Ensuring Robust Agent Control in Uncertain Environments [*]

Anita Raja Thomas Wagner Victor Lesser

UMass Computer Science Technical Report 1999-27

April 6, 1999

Abstract

Agent control involves reasoning about local problem solving activities, interacting with other agents, planning for a course of action and contingencies in the event of failure of the action and finally carrying out the actions with limited resources and uncertainty about agent outcomes and the actions of other agents. The growing complexity and dynamics of agents and the environments in which they interact requires *robust* agent control, where the chance of complete failure of an agent's plan to achieve the high-level goal is minimized. Design-to-Criteria is a soft real-time agent control process where schedules are built to meet dynamic client goal criteria (including real-time deadlines), using a task model that describes alternate ways to achieve tasks and subtasks. In this paper we describe a post-scheduling contingency analysis process that can be employed in deadline critical situations where the added computational cost is worth the expense. We describe the uncertainty representation and how it improves task models and the scheduling process, and provide empirical examples to show how robustness is built into agent control.

[*]Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number F30602-97-1-0249 and by the National Science Foundation under Grant number IIS-9812755 and number IRI-9523419. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. 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 DARPA, NSF, USAF, or the U.S. Government.

1 Introduction

Agent control involves reasoning about local problem solving activities, interacting with other agents, planning a course of action and perhaps contingencies and carrying out the actions when there limited resources and uncertainty about agent outcomes and the actions of other agents. All these activities have to be done dynamically in real-time and within real resource and cost constraints. The growing complexity and dynamics of agents and the environments they interact in requires *robust* agent control, where the chance of complete failure of an agent's plan to achieve the high-level goal is minimized.

Design-to-Criteria (DTC) scheduling[13] is the soft real-time process of finding an execution path through a hierarchical task network such that the resultant schedule meets certain design criteria, such as real-time deadlines, cost limits, and quality preferences. It is the heart of agent control in agent-based systems such as the resource-Bounded Information Gathering agent BIG [10] and the multi-agent Intelligent Home [9] agent environment. Casting the language into an action-selecting-sequencing problem, the process is to select a subset of primitive actions from a set of candidate actions, and sequence them, so that the end result is an end-to-end schedule of an agent's activities that meets situation specific design criteria.

The general Design-to-Criteria scheduling process is designed to cope with exponential combinatorics and to produce results in soft real-time. However, its somewhat myopic approximation and localization methodologies do not consider the existence of recovery options or their value to the client. In the general case, explicit contingency analysis is not required. In the event of a failure, the scheduler is reinvoked and it plans a new course of action based on the current context (taking into consideration the successes as well as the failures, considering the value of results that been produced to the particular point). In hard deadline situations, however, the scheduler may not be able to recover and employ an alternative solution path because valuable time has been spent traversing a solution path that cannot lead to a final solution. Our uncertainty based contingency analysis tools can help in this situation by pre-evaluating the likelihood of recovery from a particular path and factoring that into the utility associated with a particular schedule. The improved estimates (based on the possibility of recovery options) can result in the selection of a different schedule, possibly one that leads to higher quality results with greater frequency. We return to contingency analysis in Section 3.

While Design-to-Criteria is a research focus in its own right, it is often incorporated into other research projects or used as an analysis expert by other tools. For example, in multi-agent systems research, Design-to-Criteria is coupled with the GPGP [5] coordination module enabling an agent to coordinate its activities with the activities of other agents. GPGP operates by exchanging the agents' *local views*, detecting task interactions, then forming commitments over the interactions to handle temporal sequencing of activities. GPGP *modulates* Design-to-Criteria through hypothetical/proposed commitments and firm commitments that have been given or received. When used in a single agent system, such as the BIG information gathering agent, the local problem solver simply enumerates its problem solving options in the TÆMS [4] language and passes them to the scheduler for analysis along with a set of design criteria that describes the type of schedules that the problem solver would prefer. For example, if there is little time to gather information and produce a result, the information gathering problem solver may specify a desired deadline and the idea that trading-off quality in favor of shorter duration is preferred.

This paper is structured as follows. Section 2 discusses how uncertainty is integrated and leveraged in the main Design-to-Criteria scheduling process. In Section 3 we step outside of the main scheduling process and discuss secondary contingency analysis methodology that uses Design-to-Criteria to explore uncertainty and the ramifications of schedule failure. Experimental results illustrating the strength of contingency analysis, relative to Design-to-Criteria's myopic view, for certain classes of task structures are provided in Section 4.

