SPRIE: System for Plan Recovery in Intelligent Environments

by

Stefanie Chiawhei Chiou

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2004

(C) Stefanie Chiawhei Chiou, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.
SPRIE: System for Plan Recovery in Intelligent Environments

by

Stefanie Chiawhei Chiou

Submitted to the Department of Electrical Engineering and Computer Science on May 20, 2004, in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science and Engineering

Abstract

As intelligent spaces have become more prevalent, the need for diagnosis and recovery from failures in these spaces has also grown. In an effort to make these types of spaces more useable for people who neither care about nor want to understand the underlying technology, encapsulation of high-level ideas into plans is being explored. Unfortunately, these plans are not always executed flawlessly. To deal with these failures, we have developed a system for plan recovery in intelligent spaces called SPRIE. SPRIE is an automated system that uses a two-stage process for recovery from plan execution failures. In the first stage, the failure is diagnosed using Bayesian networks. In the second stage, an alternate plan is selected to recover from the failure. By automating the recovery process, the burden of understand the underlying architecture of the intelligent space is removed from the user.

Thesis Supervisor: Howard E. Shrobe
Title: Principal Research Scientist
Acknowledgments

I would first like to thank Dr. Howard Shrobe, my advisor, for his guidance and support throughout this project. His perspective and insights were invaluable especially in the early stages of design and development of SPRIE.

I would also like to thank Gary Look, Robert Kochman, and Kevin Quigley for their input and help in the initial version of the diagnosis system, which later evolved into the system SPRIE. I am also grateful for all of the help that Stephen Peters gave in a variety of areas.

I would also like to thank Henry Wong for both his coding advice and editing of an early draft of this thesis.

Finally, I would like to thank my parents for their constant and unfailing support.
# Contents

1 Introduction ........................................................................... 1

2 Design Overview ................................................................... 5

3 Implementation
   3.1 World Model ................................................................. 9
   3.2 Diagnosis System .......................................................... 11
   3.3 Recovery System ............................................................ 13
   3.4 Plan Library .................................................................. 16

4 Discussion
   4.1 World Model ................................................................. 17
   4.2 Diagnosis System .......................................................... 18
   4.3 Recovery System ............................................................ 18
   4.4 Plan Library .................................................................. 19

5 Related Works ...................................................................... 21

6 Conclusions ......................................................................... 23

7 Further Work ....................................................................... 25
# List of Figures

1-1  CSAIL's Intelligent Room. .......................... 2

2-1  The form of each resource entry in the world model. .......... 6

2-2  The communications and interactions between the systems in SPRIE 8

3-1  The format for the world model entries. .................. 10

3-2  The reliability relationships between the power source, the computer,  
       and the projector for a plan trying to show a slide with the projector. 13
Chapter 1

Introduction

Over the last twenty years, computing has evolved from massive mainframes that were only available to large organizations with large budgets to small desktop personal computers that can be found in most homes today. Computing has become steadily more and more pervasive in the everyday lives of people everywhere. As the prevalence of computers continues to grow, intelligent spaces, such as the Intelligent Room in MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) will become more common as well. A picture of this space is shown in Figure 1-1. These intelligent spaces consist of a collection of sensors and actuators, which allow human users to interact with the space naturally through speech and vision. For example, the CSAIL Intelligent Room uses a series of cameras to track users as they move through the space to provide input for agents that react to user movements. In addition to the visual information from cameras, the room can also gather audio information from wireless microphones, which are worn by users. Audio communication is useful, because it allows users to request services such as light adjustment. These modes of communication are more familiar to people because they more closely mimic human-human interactions.

Unfortunately, the Intelligent Room, like most of these spaces is controlled by a complex system of hardware and software that is difficult if not impossible for the average user to understand. Since it cannot be assumed that average users will have an in-depth understanding of the system architecture, these spaces need to provide
mechanisms that form a layer of abstraction between the user and the inner workings of the system. The Intelligent Room currently uses Metaglue\cite{10}, a distributed agent system written in Java to perform all tasks in the room. Metaglue agents are designed to perform simple but specialized tasks and can communicate and cooperate with each other to perform more complex tasks for a user. These agents are typically autonomous, intelligent, and robust.

