be used infrequently) against the possibility that errors will be introduced when users accidentally change relatively hard constraints.

Because these rules considered to be fixed, the user is not permitted to turn them on or off. However, these rules have been made available in Weasel's interface for users to examine should they wish to do so. We feel it is important to make the rules controlling the planner's behavior accessible to the users, and to express them in the users' domain vocabulary, thus de-mystifying the software engine. All too frequently, automated planners and problem solvers are effectively "black boxes" from the users' perspective.

Constraint-Based ECOA generator. Once all constraints and assumptions have been entered, the user can request that Weasel generate all ECOAs consistent with those assumptions. The planner is a simple constraint-based planner that generates all permutations of the resources consistent with the constraints. Although the planner is not complex, it is more systematic about generating all combinations than most humans are, particularly when there are many combinations.

For this example, there are 6 possible ECOAs shown in Figure 8, consistent with the assumptions specified. These ECOAs have been rotated so that they are in the same orientation as they would appear when displayed on the terrain shown in Figure 2.

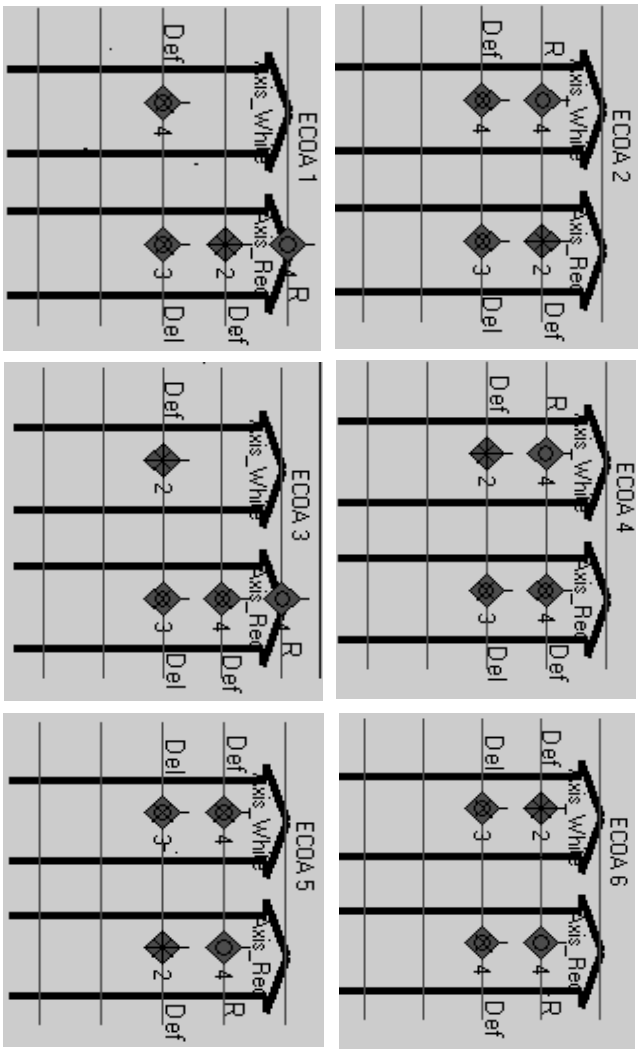

Figure 8: Six ECOAs generated by Weasel which are consistent with the constraints and assumptions in Figure 6.

ECOA Editor. Once ECOAs have been generated, analysts can view them in the plan editor (Figure 8). If he or she is mostly satisfied with the ECOAs but wishes to change a few of their properties, the enemy units can be repositioned by dragging and dropping them. Additionally, specific ECOAs can be selected and viewed in the context of the terrain as shown in Figure 2.

Changing assumptions, repeating the cycle. An important part of the annalist's problem solving is to consider what the enemy might do under a *variety* of different assumptions. For example, what might the enemy do if they decided *not* to leave a defense unit alone on an AA, or *not* to defend all AAs? Weasel's interface allows users to try different "what-if" scenarios defined by sets of assumptions and intelligence constraints, and rapidly see the impact on the likely ECOAs. This is an important function of Weasel's interface because it makes a specific and important task easier.

