Expectation Failure as a Basis for Agent-Based Model Diagnosis and Mixed Initiative Model Adaptation during Anomalous Plan Execution*

Alice Mulvehill, Brett Benyo, Michael Cox, and Renu Bostwick
BBN Technologies
Intelligent Distributed Computing Department
10 Moulton Street, Cambridge, MA 02138
amm@bbn.com, bbenyo@bbn.com, mcox@bbn.com, renu@bbn.com

Abstract

Plans provide an explicit expectation of future observed behavior based upon the domain knowledge and a set of action models available to a planner. Incorrect or missing models lead to faulty plans usually characterized by catastrophic goal failure. Non-critical anomalies occur, however, when actual behavior during plan execution differs only slightly from expectations, and plans still achieve the given goal conjunct. Such anomalies provide the basis for model adjustments that represent small adaptations to the planner's background knowledge. In a multi-agent environment where 1000 or more individual plans can be executing at any one time, automation is required to support model anomaly detection, evaluation and revision. We provide an agent-based algorithm that generates hypotheses about the cause of plan anomalies. This algorithm leverages historical plan data and a hierarchy of models in a novel integration of hypothesis generation and verification. Because many hypotheses can be generated by the software agents, we provide a mechanism where only the most important hypotheses are presented to a user as suggestions for model repair.

1 Introduction

Planning in most domains requires a description of relevant entities and processes. These are typically described by a set of models. Feedback collected during or after plan execution can be used to refine models so that the planner performance becomes more adaptable to the environment and improves with experience. The research presented here shows how small discrepancies between what is expected in a plan and what actually happens during plan execution can determine an effective refinement to the models upon which plans are based and hence improve plan performance.

Plans themselves provide an expectation about what should happen when the plans execute. An *expectation failure* occurs when the expected performance diverges from what the models of the domain predict, and such failure provides the basis for learning [Birnbaum *et al*., 1990; Ram *et al.*, 1995]. Diagnosis is a symptom to fault mapping that links the expectation failures detected during execution to the model changes necessary to prevent such failures from recurring in the future. However this blame assignment problem is nontrivial, because many factors can contribute to a single error, multiple errors can co-occur, and the diagnostic mapping between symptom and fault can be very indirect. Indeed catastrophic plan failure may result from so many factors that diagnosis is not practical. But plans can also fail in nondisruptive ways that result in suboptimal performance while still achieving the goal set. It is this type of expectation failure we term an *anomaly*.

The dynamic and continuous nature of our planning domain and the complexity of the models of this domain has led us to develop an agent-based model diagnosis tool that handles multiple types of plan anomalies (*e.g.*, consumable, spatial, and temporal). The agents use a model hierarchy with spreading-activation likelihood functions to control hypotheses generation. The agents reference historical plan data to recognize past anomaly patterns. When an anomaly has occurred in the past and all planned activities have completed, the agents decrease the likelihood that a model repair is required. In addition, the agents use a temporal link analysis to localize failure symptoms within model processes. The diagnosis system includes a mixed-initiative component that provides model error and repair suggestions (resulting from the most likely hypotheses) to a user who can accept, ignore, or reject them. Others have examined agent-based [*e.g.*, Roos and Witteveen, 2005] and model-based [*e.g.*, Williams and Nayak, 1999] diagnosis for planning as well as mixed-initiative approaches [*e.g.*, Mulvehill

* This research is sponsored by the United States' Defense Advanced Research Projects Agency (DARPA) under contract FA8750-04-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsoring institutions, the U. S. Government or any other entity.

and Cox, 1999]. Here we present a novel combination of the three.

In this paper, we describe our technical problem and provide detail on our approach. We also provide the results of a study that compares a hand-coded approach (as described in this paper) with our agent-based approach.

2 Model-Based Mission Planning and Execution Monitoring

Our planning domain is focused on military air operations and our tool suite for this domain includes a generative planner, a plan monitor and a model adaptor component. The planner creates a hierarchical task network that allocates resources for specific tasks that achieve a set of input objectives. The monitor tracks the behavior of a simulator that executes the activities in the plan. The model adaptor provides and refines models to account for changes in the world state and to improve plan performance.