2 Integrating Uncertainty Into Design-to-Criteria

The Design-to-Criteria scheduling problem is framed in terms of a TÆMS task network, which imposes structure on the primitive actions and defines how they are related. The most notable features of TÆMS are its domain independence, the explicit modeling of alternative ways to perform tasks, the explicit and quantified modeling of interactions between tasks, and the characterization of primitive actions in terms of quality, cost, and duration. To ground further discussion consider the TÆMS task structure shown in Figure 1. The task structure is a conceptual, simplified sub-graph of a task structure emitted by the BIG information gathering agent; it describes a portion of the information gathering process. The top-level task is to construct product models of retail PC systems. It has two subtasks, *Get-Basic* and *Gather-Reviews*, both of which are decomposed into primitive actions, called *methods*, that are described in terms of their expected quality, cost, and duration. The *enables* arc between *Get-Basic* and *Gather* is a non-local-effect (nle) or task interaction; it models the fact that the review gathering methods need the names of products in order to gather reviews for them. *Get-Basic* has two methods, joined under the *sum()* quality-accumulation-function (*qaf*), which defines how performing the subtasks relate to performing the parent task. In this case, either method or both may be employed to achieve *Get-Basic*. The same is true for *Gather-Reviews*. The qaf for *Build-PC-Product-Objects* is a *seq_sum()* which indicates that the two subtasks must be performed, in order, and that their resultant qualities are summed to determine the quality of the parent task; thus there are nine alternative ways to achieve the top-level goal in this particular sub-structure.

Primitive actions are characterized statistically via discrete probability distributions rather than expected quality values. The quality distributions model the probability of obtaining different quality results and the possibility of failure (indicated by a zero quality result).

The schedules shown in Figure 2 illustrate the value of uncertainty in this model from a scheduling perspective. Schedule A$'$ is constructed for a client who needs a high quality solution, requires the solution in seven minutes or less, and who is willing to pay for it. Note that the quality distribution for Schedule A$'$ includes a 20% chance of failure. Schedule O (Figure 2) is the optimal schedule for the given criteria. Even though the *PC-Connection* method has a higher expected value, the *PC-Mall* method has a lower probability of failure. Since a failure in one of these methods precludes the execution of *Query-Consumers-Reports* (via the task interaction), the issue of failure is not local to the methods but instead impacts the schedule

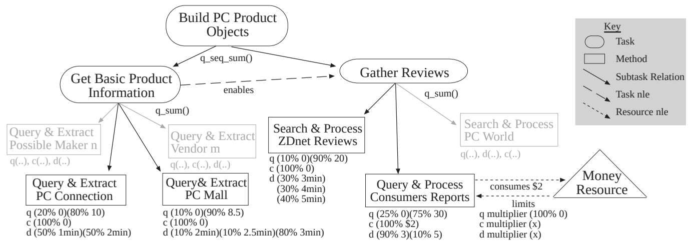
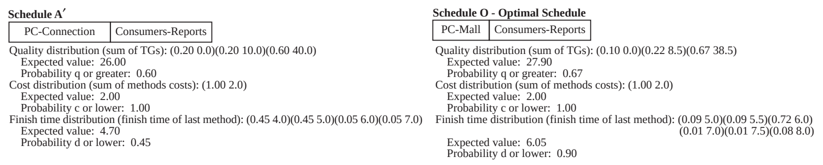


Figure 1: Simplified Subset of an Information Gathering Task Structure



```
Schedule A′
| PC-Connection | Consumers-Reports |
Quality distribution (sum of TGs): (0.20 0.0)(0.20 10.0)(0.60 40.0)
   Expected value:  26.00
   Probability q or greater:  0.60
Cost distribution (sum of methods costs): (1.00 2.0)
   Expected value:  2.00
   Probability c or lower:  1.00
Finish time distribution (finish time of last method): (0.45 4.0)(0.45 5.0)(0.05 6.0)(0.05 7.0)
   Expected value:  4.70
   Probability d or lower:  0.45
```

```
Schedule O - Optimal Schedule
| PC-Mall | Consumers-Reports |
Quality distribution (sum of TGs): (0.10 0.0)(0.22 8.5)(0.67 38.5)
   Expected value:  27.90
   Probability q or greater:  0.67
Cost distribution (sum of methods costs): (1.00 2.0)
   Expected value:  2.00
   Probability c or lower:  1.00
Finish time distribution (finish time of last method): (0.09 5.0)(0.09 5.5)(0.72 6.0)
                                                        (0.01 7.0)(0.01 7.5)(0.08 8.0)
   Expected value:  6.05
   Probability d or lower:  0.90
```

Figure 2: Uncertainty Representation in Schedules

as a whole. Thus, when uncertainty is modeled and propagated during the scheduling process, Schedule O is the optimal schedule as it has the highest net expected quality value and it still meets the client's deadline constraint.

In general, the different implications of uncertainty to the scheduling process manifest themselves in two primary ways. One is with respect to the general scheduling process. By integrating and leveraging uncertainty within the framework of coping with combinatorics and generating custom schedules, we can produce better schedules in situations where certainty is important; this is documented in [13, 14]. The other use of uncertainty in our work is to step outside of the soft real-time schedule generation context and to focus instead on detailed analysis that considers schedule recovery options and revises schedule expectations to reflect this more detailed analysis.

3 Uncertainty-based Contingency Analysis

In the previous sections we explored uncertainty as it is integrated into the standard Design-to-Criteria scheduling methodology. However, in situations where hard deadlines exist, a mid-schedule failure may preclude recovery via rescheduling because sufficient time does not remain to explore a different solution path. In these situations, a stronger analysis that considers the existence of possible recovery options may lead to a better choice of schedules. To address such situations, we have developed a contingency analysis methodology that functions as an optional back-end on the Design-to-Criteria scheduler.

