N88-17250

ADVANCED DECISION AIDING TECHNIQUES APPLICABLE TO SPACE

Robert J. Kruchten, Major, USAF
Advanced Technologies Program Manager
Rome Air Development Center (RADC/COAD)
Griffiss Air Force Base, New York 13441-5700

1. ABSTRACT: The Command and Control Directorate of RADC has had an intensive program to show the feasibility of applying advanced technology to Air Force decision aiding situations. Some aspects of the program, such as Satellite Autonomy, are directly applicable to space systems. Other parts of the program, while not directed toward space applications, have developed techniques which could be used in space applications. For example, RADC has shown the feasibility of decision aids that combine the advantages of laser disks and computer generated graphics; decision aids that interface object-oriented programs with expert systems; decision aids that solve path optimization problems; etc. The purpose of this paper is to review some of the key techniques that could be used in space applications. It reviews current applications, their advantages and disadvantages, and gives examples of possible space applications. The emphasis is to share RADC experience in Decision Aiding techniques.

2. INTRODUCTION:

RADC has undertaken a major effort in the area of decision aid development. As part of this effort, RADC has developed a variety of decision aids for specific problems. These aids were meant to prove the feasibility of using high and low technology techniques to improve the decision process. In addition, RADC has extensively studied the problem of satellite autonomy through both in-house research and through contracted efforts. The techniques used in all these programs ranged from relatively simple operations research techniques (eg multi-attribute utility analysis, etc) to advanced artificial intelligence techniques (eg model-based reasoning, etc). Interestingly, it was often found that the inclusion of a known technique into a decision aid offered benefits far beyond the immediate problem being addressed. Unfortunately, it was sometimes also found that there were subtle disadvantages to some of these techniques. Often the benefits and disadvantages remain hidden until evaluation by the prospective user. The remainder of this paper examines some techniques that have been especially useful.

3. ADVANCED DECISION AID TECHNIQUES:

3.1. Object-Oriented Programming

3.1.1. DESCRIPTION

Object-oriented programming is a method of programming that associates code with real world objects. The individual objects are things that have behavioral characteristics. These characteristics are coded in the form of methods and procedures. The behavioral characteristics are invoked by sending messages to the objects. For example, a variety of objects may have procedures to move. A message "move" (with appropriate arguments) sent to one of the objects will automatically invoke the proper procedures to move that particular object. The sender of the message does not have to have any knowledge about how the move is accomplished. A key feature of object-oriented programming is that individual objects can inherit characteristics from other objects. Programs written in object-oriented programming are generally easy to create, extremely easy to change, and dramatically reduce memory requirements. RADC has a number of programs using object-oriented programming some of which are described below.

One RADC program uses objects for map representations. Normally a digital map data base will indicate what features are located at each location. Thus if the system is at a location that is the intersection of two roads, it will know the roads it is on. A person looking at the map could tell basically where the roads went, what they connected with, distances, etc. Unfortunately, the computer would have to find the answer to these questions via exhaustive searches. RADC has a program that captures key map features as objects. Each object knows about itself and has procedures to determine connections with other objects and to answer questions about itself. In this system, when the computer is at a road intersection, it can send messages to each road object. It is thus possible to immediately determine which road goes to what city, the distances involved, and the route of travel.

Another RADC program uses object-oriented programming to find the best path for an aircraft through a dense series of ground to air threats. In this program, the threats are "smart" objects that know their capabilities against a variety of different aircraft and, more importantly, know the best means for an aircraft to avoid the threat. As an aircraft approaches the threat, it sends a message to the threat requesting information on paths to avoid the threat. The message contains arguments giving the aircraft type and capabilities. The threat returns alternative paths, their lethality, and their distance. The aircraft can use this information to select the best path.

Finally RADC has an object-oriented program that projects enemy activity into the future. It has various enemy units defined as objects that have behavior characteristics similar to real enemy units. These characteristics allow the units to move about within an area, given an objective. Each unit knows how to find its own path to its objective. It knows its rate of travel over various terrain, the impact of other units using the same roads, etc. All units are given objectives and then the system is queried to show unit locations over time. This system interacts with a rules-based mission planner to plan aircraft missions. It allows the planner to plan based on realistic projections of where the enemy will be.

3.1.2. Advantages

Another advantage is that an object-oriented program tends to be very compact. For example, a complex simulation using object-oriented programming could be a factor of ten or more smaller than a similar program using traditional techniques.

Another potential advantage of the object-oriented concept is that it will be possible to combine it with parallel processing techniques. To date, this has been done in only a most rudimentary fashion, but it appears to be feasible to hand off separate object processes to separate processors.