Next problem solving steps. The analyst's work is not yet complete even after a satisfactory set of ECOAs have been developed. They must select a small set of ECOAs (for computational reasons -- usually between 3 and 6) which they judge to be most relevant or important. This selected set of ECOAs will be used while generating friendly COAs (step 5 in Figure 1) to assess the appropriateness and possible performance of each FCOA considered.

Evaluation Method

Subjects. Eighteen subjects participated in the experiment (9 Air Force and 9 Army subjects). All had between 1 and 21 years of experience in the U.S. armed forces. Five subjects were categorized as experts, and 13 as intermediates; experts were those having at least 6 years of military experience on active duty, in the National Guard or Reserves. Domain novices (those having less than a year experience with the domain) were not used in the evaluation because they lacked sufficient knowledge to perform the task even with Weasel's assistance. The average length of experience of all 18 subjects was 5.03 years.

Scenarios. Subjects were asked to generate ECOAs for 3 different scenarios. Scenario 1 was designed to be difficult, requiring subjects to generate many possible ECOAs. Scenario 2 was designed to be relatively easy, and Scenario 3 was one for which Weasel generated a "brittle" solution set, in that it was incomplete. Solutions in which the enemy protected all possible approaches were not included. Weasel generated eight ECOAs for Scenarios 1, two for Scenario 2, and four for Scenario 3.

Solution Methods. Subjects were asked to generate solutions by three different methods, A, B and C. In Method A, subjects first generated ECOAs by hand, then were shown the ECOAs generated by Weasel and asked to pick between their own solution set and Weasel's. In Method B, subjects again generated solutions first by hand, and then were shown Weasel's solutions. However, this time they could revise their solution set if they so desired. Examples of revisions include copying one of Weasel's ECOAs or incorporating elements of it in one of their own. In Method C, subjects were shown Weasel's solutions first, and then they were asked to generate their own, which could include Weasel's ECOAs, or ECOAs based on them.

Design. All subjects solved all scenarios, and applied all methods. However, to eliminate learning effects, the order in which subjects saw the scenarios and applied the methods was counter-balanced. Given that there are 6 permutations of three items, this suggests a 6x6 experiment requiring 36 subjects. Instead we applied a lattice design

(Montgomery 1991) which reduced the required subjects by half (to 18).

Evaluators. Two evaluators assessed the quality of the ECOA sets generated by the subjects. The evaluators were selected for their expertise in Army battlefield strategy as well as their specific knowledge of current battlefield simulations used in the U.S. Army. One had 9 years U.S. Army experience, and the other 5 years.

Procedure. First, subjects were given familiarization training by the experimenter on a computer workstation. Materials given to subjects included: a scenario instruction page, three pages each describing the scenario, pen, and a one-page list of "required" (hard) constraints used by Weasel to generate ECOAs so that they may understand the computer's behavior.

Next, subjects were given scenario descriptions and asked to generate a set of ECOAs appropriate for each of the three scenarios. An experimenter was present at all times to answer questions. When a subject finished each scenario, they were then asked to provide verbal explanations of their solution choices. Upon completion of all three scenarios subjects, they completed a short questionnaire. Lastly, after all subjects had completed all scenarios, evaluators "scored" all solutions sets (including Weasel's) for each scenario, where best was 10 and worst was 1.