In this section we discuss contingency scheduling issues and formalize five different measures of schedule *robustness*, where robustness describes the quantity of recovery options available for a given schedule. In Section 4 we then present experiments comparing the use of the contingency algorithms to the standard Design-to-Criteria scheduling approach.

This work in contingency analysis of schedules is closely related to recent work in conditional planning. However, the planning-centric research

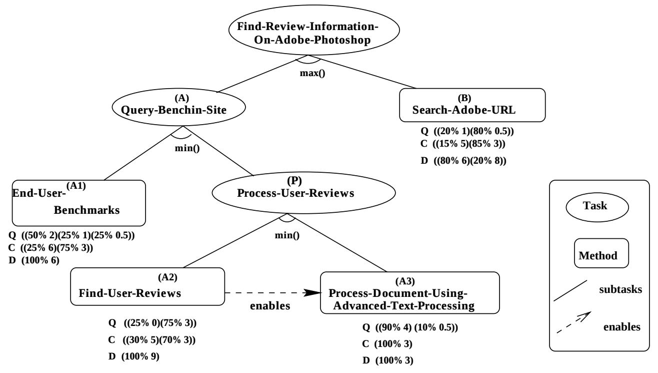


Figure 3: Gather review information on Adobe Photoshop.

focuses on solving problems which involve uncertainty by probabilistic reasoning about actions and information on the value of planning for alternative contingencies [6, 8] and using utility models [7]. Other approaches use partial Markov decision processes and decision theoretic planning approaches [1, 3] which prune the search space using domain-specific heuristic knowledge. [11] describes a partial-order planner called *Mahinur* that supports conditional planning with contingency selection. The authors concentrate on two aspects of the problem, namely, planning methods for an iterative conditional planner and a method for computing the negative impact of possible sources of failure.

[2] discusses an algorithm for a specific domain namely a real telescope scheduling problem where the stochastic actions are managed by a splitting technique. Here the Just-In-Case scheduler pro-actively manages duration uncertainty by using the contingent schedules constructed by analyzing the problem using off-line computations.

To better illustrate the power of contingency analysis, consider a simple example. Figure 3 shows a task structure for gathering information on Adobe Photoshop. The top-level task can be achieved by either completing task *Query-Benchin-Site (A)* successfully or executing the method *Search-Adobe-URL (B)*, or both. If both A and B are executed the maximum quality of these two is the quality propagated to the parent node (per the $max()$ qaf). The quality, cost and duration distributions for the executable methods denote expectations about method performance. For instance, the quality distribution of method *End-User-Benchmarks* indicates that it achieves quality value of 2 with probability 0.5, quality of 1 with probability 0.25 and 0.5 with probability of 0.25. Lets assume the client design criteria specifies that the task should achieve the maximum possible quality within a hard deadline of 18 minutes. The Design-to-Criteria scheduler first enumerates a subset of the alternatives that could achieve the high level task. A subset of these alternatives are selected and schedules are created using the one-pass method-ordering techniques. The set of candidate schedules are then ranked using the multi-dimensional evaluation mechanism [12] which compares the schedules' statistical attributes to the client design criteria.

We will use the term *expected lower bound* (ELB) to denote a slightly modified schedule utility rating returned by the standard Design-to-Criteria scheduler. In the ELB computation, the standard utility value associated with the schedule is computed without any relative scaling components. This enables comparison between the ELB for a schedule belonging to one set, e.g., S_1, and a schedule belonging to a different set, S_2. For the purposes of illustration simplicity, we will discuss the ELB in this document as being directly related to the expected quality of a given schedule, i.e., in this document, the ELB *is* the expected quality of a given schedule assuming no rescheduling. In terms of the design criteria, this is equivalent to a client specifying a preference for maximizing quality within a given deadline – no weight or value are given to any of the other criteria dimensions. The algorithms presented in the following sections operate on more interesting criteria settings, but, the analysis is more easily understood if the metrics are cast in terms of expected qualities rather than a multi-dimensional objective / utility function.

For the example in Figure 3, the two possible schedules are {A1,A2,A3} and {B}. Figure 4 describes the computation of the ELB for the schedule {A1,A2,A3}. Consider the first entry in the table. It describes the case when method A1 achieves a quality of 2, which occurs with a probability of 0.5 as described in the TÆMS task structure. Method A2 achieves a quality of 0 with probability 0.25. [1] The probability of the methods achieving these qualities in a single execution is 0.125, given in column 4. The expected quality of the schedule {A1,A2,A3} is 0 in this case, described in column 5. The duration and cost distributions and their expected values are computed in a similar fashion. The ELBs for schedules {A1,A2,A3} and {B} are as follows:

[1] Failure of A2 (where quality= 0) results in zero quality for the schedule due to the way in which the task structure is defined, i.e., under $min()$ qafs, failure results in zero quality for the parent task as well. Hence the quality of A3 is a not a determining factor and is represented by nil.

