Failure Recovery: A Model and Experiments

Adele E. Howe, Paul R. Cohen

Computer Science Technical Report 91-66

Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

This paper presents a model of failure recovery from which we have designed and tested sets of failure recovery methods in the Phoenix system. We derive the model, document its assumptions, and then test the validity of the assumptions and predictions of the model. We present three experiments. One derives baselines for failure recovery in the Phoenix environment. The second compares the performance of two strategies for selecting failure recovery methods. The third compares the performance of an initial set of failure recovery methods with a redesigned set that is predicted to have lower expected cost.

This research was supported by DARPA-AFOSR contract F49620-88-C-00113, by a University Initiative Grant, ONR N00014-86-K-0764, by ONR contract N00014-88-K-0009, and by the National Science Foundation under an Issues in Real-Time Computing Grant, CDA-8922572. We wish to thank David Westbrook and Scott Anderson for their invaluable assistance in running the experiments and members of the Phoenix project for the construction of the testbed.

1. Problem Description

Planners that commit to future action inevitably fail. Plan failures may be caused by actions not having their intended effects, by unexpected environmental changes or events, or even by inadequacies in the planner itself. Ever since planners have been applied in dynamic environments, researchers have adopted numerous approaches to recovering from plan failures. For the most part, these approaches classify the failure and select among a set of domain-independent methods for adapting the plan in progress to recover from the failure (Hayes 1975, Hammond 1987, Ambros-Ingerson & Steel 1988, Simmons 1988, Wilkins 1988, Simmons 1990). These approaches differ both in their classification of failures and their method sets; so how does a designer identify and evaluate possible recovery techniques for a *new* domain? In this paper, we will describe an approach to designing and evaluating recovery methods based on a model of the application of failure recovery methods within an environment.

1.1 Environment Analysis

In general, environment analysis identifies those aspects of the environment that necessarily constrain the behavior of agents within it. Plan failures arise from an interaction between the environment, which changes, and the plan, which is based on expectations of change (or lack thereof) in the environment. The environment does not by itself produce failures -- only changes. Thus, any environmental analysis in support of understanding failure recovery needs to be an analysis of how environmental changes manifest as failures for the planner.

A complete model of failure recovery requires modelling how failures occur and are detected, how much the planner knows about the failure, and how recovery methods address the failure. For the purposes of evaluating the performance of failure recovery methods, however, the environment analysis need only include the constraints on the applicability of the recovery methods, i.e., the types of failures and the information available to the planner about them. The task of the designer is to define a set of recovery methods that can together address those failure situations. Consequently, in this paper, we will rely on failure descriptions already provided by the test environment as the preliminary environment analysis, but acknowledge from the outset that a semantic model of how failures happen is the next step.

1.2 The Test Environment and Agent Architecture

Our test environment for failure recovery is a simulation of forest fire fighting in Yellowstone National park. The goal of forest fire fighting is to contain fires. In this simulation, a single agent, the fireboss, coordinates the efforts of field agents to build fireline, cleared areas that halt the spread of fire. Fire spread is influenced by weather (e.g., wind speed, wind direction, temperature) and terrain (e.g., ground cover, elevation, moisture content). While terrain remains constant, weather changes constantly: agents planning to contain the fire must be prepared for these changes to invalidate their expectations.

The Phoenix system provides the forest fire simulator and a basic agent architecture that support the experiments in failure recovery design and evaluation (Cohen et al. 1989). The agent architecture includes a set of sensors and effectors for this environment and two components for control: reflexes and the cognitive component. Reflexes make small short-term adjustments to the sensor/effector settings to keep them tuned or to remove the agent from immediate danger. The cognitive component includes a planner which makes nearly all decisions about present and future action. The planner operates by lazy skeletal expansion: it selects plans from a library and begins to execute them, expanding some into more detailed action sequences only as the need arises. Its plan language is specific to this style of planning and its plan library is specific to fire fighting.

