

On the Role of Humans in Enterprise Control Systems: the Experience of INSPECT

Andre Valente, Jim Blythe, Yolanda Gil and William Swartout
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{valente,blythe,gil,swartout}@isi.edu

Abstract

In this paper, we use the example of a successful mixed-initiative plan evaluation tool for the domain of air campaign planning to argue that the human-in-the-loop is an important feature of enterprise control systems. Our tool, called INSPECT, evaluates air campaign plans and alerts the user about inconsistencies and potential problems. A generalization of INSPECT called PSMTool is also capable of limited interaction with a subject matter expert to capture new critiques of plans. The paper describes our work on INSPECT and PSMTool, analyzes the key contributions of these tools, and draws some conclusions about the role of mixed-initiative tools in enterprise control systems.

1 Introduction

Working with Air Force experts from the “Checkmate” Group at the Pentagon (HQ USAF XOOO), we developed INSPECT, a tool for evaluating and critiquing the air-related portion of a military campaign plan. INSPECT integrates several AI technologies. It was built using the EXPECT framework for knowledge-based systems development, that incorporates knowledge acquisition techniques, a description logic-based knowledge representation system, and a sophisticated problem-solving language and reasoner.

Our experience in developing INSPECT and several other systems led us to argue for *mixed-initiative systems* as the right approach for building enterprise control systems. The air campaign planning domain experts we worked with felt that it was essential that any system should keep human users “in the loop.” For example, they argued that tools for air campaign plan evaluation should allow users to understand their plans, identify critical factors, and analyze tradeoffs among options. More importantly, these domain experts argued that evaluation criteria need to be adapted in response to new situations, user preferences, and institutional prac-

tices. These characteristics indicate that enterprise control tools need to be *adaptable* and *responsive to changes in context*.

Based on this insight we drew from our experiences with INSPECT and other systems to design a reusable system for critiquing and evaluating plans, which can be instantiated and applied to any planning domain, such as air campaign planning or army course of action critiquing. We used this as a platform to develop a knowledge acquisition tool called PSMTool that allows domain experts to enter new evaluation criteria for plans. PSMTool makes use of general ontologies that we developed for planning and plan critiquing to help users define and organize the criteria through a simple dialog. In preliminary experiments at Fort Leavenworth BCBL conducted through DARPA’s HPKB project, we found that end users were able to use PSMTool with very little training to add new critiques that without the tool could only be added by a knowledge engineer. This system extends the approach of keeping human users in the loop and building tools that can easily be adapted in response to new conditions.

In this paper, we first describe the INSPECT system: the problem it solves, its design, architecture and implementation. We emphasize some of the lessons learned in this respect. Then we describe PSMTool and the further lessons from building such a highly adaptable tool. Finally we present our lessons learned for the construction of enterprise control systems.

2 The Air Campaign Planning Domain and the Design of INSPECT

INSPECT was developed as a tool to support the air campaign planning process. The tool had to provide support to the adoption and use of a methodology for military planning called strategies-to-tasks [12, 11]. In this approach, high-level national objectives are refined into lower level military objectives and then into operational tasks. Figure 1



Figure 1. Strategies-to-Tasks Example (from [Todd 94])

illustrates the hierarchy of objectives and how it links low level operational activities to higher level objectives. By encouraging planners to think about how low-level decisions relate to higher level objectives, this approach promotes air campaign planning that is more rational and helps avoid the sorts of mistakes that can arise from thinking about targeting too locally. The result should be air campaign plans that achieve the desired results while reducing risks and minimizing unnecessary damage. These plans are also more structured than traditional plans and capture more of the rationale behind the plan.