The domain is defined by two sets of hierarchical models. The entity models describe objects in the domain, and the process models define the actions that entities perform. Entities include hundreds of configurations of munitions and fuel tanks that may be loaded on an aircraft for specific missions. Processes encode the different mission variations for which the aircraft may be deployed, *e.g.*, the sequence of activities the aircraft perform varies depending on the mission being performed and the aircraft's configuration load at the time. In addition the entity hierarchy defines specific types of aircraft using default parameters valid for whole families of aircraft types when applicable and uses specific overridden individual parameters when necessary. Finding and fixing model problems in such a large and complex model is difficult.

As an example, consider a plan servicing 300 objectives, ranging from targets that need to be destroyed to surveillance tracks that must be flown. During execution of this plan, the plan monitor can detect hundreds of anomalies. Figure 1 shows a data set of expected and actual fuel values during various activities of a close air support mission flown by A-10 aircraft.

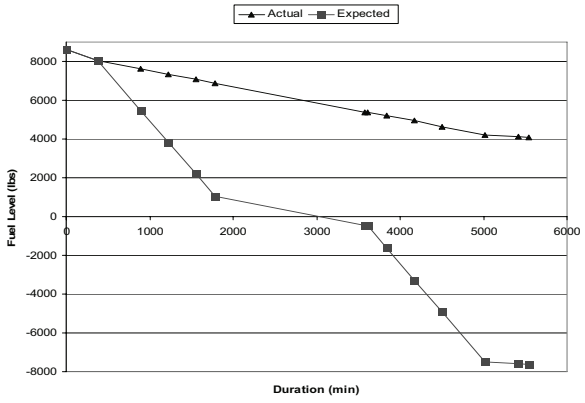

Figure 1. Data set 1

During takeoff and climb (the first two points), the values are equal, but as the mission continues the values begin to diverge at successive navigation points. The difference between planned (square) values and simulated (triangle) values at point three, however, is not great enough to generate an anomaly by the monitor. At point four and all subsequent locations the difference exceeds a threshold, and the monitor sends anomaly messages to the model adaptor component. Yet during the midpoints of the mission and at the very end, the difference between actual and expected fuel levels remains constant, thus indicating that no problem exists in those segments of the mission, despite anomaly messages from the monitor. In actuality, the fuel level discrepancy was introduced by problems in previous mission segments.

3 Agent-Based Diagnosis and Adaptation

The problem for the model adaptation component is to map failure symptoms from the monitor such as fuel consumable anomalies to causal faults such as incorrect model parameters like high fuel burn rates on navigate activities for A-10s. The reported anomalies can be simple or severe and can be caused by many sources, including the models. As anomalies are detected, they are classified and provided to the model adaptor component. Here users, who are familiar with the models, use the model adaptor software to determine if a model is responsible for an anomaly, and if so, repair and republish it to the other planning components. Because this particular planning application has the potential to scale to over 1000 concurrently executing missions, an agent-based diagnosis system was developed to support anomaly management, interpretation and model repair suggestion generation.

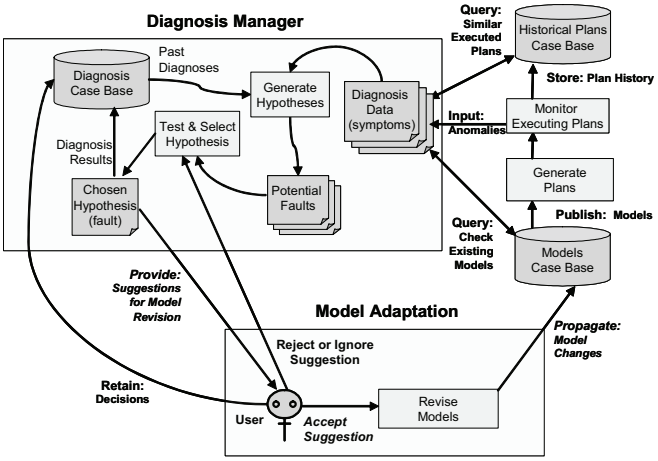

Figure 2. Agent-based diagnosis and mixed-initiative model adaptation processes

The diagnosis system is composed of a set of agents. Each agent is a java class and is associated with a hypothesis about a model failure. The Diagnosis Manager (detailed in Figure 2) is the component responsible for hypothesis generation, along with the creation and execution management