Results

The first steps in analysis were to check 1) the level of agreement between the evaluators and 2) whether there was a significant performance difference between the intermediate and expert subjects when they generated ECOAs by hand, *without* Weasel's assistance. The purpose of the first check was to assess whether evaluators had been chosen appropriately, the assumption being that there is a very low probability that independent evaluators will produce similar quality rankings for many solutions unless they have sufficient experience to assess quality. The purpose of the second check was to assess whether the division between the intermediates and experts was a meaningful one. The correlation for scenario 1 was 0.94, for scenario 2: 0.90 and for scenario 3: 0.99, indicating a high level of agreement between evaluators. In the second check, we compared the average quality ranking given by the evaluators to the intermediate and the expert groups. An ANOVA indicated that the difference between average expert and intermediate quality rankings was very significant, $p = 0.001$, indicating the experts performed significantly better than intermediates. Once these two issues had been established, we investigated the four questions posed in the introduction:

1. *Did use of Weasel improve the quality of ECOAs generated?* Yes. Overall there was a significant improvement in quality scores when ECOAs generated *without* Weasel's assistance were compared to ECOAs generated *with* Weasel's assistance ($p = 0.018$). Figure 9 shows the average quality scores received by users without and with Weasel's assistance, as well as the computer's quality scores. As expected, the quality score on the brittle scenario (Scenario 3) was very poor.

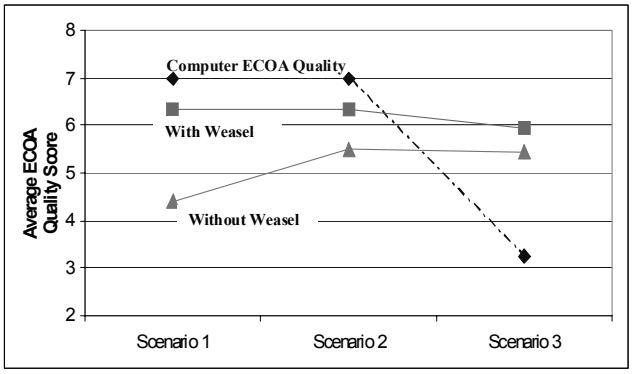

Figure 9: Ave quality scores received by subjects without and with Weasel's assistance (where 10 is best and 1 is worst score).

2. *Did use of Weasel change intermediates' performance more than experts'*? Yes. It improved intermediates' quality scores significantly ($p = 0.0002$), but did not significantly change experts' quality scores ($p = 0.251$). Furthermore, differences between experts and intermediates were leveled when both groups used Weasel; there was no significant difference between intermediate and expert quality scores when using Weasel ($p = 0.366$). This implies that use of Weasel elevates intermediates' ECOA quality to closer to the level of experts.

3. *Did ECOA quality decline when Weasel exhibits brittle behavior?* No. For scenario 3, there was no significant difference in the quality of ECOAs generated without or with Weasel's assistance ($p = 0.51$). In fact, ECOA quality scores increased on average for all scenarios when users employed Weasel's assistance. However, closer examination of individual subjects performances revels that there is more to the story. When using Weasel's assistance on Scenario 3, *more* subjects' (three out of 18) tended to repeat the mistake made by Weasel on Scenario 3 (i.e. omission of ECOAs that "cover" all avenues of approach). Furthermore, three of the five who made the omission were experts. In contrast, only one subject (an intermediate) made this same mistake when producing solutions manually. This implies that use of Weasel may have "biased" some users towards flawed solution sets when it exhibited brittle behavior, just as FTP biased users towards unnecessarily risky solutions when it exhibited brittle behavior.

4. *Did presentation order change users' the tendency to repeat Weasel's mistakes*? In Smith, McCoy and Layton's study of FPT, they reduced the tendency of users to adapt

the computer's flawed solutions by delaying presentation of the computer's solution until they had explored the problem on their own. However, we did not find a similar effect in this domain. Of the five users who "copied" the computer's mistake on Scenario 3, four generated their own solutions first and only one saw Weasel's solutions first.

Future Work

This work represents a positive start in the right direction. However, we are not going to declare victory yet; there is still much maturation of Weasel that needs to occur (through further development and evaluation) before Weasel can be installed and assessed in the context of a daily work environment. Many issues still need to be investigated and incorporated into system designs. For example, does explicit display of the ECOA generators' fixed constraints allow users to better understand Weasel's behavior, results and limitations? Or do they persist in ascribing highly-nuanced human-like reasoning to the computer, possibly leading to failure to recognize brittle behavior. To what extent does allowing users to manipulate Weasel's soft constraints increase its utility, or decrease its usability? Would allowing users to control more constraints add to Weasel's utility or is there a point where the added complexity of the interface becomes more of a burden than a help to users?