Our goal was to design an application that integrated well with this methodology and the air campaign planning process. Our domain experts had a number of useful suggestions about the design of the tool, most notably about issues regarding its interaction with the user. First, they requested that the user should not be locked in by the tool: the tool should present suggestions, not definitive answers, and users should be encouraged to think of alternatives. Second, they requested that all problems identified in the plan should be accompanied by an explanation and justification, so that the user could understand the reasoning behind the tool's recommendation and make an informed decision. Third, they asked that the problems should be presented as constructive

criticism, including suggestions of possible fixes to the problem. Not surprisingly, following these suggestions allowed us to make our tool more attractive to users.

3 INSPECT: An Air Campaign Plan Evaluation Tool

Based on the design decisions described above, we developed INSPECT (INtelligent System for air campaign Plans Evaluation based on expeCT). The architecture of INSPECT is shown in Figure 2.

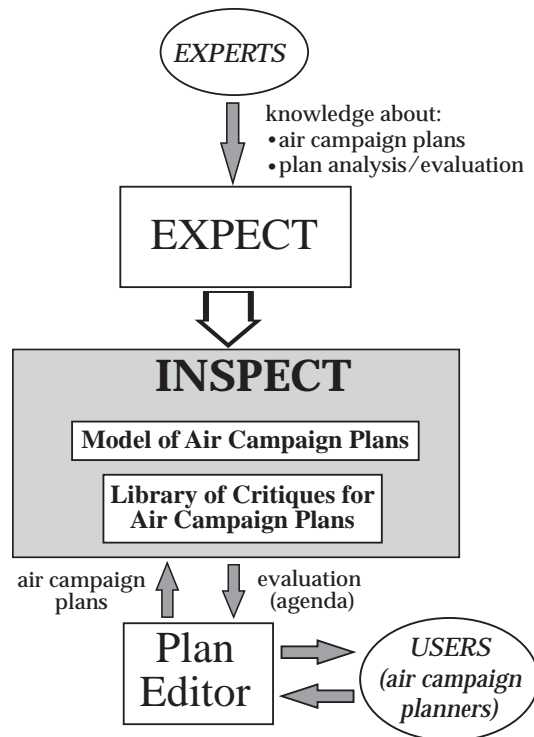


Figure 2. Architecture of INSPECT.

After entering air campaign objectives with a plan editor, a user invokes INSPECT to evaluate the plan. INSPECT looks for several types of problems in the plan. For example, it checks the hierarchical structure, identifies incomplete or incoherent objectives, and performs rough feasibility estimates based on the resources available for the campaign. INSPECT shows the user an agenda with all the problems found with the plan. The agenda items are marked according to their seriousness using a convention very familiar to Air Force pilots: WARNING, CAUTION, and NOTE (warnings requiring immediate attentions, and notes being non-critical). In addition to pointing out these problems, INSPECT can provide a detailed explanation of each problem and also suggest ways to fix it. Figure 3 shows a snapshot

of the agenda, including an explanation and suggested fixes for one of the agenda items.

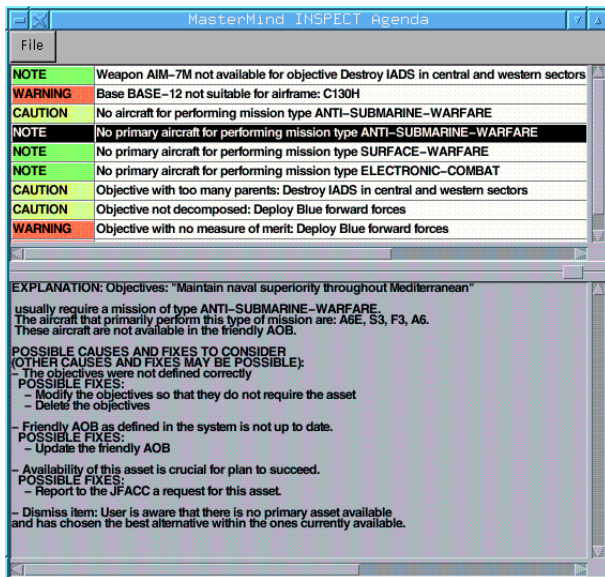


Figure 3. INSPECT’s Evaluation of an Air Campaign Plan. The explanation (lower frame) refers to the selected item on the agenda (upper frame).

