INSPECT: A Tool to Evaluate Air Campaign Plans

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

Abstract

INSPECT is a mixed-initiative plan evaluation tool in the domain of air campaign planning that has been a central component of several major DARPA demonstrations of integrated planning environments and tools. The creation process of an air campaign plan is manually driven at its higher levels, and because plans are complex and always changing they often (our experience says always) contain errors or inconsistencies. INSPECT evaluates user-generated plans and alerts the user about inconsistencies and potential problems. INSPECT has received wide acceptance by air campaign planning experts, and is currently undergoing new extensions and further integrations with other tools in this domain. The paper describes our work on INSPECT, analyzes the key contributions of this tool, and draws some conclusions about the design and integration of planning applications.

Introduction

EXPECT is a framework to develop knowledge based systems that we have applied to build applications in several planning domains. Working with Air Force experts from the "Checkmate" Group at the Pentagon (HQ USAF XOOC), we developed INSPECT, a tool for evaluating and critiquing the air-related portion of a military campaign plan. Although INSPECT was originally intended to be just a testbed, it has succeeded beyond our expectations. INSPECT has been enthusiastically received by domain experts and has been a central component of the Fourth Integrated Feasibility Demonstration of the DARPA/Rome Laboratory Planning Initiative (ARPI), The Jumpstart Demonstration of the DARPA JFACC program, and more recently the ARPI TIE97-1 demonstration. Subsequently, INSPECT received independent funding to extend its capabilities under the DARPA Joint Force Air Component Commander (JFACC) program.

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. Further, in the development of INSPECT we defined a specialized representation for air campaign objectives based on case grammars (Fillmore 1968). Finally, because INSPECT was part of a larger integrated air campaign planning system, integration issues arose in coupling INSPECT with other AI and conventional systems.

There were several important factors that contributed to INSPECT's success and are described throughout the paper: 1) taking a mixed-initiative approach, 2) working in a domain where experts had found the value added in capturing information about the plan structure and rationale, 3) integrating our application with an existing successful tool, 4) designing the application together with the experts and future users, 5) surprising the experts by finding ways to improve every one of their plans by using our tool, and 6) developing a standard representation of planning objectives that is having impact in the development of air campaign planning tools.

In this paper, we describe the INSPECT system: the problem it solves, its design, architecture and implementation. We emphasize some of the lessons learned in designing and developing this system that we hope will be useful to developers of planning applications. First, we discuss our view on decision support tools for planning. Second, we describe the air campaign planning domain, and how it imposes requirements on the design of INSPECT. Third, we describe the INSPECT system. Fourth, we discuss integration issues in building INSPECT. Fifth, we discuss the benefits of INSPECT. Finally, we present our conclusions.

Decision Support Tools for Planning

The automatic generation of plans has traditionally been a major focus of AI planning research. However, in large-scale, real-world domains, completely automated planning is often impracticable due to the scope and breadth of knowledge required. Instead, a mixed-initiative approach where machines and people work together is often more desirable to solve planning problems (Ferguson, Allen, & Miller 1996; Myers 1996). Because people will understand the planning problem and context more broadly than machines, they will be able to make better judgments about certain decisions. Machines, on the other hand, will be able to carry out mechanical tasks more effectively. For the collaboration to work, people need to be able to input their own plans and decisions, and computerized modules then need to be able to make use of those plans and decisions in their own processing.

As a result of this view on planning, the kinds of tools that we have seeked to develop have the following characteristics:

- Decision *support* tools that help people make a decision are often more desirable than tools that automatically *make* a decision for them.
- The tools we construct 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.
- 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.

The domain experts we worked with endorsed these views and felt that it was essential that any system should keep human users "in the loop." They also recognized that the generation of an initial plan is only part of the planning process. Another very important area where decision support tools are needed is *plan evaluation*. Plan evaluation tools allow users to understand their plans, identify critical factors, and analyze tradeoffs among options. Evaluation criteria need to be adapted in response to new situations, user preferences, and institutional practices. These characteristics indicate that plan evaluation tools can benefit from automated knowledge acquisition tools, making them an interesting area of application for applying and experimenting with the EXPECT framework.