Object-oriented programming allows the developer to use more natural thought processes to visualize the problem and develop a program approach. This is possible because it releases him from many of the mundane tasks of linking together the pieces of an object.

Finally, the object-oriented system is easy to code and maintain. Changes made to an object are automatically propagated to all the appropriate instances of the object. Code is easier to write because the problem is broken down into smaller, more reasonable pieces. Thus, overall it is quicker to develop and get running than more conventional methods of programming.

3.1.3. Disadvantages

The primary disadvantages of object-oriented programming is the lack of robust language implementations to support it. The primary work is in ZETALISP FLAVORS, ZEROX LOOPS, and SMALLTALK. Although these implementations are very good, they do not fully implement the concepts. Other, more common languages offer only minimal support to this concept.

Another disadvantage is that it is difficult to fully predict the runtime performance of an object-oriented system. The problem occurs because of the very advantages of the approach. Object-oriented programming greatly simplifies the developer's tasks and allow him to actually build a program that exceeds the capabilities of his processor to execute in real time.

3.1.4. Space Applications

Object-oriented programming combining with some research in truth maintenance systems (TMS) offers hope of resolving unanticipated anomalies. The traditional approach to anomaly resolution is to define the anticipated anomalies, their indications, and the proper corrective actions. Unfortunately, most complex anomalies are unanticipated (we tend to design out the ones we anticipate). However, using object-oriented programming to create multiple models of a system (thermal, electrical, structural, etc); we define how the system should work. When unexpected situations arise, it will possible to use TMS concepts to reason about the differences between the model predictions and the actual data thus defining the problem. Goal directed search techniques can then be used to generate a solution and the models can be used to test the solutions.

One key application of the object-oriented system is that it can be used to create a highly interactive simulation of any system. Most systems can be broken down into separate modules. These modules can, in turn, be further broken down into more detailed modules. At lower levels there is much in common between the modules. Object-oriented programming allows for generic objects (modules) to be created. Instances of these modules can then be easily made and interconnected. They can inherit characteristics as needed. The end result is a software simulation of a system. Object-oriented programming is ideal for large interactive simulations. Its modular construction allows a complex system to be rapidly built and de-bugged. In fact, it is possible to create objects that could be used in multiple space programs. A related application is to allow a satellite configuration to be designated by an interactive object-oriented model. As the configuration changes (consumable, position, failures, etc), it is easy to modify the model and thus system performance can be projected at any time.

A related application to the simulation described above is simulation used to support systems engineering design studies and tradeoffs. Complex architectures can be built using object-oriented techniques relatively easily. These can serve as system "breadboards" to check on overall system performance and design tradeoffs.The simulations themselves are not meant to be complete or to have perfect fidelity. The key is to build the simulations early in the design process in order to support the design.

Another application is in display technology. A circuit that is built and displayed as an object can be easily designated by an operator. This would give him immediate access to all relevant information on that circuit. Object-oriented representations can also be used in search routines by a computer. Thus a complex network of objects could be searched. When the system is at any particular node in the net it has the immediate and automatic capability to query that node.

3.2. Natural Language

3.2.1. Description

Traditional man-machine interface for software has been menu driven, special function keys, touch-screen, mouse/track-ball, or rigid syntax typed input. There has been much research into a more "natural" form of input. Natural Language is the machine analysis of typed sentences. That is, the transformation of a sentence into a machine usable form. RADC has developed systems that have shown the feasibility of using Natural Language as a way of communicating between the man and the machine. These systems allow the machine to carry on a dialog with a person even though the individual sentences contain ambiguity. They do this by establishing a loose context relationship between the sentences being input, previous sentences, previous machine responses, and knowledge of the domain. The net result is a relatively free flowing, conversational type dialog.

3.2.2. Advantages

Natural Language allows an operator with virtually no training to be immediately effective. Syntactic errors are virtually eliminated and no pre-defined formats are required to be memorized.

A secondary advantage of the Natural Language research is its development of solutions to problems that must be evaluated in context. Since most of our sentences and words can only be understood in context, Natural Language researchers were forced to develop techniques for establishing context relationships. Those techniques can be fruitfully used in other applications as defined below.

3.2.3. Disadvantages

Natural Language involves both input and output. As an output medium, it allows for unanticipated, machine generated statements or queries. As the primary means of machine input, Natural Language is far from ideal. Most operators are not skilled typists and cannot input sentences quickly or accurately. In addition, if the task is repetitive, a person will be much faster and accurate with standardized input or output schemes.