The types of problems detected by INSPECT include:

- *Objective with no child/parent.* According to the strategy-to-task approach to air campaign planning, all objectives must be subordinated to higher objectives and be decomposed further (until a pre-specified “leaf” level).
- *No objective fulfilling one of the basic tenets of air power.* Air Force doctrine suggests that an air campaign plan must contain objectives for all the tenets of air power, such as force deployment and force protection.
- *Objective with too many parents.* An objective with too many parents is an indication that the parent objective is either too general (and should thus be divided) or that the connections are meant to emphasize priority rather than reflect a true decomposition.
- *Incompatible sequence restrictions.* This problem occurs when temporal constraints of two or more objectives are contradictory.
- *No adequate aircraft currently available for an objective.* This problem occurs when an objective requires a type of mission for which none of the aircraft available is considered adequate.
- *Incoherent decomposition.* In principle, an objective must be decomposed into objectives which are more specific or more detailed than their parent. This prob-

lem occurs when a parent objective is subsumed by one of its child objectives.

There are a number of benefits for air campaign planning that stem from using a tool like INSPECT:

- **Catch errors introduced during manual plan development:** Air campaign plans are very large and complex, and need to be changed over time. Because the plan creation process is manual, it is conducive to introducing errors and inconsistencies in the plan. In fact, INSPECT has found unintentional errors in every plan generated by our domain experts which would otherwise have gone unnoticed.
- **Raise the floor on plan quality:** An evaluation tool like INSPECT helps users avoid creating inconsistent or low-quality plans. Because it works with the higher levels of the plan, it can detect problems that could percolate down to the lower levels and be accentuated as the plan details are worked out.
- **Enforce “good” practices in plan construction:** INSPECT’s evaluations enforce the strategy-to-tasks methodology, and a number of style guidelines that experienced air campaign planners developed using it. Each of our domain experts liked to hear about the evaluation criteria suggested by other experts, as a new insight on how they could improve their own work on planning air campaigns. Our experts liked that INSPECT would point out that they were following their own standards as it evaluated their plans.
- **Training new planners:** INSPECT’s knowledge base captures the knowledge of experienced planners, and novice users can learn as they use the tool. INSPECT’s evaluations point out what experienced planners would see as serious flaws in the plan, and the explanations of each agenda item are designed to back up the evaluations with sources of information in the air campaign planning domain (doctrine, typical capabilities of weapon systems, etc.). INSPECT’s suggested fixes show what an experienced planner would do differently.

4 Building INSPECT with EXPECT

INSPECT integrates several AI technologies. It was built using the EXPECT framework for knowledge-based systems development, that incorporates knowledge acquisition techniques, a description logic-based knowledge representation system, and a sophisticated problem-solving language and reasoner. EXPECT also has a language to express problem solving goals that is based on case grammars. Below, we briefly describe these technologies, and how they were used.

The knowledge acquisition bottleneck is frequently cited as a major impediment to broad dissemination of AI tech-

nology. The EXPECT project [6, 7] is addressing this problem by developing a knowledge acquisition framework that empowers people to augment, modify and adapt knowledge based systems without needing to understand the details of the system's implementation. The key to EXPECT's approach is that it captures the design rationale for knowledge based systems, and uses that design knowledge to guide a user in augmenting the system. In addition to INSPECT, EXPECT has been used to build several knowledge based systems in domains such as transportation planning and battlefield assessment.

