# Features of a Decision-Theoretic Approach to Monitoring and Repair and their Applicability to Replanning

**Richard Pelavin, Perry McCarty, Jim McGuire, and Chris Toomey**
**Lockheed AI Center**
**3251 Hanover St.**
**Palo Alto, CA 94304**

**Abstract**

This paper describes a real-world application of Decision Analysis to monitoring and repair of an unmanned mini-submarine and discusses its applicability to Decision-Analytic replanning. We feel that experience on real-world monitoring and repair applications can provide valuable insight into key, largely unaddressed issues in real-world replanning. We identify key features of our monitoring and repair application that make it computationally tractable and feasible to implement. By drawing an analogy to replanning, a subset of tractable real-world replanning problems can be circumscribed. This also provides a good starting point for eventually handling more sophisticated replanning domains. We also discuss the need to re-examine the representation of actions in a Decision-Analytic framework.

## 1.0 Introduction

Recently, automated Decision Analysis has been applied to real-world problems [Henrion et.el 91]. At the Lockheed AI Center, a Decision-Analytic reasoner plays a central role in a software project funded by the Navy to monitor and respond to "unanticipated events" on an unmanned mini-submarine. The software, called the Autonomous Control Logic (ACL) system [Sullivan et.al. 92], is built on top of a conventual control system and takes the place of a traditional Fault, Detection, Isolation, and Response (FDIR) system.

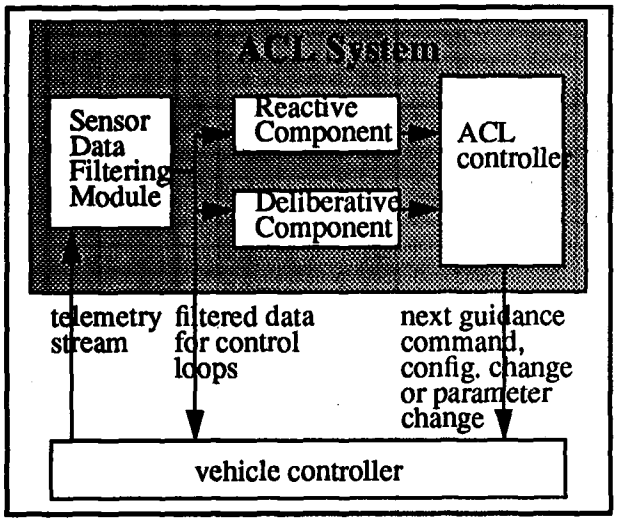


Figure 1

The ACL software has four major components as depicted in figure 1. The role of the sensor-data filtering module is to filter noisy data and to fuse data from redundant and related sensors. This filtered and fused data serves as input to ACL's reactive and deliberative components and is also injected back into the vehicle controller's feedback loops to produce better control.

The role of the reactive component is to respond immediately to time-critical problems and, if necessary and possible, put the vehicle in a "safe" state to give the deliberative component time to analyze and respond to problems.

The role of the deliberative component is to detect and respond, if warranted, to problems that can impact mission objectives. It selects responses based on a Decision-Theoretic analysis. Responses available consist of

i) modifications to the vehicle's *configuration*, which refer to switching to backup-up components, such as a backup propulsion controller, and changing control- loop parameters, such as fin limits (which affect how far the fins can travel);

ii) changes to the pre-loaded guidance-command parameters; and

iii) omission of *mission segments*, which refer to groups of guidance-commands that collectively accomplish a common high-level mission task.

The deliberative component can also request information-gathering tests to enable better decisions.

The role of the control component is to directly interact with the vehicle's control system, mediating between the vehicle controller and the reactive and deliberative components. This mediation process entails sending the next step of the pre-stored mission plan when there are no detected problems, inserting guidance plans to carry out requested tests, and transmitting requested configuration or guidance-parameter changes.

The sub-modules comprising the deliberative component are depicted in figure 2. The deliberative component operates in a continual cycle of monitoring and response -- when necessary -- with requested interjections of tests to enable better decisions.

