A. Issues and Information for Probabilistic Risk Assessment

During the operation of a nuclear power plant, conditions exist that alter the risk of operating the facility. These events that result in a change, where "change" can be either an increase or decrease in risk, fall under three general categories. First, plant activities force certain components to be incapable of performing their desired functions during operation. Examples of these activities that incapacitate components include preventative maintenance and testing (both scheduled activities), corrective repairs to failed components (an unscheduled activity), and Technical Specification actions (these activities could be either scheduled or unscheduled). Second, improper plant design or maintenance could result in an unintended reduction in plant or component reliability, potentially over long periods of time. Examples of these reductions in plant reliability include faulty component design such as undersized valve motor operators and insufficient fire-barrier protection or faulty component restoration and testing after a maintenance activity. Third, initiating events that occur during operation cause challenges to plant systems and operators. Examples of these events include losses of off-site power, miscellaneous plant transients, and loss-of-coolant pipe breaks.

To address the risk of operation, the methodology of probabilistic risk assessment (PRA) has been accepted in the nuclear power generation industry. In our analysis, we focus on Level 1 PRA to quantify sequences leading to core damage. This involves the determining events that challenge plant operation (i.e., initiating events). Then, in response to the initiating events, the plant response must be identified. This identification is modeled through the use of fault trees and event trees. From the fault trees and event trees, the accident sequences are obtained.

A.1.1 Traditional PRA Metrics

Current PRAs have evolved beyond the performance of analyses to the application of the results to day-to-day operation of the facility. While the process of doing the PRA is important (in the sense of providing value), of increasing importance is the reliance on the numerical results of the PRA (Melnicoff, 1999; Holahan et. al, 1998; Cunningham et al., 1999). From the PRA, two

primary metrics rise to dominate the risk-management (in the nuclear power generation industry) scene:

1. Overall summary results such as core damage frequency (CDF) or core damage probability (CDP)

2. Traditional importance measures such as Fussell-Vesely (F-V) or risk achievement worth (RAW).

Regardless of the metric, the issue of context when obtaining and using PRA results is important. As we have demonstrated through the formal decision making tools, the context of the decision frames the evaluation and determination of decision outcomes. This point should be obvious because the PRA and the decision models are utilized/constructed with knowledge of the problem boundary conditions. Further, we have seen that the PRA provides a convenient mechanism of "feeding" probabilistic information into the decision models. This information transfer takes place primarily by using the CDP metric from the PRA in the decision model chance nodes.

The subject of traditional PRA metrics would be incomplete without a discussion of importance measures. While the topic of importance measures has been around almost since the onset of quantitative methods in risk and reliability assessment, the use of importance measures has become ubiquitous only within the last 15 years. Conversely, the use of CDF and CDP PRA metrics has been present since the inception of formal risk and reliability studies, mainly because these two metrics are natural numerical outcomes of the analysis.

Early work in the development of importance measures is best represented by the IMPORTANCE code developed by Lambert in the early 1970s. (Lambert, 1975) This analysis tool built upon the theoretical development completed by early pioneers such as Barlow, Birnbaum, Fussell, Proschan, and Vesely (Modarres, 1993). The IMPORTANCE software calculated results for eight different types of importance measures, ranging from F-V to

Birnbaum to Barlow-Proschan metrics. While these (and others) are still in use within the risk and reliability community, we will focus on the two most popular metrics, F-V and RAW.[1]

F-V represents the fraction of the total system failure probability containing an event of interest. Or, alternatively, F-V is "the probability that event *i* is contributing to system failure, given that system failure occurred" (Henley and Kumamoto, 1981). Historically, the value of F-V for the *i*'th basic event has been found by either (a) summing the minimal cut sets containing the *i*'th event and dividing by the total (of all the minimal cut sets) or (b) summing the minimal cut sets that do not contain the *i*'th event, dividing by the total, and then subtracting this quantity from unity. These two methods are shown below:

$$\text{F-V}i = Q_i / Q_{total} \qquad \text{(A-1)}$$

$$1 - (Q_j / Q_{total}) \qquad \text{(A-2)}$$

where Q_i is the summation of the minimal cut sets containing event *i*, Q_{total} is the summation of all of the minimal cut sets, and Q_j is the summation of the minimal cuts that do not contain event *i*. Note that, in practice, software tools do not sum minimal cut sets. Instead, a generally accepted technique is to either evaluate the set of minimal cut sets using the "minimal cut set upper-bound approximation" or to calculate the set probability exactly.