Most knowledge acquisition tools have a fixed set of guidelines or expectations about how knowledge should be added to a system. The problem with this approach is that it is inflexible, and limits the range of systems that can be supported. EXPECT takes a more flexible approach: it automatically derives a knowledge-based system from abstract domain facts and problem-solving methods. The derivation process is recorded so that EXPECT captures the normally implicit dependencies in a KBS, such as what factual knowledge is needed to support problem solving, and how factual knowledge is used in problem solving. EXPECT provides tools that use this information to guide the user in adding knowledge and tools (such as a natural language explanation facility) that help make EXPECT's representations more understandable to non-computer experts. For example, the system understands how various types of instances are used in problem solving, so when a new instance is added the acquisition tools can make sure that enough information is specified about the instance so that it can be used. In this way, EXPECT allows a user to add knowledge to a knowledge-based system without requiring him to understand all the details of how the knowledge interacts.

The EXPECT system is fully integrated with the LOOM knowledge representation system [9]. LOOM is an implementation of description logics, which emphasizes efficiency and expressiveness instead of completeness. In EXPECT, LOOM is used to represent the factual and definitional knowledge about a domain. For example, in INSPECT there are LOOM definitions about what are the elements of air campaign plans, what are objectives, what are known types of aircraft, what kinds of missions they fly, etc. This knowledge has proved to be an important byproduct of the INSPECT development. It has been used as a basis for the development of a broad ontology of air campaign planning, which is being used and further developed under the JFACC DARPA Program.

INSPECT was built using EXPECT as follows. General knowledge about air campaign plans, their structure and contents, as well as general domain knowledge about air flight was coded into a LOOM knowledge base. Procedural knowledge on how to evaluate the plan according to the critiques specified was acquired and represented as EXPECT

methods. The EXPECT system then put together these two types of knowledge, indicating whether there were any gaps or problems. The result of this process is an EXPECT model that records all the dependencies between procedural and domain knowledge. This model was then passed through the EXPECT compiler, that transformed it into efficient Lisp code that is able to solve the specified problem.

4.1 Representing Air Campaign Plans and Objectives

A very prominent contribution of our work resulted from integrating INSPECT with the plan editor tool. We designed a representation for air campaign plans and objectives that would allow both users and tools to exchange information about the plan. This representation has been adopted by other planning tools in the air campaign planning domain throughout the ARPI and JFACC programs, and is now seen as an important input to an ongoing effort in the US Air Force to create a common representation of objectives and tasks for air operations planning.

In integrating INSPECT with the plan editing tool (ACPT), we found a representation gap. Objectives in ACPT were represented with a sentence like "Gain air superiority in the western region", or "Destroy petroleum distribution facilities before the 15th day of the campaign". This was an unconstrained string, and the planner could write whatever came to his/her mind.

In order to be able to automate any interesting evaluation of the plan, we needed to capture the objective statement in a formal representation language. Parsing and interpreting the natural language sentence was too complex (and a new problem by itself). At the same time, the users were not willing to write their objectives in a form substantially different from the one they already used. The representation we proposed was therefore a middle-ground: we used a *case grammar*.¹ The basic idea of case grammars is that there is normally a limited number of roles (called *thematic* or *case* roles) that an argument of a verb can play with relation to the verb. That was definitely true for the objective statements, for several reasons. First, we found that the objective statements followed a very regular grammar of the form `<verb>` `<roles>`. Second, we found out that there were several regularities on the use of this structure. For example, only a handful of verbs (less than 30) are used. Third, each of these verbs introduces limitations with respect to the types of roles that can be used. For instance, most occurrences of the action type *Destroy* refer to a (physical) object type like "missile launch sites" or "military headquarters". We were

¹While case grammars have been dismissed as a general solution for natural language interpretation, they can be an interesting and powerful device in restricted settings such as the one we have in the air campaign planning domain.

able to establish reasonably exhaustive lists of terminals for each of the main types specified for role fillers. Fourth, we found that certain roles were actually *modifiers* that are used to specify restrictions or constraints on the objective. There are three types of restrictions, for time (e.g., within 21 days), space/area (in Western Region) and resources (using B-52s from base XYZ). A diagram showing the structure of the proposed grammar is shown in Figure 4.