Since the agents each perform small, specialized tasks, most high-level tasks that a user might want to complete need the cooperation and interaction of many agents. Therefore, in order for the Intelligent Room to be effectively utilized by people who are not familiar with its architecture, a semantic translation service is needed to translate the high-level tasks that users want to accomplish into a series of low-level tasks that can be performed by the appropriate agents. One possibility that is being explored in the Intelligent Room is the use of plans to define high-level commands \cite{8}. For example, the command “Computer, configure the room for a presentation,” actually consists of a number of low-level tasks such as dimming the lights, turning on the projector, connecting the projector to the appropriate computer, etc. The use of plans to encapsulate all of these low-level tasks makes it possible for a user to configure the Intelligent Room for a presentation without needing to know the

Figure 1-1: CSAIL's Intelligent Room.
specifics about how to activate and manipulate individual agents.

At the same time, a system that uses plans to automatically perform tasks for an individual user gives rise to the need for a recovery system that deals with failures in these plans in an autonomous fashion. Since users are not required to have intimate knowledge of the underlying architecture to perform tasks in the room, they cannot be expected to understand how to diagnosis and recover from an error on an agent level. The depth of their knowledge can only be assumed to be as deep as is required to perform the high-level tasks, so an automated recovery system is an essential feature that makes the space useable for average users. The System for Plan Recovery in Intelligent Environments (SPRIE) is designed to fulfill this need for automated error recovery in the Intelligent Room.

SPRIE is designed to use a two-stage approach to recover from plan failures. The two stages utilize two different systems that work together to accomplish automated recovery. The first system is the diagnosis system, which is used to isolate the cause of failure. The second system is the recovery system, which utilizes the information gained from the diagnosis system to determine the best alternate plan to accomplish the original goal.

This thesis discusses the design and implementation of SPRIE in detail. Chapter 2 provides a design overview of the system as a whole and discusses how the different modules in the system interact. Chapter 3 is a more detailed technical description of the implementation of the various modules in SPRIE. Chapter 4 discusses the design decisions of SPRIE and cites areas for improvement. Chapter 5 explores some of the other work that has been done in diagnosis for multi-agent systems. Chapters 6 and 7 discuss what SPRIE has accomplished and further work and improvements for the system.
Chapter 2

Design Overview

In an effort to abstract low-level details away from users of the Intelligent Room, complex, high-level commands are decomposed into simple tasks, which are combined to form a plan that the space can eventually execute in small, defined steps[8]. SPRIE is designed to automate recovery from a failure in the execution of any of these plans. Possible sources of failure include a single hardware failure, multiple hardware failures, a software agent failure, or some other type of malfunction. SPRIE uses a two-part process to recover from errors. The first portion of the system is the diagnosis module, which isolates and determines the source of the failure. The second portion of the system is the recovery module that devises a plan to achieve the goal based on the assessment provided by the diagnosis system.

The diagnosis and recovery systems work closely together to generate a recovery plan. Under normal conditions, a plan is executed by an executor module. This module is responsible for walking through a plan and executing each of the steps by contacting the appropriate agents to accomplish each low-level task. The executor is also the first module notified of a failure in the plan execution. When an error is detected, the executor notifies the recovery system. The recovery system in turn requests a diagnosis for the failure from the diagnosis subsystem.

The diagnosis system uses Bayesian inferencing to probabilistically determine the source of failure in the plan, based on the error that was detected and the current worldview. The worldview is encapsulated in the world model, which is used to keep
track of how likely all of the various resources in the Intelligent Room are to be in a
good, functional state. Each resource in the world model has an entry that includes
its name, probability of being functional, and probability of being non-functional. In
Figure 2-1, the form of a resource entry in the world model is shown.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>P(Resource Functional)</th>
<th>P(Resource Non-Functional)</th>
</tr>
</thead>
</table>

Figure 2-1: The form of each resource entry in the world model.