3.2.4. Space Applications

One obvious application of Natural Language is for man-machine interfaces. Here it has the advantages and disadvantages described above. A not so obvious application is in data interpretation. Any one data channel or sensor output can only be interpreted in context of all other data, past history, and knowledge of the domain (eg. what happens during an eclipse, etc). Technically, data interpretation has many of the same problems as language interpretation. Natural Language techniques allow loose context relationships like those in involving telemetry to be created.

3.3. Video Disk Displays

3.3.1. Description

Most graphic displays are generated using digital data that is either vector plotted or bitblt'd to fill a screen. In the first case, graphics draw commands are used to draw an image using a digital data base that defines a series of vectors to be drawn. In the second case, the image is created from a large array representing the individual pixels to be drawn. Bitblt has the advantage of creating a display very quickly, but it requires large amounts of memory if there are many different displays to be created. Vector plotting is more common because it requires less memory, but it is usually quite slow to create an individual image. In addition, complex images created by a vector plot also involve large amounts of data. Normally resolution of the image is sacrificed to reduce the data overhead. The impact is that curved lines appear as stepped lines and detail is eliminated from the image.

RADC has recently developed several tactical planning aids that required tactical maps to be displayed with icons superimposed on the maps. The technique used was to combine a commercial laser disk image with computer generated icons. The laser disk is a standard commercial NTSC laser disk with each frame containing an image of a different map (or a different scale). Laser disks represent an extremely dense medium and contain up to 108,000 separate NTSC images on a single disk. It is possible to select any one frame in less than one second. The laser disk image is essentially anything that could be displayed by a commercial television (graphics, photo, etc). As implemented, the laser disk image is displayed on a high resolution screen with NTSC resolution. The computer generated graphics are displayed with high resolution on the same screen. Thus, it is possible to rapidly display many complex images using an inexpensive and small system (our system used an IBM AT).

3.3.2. Advantages

The primary advantage of the laser disk system is many, very high quality images are available on a relatively small, inexpensive system. Computer generated imaging is still available and simply overlays the video disk image. The commercial hardware for this system has hardware zoom and pan capabilities that operate much the same as hardware zoom and pan on bitblt or vector created images. However, the laser disk system has the additional capability of selecting another image instead of expanding or moving the existing image. The new image selected by the laser disk system would still retain a high resolution whereas zoom of a bitblt or vector created image loses resolution after a factor of 4-6 times. For map applications, this means using hardware zoom up to a factor of about 4; then selecting another map with a different scale.

Another advantage of the video disk is that the storage media is radiation hard and easily removable. When removed from the drive system, The individual disks are easy to store and relatively indestructible.

A final advantage of the video disk system is the high level of user acceptance. The systems are very user friendly and, most importantly, they provide images similar to the ones currently used by the user. For RADC's tactical programs, this means map images identical to those carried by the pilot. For other applications, it could mean schematics or illustrations found in other references.

3.3.3. Disadvantages

The primary disadvantage of the laser disk system is the laser image is unavailable to the computer. Thus, a line on a map representing a road is unknown to the computer. This disadvantage can be somewhat alleviated by having a separate digital representation for the computer. Whereas the operator may see a nice, smooth curve for a line; the computer may use a relatively crude jagged representation of the same feature. This requires many tradeoffs to be made between operator needs/desires versus the computer needs for maximizing performance and memory requirements.

Another disadvantage with the laser disk system is the relatively high cost of first disk. The first disk for an RADC system cost approximately $50K for 34K images. Second and later disks are considerably cheaper ($600).

Finally, this system requires the laser images to be static images. The disk represents a read-only medium and cannot currently be used for applications requiring real-time changes to the images.

3.3.4. Space Applications

The laser disk system is appropriate for any application involving many unchanging images. It could easily be integrated into a diagnostic system involving logic diagrams, illustrated parts breakdowns, photographs, etc. It can also be used for high resolution background displays.

3.4. Expert (Rule-Based) Systems

3.4.1. Description

A rule-based expert system normally has a knowledge base of facts and if-then rules plus an inference engine to make inferences. A key feature of this type system is that the system developer concerns himself with capturing the knowledge rather than the details of the inference mechanism. Rule-based systems are the most common type of expert system being developed today. Most of RADC's rule-based systems either use a rule-base as part of a larger system or are primary concerned with assessing information.

3.4.2. Advantages

Rule-based expert systems are relatively easy to construct, very easy to modify/maintain and can handle incomplete, ambiguous, or conflicting data. They are unique in their ability to capture high level human thought processes. That is they easily capture rules of thumb (heuristics). More traditional systems can perform similar functions, but they require the persons thought process to be abstracted into a more acceptable form for the machine.