Table 1 describes the possible failure situations that we have observed in Phoenix. Each failure situation has a name, such asnot-enough-resources, and a one-letter identifier, used in later tables. Table 1 also indicates which failure recovery methods (e.g., RP for replan-parent; to be presented in Section 3.3) are applicable to each of the failure situations. Table 1 shows, for example, that the RP method is applicable to all failure situations; indeed, in the baseline experiments, described in Section 5.1, most methods are applied in most failure situations.

2. Model of Failure Recovery

The goal of failure recovery is to allow a plan to continue from a failure, while incurring minimal overhead. The previous section described possible failures in terms of features. This section describes a model in which the overall cost of recovery is a function of the probability of failure situations occurring and the expected cost of recovery methods.

	WATA	RV	RAV	SA	RP	RT
A. cant-calc-path-to-road	x	x	x	x	x	x
B. not-enough-resources	x	x	x	x	x	x
C. cant-find-plan	x	x	x	x	x	x
D. resource-unavailable	x	x	x	x	x	x
E. cant-calc-path	x	x	x	x	x	x
F. no-remain-segments	x	x	x	x	x	x
G. fire-not-encircled	x	x	x	x	x	x
H. insufficient-progress					x	x
I. cant-calculate-variable	x	x	x	x	x	x
J. cant-calc-projection	x	x	x	x	x	x

Table 1. Failures and the applicabilities of failure recovery methods

2.1 Basic Model

In the basic cost model, the expected cost of recovering from any situation is a cumulative function of the probability of a situation S_i occurring times the cost of recovering from it, for all situations:

$$\mathbf{EC} = \underset{i=1}{\overset{n}{}} \mathbf{P}(S_i) \quad \mathbf{C}(S_i) \qquad (1)$$

where n is the number of situations possible in the environment. A situation S_i is a type of failure. The probability of a situation occurring $P(S_i)$ is empirically determined for a given environment (see Sec. 5.2). $C(S_i)$ is the expected cost of recovering from situation S_i.

We assume that failure recovery works by selecting a method from a set of applicable methods and running that method. If no methods are applicable to a particular situation, then $C(S_i)$ is defined to be the cost of outright failure. If we assume that a method may fail, then for a given situation, we may need to try several methods before one works or we run out of methods. Given the probability that each method will succeed in situation S_i ($P(M_j|S_i)$), and the cost of the method $C(M_j)$, the expected cost of trying two methods in the order M_j, M_k is as follows: The cost of M_j and M_k are assessed regardless of whether they succeed because methods run for the same amount of time whether they succeed or fail. If both methods fail, C(F), the cost of failure, is assessed. Thus, the expected cost of attempting to execute M_j, M_k in situation S_i, is:

$$EC(S_i) = \quad C(M_j) + (1 \quad P(M_j|S_i))\, C(M_k) + \ (1 \quad P(M_j|S_i))\, (1 \quad P(M_k|S_i))\, C(F) \qquad (2)$$

By the same argument we can find the expected cost of executing any sequence of methods in a situation S_i:

$$EC(S_i) = \quad C(M_j) + Q(M_j|S_i)\, [C(M_k) \ + Q(M_j|S_i)\, [C(M_l) + \ldots Q(M_z|S_i)\, C(F)] \ldots] \qquad (3)$$

where $Q(M_j|S_i) \ = 1 \quad P(M_j|S_i)$. So the expected cost of recovery from a situation is the cost of trying the methods in some order times the probability of previous methods having failed, accumulated until the final option is accepting outright failure.

The assumptions of this model are:

1) $\forall m \; C(M_m) < C(F)$.

2) The cost of failure C(F) is independent of the situation S_i.

3) $C(M_m)$ is independent of the order of execution of the methods.

4) $P(M_m|S_i)$ is independent of the order of execution of the methods.

5) The cost of each method $C(M_m)$ is independent of the situation S_i.

Assumption 1 must be trivially true: there is no point in failure recovery if it costs more than the failure itself (i.e., the cure is worse than the disease). Assumption 2 is very difficult to test, and is not dealt with here. The validity of the other assumptions depends on the design of the set of methods, the environment in which they are used, and the strategy by which the methods are selected. Assumptions 35 are tested in subsequent sections. We begin with the method selection strategy.