of these agents. Each agent executes a set of tasks, gathering and performing computations on data from past and current runs in order to determine whether its hypothesis is likely or not. When an agent attains a high degree of confidence in its hypothesis, it will further generalize or refine the hypothesis in a mixed-initiative process with the user. Figure 2 shows both the agent-based manager and the mixed-initiative component. Specialized agents combine and evaluate hypotheses derived from single agents. These specialized agents output model revision suggestions, if warranted.

3.1 Hypothesis Generation

Currently the Diagnosis Manager uses four methods to generate new hypotheses and associated agents. First, a set of general hypotheses that should always be investigated is provided at initialization time. Examples of such general hypotheses are that the planner is using an old model version, or that there is a non-model related error, such as a faulty sensor or a bug in the world simulator. The second type of hypothesis generation is initiated by the arrival of a plan anomaly message. The Diagnosis Manager analyzes the anomaly within the context of the plan activity in which the anomaly occurred and within the context of a network of related activities and entities; it generates a list of the model classes and parameters that were used to model the activity. Each of these classes and parameters within the model is suspected as a potential source for the error, and an appropriate hypothesis is generated. Consider one example plan anomaly referencing an altitude discrepancy on a navigate activity for an A-10 flight group. Related models involved in the generation of this planned activity include the A-10 entity model, the navigate activity model, the CAS (close air support) mission process model, a set of entity models for objects carried on the A-10 (missiles, bombs, and fuel tanks), and a set of geometric object models defining the space through which the aircraft flies. For each of these models, a hypothesis that there is an error in that model is generated by the Diagnosis Manager.

The third type of hypothesis generation is triggered when the planner is unable to satisfy a particular goal. This problem can point to an inconsistency between the capabilities modeled for the actor entity, the process models used by the planner to support reasoning, and a potential change in the world state. In this case, model error hypotheses will be generated for each available actor model and any related models referenced in the unsatisfied goal. The fourth type of hypothesis generation is hypothesis generalization and refinement. Specifically, when an agent's belief that its hypothesis is true (called the agent's likelihood) reaches a certain value, rules can fire that generate more general hypotheses by traversing up the model hierarchy.

3.2 Hypothesis Refinement

In addition to generalizing hypotheses by utilizing the model hierarchy, agents can generate new hypotheses by refining their current hypothesis. This refinement can take two forms. First, the Diagnosis Manager can specify what part of the model might be in error, either giving a specific parameter or a specific constraint of a process model. Second, the Diagnosis Manager can specify an interaction between models, such as an error for the A-10 model on a navigate activity flying a CAS mission.

Parameter refinement is accomplished by data mining the model case base to generate a list of parameters that have adaptable values associated with the model in question. For each such parameter we reference a predefined knowledge base to determine if that parameter can cause the anomalies the agent has seen. If so, a new diagnosis agent with a refined hypothesis is generated. The new agents compare the current anomalies with historical data and with expected data for each such parameter type. These newly specified hypotheses are then further refined by looking for interactions between models. For this domain, parameters can have different values for sets of aircraft, activities, and missions.

For example, consider consumable fuel anomalies for an A-10 aircraft. When the "A-10 Aircraft model error" agent reaches a high likelihood, the hypothesis is refined. An examination of the A-10 model data reveals a list of adaptable parameters, and the causal model narrows this list to the *maxFuel* parameter, and the *fuelBurnRate* parameter. For *fuelBurnRate*, there are multiple values in the models based on the aircraft, activity, and mission. Thus, we can further refine these hypotheses for the A-10 Aircraft by tracking the highest likelihood activity and mission. In our example, the consumable anomalies occur on Navigate activities for any mission type, so the Navigate model error agent has a high likelihood, whereas no mission model error agent is very high. Therefore, we refine the hypothesis to suspect an "Error with *fuelBurnRate* on A-10 aircraft performing Navigate activities on any mission type." Similar refined hypotheses for the other parameter choices are also generated.