The RAW importance measure represents the increase in risk (or reliability) conditional upon event *i* having a failure probability of one. The increase in risk (or reliability) in this case is relative to the nominal risk (or reliability). Thus, the resulting RAW has traditionally been found by

$$\text{RAW}i = Q_{i=1} / Q_{total} \qquad \text{(A-3)}$$

---

[1] A third metric, the "risk reduction worth" importance measure, is used frequently in modern risk or reliability assessments. This measure provides the same rank ordering as found by the F-V measure. As such, it will not be discussed.

where $Q_{i=1}$ is the summation of all the minimal cut sets where the *i*'th event's probability is set to a value of one and Q_{total} is the summation of all of the minimal cut sets (at their nominal probability values).

These importance measures (F-V and RAW) have been presented, discussed, and computerized in numerous texts, articles, and software packages since their discovery. Since F-V and RAW are deceptively simple computationally, it would be natural to assume that their use and meaning in conjunction with "real" PRA models would also be simple. Unfortunately, this ease of use or understanding has not been realized. One would like to be able to use these simple summary metrics from the PRA to guide decision-making appropriately in contexts such as the determination of "important" plant components, where here the term "important" implies a significance to either plant safety or risk. As others have stated (possibly understated), "The identification and categorization of the safety-significant SSCs is not straightforward." (Cheok, Parry, and Sherry, 1998)

In general, there are many ways to quantify minimal cut sets. But, in PRA, it is standard to use one of three methods, which include

1. Rare event approximation. – This calculation approximates the probability of the union of minimal cut sets. The equation for the rare event approximation is:

$$P = \sum_{i=1}^{m} C_i \qquad \text{(A-4)}$$

   where P is the probability of interest, C_i is the probability of the i'th cut set, and m is the total number of cut sets.

2. Minimal cut set upper bound – This calculation approximates the probability of the union of minimal cut sets. The equation for the minimal cut set upper bound is

$$P = 1 - \prod_{i=1}^{m} (1 - C_i) \qquad \text{(A-5)}$$

where P is the probability of interest, C_i is the probability of the i'th cut set, and m is the total number of cut sets. Note (1) that the capital pi symbol implies multiplication and (2) most PRA tools utilize this equation as the default method of quantification.

3. Exact – There are various methods of determining the exact probability given a set of cut sets. One common approach is commonly referred by the name "inclusion-exclusion" while a second approach utilizes binary decision diagrams.

   The inclusion-exclusion rule works by obtaining various combinations of the minimal cut sets to form higher-order sets. Then, one finds the probability from the "first order" set (which are just the cut sets you started with). From this value, you subtract the probability from the "second order" set. From this value you add the probability from the "third order" set. From this value you subtract the probability from the "fourth order" set, etc., with as many passes as you have cut sets.

   Note, when there are many cut sets, the exact probability calculation performed in this fashion may be impossible to compute.

A.2 PRA Calculation Needs for the Decision Analysis Prototype

The incident management prototype will use decision models to provide the quantification method for ranking decision options. As part of this analysis, the PRA will need to provide input where applicable. Two primary methods of incorporating the PRA is through (1) the decision nodes by suggesting decision alternatives or important decision points and (2) the chance nodes by providing conditional probabilities related to plant operation.

The PRA normally provides a “core damage frequency.” This metric is the rate at which core damage events appear and is usually measured on a per year basis. The decision model,

however, deals in conditional probabilities, not frequencies. Consequently, one must translate the frequency to an applicable probability, which implies that one must know the time period over which the frequency is defined. The equation that will translate the rate into the desired probability is:

$$P(\text{event} \mid \text{conditions}) = 1 - \exp(-\lambda T) \cong \lambda T \qquad \text{(A-6)}$$

where λ is the conditional core damage frequency and T is the mission time (i.e., the time of interest). This equation assumes that the event of core damage is a Poisson process with constant rate. Thus, the probability that we require from the PRA is both "time conditional" (via the mission time parameter) and "configuration conditional" (via the conditions specified with calculating the core damage frequency).