2.2 Deriving the best control strategy for trying methods.

We can see from Eqs. 2 and 3 that the order in which we try methods determines the expected cost of failure recovery. For the case of two methods, we can easily derive a rule for the minimum-cost ordering. First we expand the expressions for the expected cost of order M_j, M_k and M_k, M_j, and then remove common terms:

$$EC_{jk} = C(M_j|S_i) + (1 \; P(M_j|S_i))\, C(M_k|S_i) + (1 \; P(M_j|S_i))\,(1 \; P(M_k|S_i))C(F)\;)$$
$$= P(M_j|S_i)\, C(M_k)$$
$$EC_{kj} = C(M_k|S_i) + (1 \; P(M_k|S_i))\, C(M_j|S_i) + (1 \; P(M_k|S_i))\,(1 \; P(M_j|S_i))C(F)\;)$$
$$= P(M_k|S_i)\, C(M_j)$$

So if $\dfrac{P(M_j|S_i)}{C(M_j)} > \dfrac{P(M_k|S_i)}{C(M_k)}$, $EC_{jk} < EC_{kj}$.

Simon and Kadane (Simon and Kadane 1975) have proven in the general case that expected cost of a sequence of methods m is minimized by the strategy of trying the methods in order of descending $\dfrac{P(M_m|S_i)}{C(M_m)}$.

3. Design of Failure Recovery in Phoenix

The model of the previous section recommends minimizing the cost of failure recovery by designing cheap methods that always work. Since that it rarely possible, the model can be used to focus design effort in several ways: maximizing the coverage of methods so as to avoid incurring the cost of failure; generalizing cheap methods to apply in more situations, and adding new, cheaper methods to the method set. We have designed and implemented a recovery mechanism and a set of recovery methods for Phoenix that attempt to follow these design guidelines.

3.1 Detecting Failures

The Phoenix agent architecture includes three mechanisms for detecting failures: execution failures, reflexes, and envelopes[1]. Execution failures occur when a plan action cannot execute to completion because conditions in the environment or plan do not match the expectations of the current action. Reflexes are a reactive component of the agent architecture that trigger timely responses to threatening situations; the execution of a reflex response flags an execution time failure in the on-going plan. Envelopes detect impending failures (Hart, Anderson & Cohen 1990). They monitor the plan's progress to determine whether the plan can complete given expectations about the environmental conditions and its resources.

Because two of these mechanisms, reflexes and envelopes, operate as adjuncts to the planning actions, they tend to provide little information about the failure and its impact on the plan. Moreover, the Phoenix plan language itself is impoverished in its representation of cause and effect. Thus we have adopted a failure classification scheme that does not require explanations of the cause of failure, but only the situation that triggered the failure and whatever information is available about its scope in the plan. We have avoided more knowledge intensive, and therefore more predictable, approaches to explaining failures, such as in Hammond's CHEF (Hammond 1987) and Simmons' GORDIUS (Simmons 1988), because the environment and plan interactions that cause failures are difficult to model and analyze (see Sec. 5).

3.2 Selecting a Response

When a failure is detected, an action to deal with it is added to the agent's agenda of actions and plans. Executing this action results in calling the planner to find a plan (i.e., method) to address the failure. The planner searches a plan library for methods applicable to the failure situation and selects among them. In the experiments described below, the planner is made to either select randomly among the methods without replacement, or to select methods in the optimum order, specified in Section 2.2, above.

3.3 The Basic Recovery Method Set

To test the model, we defined a core set of basic recovery methods. These are the methods shown at the top of Table 1. They make either global or local repairs to plans:

WATA:	Wait and try the failed action again.
RV:	Re-calculate one variable used in the failed action.
RAV:	Recalculate all variables used in the failed action.

[1](Howe & Cohen 1990) describes in further detail aspects of the agent architecture designed to respond to change in the Phoenix environment.

SA: Substitute a similar plan step for the failed action.

RP: Abort current plan and re-plan at the parent level (i.e., the level in the plan immediately above this one).