The Air Campaign Planning Domain and the Design of INSPECT

There were two important aspects that made the domain of air campaign planning an excellent environment for our research: the strategies-to-tasks methodology and the Air Campaign Planning Tool (ACPT). The strategies-to-tasks methodology represents a move towards capturing more information about the rationale and the decisions behind the air campaign plan, and ACPT is a tool that allows users to specify this information. As a result, our systems had access to information about the plan's objectives and their realization that turns out to be very useful to evaluate the overall plan. As we will explain in our final discussion, we still had to extend significantly the ACPT representations (and still continue to do so) in order for our evaluations to be useful and more meaningful. What turned out to be crucial is that the strategies-to-tasks framework and ACPT provided a core structure that turned out to be very valuable. Moreover, our domain experts were already practitioners of this approach and already understood the importance of representing and capturing information about a task. The rest of this section describes more details about strategies-to-tasks and ACPT.

Traditionally, planning an air campaign has involved identifying a set of possible targets, selecting from that set a subset of targets to attack, and then assigning aircraft, crews and munitions to actually carry out the missions. The problem with this approach has been that this process does not encourage planners to think about how the selection of a particular target will help the overall goals of the military campaign. Because targets are selected without relating them back to higher objectives, sometimes the targets selected actually hinder the operation. For example, an enemy bridge might be destroyed early in a campaign that could have been used by friendly forces later if it had been preserved.

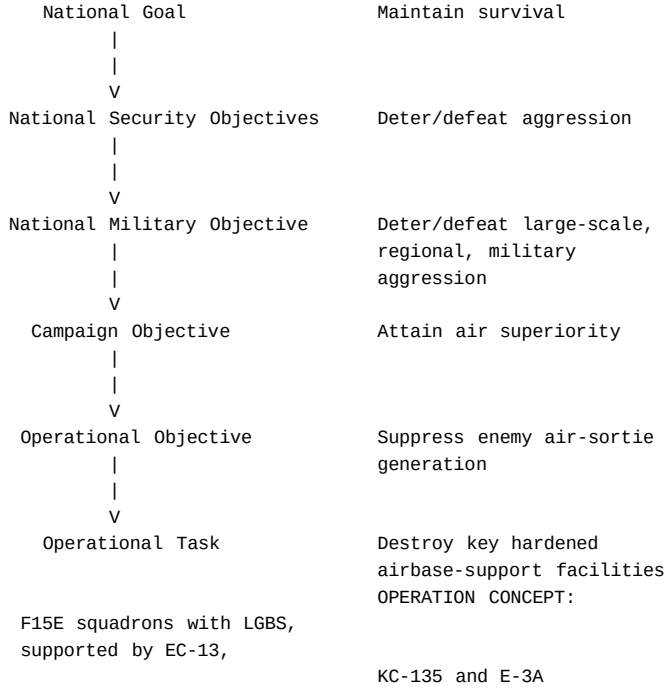

Figure 1: Strategies-to-Tasks Example (from [Todd 94])

Recently, the Air Force has begun to investigate a new approach to air campaign planning called the strategies-to-tasks methodology (Todd 1994; Thaler 1993). In this approach, high-level national objectives are refined into lower level military objectives and then into operational tasks. Figure 1 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.

Several editors have been developed to create plans, using the strategies-to-tasks approach. The original one was the Air Campaign Planning Tool (ACPT). Several other tools based on ACPT were developed later, among them the Joint Planning Tool (JPT), the Joint Operations Editor (JOE), and the Mastermind+ Plan editor.[1] Each of these tools is ba-

[1] INSPECT has been successfully integrated to date with ACPT, JOE, and Mastermind+

sically a browser/editor for air campaign plans. It allows users to create objectives and decompose them into subobjectives. The user can specify temporal constraints to sequence plan steps. The editor allows a user to view his plan in a textual format or as a graph showing the interdependencies between the steps of the user's plan. These editors provide a way of browsing and editing a plan, but they do not understand the plan in any sense: they provide no help in refining objectives or in finding or correcting possible errors in the plan.