Incidents involving an initiating event are complicated by the fact that the initiator actually happened. Of course, any components or systems that are inoperable at the time the initiator occurs will increase the overall risk of the event. The PRA would need to be adjusted by setting the specific initiator that occurred to a probability of one and the other initiators to zero. For this situation, the risk that is estimated by the PRA is like a "risk spike" for the event since the initiator actually happened (as opposed to its average frequency as nominally modeled). Figure A-1 illustrates the concept of a risk spike for an initiating event. In this figure, the horizontal axis tracks the time during the plant operation when the risk spike occurred. The "risk" or vertical axis measures the conditional core damage hazard rate, which is the product of the initiating event rate [$\lambda(t)$] and the conditional probability of core damage given the initiating event [$\phi(t)$]. This conditional core damage could be used to calculate a conditional core damage probability that would be fed into our decision model.

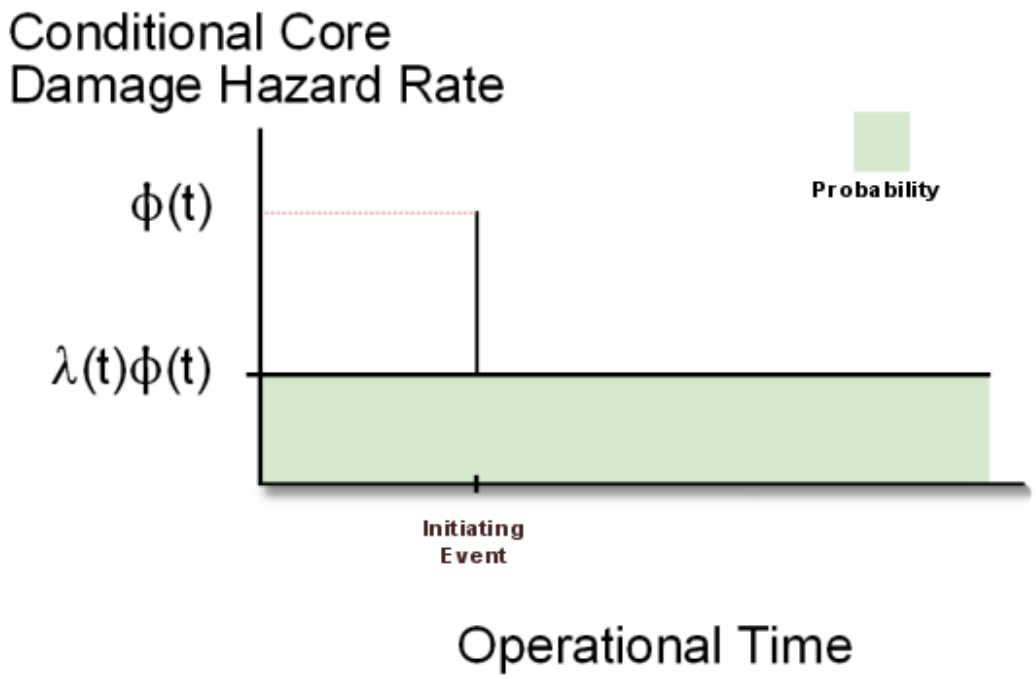

Figure A-1. A risk "spike" as measured by the PRA conditional upon an initiator occurring.

For incidents where an initiator does not occur (which are the majority of the events), the plant risk arises due to a component or system (or more than one) being inoperable for a certain length of time. **Figure A-2** illustrates the risk increase that can be estimated over the duration of time. Again, the horizontal axis is operational time and shows the start and stop of the particular condition being evaluated. The "risk" or vertical axis measures the conditional core damage hazard rate, which is the product of the initiating event rate [$\lambda(t)$] and the conditional probability of core damage given the initiating event [$\phi(t)$].