We call the refined hypotheses with the highest likelihood a *suggestion*, because they can be presented to the user to support model repair. The operator can accept, reject, or ignore a suggestion and in doing so, provides feedback to the suggestion agent. At this time, the feedback is in the form of a selection from a list of pre-defined options. Examples are: "suggestion is too general/specific", "suggestion makes no sense", and "source of error is not in the models."

3.3 Likelihood Calculation

When a new hypothesis and associated agent are generated, the agent is loaded with a predefined set of tasks. The goal of a task is to calculate the belief that the agent's hypothesis is true, false, or uncertain by performing a certain calculation or examining a specific data source. We are experimenting with various forms of evidence combination to merge the belief calculations from various tasks into one single value for the agent, including an application of the Dempster-Schafer evidence combination theory [Dempster, 1968; Shafer, 1976] and an operator modifiable set of likelihood functions.

Examples of tasks executed by most agents include: (1) querying the model case base to determine what model changes have been made in the past, (2) querying the historical plan case base to determine if similar anomalies ad-

versely affected previously executed missions, (3) examining currently executing missions and activities to see if other activities involving this model are completing without error, and (4) performing a statistical analysis of the current anomalies with the involved models. Agents with hypotheses about activity models can execute an additional task (5) that examines the temporal successor links in the plan to determine if anomalies can be blamed on the previous activity in the mission. Refined hypothesis suggestion agents can execute a task (6) that compares the set of current anomalies and their actual data values with a theoretical set of anomaly data that we would expect to receive if the model error in question were present. This theoretical anomaly data can come from information about previous model errors or from a human-defined knowledge base.

For example, for a fuel burn rate error, the expected results (defined in the knowledge base and case base of past fuel burn rate errors) are a decreasing slope for actual values as the mission progresses, with an increasing value for the delta between actual and expected values. If the fuel burn rate is in error, we expect the difference between actual and expected fuel levels to get progressively worse as the mission executes. Alternatively, if the maximum fuel level is in error, we expect the difference between expected and actual fuel levels to remain constant," and we expect the actual fuel level value to decrease.

3.4 Example Scenario

For example, consider an A-10 model error agent. Assume that a case base query (task 2) reveals a handful of successful A-10 missions using an older version of the A-10 aircraft model. This discovery would result in a small belief value that the current A-10 model might be in error and a large belief that we are uncertain. The supporting evidence is that previous successful missions were found and the A-10 model was recently updated. Next, assume that of the missions in the currently executing plan, the ones without A-10 aircraft are experiencing few anomalies (task 3) and most of the ones with A-10 aircraft are generating fuel value anomalies (task 4). Both of these tasks would result in a significant belief value that the A-10 model has an error. Combining these beliefs using the Dempster rule of combination [Dempster, 1968] would result in a high likelihood for this A-10 agent.

Refining this hypothesis gives rise to two new agents hypothesizing that either the *maxFuel* parameter or the *fuelBurnRate* parameter of the A-10 is in error. Examination of the anomaly value data (task 6) shows that the delta between the expected and actual fuel values continued to grow as the mission progressed. Historical case base data shows that this is consistent with previous *fuelBurnRate* parameter errors; however the knowledge base data for maxFuel errors show that this delta between expected and actual fuel values should stay constant. Thus the "A-10 *fuelBurnRate* error" agent ends up with a higher likelihood than the "A-10 *maxFuel* error" agent.

Next, consider the Navigate model error agent. Similar tasks (tasks 2, 3, and 4) would result in a moderate likelihood, because the anomalies additionally occur on many types of activities, such as Climb, Descend, and Orbit. However, examining the temporal link structure of the process model (task 5) shows that the gap between expected and actual fuel values increases mostly on the Navigate legs. Because both the "A-10 *fuelBurnRate* error" and "Navigate error" agents now have a high likelihood, a refined hypothesis from combining the two will be generated.