A1	A2	A3	Frequency	Quality
50% 2	25% 0	nil	5%*25%=12.5%	0.0
50% 2	75% 3	90% 4	33.75%	2.0
50% 2	75% 3	10% 0.5	3.75%	0.5
25% 1	25% 0	nil	6.25%	0.0
25% 1	75% 3	90% 4	16.875%	1.0
25% 1	75% 3	10% 0.5	1.875%	0.5
25% 0.5	25% 0	nil	6.25%	0.0
25% 0.5	75% 3	90% 4	16.875%	0.5
25% 0.5	75% 3	10% 0.5	1.875%	0.5

Figure 4: Each row represents a possible permutation of the quality distributions of methods A1, A2, A3 in schedule {A1,A2,A3}. The first three columns represent the possible expected quality values achieved by each of the methods A1, A2, A3. The fourth column shows the probability of the particular quality distribution combination occurring and the last column shows the final expected quality of the schedule.

1. {A1,A2,A3}: ELB: 0.97 (Expected Quality)
   Quality : (25% 0.0) (24% 0.5) (17% 1.0) (34% 2.0)
   Duration : (100% 18)
2. {B}: ELB: 0.6
   Quality : (20% 1) (80% 0.5)
   Duration: (80% 6) (20% 8)

Since {A1,A2,A3} has the highest ELB (indeed, the highest rating using the standard normalized utility functions), it is chosen and executed. Suppose A1 executes successfully, but A2 fails (i.e. it results in 0 quality), which it does 25% of the time. Then A3 cannot be executed because it is not enabled (A2 failed) but there is no time left to reschedule and attempt method {B} because there is not sufficient time to execute method B before the deadline.

Because of the one-pass low-order polynomial method sequencing approach used by the scheduler to control scheduling combinatorics, the standard Design-to-Criteria scheduler will only produce one permutation of the methods A1, A2, and A3. However, if the scheduler did produce multiple permutations, the schedules {A1,A2,A3} and {A2,A1,A3} would receive the same expected lower bound value. Hence the contention is that there is no difference in performance between the two. However with more detailed evaluation of the schedules, it is clear that {A2,A1,A3} allows for recovery and contingency scheduling which schedule {A1,A2,A3} does not permit for the given deadline. If {A2,A1,A3} is the schedule being executed and A2 fails, there is time to schedule method {B} and complete task TG1. This clearly implies that schedule {A2,A1,A3} should have a better expected performance rating than {A1,A2,A3} as the schedule {A2,A1,A3} includes the recovery option from failure in its structure.

3.1 Performance Measures

In this section we formalize a general theory relating to the contingency planning concepts discussed in the previous section. The question we strive to answer formally here is the following: *What performance measure is the most appropriate estimator of the actual execution behavior of a schedule given the possibility of failure?* Our basic approach is to analyze the uncertainty in the set of candidate schedules to understand whether a better schedule can be selected or an existing schedule can be slightly modified such that its statistical performance profile would be better than that normally chosen by the Design-to-Criteria scheduler. We accomplish this analysis through the use of several performance measures. Prior to presenting the measures, a few basic definitions are needed:

1. A schedule s is defined as a sequence of methods $(m_1, m_{2,}..m_{n-1}, m_n)$.
2. Each method has multiple possible outcomes, denoted m_{ij}, where j denotes the j'th outcome of method m_i. This is part of the TÆMS definition of methods or primitive actions. Though the examples generally present methods as having quality, cost, and duration distributions, methods actually may have sets of these distributions where each set is one possible outcome. For example, if method m may produce two classes of results, one class that is useful by method m_1, and one class that is useful by method m_2, method m will have two different possible outcomes, each of which is modeled via its own quality, cost, and duration distributions. Additionally, these different outcomes will have different nles leading from them to the client methods, m_1 and m_2 respectively.
3. Each outcome is characterized in terms of quality, cost, and duration, via a discrete probability distribution for each of these dimensions and each outcome has some probability of occurrence.
4. m_{ij}^{cr} is a *CTER* when the execution of m_i results in outcome j which has a value or set of values characterized by a high likelihood that the schedule as a whole will not meet its performance objectives. For instance, m_{ij} is a CTER if the probability of the quality of m_{ij} being zero is non-zero.

5. A schedule s could have zero, one or more CTER's in it. A general representation of such schedule with at least one CTER would be $s^{cr} = (m_1, m_{2,}..m_{ij}^{cr}..m_{kl}^{cr}...m_{no}^{cr}..m_{n-1}, m_n)$.
6. f_{ij}^{cr} is the frequency of occurrence of m_i's , j'th outcome where m_{ij} is a CTER.
7. $\overline{m_i^{cr}}$ is m_{ij}^{cr} with its current distribution being redistributed and normalized after the removal of its critical outcome. In other words, the criticality of m_{ij}^{cr} is removed and the new distribution is called $\overline{m_i^{cr}}$.
8. If $s^{cr} = (m_1.., m_{i-1}, m_{ij}^{cr}, m_{i+1}, .m_{kl}^{cr}..m_{no}^{cr}..m_{n-1}, m_n.)$, then
   $\overline{s_i^{cr}} = (m_1.., m_{i-1}, \overline{m_i^{cr}}, m_{i+1}, ..m_{kl}^{cr}..m_{no}^{cr}..m_{n-1}, m_n.)$
   $\overline{s_k^{cr}} = (m_1.., m_{i-1}, \overline{m_i^{cr}}, m_{i+1}, .\overline{m_k^{cr}}..m_{no}^{cr}..m_{n-1}, m_n.)$ and
   $\overline{s^{cr}} = (m_1.., m_{i-1}, \overline{m_i^{cr}}, m_{i+1}, .\overline{m_k^{cr}}..\overline{m_n^{cr}}..m_{n-1}, m_n.)$