The diagnosis system is probabilistic instead of deterministic, because there are
often many combinations of failures that can generate the same observed symptoms,
and more than one of these possibilities needs to be considered when formulating a
recovery plan. After the diagnosis system has determined the most likely current state
of the world, these results are used to update the world model. Since the world model
is updated after each error is detected, no matter what initial probabilities are used to
populate the world model’s database, over time the probabilities will reflect the true
reliability of the resources or the real state of the Intelligent Room. For example,
over time any resources that are non-functional will be assigned a “probability of
functioning” closer and closer to zero, while resources that are working correctly will
be assigned a “probability of functioning” closer and closer to one.

After the update is made to the world model by the diagnosis system, the recovery
system begins its work. At this point, the world model reflects an accurate representa-
tion of the resources in the Intelligent Room based on symptoms observed. The
recovery system can use this information to query the plan library for possible alter-
naie plans that can bring the system from the current state to its original goal state.
For example, if the diagnosis system ascertains that a particular failure is caused
by a bad projector, then the system might attempt to select and implement a plan
that utilizes another projector in the Intelligent Room. The plan library contains a
collection of plans and the Bayesian networks that correspond to these plans. The
Bayesian networks that are associated with a given plan represent the relationships
between the resources that are being used by the plan. For example, knowing that a
projector is working correctly provides information about the reliability of the power supply for the projector and the computer that the projector is connected to. These relationships are modeled by edges in the Bayesian network.

When queried for an alternate plan, the plan library searches for plans that have a start state that is met by the current state of the world and an end state that matches the goal state. For example, if a user is attempting to play a song on a speaker, the plan that the Intelligent Room uses might consist of the following:

1. initial state: speaker off
2. Turn on speaker
3. Route audio output from computer A to speaker
4. Play song on computer A
5. goal state: song playing on speaker

If the plan failed at the second step (after the speaker had been turned on), the recovery system might look for a plan like this:

1. initial state: speaker on
2. Route audio output from computer B to speaker
3. Play song on computer B
4. goal state: song playing on speaker

All possible alternate plans are returned to the recovery system, where the one with the highest expected return is selected for execution. Expected return is calculated as follows:

\[
E[return] = P(\text{plan succeeds}) \times (\text{reward for success}) - P(\text{plan fails}) \times (\text{cost of failure})
\]
For example, if a plan has a 70% chance of success with a reward of X, and a 30% chance of failure with a cost of failure of 3X then the expected return is \( .7X - .3(3X) = -.2X \). Once a new plan is chosen, the recovery system returns that plan to the executor, which is then responsible for discarding the old plan and executing the new one. This two-step recovery process is shown in Figure 2-2.

![Figure 2-2: The communications and interactions between the systems in SPRIE](image)

The following sections discuss these individual subsystems in greater detail. The specifics of how various tasks are accomplished and how information is communicated between the different subsystems are all covered in detail in subsequent sections. In addition to an in-depth explanation of methods used by the subsystems the theory behind these methods is also discussed to provide a clear picture of how these subsystems work together to perform the tasks that SPRIE is responsible for.
Chapter 3

Implementation

SPRIE's two-stage recovery system for plan execution failures involves a number of systems and modules that work together to automate recovery in the Intelligent Room. These individual systems and modules each perform essential functions in the recovery process. The world model is responsible for tracking the reliability of all of the various resources in the Intelligent Room. The diagnosis system is used to determine and isolate the cause of failure, as well as to update the world model with this new information. The recovery system then requests a new plan from the plan library, which holds all of the plans that can be executed in the Intelligent Room. The recovery system is also given the added responsibility of selecting a plan from the library that has the highest expected return. This chapter describes the design and implementation of each of these systems, and how these different systems interact with each other to automate recovery in the Intelligent Room.

3.1 World Model

The world model is used to keep track of how likely all of the resources in the Intelligent Room are to be in a “good state”. In this case, good state is defined as functioning as expected. Likewise, a “bad state” is one in which the resource is not functioning properly. In reality, the state of a resource in the Intelligent Room cannot be represented as just “functional” or “non-functional.” For example, a resource
might only be temporarily non-functional, because it is "in use by another user" or "powering up" or might be operating at reduced capability, such as a computer under heavy load. However, for scoping purposes this thesis will model resources as either functional or non-functional.