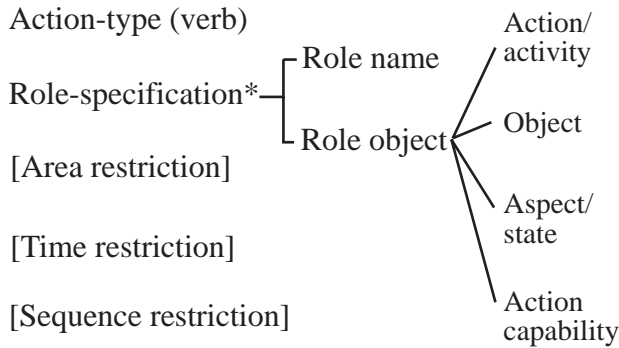


Figure 4. Overall structure of the case grammar to represent air campaign objectives.

There were several benefits to this representation. On the systemic side, it allowed the integration of ACPT and INSPECT. A syntax-oriented editor was built that helps the users enter valid sentences by offering lists of valid completions (according to the grammar) for the text being entered. This provides a proactive support for using for the grammar in the edition of objectives, without unnecessarily constraining the planner, who still has the liberty to write free text if he/she deems necessary. The case grammar became a shared representation that allowed other applications to make use of the more semantic representation.

Somewhat unforeseen were a number of methodological benefits, i.e., the benefits of using a grammar to the planning process itself, independently of any tools used. These benefits were so important that the structured representation took a life of its own and is often seen as a key contribution of our work on INSPECT. First, the knowledge acquisition process involved in building the representation forced the experts to explain and reflect over the way they write air campaign objectives. For instance, they came to the conclusion that frequently occurring objectives like “Conduct operations” should not be allowed because they in fact do not mean anything — in a air campaign plan, basically everything can be seen as conducting operations. Second, the resulting grammar embeds the notion of “reasonable” objectives, which had never been made explicit before then. Third, the additional structure provided by the grammar was considered particularly useful for training. Fourth, the case grammar became an input to an ongoing standardization process in

the Air Force on specifying valid types of tasks and objectives. Indeed, the success with this representation has led us to participate in the development of specialized representations for other elements of air campaign plans, as well as in extending the existing representation of objectives.

5 Adding new critiques with PSMTool

INSPECT supports the user in simple modification and maintenance tasks by virtue of the underlying EXPECT system. However we wanted to provide support for adding new critiques to a critiquing tool such as INSPECT. From our experiences with INSPECT as well as critiquers for logistics planning [8] and army courses of action [2] we observed that plan critiques often follow one of a set of generic patterns which could be captured using an ontology of planning and critiquing. This ontology can be used by a computer program to provide guidance for adding new critiques through dialogue with users. It can also help to organize the critiques and give a baseline estimate of the completeness of the critiquing system, all of which helps to increase user acceptance of the plans that are generated. More details about the ontologies and their coverage of real-world critiquing tasks can be found in [2].

We implemented a knowledge acquisition tool called PSMTool that uses these principles to acquire new critiques from users. The tool is domain-independent but requires that the domain has been aligned with the generic ontology of planning and critiquing, which it uses to express the new critiques. Even with this tool, specifying a new critique involves specifying problem-solving knowledge, so PSMTool also makes use of an editor developed to enter problem-solving knowledge for the Expect system using English-like syntax [4]. We tested PSMTool at Fort Leavenworth with the help of the Battle Command Battle Labs and found that Army officers with very little training were able to add significant new critiques to an Army battle course of action critiquer. In this section we describe the design and implementation of PSMTool and briefly describe the results of the user tests.

PSMTool uses two ontologies to represent general plan critiquing strategies: an ontology of plans and an ontology of critiques. Generic problem-solving knowledge is attached to the critiques as we now describe. The planning ontology, called Planet [1], was developed under the Darparran HPKB project and is a general ontology that allows both machine-generated and human-generated plans to be represented and also explicitly represents different assumptions made by planners. It has also been used as the basis of a translation service for software agents collaborating on a planning task [3].