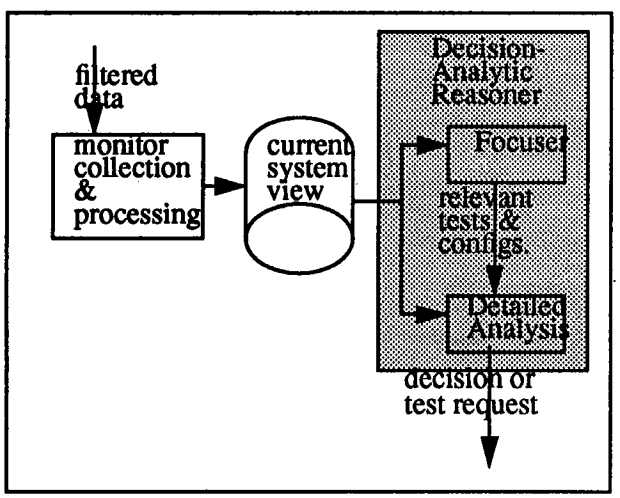


Figure 2

The deliberative component maintains an up-to-date probabilistic view of the health and status of the vehicle's physical components and functional capabilities. This system view is encoded in a discrete-state influence diagram with i) a causal model capturing the relationship between component states, subsystem states and predicted vehicle behavior, ii) value nodes encoding mission objectives, and iii) decision nodes capturing vehicle configuration alternatives and mission segment omissions.

The role of the monitor collection and processing module is to update the probabilistic system view at a 2 hertz rate. This module processes streams of real-valued data and puts them into a form that can be directly incorporated into the influence diagram (more formally, likelihood vectors). This monitoring module can be viewed as a collection of generalized Kalman filters [Dean&Wellman 92].

The Decision Analytic Reasoner, which is further broken into two sub-modules in figure 2, is responsible for continually determining whether any configuration change or test is needed and issuing requests when warranted. The focuser sub-module runs at a 2 hertz rate and is heuristic because of performance constraints. This sub-module is responsible for determining at each cycle whether any response or test may be needed and if so providing the subset of the relevant configuration changes and tests to analyze in detail. We refer to the subset of relevant configuration changes and subset of relevant tests as *candidate configuration-changes* and *candidate tests*, respectively.

The detailed-analysis component

i) computes the *value of information* for the candidate tests, requesting the highest-valued test if there is one with positive value-of-information, and

ii) computes the *expected value* for each candidate configuration change as well as for continuing without change; if there are no outstanding tests and a configuration change is the highest -valued alternative (as opposed to "continue without change"), then the highest-valued configuration change is requested.

Both *value of information* and *expected value* refer to the standard Decision Analytic computations. The detailed-analysis sub-module operates by manipulating the influence diagram encoding the current world view, possible alternatives and mission value. It is implemented using Hugin [Anderson et.al. 89], a commercially-available discrete belief-net and using Cooper's mapping [Cooper 88] for encoding influence diagrams in a belief net.

One can categorize the type of Decision-Analytic problem being handled as real-time monitoring and repair. The objective of this paper is to identify some key features in this paradigm that have made a real-world application possible, and to show how they relate to onboard monitoring and replanning. We will focus on the role played by the Decision Analytic Reasoner. We first set the stage by drawing an analogy between repair and replanning. In Section 3.0 we discuss a key heuristic typically employed in repair systems, where one looks for *minimal modifications* to address detected

problems. In section 4.0, we discuss our strategy for handling multiple faults in a tractable fashion. In Section 5.0 we identify the role of component states, showing they are integral to focusing the repair process. By drawing an analogy between component states and action representations, we identify a key inadequacy in the representation of actions in a Decision Analytic framework. In Section 6.0 we examine the role of analytic and empirical evaluation in determining the most appropriate place to repair a system. We show that we gain tractability and implementation-feasibility by relying heavily on empirical evaluation; furthermore, we identify features of our domain that make this strategy feasible, such as a minimal need for the anticipation of problems and the ability to directly monitor variables tied to objectives.