As a result of the agent processing, the user is finally presented with a set of suggestions. The highest confidence suggestion, from the highest likelihood agent is to check the value for *fuelBurnRate* for A-10 aircraft on Navigate activities for all mission types. Less likely suggestions presented to the user include "Check the *fuelBurnRate* parameter for A-10 aircraft for all FlightActivities", and "Check the *maxFuel* parameter for A-10 aircraft."

4 Evaluation

As discussed earlier, anomalies occur in the form of expectation failures when the expected value of some parameter as specified in a plan differs from an actual value as determined during monitoring of plan execution. Because most of our data is proprietary, we created an artificial data set that is representative of the real data for use in this paper. As shown by Figures 1, 3 and 4, we created three data sets for aircraft fuel levels with which to evaluate the effectiveness of our approach. Data set 1 (Figure 1) represents a typical case where a mismatch occurs, because the fuel burn rate in the model for the A-10 aircraft is mistakenly set too high. Thus the plan predicts the fuel at increasingly lower levels than what actually occurs. Data set 2 (Figure 3) represents the case where false anomalies are reported by the plan monitor. In this case, a faulty sensor reports all fuel values to be zero. Data set 3 (Figure 4) reflects the case where a fuel leak causes an A-10 to crash. In the latter two sets the generated plan is correct, but in the first it is not. Note also that superficially the third set is similar to a combination of features from the other two.

The results from our study compare a naïve method whereby a simple set of hand-coded rules is compared to the diagnosis process described earlier. The rules can be summarized as follows:

```
If Expected(current-activity)
   < Actual(current-activity) then
Is-Faulty ← parameter(current-activity)

If Was-Faulty = parameter(last-activity)
   and ThisDelta = LastDelta then
Is-Faulty ← null
```

Where

```
ThisDelta = Actual(current-activity)
            -Expected(current-activity)
LastDelta = Actual(last-activity)
            -Expected(last-activity)
```

4.1 Data Set 1

The results between the naïve method and the agent-based diagnosis method are comparable on the first data set of eleven anomalies. With both techniques, the fault is identified as a fuel burn rate parameter for the navigate activity on the A-10 model. The crucial aspect of the anomaly pattern is that the difference between expected and actual values changes for those activities related more directly to the fault. The delta remains the same for an activity that is not faulty. This distinction is evident within Figure 1 from the relative slopes of various segments of the two curves. When the slopes are the same and the curves parallel, the differences between expected and actual do not change, and any anomaly can be considered a false positive. The slopes are different in the two segments that correspond to navigation activities (points 3-6 and 9-12). That is, the plan predicts a steeper decline in fuel than actually occurs.

The temporal link analysis discussed earlier (task 5 in Section 3.3) forms the basis to determine the slopes and hence identify the correct activity that represents the diagnosed fault given the anomaly symptoms.

4.2 Data Set 2

Not all anomalies detected by the plan monitor will follow the pattern of data set 1. Data set 2 represents the case of false symptoms (in this example, where the plan simulator or fuel sensor is bad) and hence provides incorrect data for comparison to the expected values (which are good). This data set cannot be handled correctly by the naïve rules, because for each anomaly input into the system, an expectation failure exists and the deltas progressively vary.