The five statistical measures that aide in detailed schedule evaluation are:

Expected Lower Bound (ELB) The expected lower bound rating, of a schedule s, is the performance measure of a schedule execution without taking rescheduling into consideration [13]. It is an expected rating because it is computed on a statistical basis taking quality, cost and duration distributions into account, but ignoring the possibility of rescheduling. As mentioned previously, in this paper, to simplify presentation of the algorithms we will concentrate on the case in which the ELB is only the expected quality of a given schedule.

Approximate Expected Upper Bound (AEUB) The AEUB is the statistical schedule rating after eliminating all regions where rescheduling could occur. The assumption is that there are no failure regions and hence the schedule will proceed without any failures and hence no rescheduling will be necessary. The following is a formal definition of AEUB:

Suppose m_{ij}^{cr} is a CTER in the schedule $s = (m_1..m_n)$ and it occurs with frequency f_{ij}^{cr}. Let $\overline{s_i^{cr}}=(m_1, m_2..\overline{m_i^{cr}}..m_n)$. If $\frac{ELB(\overline{s_i^{cr}})-ELB(s)}{ELB(s)} \geq \alpha$, then m_{ij} is a *CTER*, where α is a percentage value that determines when a region should be classified a *CTER* and thus a candidate for more detailed analysis. The value of α is contextually dependent and should be specified by a scheduler client. For instance, if saving on computational expense is more important to the client than high certainty, α should be high, and thus the threshold for *CTER* classification is also high. However, if certainty is paramount, then α should be low, indicating that any significant change in the ELB should be explored. When this computation is done on an entire schedule for all of its *CTER's*, we call it the Approximate Expected Upper Bound. Generalizing this formula for k *CTER's* $m_{i_1 j_1}...m_{i_k j_k}$, $AEUB(s) = ELB(m_1...m_{i_1-1}, \overline{m_{i_1}^{cr}}..\overline{m_{i_2}^{cr}}.......\overline{m_{i_k}^{cr}}...m_n)$. The AEUB is thus the best rating of a schedule on an expected value basis without any rescheduling.

Optimal Expected Bound (OEB) The OEB is the schedule rating if rescheduling were to take place after each method execution. So the first method is executed, a new scheduling subproblem which includes the effects of the method completion is constructed and the scheduler is re-invoked. The first method in this new schedule is executed and the steps described above are repeated. Hence the optimal[2] schedule is chosen at each rescheduling region. For complex task structures, the calculation would require a tremendous amount of computational power and it is unrealistic to use it for measuring schedule performance in a real system. In most situations, $ELB(s) \leq OEB(s) \leq AEUB(s)$, since the $OEB(s)$ is based on recovery from a failure while $AEUB(s)$ assumes no failure.

Expected Bound (EB) Let m_{ij}^{e} be the set of actual quality, cost, duration values when method m_{ij} is executed. After each method execution the schedule is re-rated. If for some schedule $s = (m_1, m_2..m_i..m_n)$,and $ELB((m_1...m_n)) \gg ELB((m_{1j}^{e}, m_{2k}^{e}...m_{il}^{e}, m_{i+1}..m_n))$, i.e. the actual execution performance of a schedule is below expectation, then a new schedule is constructed based on the partially complete schedule $\{m_{1j}^{e}, m_{2k}^{e}, ...m_{il}^{e}\}$.

So the EB is the schedule rating when rescheduling occurs only when there is a possibility for the partial execution of the current schedule will fail to meet expected criteria as a result of the outcomes of methods already executed. This computation, like the OEB, will require extensive computational power. Again in most situations, $ELB(s) \leq EB(s) \leq OEB(s) \leq AEUB(s)$.

Approximate Expected Bound (AEB) It is the schedule rating with rescheduling only at *CTER's* and using expected lower bound of the new stable schedule for methods following the *CTER*. This is limited contingency analysis at *CTER's*. Consider a schedule s of n methods s=$(m_1, m_2..m_i..m_n)$. Now suppose m_{ij} is a *CTER* with a frequency of occurrence of f_{ij}. In order to compute the AEB of the schedule, we replace the portion of the schedule succeeding m_{ij}^{cr}, which is $m_{i+1}, m_{i+2}, ...m_n$ by $l_{i+1}, l_{i+2}......l_k$ if there exists a $l_{i+1}, l_{i+2}......l_k$ such that $ELB(m_1...m_{ij}^{cr}, l_{i+1}...l_k) \geq ELB(m_1...\overline{m_i^{cr}}, m_{i+1}...m_n)$. The Approximate Expected Bound for this instance is computed as follows: $AEB_{ij}(m_1, ...m_n)$=$ELB(m_1...\overline{m_i^{cr}}, m_{i+1}..m_n)*(1-f_{ij}) + ELB(m_1..m_{ij}^{cr}, l_{i+1}..l_k)* f_{ij}$. The new schedule rating thus includes the rating from the original part of the schedule as well the ELB of the new portion of the schedule. This is basically the calculation described when the AEB was introduced in a previous section.