## 2.0 Relating Repair and Replanning

Replanning can be naturally seen as a repair problem: Given monitored evidence that a plan is no longer meeting its objectives, the replanning system should appropriately modify, i.e., repair, the plan. The actions comprising a plan are analogous to the components comprising a physical system. The structure in a hierarchical plan is analogous to the hierarchical decomposition of a physical system into sub-systems and components.

One evident difference between repair and replanning is that actions, unlike components in a physical system, go in and out of existence. Clearly, for replanning, one must keep track of the current action (or actions if parallel execution is allowed) and only consider modifications to present or future actions. The simplest way to incorporate this progression of time is to simply remove from consideration the modification of past actions. A deeper analysis, however, may take into account the loss of potential responses as time progresses and balance this loss with the uncertainty typical of acting early and predicting farther into the future. In our repair application, although the set of possible configuration changes does not shrink with time, the effectiveness of making a change at different times can potentially vary; in Section 6.0, we briefly discuss this issue.

Another difference between repair and replanning is the "scope" typically associated with repairs in comparison to the plan modifications considered in replanning. Typically, in repair problems, if a faulty or degraded component (or subsystem) is identified, the relevant courses of action are either to replace the component or to modify some of its parameters. In contrast, modifications considered in replanning can be far more reaching in that they can refer to replacing an action or plan fragment by a new action or whole new plan fragment (in addition to removing an action or changing its parameters). For this activity, a "plan generation module" may be considered to combine actions to form new plan fragments to replace existing ones.

If one divides repair or replanning into the three processes *isolation*, *solution-generation* and *evaluation* (of relevant repairs or plan modifications), a "plan generation module" would belong to the solution-generation process. Tractability is gained in our application by taking a simple approach to solution-generation, which is potentially an unbounded activity. In our application, the possible solutions are hard-wired into the influence diagram. Furthermore, only a small number of components are considered for modification and for each of these there is a small number of pre-planned ways to modify it. Moreover, we have found that even with a small number of responses available, there is still flexibility to adequately respond to problems.

## 3.0 Nominal State and A Minimal Modification Principle

Central to repair is the notion of a system's *nominal state*, i.e., the state that a system obtains when it is initially fielded after being tested and shown to meet its requirements or objectives. In our domain, the unmanned submarine is in its nominal state after passing a sequence of check-out tests. The system's nominal state is defined to include the system's components as well as conditions normally considered external to a system that can have impact on system behavior, such as ocean current.

The nominal state can be divided into factors that can be directly controlled and those that cannot. A subset of the controllable aspects refer to the decision alternatives that may be modified during the repair process. We will refer to decision alternatives that can be changed as *configuration variables* and their setting in the nominal state as the *nominal configuration*. We will refer to probabilistic beliefs about conditions that hold in the nominal state as *nominal assumptions*.

In our application, as typical of an automated repair system, we exploit a principle of minimal modification: If changes to the monitored system need to be made,

the software tries to make minimal modifications to the current configuration. If possible, for example, changes are made to a single component or isolated to a single sub-system. This principle can be viewed as a powerful heuristic that exploits the fact that the nominal state has high value (given the nominal assumptions); consequently staying close to the nominal state, unless the nominal assumptions become far from truth, typically produces configurations with high value. This heuristic buys us tractability in that it relieves the problem solver from exploring all possible settings to the configuration variables, which is exponential in the number of configuration variables. As we will discuss in section 4, the principle of minimal modification coupled with our approach to handling multiple faults yields a set of alternatives to explore at each decision-making cycle that is linear in the number of configuration variables.

When repair is considered in a Decision-Analytic framework, one must consider the expected value of the system's nominal state and the expected value of any minimal modification in response to detected problems. In our application, the value model is based on a set of specified constraints defined on the pre-stored segments comprising the mission plan. For example, a segment containing a heading guidance command to 60 degrees (true North) may include the constraint that steady-state heading is between 50 and 70 degrees. More specifically, we employ an additive value model where total expected value for a mission $m$ is a function of the probabilities that each task in $m$ is being accomplished and each task's assigned value. We use a simple, Boolean, notion of task achievement: a task $t$ is achieved if and only if the set of constraints defined on the segments associated with task $t$ all hold. The highest total value is obtained if all the constraints for all the tasks are met. We assume that the nominal system, i.e., the submarine after check-out, meets all the constraints and hence has highest expected value (given the nominal assumptions). We do not, however, assume that modifications made in response to detected problems produce vehicle configurations with highest expected value; rather, by design we sacrifice optimality for the tractability gained from a principal of minimal modification.