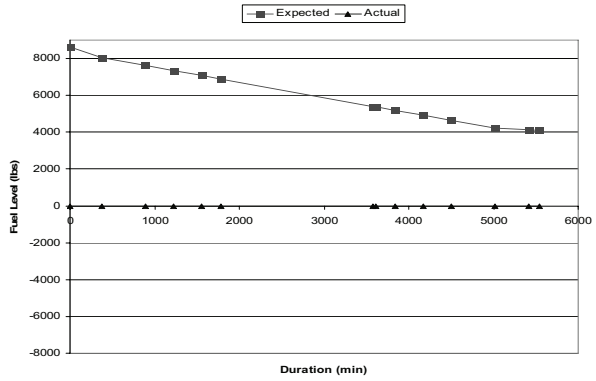

Figure 3. Data set 2

What is needed is recognition that zero fuel values represent unrealistic data and thus the input is faulty. Our diagnosis system can check for such conditions using a knowledge-based sanity check (task 6 in Section 3.3). Data Set 2 has a constant actual fuel level, and a decreasing value for the delta between actual and expected fuel values. Because this does not match the expected results for fuel burn rate or maximum fuel level errors, neither hypothesis ends up with a high likelihood. The likelihood of one of the default hypothesis agents, the "faulty sensor or simulator bug" agent, is inversely proportional to the maximum likelihood of the other suggestion agents. Because no other agent has a high likelihood, the suggestion that there is a faulty sensor or a simulator error is presented to the user.

4.3 Data Set 3

The third data set cannot be easily analyzed by the agents alone, nor can the rules diagnose the fault. Indeed the fault is outside the system and a proper diagnosis relies upon insight provided by a user. In this case, not only does the expectation failure progressively increase, but at a certain point in the input stream, the fuel ends at zero. At this point a series of spatial anomalies enter the input stream.

The expectation in the plan specified that the altitude would be a given height, but instead the altitude is zero. Using the simple rules results in spatial anomalies being most suspect, but with our system, although the agents are able to focus correctly upon the fuel anomalies as opposed to the spatial anomalies, the complete diagnosis depends upon the human ignoring the spatial anomaly suggestions from the suggestion manager. This is done using the feedback received through the mixed-initiative adaptation process described in Figure 2.

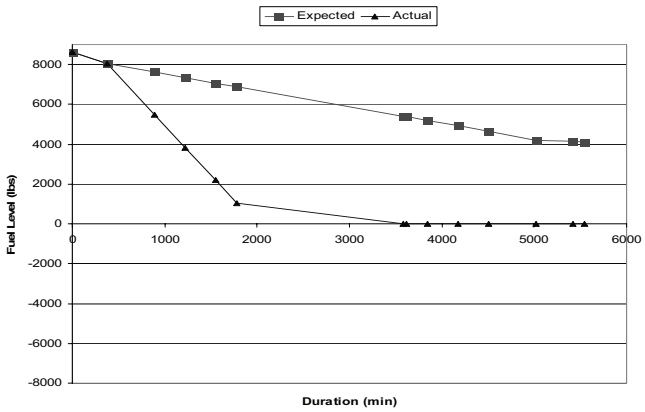

Figure 4. Data set 3

5 Conclusion

Our evaluation results indicate that while our agent-based approach is superior to the simple hand-generated method we presented, our approach is unable to handle all classes of faulty anomaly reporting unless there is a human in the loop. However, even with the mixed-initiative approach, we still are faced with the problem of focus. In other words, a user might reject or ignore a suggestion because there is a more important problem being reported by the anomalies or because visual cues from other tools show a particular state that is inconsistent with anomaly reporting. For instance, in the example where the aircraft runs out of fuel and crashes, if the reported anomalies do not result in timely adjustment of the activities to include an aerial refuel, then the plan will fail. The crash will be indicated in a spatial anomaly with a

constant altitude value of zero. Future versions of the suggestion tool need to account for this situation.

In some early work with human modelers, we have determined that our current tool is only satisfying some of the needs of the human modeler. For example, diagnosing a model problem is complicated by the fact that an error in a single parameter for a given modeled entity could result in multiple problems (like the third issue listed below), and by the fact that the models are hierarchical. For example, in the case of an anomaly related to the fuel level for a particular aircraft flying a specific activity of a mission, some questions the human modeler might have are as follows:

- Is the fuel level higher or lower than modeled? – Our agents can provide this suggestion.
- Is the fuel burn rate parameter entered in the model for a specific activity for a specific load set too high or too low? – Our agents can provide this suggestion.
- Is it a fuel level problem or is the speed specified in the model too high or too low which would cause the aircraft to burn fuel differently than expected? - Currently both suggestions would be presented.
- Is it really a modeling problem or does the specific aircraft that is flying have a fuel leak? – In the future version of the anomalies, there will be severity and source data provided so that we can more accurately answer this question.
- Is it just this specific type of aircraft (a subclass in the model hierarchy) that has this problem or is it the whole family of aircrafts of this type (a superclass in the model hierarchy) that has the problem? – In experiments performed to date, we were able to leverage the model hierarchy to satisfy this requirement.