RT: Abort current plan and re-plan at the top level (i.e., redo the *entire* plan).

The first four methods make local changes to the failed action and surrounding actions; the last two replan at either the next higher level of plan abstraction or at the top level. These recovery methods, or ones very like them, have appeared in other recovery systems. WATA is like the "retry" method described in (Hanks & Firby 1990); RV and RAV are Phoenix specific forms of SIPE's Reinstantiate (Wilkins 1988); SA is applied in GORDIUS (Simmons 1988) and the two replan methods are constrained forms of the more general replanning done in nearly all failure recovery systems.

4. Hypotheses

Several hypotheses follow from the expected cost model in Eq. 3. These include tests of our assumptions from Section 2.1:

1) $C(M_m)$ is independent of the order of execution of the methods.

2) $P(M_m|S_i)$ is independent of the order of execution of the methods.

3) The cost of each method $C(M_m)$ is independent of the situation S_i.

We also test whether the model in Eq. 3 would facilitate the redesign of a method set with a predicted lower expected cost of failure recovery:

4) Ordering the selection of failure recovery methods by $\frac{P(M_m|S_i)}{C(M_m)}$ should result in a lower average cost of failure recovery than a random method selection strategy.

5) It should be possible to change the average cost of failure recovery in all situations by

modifying the applicability conditions of failure methods to reduce the applicability of expensive methods that are unlikely to succeed

modifying the set of failure methods to include lower-cost methods

Moreover, the cost savings of these modifications should be predictable from the model in Eq. 3.

In addition, we are interested in whether different conditions in the Phoenix environment lead to different distributions of failure types.

5. Experiments

We ran three sets of experiments. In Experiment 1, we collected baseline statistics to test hypotheses 13, above, and to empirically determine values for the parameters $P(M_m|S_i)$, $P(S_i)$ and $C(M_m)$. In Experiment 2 we compared the random and optimum method-selection strategies (hypothesis 4, above). In Experiment 3, we added new recovery methods to more cheaply address expensive recovery situations as suggested in hypothesis 5 and compared the results to those of Experiment 2.

5.1 Experiment 1. Baselines.

We ran 116 trials in which Phoenix fought three fires, resulting in 2462 failure situations and 5558 attempts to recover from the failures. During these trials, the fires were set at intervals of eight simulation hours. Wind speed and wind direction were varied by 3 kph and 30 degrees, respectively at one hour intervals. For each situation we collected the following data: the failure type, the failure methods tried, the order in which the methods were tried, and the cost (in simulation-time seconds) of executing the recovery methods and the plan modifications made by the methods. The agents were given the recovery method set and applicabilities described in Table 1; in addition, the bulldozer agents were given a special method for avoiding a deadly object. The distribution of method use was essentially uniform, modulo the applicability of the two replan methods to an additional failure situation.

Hypothesis 1: $C(M_m)$ is independent of the order of execution of the methods. We want to know whether failure recovery methods have different costs depending on their position in the order in which methods are executed. We ran a two-way analysis of variance in which the factors were method and position, and the dependent variable was $C(M_m)$. We analyzed separately failures in the Phoenix fireboss and Phoenix bulldozers, since the fireboss encounters different types of failures which generally take much longer to repair than bulldozer failures.

The bulldozer data produced a main effect of method, indicating that different methods have different costs. But the analysis found no significant effect of position, nor any method by position interaction. The fireboss data yielded main effects of cost

	Failure Situation (fireboss failures only)									
	A	B	C	D	E	F	G	H	I	J
Probability of failure occurence (Exp. 1)	.075	.159	.118	.077	.268	.002	.002	.213	.006	.080
Baseline Costs (random strategy)	1373	3395	3030	3254	1932	2904	1163	3883	5710	2165
Probability of failure occurence (Exp. 2)	.043	.163	.084	.058	.212	.004	.002	.203	.007	.223
Actual Costs (optimum strategy)	1041	2838	2707	2514	1056	414	474	3977	1618	2518

Table 2. Probabilities and actual costs for baseline and strategy experiments, by situation.

and position and a significant interaction effect. Because we believed this to be due to large variance in the costs for the replan methods, we analyzed the data for the replans separately from the other methods. As separate data sets, neither analysis of variance found significant interaction effects, indicating that the replan methods behave qualitatively differently from the other methods and that for methods other than the replans, the cost of a method is independent of its order of execution.