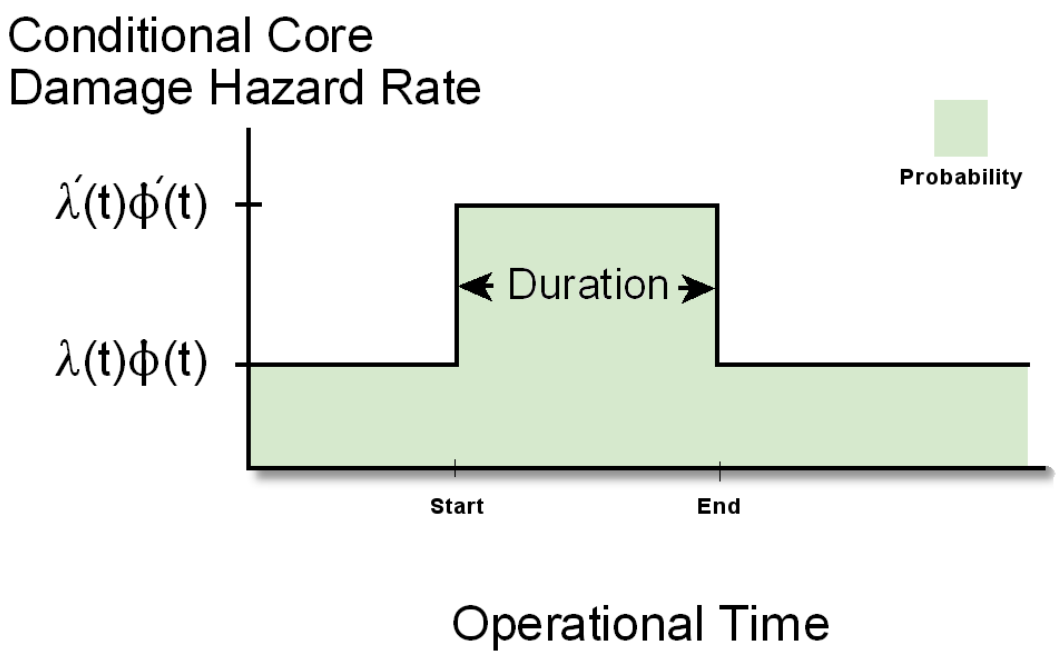

Figure A-2. Risk over a length of time as measured by the PRA conditional upon a failure.

If we integrate over the area of the curve shown in **Figure A-2**, then we would have a cumulative core damage probability (CDP). The cumulative CDP would give an estimate of the risk experienced over some length of time.

A.3 PRA Shortcomings that may Impact Decision Analysis

To calculate any PRA metric, one typically generates minimal cut sets representing CDF from the PRA. Generation of these cut sets is a relatively simple task (once the models are constructed) for modern software tools. However, once the CDF cut sets are used for the metric determination, the analysis becomes complicated. To address these complications, the focus in this section will be on the areas of:

1. The context surrounding PRA metric calculations.

2. The adjustments to the PRA model that are required when performing a conditional evaluation.

3. The limitations that are posed by the PRA model or software when performing non-nominal calculations.

4. The uncertainty that is inherent in the PRA metric calculations.

A.3.1 Context when Calculating PRA Metrics

At a high level, there are only two "contexts" that need to be discussed when evaluating the topic of the calculation and use of PRA metrics. First, there is the context related directly to the PRA model itself. This context consists of the assumptions or boundary conditions that are implicit in the PRA. Second, there is the context related to the desired use of the resultant PRA metrics. In other words, the context refers to (1) which PRA measures (if any) are applicable to a particular decision and (2) why they are applicable.

The first type of context related to the development of the PRA is important. The issues of PRA construction, standards for such construction, and the resulting PRA quality (possibly measured by the model uncertainty, another type of epistemic uncertainty) are all crucial. For example, many PRAs model standby components via a basic event with a demand failure probability rather than with a standby failure rate (and "test interval") and subsequent demand failure probability. This modeling approach simplifies the PRA but ultimately makes it more difficult to use when evaluating issues that may encroach on test intervals for standby components.