In general it would be helpful to elicit criteria to evaluate Decision-Analytic repair or replanning systems. For example, if optimality is given up as unachievable, a notion of *sufficing*, i.e., adequacy, might be developed. Along these lines, a distinction can be made between repair (replanning) systems that initiate repairs (plan modifications) only when a *problem* is detected and ones that additionally consider *potential opportunities*. By "problem", we mean cases where the expected value of the current configuration drops. By "opportunity", we refer to cases where the current configuration's expected value may not drop, but some alternative configuration becomes more attractive (i.e., obtains higher expected value).

Systems that just look for problems are more tractable than those that look in addition for opportunities. Repair or replanning systems that respond to just problems need only to concentrate on how the expected value of the current configuration is changing. Repair (or plan modification) can be triggered by a mechanism that looks for "significant drops" in expected value. For example, our application can be viewed as one that just looks for problems. Repair is considered only if the expected value of the current configuration *significantly drops*. "Significantly drops" in our case refers to the case where the expected value falls below the expected value associated with aborting the mission.

In contrast, systems that look for opportunities cannot concentrate solely on drops in the expected value of the current configuration (plan). They must also have ways to determine when an alternative has higher expected value, even when the current configuration does not drop in expected value. For example, consider a plan to take Interstate 280 to San Francisco. To determine whether it is worth switching to Interstate 101, one must do more than monitor the current plan (e.g., watching the traffic on 280); rather one must also monitor the alternative plan (e.g., listening to the radio report about traffic on 101). When looking for opportunities, one might switch to 101 when the traffic on 101 is less than originally expected even though the expected value of taking 280 does not drop. (See [Ogasawara 93] for a sophisticated approach to initiating repair, which involves balancing the estimated cost of computing and making a repair with the estimated benefit obtained by the best possible repair.).

# 4.0 Multiple Faults and Decisions per Cycle

As noted in the previous section, the number of potential configuration alternatives to consider each repair cycle is exponential in the number of configuration variables, i.e., number of components that can be individually replaced or modified. Being more precise, let $m$ refer to the number of configuration variables and $n$ refer to the average number of states each variable can

assume (which is finite and very small in our case). Potentially, the search-space at each cycle is in the order of $n^m$. Our system, however, inspects at most $m*n$ configuration alternatives at each repair cycle. More specifically, if a problem has been detected (i.e., the expected value of the current configuration significantly drops) and there are configuration changes potentially relevant to the detected problem, the focuser module (see figure 2) will identify a non-null subset of configuration variables, which we will call the *candidate variable set.* (note: in section 5, we will briefly describe how the candidate variable set is generated). The detailed analysis module then evaluates all configurations formed from the current configuration with a change in state to a *single* member of the candidate variable set. Thus, if there are $x$ members in the candidate variable set (where necessarily $x<=m$), there will be on the order of $x*n$ alternative configurations evaluated.

Implicit in the above mechanism is a strategy for handling multiple faults. Our system can handle multiple faults by sequentially fixing the problems (over multiple repair cycles), rather than simultaneously fixing the faults in a single repair cycle. We rely on the fact that the Decision-Analytic Reasoner is in a continual monitoring and repair cycle. It is assumed that if there is a multiple fault, then after one of the faults is repaired, subsequent monitoring and repair will enable the other faults to be identified and repaired. This simple approach to multiple faults is most applicable when the order in which the faults are repaired is not significant.