Hypothesis 2: $P(M_m|S_i)$ is independent of the order of execution of the methods. To test whether the probability of a method's success depends on its position in the order in which methods are executed, we counted the number of successes and failures for each method in each position in which it was executed. and constructed contingency tables from these counts.

Chi-square analyses for all but the two replan methods and the substitute action method yield the same result: P(Mi|Si) is independent of position. As in the testing of Hypothesis 1, the replan methods behave differently from the other methods. Because the two replans are designed to take the same action in some situations, when one replan method fails, the other will as well; thus, the probability of success for these methods is not independent of which method proceeded them. Similarly, we believe that the substitute action method may be interacting with one of the other actions, but we have yet to test these explanations.

Hypothesis 3: The cost of each method $C(M_m)$ is independent of the situation S_i. To test whether methods have different costs in different situations, we ran a two-way analysis of variance in which the factors were method and failure situation. As before, we analyzed fireboss and
bulldozer failures separately. The bulldozer data showed a main effect of method (i.e., different methods have different costs), no effect of failure type, and no method by failure interaction, indicating that the cost of any given method is independent of the failure situation in which it is applied. As in the analysis of hypothesis 1, the fireboss data was separated into global (i.e., the two replans) and local (i.e., WATA, RV, RAV, and SA) methods. The analyses of these groups showed no method by failure interaction.

5.2 Experiment 2.

We ran 94 trials in which Phoenix fought three fires, resulting in 2001 failure situations and 3877 attempts to recover from the failures. We collected the same data about each failure as in Experiment 1. The experiment scenario, that is, wind changes and the intervals between new fires were the same as in Experiment 1.

Hypothesis 4: Ordering the selection of failure recovery methods by $\frac{P(M_m|S_i)}{C(M_m)}$ should result in a lower average cost of failure recovery than a random method selection strategy. To test this hypothesis we used observed values of $P(M_m|S_i)$ and mean values of $C(M_m)$ derived from Experiment 1 to determine $P(M_m|S_i)/C(M_m)$, from which we determined the best order in which to try methods. Using the mean for $C(M_m)$ is justified by the fact that $C(M_m)$ is independent of situation and position.

Table 2 gives costs of failure recovery for each of the failures, identified by capital letters as in Table 1. The second row shows the costs incurred during the baseline experiments (Experiment 1). The fourth row shows the actual mean costs under the optimum method selection strategy for each failure situation. In all but situations H and J, the cost of failure recovery is much lower (11-86%) with the optimum strategy than with the random strategy, as predicted.

Table 2 also includes the probability of occurence for each of the failure types as observed in the two experiments. Given the probabilities of failure occurences and the observed costs of each failure type for each experiment, the mean recovery costs

	WATA	RV	RAV	SA	RP	RT	AAR	SPA
A. cant-calc-path-to-road			2	3	4	1		
B. not-enough-resources		4	3	2	5	6		1
C. cant-find-plan	4	5	6	2	1	3		
D. resource-unavailable					1	2		
E. cant-calc-path			1	2	3	4		
F. no-remain-segments					1	2		
G. fire-not-encircled				1	2	3		
H. insufficient-progress					2	3	1	
I. cant-calculate-variable					1	2		
J. cant-calc-projection		1	2	4	5	6		3

Table 3. The revised method set and applicabilities for Experiment 3.

are 2811 for Experiment 1 and 2530 for Experiment 2. A z-test on the differences between the failure recovery means for the fireboss data for the random and optimum selection strategies yielded a significant result ($z = -1.79$, $p < .0367$). The same test on the bulldozer data produced a highly significant result ($z = -16.6$, $p < .0001$). Thus, ordering the selection of failure recovery methods by the suggested strategy results in a lower cost for most failure situations and a significantly lower cost overall.