3.4.3. Disadvantages

Rule-based systems are not good for all applications. They often tend to be slow and require relatively large memories. Most importantly, they sometimes give wrong answers. In traditional programming, the goal is to build a system that meets some performance requirements. The expert system tries to mimic the expert. Like the expert, it makes mistakes. This is especially true near the boundary area of its knowledge. These are problems that match part of its rules, but also include facts beyond its knowledge base. In these areas, the rule-based system may give misleading and wrong answers whereas the more traditional system would crash.

Another problem with a rule-based system is in expecting them to solve problems requiring very basic knowledge. Actually they perform at their best by helping raise the performance level of a decision maker. These systems have not been able to capture basic (deep) knowledge that experts use to solve unique problems.

3.4.4. Space Applications

These kind of systems are ideal for removing the burden of controllers to check for common problems. They are also helpful for building scheduling systems. A key consideration in their use is whether the system will require frequent adjustment (change) by the operator (experts). The rule-based systems normally allow rules to be added or deleted by the user. Most of the experience at RADC has shown that they can raise the level of a decision makers to a new plain. Whereas before he was totally engrossed in mundane tasks, the rule-based system easied his workload so that he could concentrate on an assessment of the big picture.

3.5. Rapid Prototyping

3.5.1. Description

The last technique in this paper is not concerned with a software design approach. Instead, rapid prototyping refers to a development technique. Currently, DOD software programs are developed using DOD STD 2167. This standard involves an orderly systems engineering design approach where the design evolves form high level system requirements to low level detailed item requirements. Along the way The requirements are documented in the various levels of specifications (A, B, C); reviewed in formal design reviews (SDR, PDR, CDR); and finally, audited (PCA FCA). Throughout this process configuration control procedures track the system and generally the requirements are very well defined before coding commences.

Knowledge based systems cannot generally be effectively designed using the above approach. The difficulty lies in capturing human, expert knowledge (including heuristics) via the specification process. A more effective way is to build a prototype of the system and iterate its design through sessions with the experts. RADC has a number of knowledge based programs that have used a rapid prototyping development approach. The approach is made possible by many advances in the software development environment that make it easy to create data structures, stubbed interfaces, etc. It is also easy to make changes, incrementally to the system under development.

3.5.2. Advantages

The primary advantage of this approach is the ready acceptance of the user. This process has the user intimately involved in the total development cycle. His critiques of the system result in immediate feedback on the design. Another iteration of the prototype provides feedback to the user. This interaction makes the user a true part of the development process and allows him to see actual performance tradeoffs as they occur.

3.5.3. Disadvantages

A key disadvantage with this approach is the absence of formal procedures. Since the approach is contrary to the accepted development techniques, there are no standards for documentation, design reviews, or testing. This makes it extremely difficult to create contract packages with measurable end products. Disagreements between user experts create difficulties in defining the prototypes. Similarly, availability of experts becomes a critical path to the overall development. If the experts cannot devote sufficient time or the experts are constantly being replaced, the system development rambles.

Tests also tend to be more ad-hoc because there are few specific requirements (no specs) to measure against. Thus, it is difficult to quantitatively measure the software developed. The weakness in the tests of knowledge based systems is probably a primary reason for the heavy emphasis on systems that can explain their results.

3.5.4. Space Applications

It would seem that some form of rapid prototyping is necessary for successful knowledge acquisition if one is building a knowledge based system. Thus if the primary consideration of the system design is capturing a human reasoning process, then rapid prototyping should be used. For these situations, the systems engineering approach tends to be incomplete. Rapid prototyping can also be used for other applications, but it should be carefully weighed against the more traditional approaches. It is a relatively new technique and has not been evaluated against all types of development projects. Long term, some combination of a systems engineering approach, for technical requirements identification, and rapid prototyping, for knowledge acquisition and feasibility proof, will be probably be best.

4. Recommendations/Conclusions:

The purpose of this paper was not to reveal some radically new and unique approaches to space related problems. Rather, it is meant to use solutions and ideas to other problems and show how they may help solve space related problems. Although research continues, most of these techniques have been proven to be very effective in other applications. Most of these solutions and ideas have been presented in the form of programming techniques. All of the software programming concepts described here require relatively large processors (in terms of memory size) compared to existing space qualified hardware. However we anticipate future on-board hardware will be capable of using this techniques. This is feasible because each of the above techniques show how new methods of programming can simplify the problems and reduce the machine requirements compared to more traditional progamming approaches. Thus, they open the way for new software applications. More importantly, they offer hope for improved system performance and reduced support costs.