Our strategy for handling multiple faults also relates to *the issue of myopic evaluation*, an important problem cited in the Decision-Analytic literature [Horvitz et.al. 89]. As an example, assume that there are two faults present in the unmanned mini-submarine and that the expected value of aborting the mission is greater than the expected values of the two alternative where only one of the two faulty components are fixed. A naive system adopting a sequential approach to multiple faults would choose to abort the mission because aborting is more attractive than the two other available alternatives, i.e., fixing one of the two components. A more sophisticated approach would need to be optimistic in its evaluation of fixing a single component to take into account that additional problems could be corrected in later decision (repair) cycles. In our application, we use a simple heuristic that tends to give the expected value of a single repair higher value than would be warranted if this were the sole change being made. Because we are monitoring after a configuration change is made, if this optimism is unwarranted, the system will realize this fact from its monitoring. A complication is that a provision is needed to avoid "thrashing-like" behavior where the system flip-flops between two alternatives, neither of which solve the problem.

# 5.0 Component Nominal States and Degraded Modes

In our application, as typical of descriptions of physical systems, the vehicle is described in terms of subsystems and components each with a nominal state. Identifying whether a component or subsystem is in its nominal state plays a critical role in the guiding the repair process; our focuser (see figure 2) only considers changes to components or subsystems that are not in their nominal states. For planning, the analog to component nominal states relate to preconditions. For example, STRIPS-like replanning systems use precondition violations to guide repair.

If a component is not in its nominal state, its state may range the spectrum from slightly degraded to completely failed. A key problem in our application has been representing component modes with discrete states and providing a method for determining their impact on the system as a whole (i.e., impact on total expected value). It has been important to handle many cases where a component is in degraded mode, but there is minimal impact on total expected value, and consequently a repair should not be made. In planning, the analogy to component failure or degraded modes is states where an action's precondition does not hold; in this situation, the action may only be slightly affected, may partially bring about its effects, or do nothing at all. The prevalent representations of actions, which are centered around propositional preconditions and effects, do not capture these distinctions; if the preconditions do not hold, these representations say little about the effect of the action. An advance to these representations is those of Haddawy [Haddawy 91] and Chrisman [Chrisman 1992], who provide probabilistic representations of actions; these representation, however, are still deficient in that they are based upon a propositional treatment of effects and preconditions in that the objects of the probability statements are propositions and predicates. These representations, for example, cannot succinctly capture that "slight changes" to tan action's preconditions causes only "slight changes" to the action's effects. As a specific example, consider an action that corresponds to surveying an undersea

region. Suppose also that a precise real-valued metric is defined for the effectiveness of performing a search, which we take to be the action's effects. It would be desirable to capture the relationship between the accuracy of the sensor utilized in the survey and the search effectiveness and to reason about the impact of degraded sensing on search effectiveness.

Rather than having a representation of actions based on propositional effects and preconditions, what seems needed for real-world applications like ours is one where preconditions and effects are viewed as arbitrary variables, not just two-state propositions. To clarify, we note that the propositional treatment of actions can refer to real-valued variables, but only as arguments to the predicates representing effects and preconditions.

In developing a "variable-based" representation of actions, a complication that needs to be addressed is that effects are not strictly dependent on preconditions; rather, the effect of an action depends also on "how the action is to be executed." When considering the representation of low-level actions, "how the action is executed" can be encoded by simple parameters, such as the commanded heading and water speed. For abstract actions, however, the notion of "how the action is executed" is more amorphous and consequently more difficult to represent.

# 6.0 Determining the Most Appropriate Repair

Basically, there are two ways in which to determine an appropriate configuration change or plan modification: analytically, using a causal model to predict or estimate the effect of making candidate changes, or empirically by physically trying candidate changes and monitoring their effects. Ideally, one would have a computationally tractable and appropriately accurate causal model that would enable the repair or replanning software to analytically determine the best change. Pragmatically, finding such a model may be impractical because of implementation constraints, such as the current state of belief-net technology, performance considerations, or availability of expertise to produce a causal model of sufficient fidelity.