Our goal was to design an application that extended the functionality of ACPT and was well integrated with it.

Selecting an Application Area

Involving the experts and users early in INSPECT's design was critical to its ultimate success. When we first analyzed the air campaign planning domain and ACPT, we saw many possibilities for evaluation tools and had our own opinion about what kind of system we should build. However, we felt that our views were based on our limited knowledge about this domain and the planning process so we decided to consult with our domain experts, who were also among the potential users for the system. We presented them with six tasks we identified in the domain that might be automated by a plan evaluation tool:

1. prioritizing targets: determine target priority based on relevant criteria.
2. feasibility: provide a rough evaluation of the feasibility of the plan at the early stages of plan development, when only higher level objectives are specified.
3. consistency checking: detect errors in the plan that are introduced during the plan editing process.
4. compliance with rules of engagement: evaluate whether a given plan violates any constraint (things that the plan cannot include) or restraint (things that should not be done) as dictated by a commander.
5. compliance with doctrine: evaluate whether a plan complies with different aspects of military doctrine.
6. incompleteness detection: evaluate the plan with respect to completeness and present the user with an agenda[2] of missing objectives and components that would guide further plan edition.

In making the choice, we asked the domain experts to take into account several issues. First, that the *knowledge should be available*, both in terms of being agreed and understood and in terms of having subject matter experts available to us. Second, that the system would have *high impact*. Users should get a high payoff in terms of added value to ACPT. Finally, we asked that the problem should be *challenging for our research* on knowledge acquisition tools to modify knowledge bases. We were interested in an application that could have initially a reasonable amount of knowledge about the task, but that would need to be adapted and extended by users.

[2]This idea was inspired on EXPECT, where the knowledge acquisition tools notify users about missing knowledge and other problems in a knowledge base through an agenda.

Our domain experts immediately focused on the consistency checking option. They had reasons for ruling out options that we could not have anticipated ourselves. Enforcing doctrine was seen as a potential quagmire. They saw difficulty in building a useful tool to evaluate compliance with rules of engagement because we would need to use information that was not readily available online.

The most surprising reaction was to the target prioritization. Because so much of the air campaign revolves around selecting targets and sequencing them appropriately, we thought that this option would be most attractive to pursue. They had the insight of realizing that it was not clear that anyone could articulate the knowledge required to prioritize targets. Many months later, our knowledge acquisition sessions encovered that in fact the priorities of targets are derived from the higher objectives that the targets are serving. The structure of the plan can be extended to capture important information about the importance, sequencing, and priorities of these objectives. Target priorities should be derived from them.

The final decision was to develop an evaluation tool to check consistency of plans. We later extended the tool to check completeness and to analyze rough feasibility.

This decision satisfied the requirements of knowledge availability and high impact, two of the three that we outlined above. Although the evaluation knowledge that we needed to incorporate in the tool would have to be teased out from our domain experts based on their experience, they were quite confident that they would be able to ultimately articulate it to us. They often visited other groups within the Air Force to provide support on using ACPT and the strategies-to-tasks approach, and they felt they could capture with our tool the advice and explanations that they provided to these groups.

The tool would have high impact for several reasons. First, the complexity and size of air campaign plans (typically hundreds or thousands of nodes) made it difficult to see errors. A tool that could ferret out commonly occurring errors could have a big impact on plan quality. As noted above, existing tools such as ACPT provided little support for error detection. Second, because the strategy-to-task approach is new, not all planners are experienced with it, and inexperienced users frequently make mistakes, some of which could easily be detected by a tool. For inexperienced users, it was felt that the tools could also have educational benefits.

What was not so clear to us at that time is that this application would satisfy our last (and very important) requirement, i.e., that it would exercise the benefits of our technology and research in the EXPECT knowledge acquisition tools. As we explain below, there were additional benefits from using EXPECT that we had not anticipated.

Our User's Requirements

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.

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.