The diagnosis system we have developed to date produces suggestions that are consistent with some of the types of suggestions required by the human modeler. In addition, the diagnoses that are generated to date are also consistent with the theory of reasoning failure proposed by Cox and Ram (1999). Furthermore we have leveraged the work of Lesser [Horling, 2001] to develop our diagnosis hypotheses networks. Our use of historical plan data and the generalized reasoning we do with the model case base is our novel contribution as we are unaware of any work of this particular hybrid being done in the past.

The experimental data we present here shows how our diagnosis agent approach provides an improvement over a hand-coded method. In other experiments conducted to date, but not reported here, we have found that our diagnosis agents can generate suggestions that are useful to the user. For example, the agents were able to find subtle errors even when the user was distracted with other recurring problems, and we have had some measurable success in providing correct suggestions that were derived from generalizations about model class problems.

The next step in increasing fidelity of the suggestion system is keeping track of accepted, rejected, and ignored suggestions to affect the likelihood values calculated by the diagnosis and suggestion system for future anomalies that come in. As the human modeler gives the system feedback about the accuracy of the diagnoses and related suggestions, the system can improve its diagnosis likelihood calculations and future suggestions to the modeler.

In this paper, we have introduced a new agent-based diagnostic approach that is being developed to support both offline and real time model diagnosis and repair. We have presented some preliminary experimental results that indicate that this approach is adequate in our specific problem domain. This diagnostic tool is currently an integral part in a closed loop planning, execution and adaptation framework. Our goal is to provide suggestions to human modelers that have such integrity that in the future the human will allow the system to detect, recommend, and in fact automatically repair models.

7 References

[Birnbaum *et al.*, 1990] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-Based Diagnosis of Planning Failures. *Proceedings of the Eighth National Conference on Artificial Intelligence*. Boston, MA, pp. 318-323.

[Cox and Ram, 1999] M. T. Cox and A. Ram. Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence*, *112*, 1-55.

[Dempster, 1968] A. Dempster. A generalization of Bayesian inference, *Journal of the Royal Statistical Society*, Series B, Vol. 30, pp. 205-247.

[Horling, *et al.*, 2001] B. Horling, B. Benyo, and V. Lesser. Using Self-Diagnosis to Adapt Organizational Structures. *Proceedings of the 5th International Conference on Autonomous Agents*, ACM Press, pp. 529-536.

[Mulvehill and Cox, 1999] A. Mulvehill and M. Cox. Using Mixed Initiative to Support Force Deployment and Execution. In M. T. Cox (Ed.), *Proceedings of the 1999 AAAI-99 Workshop on Mixed-Initiative Intelligence* (pp. 119-123). AAAI Press, Menlo Park, CA.

[Ram *et al.*, 1995] A. Ram, S. Narayanan, and M. T. Cox. Learning to trouble-shoot: Multistrategy learning of diagnostic knowledge for a real-world problem solving task. *Cognitive Science*, 19(3), 289-340.

[Roos and Witteveen, 2005] N. Roos, and C. Witteveen. Diagnosis of plan execution and the executing agent. *Proceedings of Computational Logic in Multi-Agent Systems: Sixth International Workshop (CLIMA VI).*

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence*,. Princeton University Press.

[Williams and Nayak, 1999] B. Williams, and P. Nayak. A Model-based Approach to Reactive Self-Configuring Systems. *Proceedings of Workshop on Logic-Based Artificial Intelligence*. University of Maryland, Computer Science Dept. Washington, DC, June 14-16.