As noted however, the cost of situation J increased, as well as its probability of occuring. We believe these increases are primarily due to changes made to the underlying system during the intervening time between the two experiment sets.

5.3 Experiment 3. Redesigning the Method Set

We ran 84 trials in which Phoenix fought three fires, resulting in 1540 failure situations and 4279 attempts to recover from the failures. We used the optimum strategy to select methods. All other conditions were as they were for Experiments 1 and 2.

Hypothesis 5 is that we can redesign Phoenix's set of failure recovery methods to minimize the cost of failure recovery. We could do so in two ways:

a. modify the applicability conditions of failure methods to reduce the applicability of expensive methods which are unlikely to succeed

b. modify the set of failure methods to include lower-cost methods

Because the strategy already selects methods to produce the most efficient ordering, we chose to use the same applicability conditions as in Experiment 2 (which results in the method orderings shown in Table 3) and augment the failure method set. As in Table 1, rows represent failure situations and columns represent methods. The cells contain the strategic ordering of the methods for each failure situation, e.g., for situation G, fire-not-encircled, the method selection order is SA, then RP, and finally RT. We added two new methods for each of the agents, designed by specializing some of the existing methods to perform better in those situations that were both expensive and likely (insufficient-progress, H, and cant-calculate-projection, J, for the fireboss). The new methods, **add-another-resource** (AAR) and **substitute-projection-actions** (SPA), were based on existing methods (RAV and SA, respectively); consequently, the cost and probabilities for them were copied from those methods and reduced somewhat to reflect better expected performance, e.g., the $P(M_m|S_i)$ values were increased for the target situations.

The costs of failure recovery for the fireboss are shown in Table 4. The new method set has managed to reduce the cost of the target situations (B, H, and J) by 21%, 26% and 10%, respectively ; yet, doing so incurred higher costs in nearly all the other situations. This produced a mean recovery cost over all situations of 2370 with the new method set. Consequently, a z-

	Failure Situation (Fireboss failures only)									
	A	B	C	D	E	F	G	H	I	J
Baseline data (random strategy)	1373	3395	3030	3254	1932	2904	1163	3883	5710	2165
Actual Cost (optimum strategy, old method set)	1041	2838	2707	2514	1056	414	474	3977	1618	2518
Actual Cost (optimum strategy, new method set)	1222	2094	2866	3029	1411	266	520	3149	2208	2253

Table 4. Actual costs for Experiments 1, 2, and 3, by situation.

test on the difference between the means for Experiments 2 and 3 (2530 and 2370) was not significant ($z = -1.00$, $p = .1587$). Since we made no changes to the method set which would account for the increased costs in other situations, we did not predict this result and at present have no explanation for it (see Section 7 for further discussion). As in Experiment 2, a comparison of the new and old method set applied to bulldozer errors resulted in a highly significant result ($z = 6.57$, $p < .0001$).

5.4 Another Baseline Experiment

To test the hypothesis that the distribution of failure situations is independent of the environment conditions, we recorded the failures that occurred in two different scenarios. In the *fast-change* scenario the windspeed and wind direction changed every 30 minutes, the wind speed by 2 kilometers/hour and the wind direction by 15 degrees; three fires were set at 8 hour intervals. In the *slow-change* scenario the windspeed and the wind direction changed every 60 minutes, by 3 kph and 30 degrees, respectively, and three fires were set at 12 hour intervals. Table 5 shows the distribution of failure situations (labelled with letters, corresponding to the failure situations in Table 1) in the fast-change and slow-change scenarios.

Failure Situation (All failures)

	slow change	baseline	fast change
A	.048	.102	.078
B	.195	.218	.147
C	.064	.162	.068
D	.023	.105	.046
E	.230	.366	.203
F	.004	.003	.001
G	.009	.002	.008
H	.165	.291	.181
I	.004	.007	.005
J	.257	.101	.263

Table 5. Distributions of failure types in Phoenix scenarios

A chi-square test shows that the distribution of failure situations is *not* independent of environmental conditions ($\chi^2(9) = 329.4$, $p < .0001$). The fast-change scenario generates more errors (1 every 2.1 hours for fast; 2.8 hours for slow; and 2.4 hours for the baseline); more importantly it generates a different *pattern* of errors than the slow-change scenario. A complete model of failure recovery should explain why particular failures are more or less likely in different environmental conditions; this is the goal of our current research.