Now we describe the general case scenario. Let $m_1, m_2, m_3, ...m_i...m_n$ be a schedule s of n methods with k *CTER's* named $m_{i_1 j_1}^{cr}, m_{i_2 j_2}^{cr}...m_{i_k j_k}^{cr}$. Let

[2] "Optimal" in this case is meant in a satisficing fashion. In the context of Design-to-Criteria, the "best" schedule for a given task structure is not guaranteed to be optimal as the combinatorics prevent an exhaustive search. As it is used here, optimal means the best possible schedule within the space searched by Design-to-Criteria.

the recovery path available at each *CTER* m^{cr}_{ij} be s^{r}_{ij} and each m^{cr}_{ij} occurs with frequency f^{cr}_{ij}. The AEB of the entire schedule is described recursively as $AEB = ELB(m_1...m^{cr}_{ij}, l_1, ...l_k)*f^{cr}_{ij} + AEB(m_1...\overline{m^{cr}_i}, m_{i+1}, ...m_n)*(1-f^{cr}_{ij})$ which can be expanded out as follows:

$$AEB{=}f^{cr}_{i_1j_1} * ELB(m_1...m_{i_1-1}, m^{cr}_{i_1j_1}, l_{a1}...l_{b1})$$
$$+(1 - f^{cr}_{i_1j_1}) * f^{cr}_{i_2j_2} * ELB(m_1...\overline{m^{cr}_{i_1}}...m^{cr}_{i_2j_2}, l_{a2}...l_{b2})$$
$$+ ...(1-f^{cr}_{i_1j_1})*...*(1-f^{cr}_{i_{k-1}j_{k-1}})*f^{cr}_{i_kj_k}*ELB(m_1...\overline{m^{cr}_{i_1j_1}}...\overline{m^{cr}_{i_2}}...\overline{m^{cr}_{i_3}}...m^{cr}_{i_kj_k}, l_{ak}...l_{bk})+$$
$$(1 - f^{cr}i_1j_1) * (1 - f^{cr}_{i_2j_2}) * ... * (1 - f^{cr}_{i_kj_k}) * \underbrace{ELB(m_1...\overline{m^{cr}_{i_1}}...\overline{m^{cr}_{i_2}}...\overline{m^{cr}_{i_k}}...m_n)}_{AEUB}$$

The above computation produces an approximate measure since we use the $ELB(m_1..m_{ij}, l_{i+1}..l_k)$. A better and more exact computation would be to use the
$AEB(m_1..m_{ij}, l_{i+1}..l_k)$. So if we recursively refine the $ELB(m_1..m_{ij}, l_{i+1}, ..l_k)$, the schedule rating approaches the expected bound (EB). Thus, the deeper the recursion in the analysis of *CTER's*, the better the schedule performance measure and the closer it is to the actual performance measure when rescheduling occurs. This describes the potential anytime nature of the AEB computation. Thus, in most situations, $EB(s) \geq AEB(s)$ and the $AEB(s) \geq ELB(s)$ by definition.

Here we would like to add that all computations above are based on heuristics and hence are approximations including the OEB and EB. We could define AEUB',OEB',EB', AEB' and ELB' which would involve complete analysis of all paths by the scheduler. The resulting schedules would display higher performance characteristics and meet goal criteria better but will also be computationally infeasible to generate [13].

4 Experimental Results

Using the measures described above, effective contingency planning is a complex process. It involves taking into account a number of factors, including task relationships, deadlines, the availability of alternatives, and client design criteria (i.e., quality, cost, duration, and certainty trade-offs). In this section, we evaluate the performance of the contingency analysis tools by comparing them to the standard Design-to-Criteria scheduler. Comparison is done by examining the ELB (standard scheduler metric) and the AEB (contingency analysis metric) and comparing schedules selected on the basis of these metrics to the actual results obtained by executing the schedules in a simulation environment.

The experiments in this section were conducted by randomly generating task structures while varying certain characteristics. Intuitions of which characteristics would lead to structures that are amenable to contingency analysis were used to seed the search for interesting test cases. Since method failure is a crucial factor for the contingency analysis argument, the generation of task structures was designed to concentrate on the variance of two factors, namely, the effects of failure location and failure intensity (probability of failure) within a task structure. Ten randomly generated task structure classes were then modified to varying degrees with respect to these two factors. The design criteria in these experiments is to maximize quality given a hard deadline on the overall schedule. This simple design criteria setting is one that lends itself to contingency analysis as the existence of a hard deadline (in contrast to a soft preference, e.g., soft deadline) may preclude recovery via rescheduling in certain circumstances.

The results for the experiments are shown in Figure 5. For each task structure instance, 100 simulated executions were performed using the schedule with the highest ELB and with the schedule having the highest AEB. Each row in the table indicates a different *(failure location, failure probability)* parameter setting for the ten task structures; each row is also an aggregation of results for the ten task structure instances. Of the two factors used to differentiate the task structures in each row, failure location (Lo) (found in the first column of the table) refers to the position of critical method(s) in a task structure and hence in the schedule. Failure intensity (In) (second column) refers to the probability of a method failing. Three different classifications of failure location are used in the experiments: early(E), medium(M), and late(La). Similarly, three different settings for failure intensity are used in the experiments, namely, low(L), medium(M) and high(H) where low is 1%-10% probability of failure, medium is 11%-40%, and high is 41%-90%.