The model consists of a mapping of resources to the probabilities that they are in a good state and in a bad state. An example of a world model of a system that has the following resources: Computer A, Computer B, Projector A, Projector B, and Camera A, is shown in Figure 3-1.

<table>
<thead>
<tr>
<th>Resource Name</th>
<th>P(Resource Functional)</th>
<th>P(Resource Non-functional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer A</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>Computer B</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Projector A</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>Projector B</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Camera A</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 3-1: The format for the world model entries.

The diagnosis system is responsible for constantly updating these probabilities, so that over time the probabilities more and more closely reflect the true reliability of a resource. An update occurs each time the diagnosis system is called to make a diagnosis and consists of a listing of resource and their new probabilities of being functional given the diagnosis. Probabilities in the world model are replaced as newer probabilities for resources are calculated by failures that are diagnosed by the diagnosis system. Through these updates, the world model can record the new information that is gained for diagnosing each failure and stay current.

It is important for the world model to stay current, because it supplies resource reliability information both to the diagnosis system and to any resource managers that might be running in the space. The diagnosis system utilizes the world model for making diagnoses, and the resource manager consults this model before selecting resources for a plan to increase the likelihood of that plan’s success.
3.2 Diagnosis System

This section discusses the underlying architecture of the diagnosis system. The goal of this subsystem is to ascertain the current state of the various resources involved in a failed plan. Using Bayesian inference, this system determines the current state and the associated probability of that state. This information is used to select a new plan to recover from the failure.

When the diagnosis system receives a request for a diagnosis from the recovery system, it determines the current state of the world. This is done using Bayesian networks where the nodes in the network are the resources being used in the failed plan, and the edges are the relationships between these resources. This network is solved, and the combination that is found to have the highest probability based on the observed behavior of all of the resources is then taken as the state of the world. This diagnosis is used to update the world model which makes it possible for future diagnoses to take into account all previous failures. Over a long period of time, this world model will be able to closely reflect the actual reliability of the resources available in the Intelligent Room.

When a plan execution error occurs, the cause is usually a resource that, for some reason, is not in the state that the plan expects it to be in. For example, a resource might be in use by another user, powered off, non-functional etc.

In order to recover from such a failure, it is imperative that the error in assumption be corrected. Otherwise, an alternate plan will have the same problem as the original failed plan, when it attempts to use a resource that is faulty or in some other state that makes it impossible for it to accomplish the task set out for it. To correct this error, the diagnosis system uses Bayesian networks to probabilistically determine the current states of all of the resources involved in a failed plan.

Bayesian networks or belief networks consist of a set of variables or nodes and a set of directed edges that connect the nodes. These two sets combine to form a directed acyclic graph (DAG). Each node has a finite set of mutually exclusive states and holds a truth table that maps each state to the probability that it is the current state of
the node based on observations of the system. For example, a plan with resources A, B, and C will have a Bayesian network with three nodes. If there are two directed edges, one between A and C and the other between B and C, then A and B will both have truth tables with only two entries the first is P(A) and P(B), and the second is the P(¬A) and P(¬B). However, C’s truth table will have four entries. Node C’s entries will be as follows: P(C|A∩B), P(C|A∩¬B), P(C|¬A∩B), and P(C|¬A∩¬B).

The conditional probabilities for ¬C can be calculated by subtracting the conditional probability for C from one. For example, P(¬C|A∩¬B) is equal to 1-P(C|A∩¬B).

[6]

In the case of the diagnosis system, the nodes in the Bayesian network are resources and the directed edges are the reliability relationships between the resources being represented by the nodes on either side of an edge. For example, a plan with a power source, a computer, and a projector would have the Bayesian net shown in Figure 3-2. In this Bayesian network, there would be an edge between the power supply and the projector and another edge between the computer and the projector.