To illustrate this issue of standby components further, let us look at a specific case. For the situation where two standby components are in parallel, the cut set representing failure of the components would look like A * B, where A is the first standby component and B is the second standby component. For an individual component (say for A), its average unavailability is:

$$P(A) = (1/2)\ \lambda_A T \qquad \text{(A-7)}$$

where λ_A is the standby failure rate and T is the test interval. So, now the minimal cut set would be evaluated as the product of A and B, or

$$P(\text{system}) = (1/2\ \lambda_A T)(1/2\ \lambda_B T) = (1/4)\ \lambda_A \lambda_B T^2 \qquad \text{(A-8)}$$

The actual P(system) must be evaluated by integrating over the test interval. When one does this, it can be shown that the probability should be:

$$P(\text{system, exact}) = (1/3)\ \lambda_A \lambda_B T^2 \qquad \text{(A-9)}$$

Further, note that the probability is the unavailability over a single test interval. For incidents where the time period is portions of a test interval, the calculation becomes more difficult since the integration would need to be over non-integer intervals.

The second type of context, related to the proposed use of PRA metrics, is also of interest. Historically, the method of calculating PRA metrics receives much greater attention than the end use of the metrics themselves. For example, in the U.S. Code of Federal Regulations governing nuclear power plant operation related to implementation of the Maintenance Rule, the prescription guidance is focused mainly on measures such as F-V and RAW. Use of these measures may be questioned, including

1. Why are the F-V and RAW importance measures adequate to capture “risk significant” components?

2. Could there be situations where the prescribed thresholds of F-V > 0.005 and RAW > 2 be inadequate? Why are these thresholds adequate to begin with?

3. What determination should be made if a component’s RAW uncertainty distribution is partly above and partly below a value of 2?

4. Is the calculation for RAW too draconian since it changes a component's nominal failure probability to a value of 1.0?

Regulators typically focus on critical incidents after establishing a baseline for public health and safety. For example, in the U.S., both the EPA and the NRC have penalty systems in place that utilize an escalating fine (i.e., an economic loss) as a disincentive for jeopardizing safety. Situations involving willful misconduct can result in large penalties. Conversely, negligible (but positive) impacts to public safety are accepted as a part of operation. This "degree effect" implies then that regulatory agencies may not worry about small changes in items such as component failure rates or probabilities.

A part of the PRA context issue is the matter of determining which events to model and which events to leave out. As part of the incident advisor prototype, one of the first steps in the process is to determine whether or not the component of interest is in the PRA. If the component is not directly in the PRA, one must decide whether a surrogate component is in the PRA. In this section, we discuss this process of querying the PRA for the component or surrogate component.

Traditionally, components in a PRA are represented by one or more basic events. For example, a diesel generator may be modeled by fails-to-run, fails-to-start, and maintenance events. Further, these basic events follow a well-defined nomenclature for the event name, thereby facilitating their use. Consequently, it should be relatively simple to determine if a component is directly modeled in the PRA. Toward this end, the prototype will provide mechanisms to search and subdivide lists of basic events.

The issue of finding surrogate components (if applicable) is of greater interest though. A nuclear power plant has, as a rough estimate, over 50,000 components. But, modern PRA typically model between 1,000-4,000 basic events. Consequently, the number of components directly modeled in the PRA is substantially less than that in the plant itself. So, a question arises as to "where did these components go?" As we will demonstrate via an example, these "missing" components are either (1) subsumed by other components or (2) are omitted from the PRA model. An example of a component that is subsumed in the PRA includes all of the components

making up the diesel generator. A diesel generator consists of hundreds of parts, yet the PRA only includes a handful of basic events representing failure of this “component.” An example of a component omitted from the PRA includes many of the passive (e.g., piping, wiring) components prevalent at the plant. Many times these components are omitted because they (nominally) play a small part in the overall plant risk or are risk-dominated by other, related components.