5.5 Experiment Recap

We tested five hypotheses about the behavior of failure recovery methods in a planner and one hypothesis about the distribution of failure situations in the environment. The independence hypotheses described in Section 4 clearly hold for the bulldozer agents, but the results are more complicated for the fireboss: The two replan methods behave qualitatively differently from the local methods. We predicted and found that the optimal ordering strategy results in a lower overall cost than the randon strategy. We predicted that costs can be reduced by modifying the failure recovery method set. While the method set modifications did result in lower costs for the targeted situations and in a lower overall cost, the difference is not statistically significant due to increased costs in other situations. Consequently, this hypothesis has yet to be conclusively demonstrated or refuted. The last hypothesis assumed that the distribution of failure situations is independent of the environment conditions. In fact, we showed that the distribution of failures is not independent of environmental conditions, in particular, the rate of change in the environment.

6. Discussion

The goal of these experiments was to test a model of failure recovery performance and demonstrate that the model could be used to direct the design of failure recovery in novel environments. To that end, the most important result from these experiments is the insensitivity of certain properties of general methods to aspects of their execution context: cost is independent of position and failure situation, and probability of success for a situation is independent of position of execution. While the independence assumptions have been tested only in the Phoenix environment, they held constant across the three different environmental change scenarios described in the last section[2].

The experiments also disclosed basic differences between the behavior of local (e.g., WATA, RV, RAV, and SA) and global methods (e.g., RP and RT). Local methods are far more predictable (less variance in cost), but have correspondingly lower probabilities of success. This leads to a trade-off between predictability and power (probability of success in this case). We believe that this trade-off is general because so long as the scope of changes is small and well-known (i.e., predictable), the probability of success will be limited to the likelihood that the source of the failure is within the limited scope of the changes. Conversely, as the scope of changes increases, the probability of encompassing the source of the trade-off increases, but the predictability in cost reduces correspondingly. For the designer, the implications are obvious: Lack of predictability may not be tolerable in environments with hard real-time deadlines; just as lack of recovery success may be intolerable in environments with extremely high costs for outright failures. For the Phoenix environment, the optimum strategy combined with the recovery method set modifications has led to a strategy of trying the local methods first, when they are applicable, and resorting to the global methods only after the local and specialized methods have been exhausted.

Over the course of these experiments, the model in Eq. 1 guided the design of the recovery method set for Phoenix. agents. We started with a core set of general methods that performed reasonably well. In fact, the overall recovery rate (percentage of failures that are repaired) for the core set of methods was at least 70% for all but one failure situation in the baseline experiment; the lowest recovery rate increased from 24% to 46% in the second experiment with smaller improvements in most of the other failure types, and increased to 56% in the last (producing overall recovery rates of 81%, 88% and 90% in Experiments 1, 2 and 3, respectively). Moreover, as stated earlier, the overall cost of recovery decreased from 2811 to 2530 to 2370 with the refinements to the recovery method set. The high level of basic performance, subsequent improvements produced from modifications suggested by the model, and the independence results demonstrate that this model is effective for iterative design of recovery method sets for new environments.

We tested the modifications to the control of failure recovery amd the set of methods in a single environment. Since we know now that different environmental conditions lead to different distributions of failures, it remains to be seen whether a method set designed for one set of conditions is appropriate to another. Based on the results of the baseline experiments, it appears that an untuned set of methods perform reasonably well, but we have yet to predict how well a method set will do when the environment changes.

7. Conclusions

The agent architecture for the Phoenix system includes many components. In this paper, we have described the analysis and design of just one: failure recovery for the planner. In designing this component for the Phoenix environment we analyzed the environment, modelled the behavior of the component with respect to its environment, designed the component, hypothesized its behavior and tested the hypotheses. These five steps, along with additional steps to revisit some of hypotheses based on experiment results and to generalize those results, constitute the MAD methodology (Cohen 1991). We believe these steps are necessary to the design and analysis of AI systems.