This method is useful for modeling the effect of resources on one another. In modeling the reliability of a resource, the reliability of the resources that interact with that resource need to be considered. As shown in the example in Figure 3-2, the probability that the projector is working correctly is affected by the probability that both the power supply and the computer are working correctly. The uncertainty in this model is similar to the uncertainty in error detection in the actual physical system. For example, if an error is detected in a projector trying to show a slide, it is possible for the error to lie with the computer, with the power supply, or with the projector itself.

During normal operation, the executor sends any observations it has to the diagnosis system, so that the system can record the observations in the Bayesian net. These observations are stored in the nodes and affect the probability of that node being in a good working state. When an error occurs, the Bayesian network is solved. After solving the Bayesian network, the diagnosis system uses the new probabilities to update the world model. This update makes it possible for future diagnoses to take
Figure 3-2: The reliability relationships between the power source, the computer, and the projector for a plan trying to show a slide with the projector.

in account previous failures. Once this update is accomplished, the recovery system once again takes control of the situation.

3.3 Recovery System

The recovery system is used both to regulate and control communication between the executor and the subsystems of SPRIE as well as to make the final decision as to which alternate plan should be returned to recover from the failure. This section details the communication process once the failure has been detected by the executor and the selection process for alternate plans. The calculations for determining which alternate plan has the highest expected return as well as the cost functions for calculating reward for success and cost of failure are discussed in detail.

Once the recovery system has received the diagnosis from the diagnosis system, it requests a new plan from the plan library. This new plan still has the same final
goal that the failed plan had, but due to the new world state, a new plan needs to be selected. Since the plan library may contain many plans that begin at the current world state and end with the desired goal state, the recovery system must select one that has the highest expected return.

During normal execution of a plan, the recovery system is in standby mode. It has no responsibilities. However, when the executor detects an error, it notifies the recovery system, which then becomes active. The recovery system is responsible for managing the process of determining an alternate plan. To accomplish this task, the recovery system must first ascertain which resource is malfunctioning and what the current state of the world is as a result. To isolate the cause of failure, the recovery system contacts the diagnosis system. The diagnosis system uses Bayesian networks as described in Section 3.2 to determine the current state of the world and update the world model.

Once this update has been completed, the recovery system regains control of the situation from the diagnosis system. The recovery system then queries the plan library for an alternate plan that can bring the world state from the current state to the desired goal state. The plan library is a collection of plans and their corresponding Bayesian networks. The new plan has the same final goal state as the original plan, but the new world state provides new initial conditions that need to be incorporated into the plan. Since it is possible to have multiple plans that can accomplish a given goal, the plan library returns a set of possible alternate plans that can all accomplish the goal. It is the responsibility of the recovery system to select one recovery plan from this initial set of possible plans.

To select one plan from this collection of possible plans, the recovery system calculates the expected return for each of the possible plans. The expected return can be calculated using the equation shown below:

\[
E[return] = P(plan\_succeeds)*(\text{reward\_for\_success}) - P(plan\_fails)*(\text{cost\_of\_failure})
\]

The expected return is affected by the reward of success, the cost of failure, the
probability of success, and the probability of failure.

The probability of a plan succeeding is calculated using the probability that all of the plan’s resources are in the expected state and the probability of each step succeeding. This calculation is done dynamically using the current world model. Similarly, the probability of a plan failing is equal to one minus the probability that the plan succeeds. These calculations to find both the probabilities of a plan succeeding and a plan failing can be done using the equations shown below:

\[ P(\text{plan succeeds}) = \prod_{i=0}^{n} P(\text{resource}_i) \times P(\text{step}_i \text{ succeeds}) \]

\[ P(\text{plan fails}) = 1 - P(\text{plan succeeds}) \]

The reward for success describes the quantitative benefit of having the plan succeed. Parallely, the cost of failure is the quantitative loss when the plan fails. At first glance, the cost of failure may appear to be inversely proportional to the reward of success. However, the cost of failure is not related to the reward of success and needs to be considered separately.