For each problem instance, the execution results produced by the AEB selected schedule were compared to the results for the ELB selected schedule via statistical significance testing. The third column, *N.H. valid count*, identifies the number of problem instances for which the null hypothesis of equivalence could not be rejected at the .05 level via a one-tailed t-test. In other words, *N.H. valid count* identifies the number of experiments for which the results produced via AEB are not statistically significantly different from the results produced by the ELB. These experiments are omitted from subsequent performance measures.

The fourth column indicates the number of task structures of the ten possible whose data is compared. These are task structures that led to schedules for the ELB case and the AEB case that produced execution results that are statistically significantly different, i.e., the null hypothesis of equivalence was rejected at the .05 level. The remaining columns compare the AEB and ELB selected schedules' execution results for the these task structures from an aggregate perspective.

Columns five and eight, titled *Contingency A.Q* and *Normal A.Q.* respectively, show the mean, normalized quality that was produced by the AEB and ELB selected schedules respectively. In other words, the best schedule per the AEB metric was selected and executed in an unbiased simulation environment, when failure occurred the scheduler and contingency-analysis tools were reinvoked and a new schedule generated that attempted to complete the task. The resultant quality was measured and recorded and the experiment repeated 100 times. The same procedure was done for the ELB selected schedule, though when rescheduling occurred, the contingency analysis tools were not invoked (nor were they invoked in the production of the initial schedule). The overall maximum quality produced by either the AEB or the ELB simulation runs was recorded and all resultant quality then normalized over the maximum, resulting a quality value that expresses the percentage of the maximum observed quality that a given trial produced. This procedure was then repeated for the other task structure that produced statistically significantly different results, and the normalized quality values averaged. Thus, the 0.73512 A.Q. from the first row of Table 5, column four, indicates that contingency analysis yielded schedules that produced approximately 74% of the maximum observed quality on average. Column seven indicates that the standard Design-to-Criteria scheduler produced approximately 63% of the maximum observed quality, on average, for the same set of task structures. Thus, contingency analysis yielded a 14.24% percentage increase in resultant quality over the standard Design-to-Criteria scheduler, as shown in column 11.

Columns six and nine show the number of times a given selected schedule failed to produce any result, that is, recovery before the deadline was not possible, for the AEB and ELB cases respectively. It is interesting to note that the contingency selected schedule failed to produce a result with somewhat greater frequency for rows one and five. This is because both the contingency selected schedule and its recovery option had some probability of failure, though, we do not actually consider the failure rate in these cases to be statistically significant. The failure rate in row three illustrates the classic case in which recovery before the deadline is often not possible for the schedules chosen by the standard Design-to-Criteria scheduler, whereas it is more often possible for the schedules selected by contingency analysis.

Columns seven and ten show the number of times rescheduling was necessary during execution. These results are somewhat counter intuitive as the contingency analysis selected schedules generally resulted in more rescheduling during execution due to failure. This is because the contingency analysis tools explore the possibility of recovery and do not seek to avoid the failure in the first place. Relatedly, because the contingency analysis considers the existence of recovery options, it may actually select a schedule more prone to initial failure than the standard Design-to-Criteria scheduler because the schedule has a higher potential quality. For example, say two schedules s_1 and s_2 have the following respective quality distributions: $q_1 = (25\%\ 0)(75\%\ 10)$ and $q_2 = (50\%\ 0)(50\%\ 14)$. The expected value of s_1 is 7.5 whereas the expected value of s_2 is 7. The standard scheduler will prefer s_1 over s_2 because it has a higher expected quality value (assuming that the goal is to maximize quality within a given deadline). However, the contingency analysis tools might actually prefer s_2 over s_1 if there are recovery options, e.g., s_3 for s_2, because s_2 has the potential for a higher quality result than s_1. If s_3 has a quality distribution like $q_3 = (100\%\ 7)$, then the s_2 / s_3 recovery scenario has a higher joint expected quality than does s_1 alone. Associating a cost with rescheduling in the contingency algorithms could modulate this opportunistic risk-taking type of behavior. If a cost were associated with rescheduling, the utility of a recovery option could be weighted to reflect such a cost.

The last column shows the mean normalized OEB of the AEB selected schedule. This is the measure where rescheduling is invoked after every method execution irrespective of the execution outcome. It describes the optimal performance of a schedule since the best possible path is selected every step of the way. The quality value shown is the average of 100 executions of the OEB schedule, normalized by the maximum observed quality over all the AEB selected and ELB selected schedules' executions. The OEB is higher than both *Contingency A.Q.* as well as *Normal A.Q.* for each class of task structures. This is as it should be, as the OEB is a computationally intensive performance measure which strives to obtain the optimal schedule at every point of the plan.