The critiquing ontology represents domain-independent critiques of two types: those based on the structure of the

plan and those based on its use of resources. Consider, for example, the problems detected by INSPECT that are described in the previous section.

- The critiques *objective with no child*, *objective with too many parents* and *objective with incompatible sequence restrictions* are based on the structure of the plan. They are independent of the air campaign domain and the first two are included in our ontology of critiques along with the problem-solving knowledge required to implement them.
- The objective *incoherent decomposition* is also based on plan structure but requires domain knowledge to know if a parent objective is subsumed by one of its parents. In this case the critique ontology provides support for evaluating a set of objectives with respect to their parents, and the details are fleshed out as part of the domain definition.
- The objective *No adequate aircraft currently available for an objective* is a resource-based critique that requires domain knowledge to implement. Again, the critiquing ontology provides partial support.

PSMTool uses these ontologies to classify a new critique added by the user in a question-answering process. Once the critique is classified in the ontologies, the appropriate generic problem-solving knowledge can be applied and the domain-dependent knowledge that is required can be identified. In many cases, this process also breaks down the knowledge that needs to be acquired into small chunks that are easier for a domain expert to express.

Figure 5 shows the PSMTool interface while a new critique is being added. This is a critique from the army course of action domain that checks that the friendly forces have enough force ratio for each of the tasks in the battle, according to standard practice. The window on the left shows how a three-part script is being followed to add the critique. In the first part, four questions were answered that allowed PSMTool to classify the critique, showing that it is a quantity-based check made on each task in the plan. In the second part, PSMTool explains how the critique will be implemented based on this classification, and identifies pieces of problem-solving knowledge to acquire from the user to complete the critique. In the third part, which has not yet been reached, PSMTool will run the critique on an existing course of action so the user can check the results.

In the case of the force ratio critique, PSMTool asks for two pieces of problem-solving knowledge: how to estimate the amount of force ratio required for a task and how to estimate the amount available for a task. In the right-hand window in Figure 5, the English-based editor is being used to specify how to estimate the amount of force ratio available to a task. More details about the editor, how it produces an English paraphrase of the procedure body and a set of

alternatives that allow users to create and modify procedures through navigation can be found in [4].

6 Experiences with PSMTool

Preliminary experiments were run with PSMTool at Fort Leavenworth. Four subjects who had been given one day of training with Expect and the representation of the course of action (COA) domain were presented with four new critiques to add to the system. The subjects were army officers who were reasonably familiar with the COA generation and whose use of computers ranged from reading email to having been exposed to java. Two of the critiques were added using PSMTool and two without the tool. In order to minimize learning effects on our results, two of the subjects worked without PSMTool for two critiques and then worked with the tool, and two of the subjects worked the other way around.

Our results, although preliminary, show that not only can users add more critiques using the tool than without, they are also faster and less error-prone when using the tool. Moreover, the subjects reported the critiques that they added using the tool as being simpler, even though they were in fact isomorphic. This provides some support for the idea that, by classifying the new critiques with the ontologies as they are added, PSMTool helps users to organize the critiques in their own minds.

In a second experiment we tested whether PSMTool could be used with little or no training. A fifth subject, an army officer with no programming experience, was asked to add two of the same critiques as in the previous experiment, but without the day of training in Expect and the domain, instead the tool was demonstrated and explained for approximately 45 minutes. The subject successfully added the critiques, in time comparable with that of the earlier subjects who had received training.

We are currently designing more thorough user experiments to test these highly encouraging results. However they seem to indicate that the goal of allowing subject matter experts to add new evaluation criteria to a plan critiquer by directly interacting with an automated system is attainable, even if the experts have little training with the system. If this is the case, we can hope to greatly increase the range of scenarios in which an enterprise control system can be usefully applied, by allowing it to be modifiable by its users.