The reason that these two quantities are not related is that they are both affected by different factors. For example, reward of success is affected by how efficiently a plan is executed, whereas the cost of failure is not necessarily affected. In SPRIE, the cost of failure for a plan is calculated by summing the cost of all of the steps in a plan. This calculation can be done using the following equation:

\[ Cost \text{ of Failure} = \sum_{i=1}^{n} cost \text{ of step}_i \]

The reward for success is not a calculated value. Since the reward for success is a value that is associated with the plan as a whole, it is not necessarily related to the completion of any particular step. Thus, in SPRIE the reward for success of plan is a value that is manually assigned to a plan.
3.4 Plan Library

The plan library holds a collection of plans that can be searched when a plan is requested for a specific goal. When a request for a plan is made, the whole library is searched for all plans that are a match for the starting state and the goal state. There are three ways to search for plans. The first way is by goal state. The second is by starting state. The final way is by both starting state and goal state. All plans that match the specified criteria are returned to the requesting agent. The library can be consulted for both new plans and recovery plans.
Chapter 4

Discussion

This chapter discusses some of the design decisions that went into SPRIE. The discussion includes several places where SPRIE was simplified for scoping purposes or where an inefficient algorithm was used, and briefly details how these places might be improved. The chapter also discusses alternate strategies that might be used and what the various trade-offs to each approach are.

4.1 World Model

The world model is used to keep track of all of the resources in the Intelligent Room and the probabilities of how likely they are to be functioning or not. Although, the current design and implementation effectively handles two states, functional and not functional, this model is idealized. Resources actually have multiple states, because there are more aspects of a resource that can be taken into account. The more accurately resources in the space are modeled, the more effectively they can be used. For example, a resource might be functional but operating under reduced capacity or under compromised security. In both of these cases, there may be tasks which would regard such a state as functional, while other tasks might regard such a state as non-functional. Differentiating between the two could result in a resource being used despite some limitations.

If additional states are recorded and tracked by the world model, the diagnosis
system must also employ reasoning that takes these states into account. The task of considering more states when diagnosing a failure is discussed in detail in the next section.

4.2 Diagnosis System

The diagnosis system uses Bayesian networks to calculate the likely state of the world. Bayesian nets offer several advantages over similar systems such as partially observable Markov decision processes (POMDPs) or neural nets. Both POMDPs and neural nets require training datasets in order to learn\cite{3, 5}. Although Bayesian nets require a predetermined set of relationships and truth tables, they can be run with no training data. In addition, the scope of this thesis prevented the inclusion of an extremely complex algorithm in the diagnosis system; the brute force algorithms available to solve Bayesian nets are easier to implement and less computationally expensive than similar brute force algorithms to solve POMDPs or neural networks.

An alternate approach might be to use some type of rule based system or other expert system to diagnose likely failures. However, this would require including a comprehensive set of rules and assertions which would limit the ability of the system to handle error conditions for which it was not especially designed. Due to the various limitations of the alternate diagnosis strategies examined here, Bayesian nets were selected for SPRIE.

4.3 Recovery System

The recovery system is centered around the idea of evaluating plans based on their expected return. This requires calculating three values: the "reward for successful plan completion", the "cost of failure" and the probability of plan success. The probability of plan success can be derived from the probability that all of the plan's resources are in the expected state and the probability of each step in the plan succeeding. The probability of plan success can be dynamically calculated based on the current world
model and the plan being used.

However, the "reward for plan completion" and "cost of failure" are less well defined. In this thesis, the user can assign a "cost" to various steps in a plan; the total cost of the plan is just the sum of all of the costs of these steps. Thus, for example, a plan that involves many steps is generally more expensive than a simpler plan. The user is also expected to predetermine the reward for successful plan completion. While this is a simple solution to the problem, it's not very clear how the user is expected to assign rewards or costs to actions or how a consistent policy can be enforced across all plans or even what the "units" for these numbers are. Applying different costs or rewards to the same action in different plans will lead to some plans being incorrectly chosen more frequently, but there is no mechanism in place to prevent this.

4.4 Plan Library

The plan library consists of a collection of plans that can be searched when a plan with a specified goal state is requested. The library associates each plan with a Bayesian network that represents the relationships between the resources that are used by that plan. Although, new plans can be added as new functions are needed. This static collection is inflexible to the changing environment of the Intelligent Room.