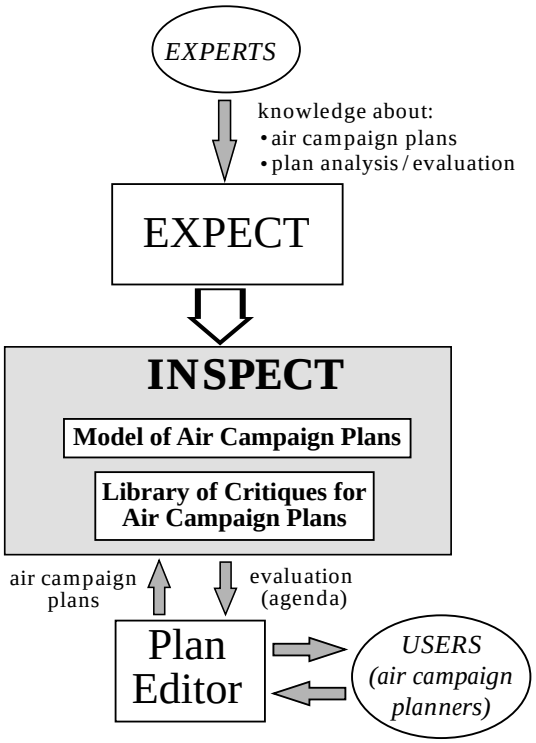

Figure 2: Architecture of INSPECT.

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 our 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.

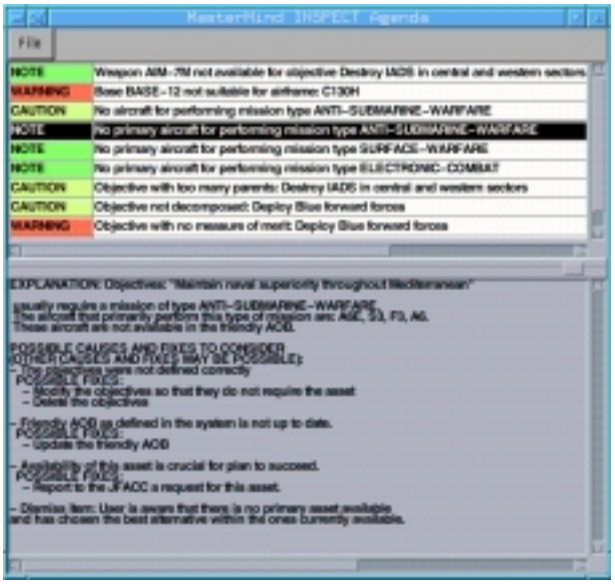

Figure 3: INSPECT's Evaluation of an Air Campaign Plan. The explanation (lower frame) refers to the selected item on the agenda (upper frame).

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

Our domain experts suggested these types of problems because they anticipated that novice air campaign planners would make these kinds of mistakes. What they were not

expecting is that *INSPECT found these kinds of problems in every plan created by the experts themselves*.

Currently, we are extending INSPECT to cover several new areas. These include being able to critique other types of objectives (initial versions were restricted to force application ("attack") objectives, being able to critique logistics issues (e.g. if an objective requires an aircraft to land in a base that has inadequate runways for that type of aircraft), and having more "global" critiques, that evaluate the overall plan (e.g., checking for balance in the use of forces).

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 (Fillmore 1968). Inspired in this representation, we developed a specialized representation for air campaign objectives. 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 technology. The EXPECT project (Gil & Melz 1996; Gil & Swartout 1995) 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 (MacGregor & Bates 1991). 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 fight 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.

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*.[3] The basic idea of case grammars is that there is

[3]While case grammars have been dismissed as a general solu-

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 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 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.

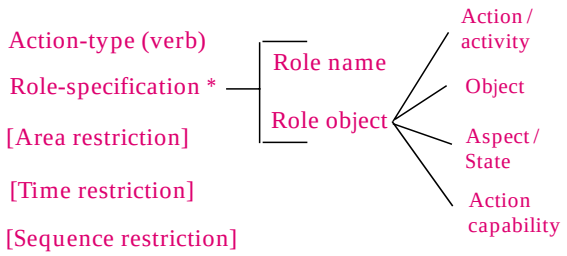

Figure 4: Overall structure of the case grammar to represent air campaign objectives.