Discussion and Conclusion

We have first examined what we view as the most important "bottom-line" issue: does Weasel improve decision making performance, and if so, for what users? Results show that Weasel results in solution quality gains for users with an intermediate level of domain experience (i.e., 1 ñ 6 years). Based on this result we see potential for use of Weasel in providing practice and training for analysts with an intermediate level of domain experience. However, with supervision from domain experts and with training on how to interpret Weasel's results; users of Weasel, and probably most MIPAS systems, should be trained to regard them as sometimes fallible suggestion generators rather than as oracles, just as they should regard their human counterparts. How successful this training is likely to be is yet another question: will it always be an uphill battle to prevent users from inappropriately regarding computer systems as infallible oracles?

Weasel may also provide benefits to domain experts, for example by reducing the number of times they are "surprised" by unexpected enemy actions. However, further evaluations are needed to determine what, if any benefits domain experts may derive. Lastly, when Weasel exhibited brittle behavior, it still resulted in an average solution quality increase, not a decrease as in the Layton, et al. experiment with FTP. We conclude from this that not all brittle solutions are created equal; the brittle solution examined in the Weasel study was an incomplete solution set. The one examined in the FTP study was a risky point solution. The latter may be a more dangerous form of brittleness than the former.

Decision support systems, of which MIPAS systems are an example, can have both positive and negative impacts on users' performance. The point for readers to take away is that it is that designers and users of MIPAS systems need to be aware that even the best designed system will sometimes exhibit brittle behavior, and both the positive and negative impacts of such systems must be carefully weighed in considering how the system should be used.

References

Ericsson, K. A. & H. A. Simon, "*Protocol Analysis: Verbal Reports as Data*" 1984, MIT Press, Cambridge, MA.

Hayes C.C.; and U. Ravinder, "Weasel: An Automated Planner that Users Can Guide," *IEEE Systems, Man and Cybernetics* (Washington, D.C.; October 5-8, 2003) Paper No. HMS Special P1-5,

Jones, P.M.; C.C. Hayes, D.C. Wilkins, R. Bargar, J. Sniezek, P. Asaro, O. Mengshoel, D. Kessler, M. Lucenti, I. Choi, N. Tu, O. Chernyshenko, M. Liang and J. Schlabach, "CoRAVEN: Modeling and Design of a Multimedia Intelligent Infrastructure for Collaborative Intelligence Analysis," *Federated Laboratories Annual Research Symposium* (Aberdeen, MD; Feb 1999) pp. 3-8.

Montgomery, D. C. (1991). *Design and Analysis of Experiments*. Wiley and Sons, pp. 176-194.

Parasuraman, R. (1997). "Humans and Automation: Use, Misuse, Disuse and Abuse," *Human Factors*, **39**(2): 230-253.

Schlabach, J.L.; C.C. Hayes, and D.E. Goldberg, "FOX-GA: A Genetic Algorithm for Generating and Analyzing Battlefield Courses of Action," *Evolutionary Computation,* **7**(1), pp. 45-68 (1998).

Smith, P. J. and N. D. Geddes, "A Cognitive Systems Engineering Approach to the Design of Decision Support Systems, in "*The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*," Jacko, J. A. and A. Sears (Eds) 2003, Lawrence Erlbaum Associates, Inc, pp. 656 ñ 676.

Smith, P. J., McCoy, C. E., & Layton, C. (1997). "Brittleness in the Design of Cooperative Problem-Solving Systems: The Effects on User Performance." *IEEE Transactions on Systems, Man, and Cybernetics ñ Part A: Systems and Humans*, **27**(3): 360-371.