Irrespective of rescheduling, in general, for the task structures that lead to statistically significantly different results, contingency analysis produced schedules that yielded higher average quality than did the standard Design-to-Criteria scheduler. However, as illustrated by the large number of task structures that lead to results that were not statistically significantly different, very few of the candidate task structures were suitable for contingency analysis (about 20%).

Fail		N.H valid	T.S.	Contingency			Normal			Perf.	OEB
Lo	In	count	count	A.Q.	F.R.	R.C.	A.Q.	F.R.	R.C	Impr.	
E	M	8	2	0.73512	0/200	72	0.63041	0/200	0	14.24%	0.75227
M	M	8	2	0.70125	2/200	64	0.63883	0/200	0	8.89%	0.71222
La	M	8	2	0.79936	21/200	100	0.66246	38/200	48	17.12%	0.84531
M	L	10	0	0	0	0	0	0	0	0%	0
M	M	8	2	0.70125	3/200	64	0.63883	0/200	0	8.89%	0.71222
M	H	10	0	0	0	0	0	0	0	0%	0

Figure 5: *Fail Lo* is the failure location; *Fail In* is failure intensity; *N.H. valid count* is number of task structures that fail to produce results for the contingency and standard scheduler cases that are statistically significantly different; *T.S. count* is number of task structures whose performance qualities will be compared; *Contingency A.Q.* is average, normalized quality of AEB selected schedule; *Contingency F.R.* is the failure rate is number of times AEB selected schedule fails to achieve any quality; *Contingency R.C.* is the reschedule count which is the number of times the AEB selected schedule reschedules due to failure of a method to achieve quality. *Normal A.Q.* is average, normalized quality of ELB selected schedule; *Normal F.R.* is the number of times ELB selected schedule fails to achieve any quality; *Normal R.C.* is the number of times the ELB selected schedule reschedules due to failure of a method to achieve quality. *Perf. Impr* is the average improvement in performance of contingency analysis over normal scheduling. *OEB* is the average, normalized quality of AEB selected schedule.

5 Conclusions and Future Work

Ensuring robust agent control requires dealing with uncertainty as a first class object both within the scheduling process and via the secondary contingency analysis is beneficial. The addition of uncertainty to the TÆMS modeling framework increases the accuracy of TÆMS models. Including explicit models of uncertainty improves the scheduling process not simply by increasing modeling power, but also by increasing the representational power of all the computations in the scheduling process.

The secondary contingency analysis procedures presented in Section 3 step outside of this context to perform a more detailed analysis of schedule performance based on the existence of recovery options. Since the algorithms explore the schedule recovery space using the Design-to-Criteria scheduler, they still exhibit a satisficing, approximate, resource conservative nature. It is interesting to note that even the coarse analysis performed in the AEB and AEUB computations is beneficial in certain circumstances. Future efforts in contingency analysis will involve explicitly bounding and controlling the complexity of the contingency analysis process. Intertwined with this research objective is the ability to classify particular problem solving instances.

Another area of future exploration in contingency analysis lies in the area of determining critical regions, *CTER*s, within schedules. One aspect of this is determining *CTER* status based on the existence and types of task interactions.

Another area to be explored involves leveraging the uncertainty-enhanced TÆMS models in multi-agent scheduling and coordination. In multi-agent systems the scheduler is typically coupled with a multi-agent coordination module that forms commitments to perform work with other agents; local concerns are thus modulated by non-local problem solving.

Other, more general, future efforts in Design-to-Criteria include using organizational knowledge to guide the scheduler decision process when operating in multi-agent environments and to support negotiation between the scheduler and its clients, which may be other AI problem solvers or humans.

References

[1] C. Boutilier, T. Dean, and S. Hanks. Planning under uncertainty: Structural assumptions and computational leverage. In *Proceedings of 3rd European Workshop on Planning (EWSP'95)*, 1995.

[2] J. Bresina, M. Drummond, and K. Swanson. Just-in-case scheduling. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[3] Thomas Dean, Leslie Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1-2):35–74, 1995.

[4] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD thesis, University of Massachusetts, 1995.

[5] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[6] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, pages 31–36, 1994.

[7] P. Haddaway and S. Hanks. Utility models for goal-directed decision-theoretic planners. *Computer Intelligence*, 14(3), 1998.

[8] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[9] Victor Lesser, Michael Atighetchi, Bryan Horling, Brett Benyo, Anita Raja, Regis Vincent, Thomas Wagner, Ping Xuan, and Shelley XQ. Zhang. A Multi-Agent System for Intelligent Environment Control. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*, 1999.

[10] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, July 1998. See also UMass CS Technical Reports 98-03 and 97-34.

[11] N. Onder and M. Pollack. Contingency selection in plan generation. In *Proceedings of the Fourth European Conference on Planning*, 1997.

[12] Thomas Wagner, Alan Garvey, and Victor Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 294–301, July 1997. Also available as UMASS CS TR-1997-10.

[13] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19:91–118, 1998. A version also available as UMASS CS TR-97-59.

[14] Thomas A. Wagner, Anita Raja, and Victor R. Lesser. Modeling Uncertainty and its Implications to Design-to-Criteria Scheduling. Under review, Special issue of the AI Journal, 1998. Also available as UMASS CS TR#1998-51.