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, because 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.

tion 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.

Benefits of INSPECT

There are a number of benefits for air campaign planning that stem from adopting the user of 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 manually done, 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.

INSPECT benefits from several AI technologies that are integrated in EXPECT:

- **Explicit representations of knowledge**: A cornerstone to the EXPECT approach is to represent different types

of knowledge separately and explicitly. For example, INSPECT checks that a plan fulfills all the basic tenets of air power. Instead of hand-coding a rule that checks for all five tenets, EXPECT represents the tenets as a separate piece of knowledge and in doing so it represents not only the knowledge required to produce the evaluation but it is capturing first principles behind these evaluations. Another example is representing explicitly the structure of each objective in the plan. For example, an objective such as "Destroy enemy naval activity in the Western Sea throughout conflict" is referring to "the Western Sea", which is a kind of geospatial region that is related to the theater of operations and so on. EXPECT's representations are tightly coupled with ontologies of the air campaign planning domain.

- **Description-based reasoning**: A significant part of the reasoning that takes part in generating INSPECT's evaluations is to detect the presence and the absence of many types of objective patterns in the plan. INSPECT's knowledge base includes descriptions of many different types of objectives expressed as LOOM class descriptions. INSPECT relies on LOOM's description logic technology to classify the objective definitions and match them against the objectives in the plan at hand.
- **Explanation and knowledge acquisition**: EXPECT's explicit representations were designed to support explanation and knowledge acquisition. INSPECT's explanations of its evaluations were initially hand-coded, and work is under way to generate them automatically based on EXPECT's representations and reasoning behind INSPECT's evaluations. Another important area of our current work on INSPECT is to use EXPECT's knowledge acquisition tools to extend and update INSPECT to add new evaluation criteria and to adapt it to different crisis, different users and command structures, and different institutional practices.
- **Knowledge compilation**: Using a high-level language to represent knowledge has many advantages, but can slow down execution significantly. EXPECT's language is designed to support explanation and knowledge acquisition, yet it does not affect performance because it can be compiled into efficient code. We were able to reduce the execution times of INSPECT by two orders of magnitude when we run the EXPECT compiler and executed the generated code.

Conclusion

Our work in INSPECT has clearly demonstrated the value of a plan evaluation tool. An additional conclusion from our experience with INSPECT is the value of shared plan representations. In our current work, we continue to develop and extend current air campaign plan representations to capture additional information, including strategy, additional interdependencies and constraints. At the same time, INSPECT is being extended to produce new evaluations based on this information about the plan. A very interesting new direction is the tight integration of the manually-driven plan creation process that ACPT and INSPECT facilitate with automated generative planning tools, whis is encovering new challenges for plan representation and sharing across applications.

Acknowledgments

INSPECT was initially developed under contract DABT63-95-C-0059, as part of the Fourth Integrated Feasibility Demonstration (IFD-4) of the DARPA/Rome Laboratory Planning Initiative. Knowledge acquisition was a critical task in building INSPECT, and demanded much effort from the experts. 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. Many thanks to all other participants in IFD-4, specially Lou Hoebel and the ISX Team (in particular Jim Shoop, Doug Holmes, Gary Edwards and Joe Roberts).

References

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95 towards a mixed-initiative planning assistant. In *Proceedings of AIPS-96*.

Fillmore, C. 1968. The case for case. In *Universals of Linguistic Theory*. Holt, New York.

Gil, Y., and Melz, E. 1996. Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Gil, Y., and Swartout, W. 1995. Expect: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop*.

MacGregor, R., and Bates, R. 1991. Inside the LOOM description classifier. *SIGART Bulletin* 2(3):88–92.

Myers, K. 1996. Strategic advice for hierarchical planners. In *Proceedings of KR-96*.

Thaler, D. 1993. Strategies to tasks, a framework for linking means and ends. technical report, RAND Corporation.

Todd, D. 1994. Strategies-to-tasks baseline for usaf planning. Internal document, Strategic Planning Division, HQ United States Air Force.