7 Lessons Learned: Mixed-Initiative Tools for Enterprise Control Systems

In order to obtain cost savings and scalability, the ideal scenario for enterprise control systems seems to argue for a completely automated control system. However, our experience with INSPECT has shown that in large-scale, real-world

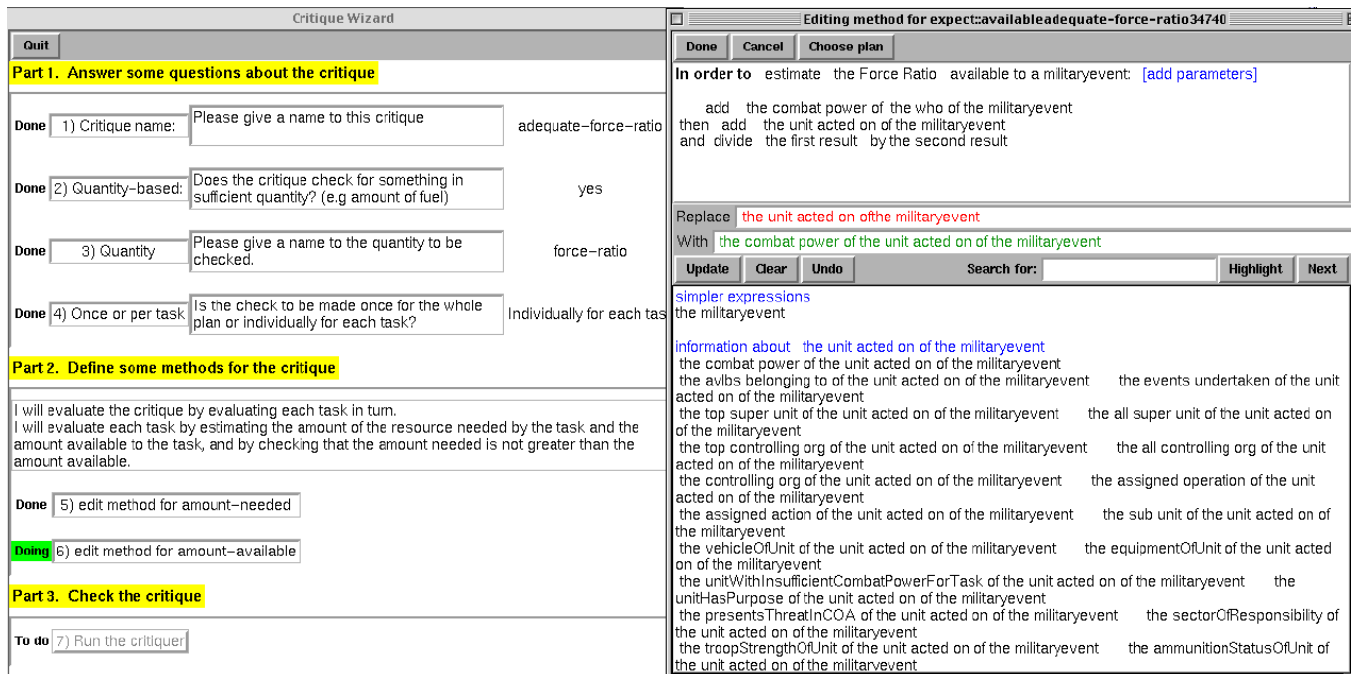


Figure 5. Adding a new critique with PSMTool. The window on the left shows the questions that were asked to define the critique and the new knowledge that is required. In the window on the right, some of that knowledge is entered using the English-based editor.

domains, completely automated control is often impracticable due to the scope and breadth of knowledge required. We are not alone with this view. For example, [5, 10] argue that for planning — a key task in enterprise control — a mixed-initiative approach where machines and people work together is often more desirable.