Since plans are static entities that need to be executed from beginning to end, their usefulness is limited. For example, if the current world state is the same as an intermediate state in a plan, the library system would not return the plan as a match, because it cannot be partially executed. Without partial execution, additional plans for tasks, which can be accomplished by partial execution of plans, need to be added individually to the library.

In addition to being inflexible, the library system also requires a large number of plans that might have many identical steps. For example, two plans might be identical except for the fact that one plan has one additional step in the beginning to toggle a resource that might have a different initial state. This small variation would require a separate plan even though it is virtually the same as another plan already
in the library. For this reason, the plan library would need to be large and repetitive in order to provide plans that can cover a variety of situations.

To improve the flexibility and reduce the repetitive nature of the plan library a more dynamic system could be utilized for plan generation. A dynamic system that is able to build a plan from smaller steps could be a valuable improvement to SPRIE.
Chapter 5

Related Works

There have been many different approaches to error recovery in multi-agent systems. Most of these efforts have revolved around collections of robots or other independent, autonomous systems, but the principals involved are similar to those used by SPRIE in the Intelligent Room. For example, model based diagnosis, as opposed a simple hardcoded detection routine, is an approach that has been applied to various other systems [9, 7, 1].

Simmons et. al.[11] use a model based approach to accurately determine the state of the world. As in SPRIE, their model is slowly built up over time by changing parameters to match the observed behavior of the system. Instead of SPRIE’s Bayesian network system, Simmons’ system uses a partially observable Markov decision process (POMDP) as the diagnosis engine. Using a POMDP requires iteratively increasing the accuracy of the model; multiple errors (steps where the expected result did not occur) may be required to correctly diagnose the problem.

Hamilton et. al.[4] use an approach very similar to SPRIE. They combine observations from various embedded sensors with a predefined model of the system to determine likely sources of error. Unlike SPRIE, Hamilton’s RECOVERY system makes use of the entire history of the system to determine faults; it logs the past behavior of all components and attempts to use that information to determine the error. For example, RECOVERY will notice a drift in component temperature over time and base its diagnosis on this information. Because it deals with a much larger
volume of information, RECOVERY uses a set of a priori knowledge to guide its diagnosis rather than an adaptive, but computationally more expensive system like SPRIE's Bayesian nets.

Bayana et. al. [1] combine a model based diagnosis system with a set of randomized algorithms to diagnose potential problems. This system is much less computationally expensive than a deterministic engine such as a Bayesian net and does not rely on an a priori plan library, but it requires multiple unexpected results to iteratively narrow its diagnosis and, due to its random nature, it may miss certain key elements in the diagnosis.

Lopes [9] takes a longer view towards model building. Rather than simply building a model of the system based on current observations and then accessing a predetermined plan library, Lopes' system builds a library of recovery plans over time based on the success of recovery attempts in similar previous situations; new plans are constructed based on an abstracted model built up from previous attempted plans. Lopes' system does not deal with the actual error diagnosis.

Although, there are many different approaches to diagnosis and recovery in multi-agent environments, these approaches are not desirable for intelligent spaces. Some of the approaches require large bodies of information about the system or that can be collected about the system, that are not available in the Intelligent Room at this time. Other approaches contain random elements, which are not desirable. The closest system to SPRIE is the model based approach created by Simmons et al. that utilizes POMDP's to perform diagnosis on the system.
Chapter 6

Conclusions

SPRIE is designed to provide an automated recovery system for plan execution failures in the Intelligent Room. The motivation behind this system is to increase the usability of the highly complex Intelligent Room for users without knowledge about the underlying architecture in the room. SPRIE fulfills the following needs:

- Diagnosis of sources of plan execution failures in real time.
- Tracking of reliability of resources over time.
- Recovery from plan execution failures in real time.
- Performing all of these tasks automatically with no user input.

SPRIE accomplishes all of these tasks by using the two-stage system described in the Implementation chapter. The Bayesian network diagnosis system is capable of isolating plan execution failures in real time. The diagnosis system's world model updating procedure also allows it to accomplish the second task, which is to track the reliability of resources in the space over time. Since the world model is shared by all plans, the information about the reliability of the resources in the space is constantly being updated. Over time the world model will grow to accurately reflect the reliability of the resources in the room.