Most repair and replanning systems will need some combination of analytic and empirical evaluation. Some form of empirical analysis is desirable in order to determine if candidate changes are meeting the objectives. On the other hand, a causal model of some sort is needed in order to select the candidate changes, i.e., changes that are *relevant* to the monitored symptoms (the role played by our focuser module). Considerations in deciding where to be on the analytic/empirical spectrum are how much anticipation of upcoming problems is needed and whether the objectives can be directly monitored. In our application, anticipation is not critical and many of the objectives can be directly monitored. These features provide us with the option of developing a system that physically tries a candidate solution, directly monitors whether the objectives are being met, and if not, trying another solution; that is, we have the luxury of being able to adopt a system on the empirical side of the analytic/empirical spectrum. For example, an objective in our value model refers to a constraint on vehicle ground speed. Our system directly monitors this objective using data from the vehicle's sonar, and responds when sensed ground speed is out of the constraint's bounds. Direct monitoring also facilitates the identification of relevant component to repair; as noted in the last section, the focuser only considers repairs to off-nominal (i.e., degraded or failed) components. By directly monitoring components, one can get a more certain assessment of their states in contrast to diagnostically computing their probabilities from monitored symptoms and a causal model linking component states to possible symptoms.

A belief net provides a framework that is flexible as to where on the analytic/empirical spectrum one wants to be. In a strictly analytic system, one would strive for a belief net with a deterministic or nearly deterministic relation between decision nodes and value-model objectives. In a more empirical system, a more probabilistic relationship between these variables is sufficient.

A key issue to examine when looking at the relationship of the empirical/analytic spectrum to replanning problems stems from the fact that objectives in the planning domain often refer to conditions that hold at the completion of plan execution, and thus cannot be directly monitored. Consequently some form of projection seems needed. It may be sufficient, however, to use a simple projection model, such as the one employed in our system to estimate whether future constraints hold. In our model, if all the subsystems are nominal, then the future constraints are assumed to hold with high certainty. For severe problems, the projection model captures that the relevant constraints do not hold with high certainty. For the cases between the nominal state and severe state, the causal model provides little predictive power, in which case future constraints are assumed to hold with high probability,

which are their *a priori* probabilistic assessments. The penalty of not more accurately predicting future constraints is that the system may be responding to problems later than it potentially could have, a relatively small penalty in our domain. However, in our application, for constraints where anticipation is important, such as energy usage constraints, a more sophisticated predictive model is employed.

Another consideration in applying our results to the planning domain relate to a distinction made by Keeney [Keeney 92] between *fundamental* and *means-end* objectives. Fundamental objectives are those which one ultimately wishes to achieve owing to their intrinsic value, while means-end objectives are those which one wishes to meet only because they are the means to meeting more fundamental objectives. For example, "win the lottery" could be a means-end objective for the fundamental objective "become wealthy."

Typically, planning goals tend towards fundamental objectives. In contrast, we have made extensive use of means-end objectives in our value model. For example, an objective in our model refers to a constraint on propeller rpm, which appears because the noise that propellers produce can lead to detection by the enemy. Clearly, the objective "propeller RPM" is a means to the more fundamental objective "avoid detection." The reasons for adopting these means-ends objectives in our application is three-fold. First estimating the value of more fundamental objectives can require a sophisticated causal model. For example, to estimate whether the vehicle will get detected would require a causal model that relates propeller RPM and enemy location to the objective "get detected". Secondly, means-ends objectives facilitate the knowledge acquisition process. Means-ends objectives are more "operational" and as a consequence we have found that people are much more comfortable providing them than fundamental objectives. For example, in our model there is a constraint on the vehicle's pitch angle, which has impact on the effectiveness of a vehicle search procedure. It is much easier to elicit constraints on acceptable pitch angle, in contrast to treating a more fundamental objective, such as "search effectiveness", which is difficult to measure. The third advantage of means-ends objectives is that they reduce the need for anticipation. For example, our model includes a constraint on vehicle depth and depth rate, which can indicate whether the vehicle may potentially hit the bottom. Specifying a bound on acceptable depth and depth rate, alleviates the need to predict whether the vehicle will hit the bottom. This use of means-ends objectives suggests an enhancement to a Decision-Analytic representation of actions. Actions can be associated with means-ends objectives much in the same way that unit requirements or objectives are distilled from system objectives. Having mean-ends objectives associated with actions can facilitate the process of identifying which actions to modify when a plan is net meetings its objectives.