The key insight is that people can understand the enterprise problems and their *context* more broadly than machines, and thus will be able to make better judgments about certain decisions. Machines, on the other hand, are able to carry out tasks with a well-defined context much more effectively. Another issue is *adaptability*: humans are able to understand that their knowledge about a certain kind of control process is no longer valid and seek to revise this knowledge at the light of the new information. Machines simply do not have that capability at the moment, and thus behave with brittleness.

Morover, even if we do develop techniques that can overcome these limitations, it is an open question whether giving all the power to the machine is a desirable choice. Humans tend to be afraid of giving up control on certain key decisions, and particularly so when they do not have access to an explanation as to why the decision was made, what other decisions were possible, and what are other choices. Black-boxes are simply not acceptable for many key control decisions, particularly in applications such as military air

campaign planning, where lives are at stake.

As a result of this view on enterprise control systems, we argue that these systems should adopt a *mixed-initiative approach* with two main characteristics. First, they must be designed to work with people, rather than completely taking over processing. This in turn means that the products of our tools must be *understandable* by people and it must be possible for people to easily input information and decisions into the tools. Second, because it is impossible to anticipate in advance all the knowledge a system might need in a broad domain, and because knowledge frequently changes, our goal is to provide *knowledge acquisition tools* that allow for users to augment and adapt a system’s knowledge in response to new situations and new needs.

Acknowledgments

The work reported here relates to research sponsored by the Defense Advanced Research Projects Agency (DARPA) under Ft. Huachuca Contract DABT63-95-C-0059 and Air Force Research Laboratory Agreements F30602-97-C-0118 and F30602-97-C-0068. Many thanks to all Checkmate members who helped us in the process; most Col Plebanek, who allowed us to have this interaction, LtCol Cardenas, who coordinated the knowledge acquisition sessions and

was a central expert, and the experts we worked with more closely: Maj Allison, LtCol Alred, LtCol Cardenas, Maj Cunico, and Maj Jackson. We are also grateful to the staff of the Battle Command Battle Labs at Fort Leavenworth for their assistance in running user experiments, especially Capt Rasch and Col Duquette. The views and conclusions contained in this article are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

- [1] J. Blythe and Y. Gil. Planet: A shareable and reusable ontology for representing plans. Technical report, Expect internal report, 1999.
- [2] J. Blythe and Y. Gil. A problem-solving method for plan evaluation and critiquing. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1999.
- [3] J. Blythe, Y. Gil, H. Chalupsky, and R. MacGregor. Supporting translation among planning agents. In *Submitted to the Fifth International Conference on Artificial Intelligence Planning Systems*, 2000.
- [4] J. Blythe and S. Ramachandran. Knowledge acquisition using and english-based method editor. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1999.
- [5] G. Ferguson, J. Allen, and B. Miller. Trains-95: Towards a mixed-initiative planning assistant. In B. Drabble, editor, *Proc. Third International Conference on Artificial Intelligence Planning Systems*, University of Edinburgh, May 1996. AAAI Press.
- [6] Y. Gil and E. Melz. Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proc. Thirteenth National Conference on Artificial Intelligence*. AAAI Press, 1996.
- [7] Y. Gil and B. Swartout. Expect: Explicit representations for flexible acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1995.
- [8] Y. Gil and W. R. Swartout. Expect: A reflective architecture for knowledge acquisition. In *Proceedings of the 1994 Workshop of the ARPA-Rome Laboratory Knowledge-Based Planning and Scheduling Initiative*, Tucson, AZ, February 1994.
- [9] R. MacGregor and R. Bates. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, June 1991.
- [10] K. Myers. Strategic advice for hierarchical planners. In *Proceedings of the International Conference on Knowledge Representation*, 1996.
- [11] D. Thaler. Strategies to tasks, a framework for linking means and ends. technical report, RAND Corporation, 1993.
- [12] D. Todd. Strategies-to-tasks baseline for usaf planning. Internal document, Strategic Planning Division, HQ United States Air Force, 1994.