The recovery stage of SPRIE is responsible for finding a plan that can take the space from the current state to the desired goal state. Utilizing the plan library, which
is a repository for all of the plans that can be executed by the space, the recovery system can quickly get an alternate plan that will allow the room to move to the goal state. So, the recovery system accomplishes the third goal of being able to recover from the failure in real time.

Finally, since all of the steps of the system occur without needing the user to supply any information, the fourth goal is also met. SPRIE recovers from plan execution failures completely transparently to the user. No understanding of the underlying architecture or additional information needs to be provided by the user in order to find and execute an alternate plan.
Chapter 7

Further Work

The current design and implementation of SPRIE fulfills the goals of providing the Intelligent Room with a way to track reliability of resources, and to diagnosis and recover from plan execution failures automatically without user input. However, there are some areas that can be improved and optimized. All of the systems can be improved so that SPRIE can run more efficiently and effectively.

This thesis models resources as being in only two possible states: “functional” and “non-functional.” The probabilities assigned to the two states model the reliability of the resource. In reality, resources might be in any one of a number of other states such as “temporarily unavailable”, “working at reduced capacity” or “about to fail” (such as from low battery power). Modeling resources in states such as these would allow SPRIE to more accurately assess the state of the world and thus the plan (and resources) that should be used for the error recovery.

There are two main improvements that can be made to increase the performance of the plan library. The first improvement is to expand the plan library. SPRIE currently searches the library for alternate plans each time an error occurs. Expanding this library would mean that the library would include a greater variety of plans, which can then be used to accomplish additional goals. Expanding the library will also mean that there will be more alternate options in the event that a failure renders more than one plan useless for solving the problem.

The second improvement to the plan library is to create a dynamic plan generator
that would be able to dynamically create a plan to accomplish a goal by combining a number of smaller plans, each of which would complete a portion of the goal. For example, one could construct a plan to “setup a home theater” out of a sub-plan to select and show a movie as well as a few discrete actions such as dimming the lights and starting the popcorn maker.

This type of plan generation will make storing plans more efficient, because smaller plans could be stored and used by the generator to piece together a larger plan on the fly to accomplish the goal. This flexibility will also make it possible to utilize a resource that might be in an unexpected initial state simply by adding a few steps to the beginning of the plan to toggle the state of the resource. This change would allow for more flexibility in plans and more efficient storage of plans.

For the diagnosis system, there are two major changes that would be beneficial to both the efficiency and the usability of the system. The first change is to optimize the algorithm to solve the Bayesian networks. The diagnosis system currently uses a brute force approach to solve Bayesian networks. This approach runs in $O(2^n)$ time, which is clearly unsuitable for non-trivial networks. Solving Bayesian networks efficiently has been the focus of much research[2], and implementing a faster solution would be an excellent way to speed up the diagnosis system.

The second change is to develop a Bayesian network generator that can automatically generate a Bayesian network from a plan. The current system relies on a human to do the translation of a plan into a Bayesian network. This method of translation is both tedious and slow. Creating a mechanism that can construct a Bayesian network to match a plan dynamically would save time and would also allow for greater flexibility. This improvement is also important because the plan generator, which dynamically pieces together plans as discussed earlier in this section requires a generator to create the Bayesian networks that corresponds to the new plan on the fly.

In the recovery system, a possible improvement would be to develop a more accurate function for the reward for success and cost of failure. The reward for completion of a plan is currently arbitrarily assigned when the plan is created. The cost of failure
is the summation of the cost of failure at each step in the plan. The cost of failure for each step is also assigned arbitrarily when the plan is created. Assigning these values without a convention makes it difficult to compare plans and select the one with the highest return. An improvement to this system would be to develop either guidelines for making these assignments or finding an accurate function that can be used to calculate these values.

In each of the different systems in SPRIE, there are improvements and optimizations to be made. These new augmentations increase the flexibility and performance of the system as a whole. Together, these changes can make the interaction between the Intelligent Room and the human users in it more transparent.
Bibliography