As an understanding of failure recovery, our model is weak. It is concerned only with the design and selection of a method set for an environment. Other models of failure recovery have addressed the role and form of individual methods, specifically, replanning. Morgenstern's model defines a logical formalism for when it is necessary, desirable and possible to replan (Morgenstern 1987). Kambhampati's model of plan modification (Kambhampati 1990) guarantees completeness, coverage and efficiency for replanning in hierarchical planners. Yet, we are still some distance from a complete model of failure recovery.

It is an open question how complete our models must be: To expedite design, we used mean costs from data with considerable variance, and still managed to improve the design. In fact, the measures of cost and success were subject to variation in part because their definitions are not always clear. Local methods exert a local influence on the plan; thus, we can easily determine the scope of their effects (e.g., cost and success). Global methods exert far-reaching influence and so it is more difficult to arbitrarily assign a horizon of influence to them. As suggested in the discussions of independence results, global methods tend to interact with one another and may produce downstream effects that should change the evaluation of performance. The next phase in this project will be to decompose these measures into parameters that allow us to better predict performance, and through a semantic model, eventually understand how failures arise and how they are best confronted (Howe, in preparation).

---

[2] We are eager to hear from other researchers whether the same results hold in their environments.

Acknowledgments

This research was supported by the Advanced Research Projects Agency of the Dpeartment of Defense and was monitored by the Air Force Office of Scientific Research under contract No. F49620-89-C-00113, the Office of Naval Research, under a University Research Initiative grant, ONR N00014-86-K-0764, and also under ONR N00014-88-K0009, and the National Science Foundation under an Issues in Real-Time Computing grant, CDA-8922572. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. We also wish to thank David Westbrook and Scott Anderson for their invaluable assistance in running the experiments and members of the Phoenix project for the construction of the testbed.

References

Jose A. Ambros-Ingerson and Sam Steel. 1988. Integrating planning, execution and monitoring. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, Minnesota.

Paul R. Cohen. 1991. A survey of the eighth national conference on artificial intelligence: Pulling together or pulling apart? *AI Magazine*, 12(1).

Paul R. Cohen, Michael Greenberg, David M. Hart, and Adele E. Howe. 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3).

Kristian J. Hammond. 1987. Explaining and repairing plans that fail. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 109-114, Milan, Italy.

Steve Hanks and R. James Firby. 1990. Issues and architectures for planning and execution. In Katia P. Sycara, editor, *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 59-70. Morgan Kaufmann Publishers, Inc.

David M. Hart, Scott D. Anderson, and Paul R. Cohen. 1990. Envelopes as a vehicle for improving the efficiency of plan execution. In Katia P. Sycara, editor, *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 71-76. Morgan Kaufmann Publishers, Inc.

Philip J. Hayes. 1975. A representation for robot plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tiblisi, Georgia, USSR.

Adele E. Howe and Paul R. Cohen. 1990. Responding to environmental change. In Katia P. Sycara, editor, *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 85-92. Morgan Kaufmann Publishers.

Adele E. Howe. Adapting Planning to a Complex Environment. PhD Dissertation, Dept. of Computer and Information Science, Univ. of Massachusetts, forthcoming.

Subbarao Kambhampati. 1990. A theory of plan modification. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 176-182, Boston, MA.

Leora Morgenstern. 1987. Replanning. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 5-1 - 5-0, Austin, TX.

Reid G. Simmons. 1988. A theory of debugging plans and interpretations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, Minnesota.

Reid G. Simmons. 1990. An architecture for coordinating planning, sensing and action. In Katia P. Sycara, editor, *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292-297. Morgan Kaufmann Publishers, Inc.

Herbert A. Simon and Joseph B. Kadane. 1975. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence Journal*, 6:235-247.

David E. Wilkins.1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann Publishers, Inc, Palo Alto, CA.