There is, however, an inherent trade-off between the operationality or efficiency of means-end objectives and the flexibility or generality of fundamental objectives. By using exclusively means-end objectives, one restricts the means by which the system will consider achieving the fundamental objectives. Thus, the extent to which means-end objectives are employed needs to be balanced against the desired flexibility in and awareness about achieving the fundamental objectives.

# 7.0 Conclusions

We feel that experience on real-world monitoring and repair applications can provide valuable insight into key, largely unaddressed issues in real-world replanning. The issue of tractability has riddled the planning literature. Tractable approaches have been developed for repair problems to a large extent due to the principle of minimal modification. By starting with a nominal system with high value and only looking for minor changes, one has a powerful search heuristic. Our research plan is to first consider replanning, rather than planning from scratch. By tackling replanning before planning from scratch, we have hope of developing a real-world application and gaining insight into the real-world representational issues unique to the application domain. These same representational issues would need to be understood before tackling realistic planning from scratch.

By viewing repair in general terms, we are able to categorize features applicable to replanning that have led to a real-world solution in our application,

such as a minimal need to anticipate problems and the ability to directly monitor objectives. These features can be used to identify the subclass of replanning problems that may be solvable in the near future.

Lastly, recent work in knowledge representation bridging the gap between planning and Decision Analysis has focused mainly on the relationship between value or utility functions and planning goals [Haddawy&Hanks 90] [Wellman&Doyle 91]. We feel there is also a great need to re-examine the representation of actions in a Decision-Analytic framework. The prevalent representations of actions, which are based on propositional effects and preconditions, are impoverished, for example, because they do not capture the effect of actions that are slightly, partially, or completely failed. It is also worth exploring the use of means-ends objectives [Keeney 92] defined on individual actions in a plan to enable *direct monitoring* of objectives, reducing the need for a causal model to project whether the fundamental objectives will be met.

## 8.0 Acknowledgments

We would like to thank Gary Ogasawara and Don Howells for comments on earlier drafts.

## 9.0 References

Andersen, S. K. and Olesen, K. G. and Jensen, F. V. and Jensen, F. 1989. HUGIN--- a shell for building belief universes for expert systems. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers.

Chrisman, L. 1992. Abstract Probabilistic Modeling of Actions. In Proceedings of the First International Conference on Artificial Intelligence Planning Systems. pp.28-35.

Cooper, G. F. 1988. A method for using belief networks as influence diagrams. In Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence, pp.55-63.

Dean, T. L. and Wellman, M. P. 1991. *Planning and Control.* Morgan Kaufmann Publishers.

Haddawy P. 1991. Representing Plans Under Uncertainty: A Logic of Time, Chance, and Action. Dissertation, Computer Science Department University of Illinois, Urbana-Champaign.

Haddawy, P., and Hanks, S. 1990. Issues in Decision-Theoretic Planning: Symbolic Goals and Numeric Utilities. In Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control, pp.48-58. Morgan Kaufmann Publishers.

Henrion, M. and Breese, J. S. and Horvitz, E. J. 1991. Decision Analysis and Expert Systems. In AI Magazine. v12. n4., pp.64-92.

Horvitz, E. J. and Cooper, G. F. and Heckerman, D. E. 1989. Reflection and action under scarce resources: Theoretical principles and empirical study. In Proceedings of Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers.

Keeney, R. L. 1992. *Value-Focused Thinking.* Cambridge, Massachussetts: Harvard University Press

Ogasawara, G. H. 1993. RALPH-MEA: A Real-Time Decision-Theoretic Agent Architecture, Dissertation (in preparation), Computer Science Division, University of California, Berkeley.

Russell, S. J. and Wefald, E. H. 1989. Principles of Metareasoning. In Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann Publishers.

Sullivan J. W., McCarty P., Yoshimoto G., Gargan R., and Pelavin R. 1992. In Intelligent System for Autonomous UUV Monitoring, Diagnosis and Control. Proceedings of The American Institute of Aeronautics and Astronautics.

Wellman, M. P. and Doyle, J. 1991. Preferential Semantics for Goals. In Proceedings of the Ninth National Conference on Artificial Intelligence, pp.698-703.