A.3.2 Adjustments Required When Using a PRA

PRA metrics of CDP, F-V, and RAW all rely on the existence of knowing the CDF (e.g., see Equations A-1 and A-2). It is important to note that these base-line metrics are predicated upon "nominal" plant conditions. In other words, the cut sets that come out of our PRA assume that components are not failed, but instead have some average failure probability over the period of interest. Unfortunately, for decision analysis, the analyst may have to face the situation where a non-nominal condition is encountered. Examples of these non-nominal conditions include the assumption of a failed basic event for the RAW calculation and a CDP calculation conditional upon an initiating event occurring. These non-nominal conditions complicate the metric calculation since the PRA will have to be adjusted or modified specifically to account for the new contextual information. The main complications that arise when using a PRA are:

1. Determining the appropriate nonrecovery probabilities to be applied as a boundary condition to the PRA model.

2. Adjusting common-cause failure probabilities specific to the plant configuration represented by the boundary condition.

3. Adjusting the default plant line-up configuration to represent the actual mode of component/system operation.

Nuclear power plant PRAs include basic events representing failure of an operator or other plant personnel to repair or recover an inoperable component. These basic events are called “recovery

actions" and are quantified by determining a probability for the operator to *fail* to recover an inoperable component. In the PRA model for the nominal case, these recovery actions are intended to cover the general situation of a component failing at any time during the mission time defined for the analysis. Consequently, the nonrecovery probability is an "average" type of value and is not specific to a particular component failure.

Common cause failure is an important part of most PRAs. Modern PRAs contain basic events representing common-cause failure for a set (i.e., two, three, or four) of redundant components. For example, if three motor driven pumps are in parallel, and failure of all three pumps is required to fail the entire pumping system, then a common-cause basic event would appear in the fault tree model and would represent failure of all three pumps due to some common failure mechanism. Further, this common-cause basic event may be quantified by use of one of the typical parametric common-cause models such as the multiple Greek letter model or alpha-factor methods. Assuming that this event were quantified with the multiple Greek letter model, the basic event probability would be quantified using an equation like:

$$P(\text{common-cause} \mid \text{three components}) = Q_t \, \beta\gamma \qquad \text{(A-10)}$$

where Q_t is the total failure probability of an individual component, β is the conditional probability that the cause of the first component failure will be shared by at least one other redundant component, and γ is the conditional probability that the cause of the first component failure will be shared by at least two other redundant components.

The quantification of the base-line common-cause failure probability is applicable *only* for the nominal case. During the calculation of a risk metric, additional information is known about the configuration of the plant that could invalidate the nominal common-cause probability. For example, in the RAW calculation, the RAW calculation for one of the three redundant components assumes that the component has failed. Traditionally for importance measures like RAW, the common-cause event related to failure of the *i*'th basic event was not adjusted at the same time the *i*'th event is set to a probability of one. If only one of the three components has

failed, it is not correct to have a contribution to CDF representing failure of all three components. After failure of one redundant component, only two more components are needed to fail.

One must then consider the size of the possible error if the calculation is not exact for metrics such as RAW. It has been discussed that the error in CDF could be significant. (Smith, 1998) The magnitude of the potential error will depend on the structure of the PRA model and the probabilities of the common-cause parameters.

On the surface, a nuclear power plant PRA model consists of a set of event trees (representing accident sequences), associated fault trees (representing system failures), and basic events (representing component failures). Because a nuclear power plant is a complicated system for the purpose of power generation, it follows that models capturing the risk of such a facility will also be correspondingly complicated. The result of this complication in modeling is that while the high-level parts of the PRA (e.g., fault and event trees) model appear straightforward, the manner in which the pieces of the PRA interact is quite complex. Consequently, the assumptions and context embedded in the PRA sometimes become quite complex, especially for non-trivial situation like a loss of offsite power. While modern PRA software tools have helped the analyst manage the context of a PRA, the fact is that having effective tools to manage the pieces of a PRA has resulted in even more complex, intertwined models. Many PRA models have progressed to the point where a specialized "PRA programming language" is used to take preliminary results through a processing stage before finalizing the results. In addition, one should not forget that the results of these complex models are used to manage risk through decision making.

Historically, researchers in the fields of risk and reliability analysis have found and discussed pitfalls that abound when performing PRA types of calculations. The issues that have been revealed are, for the most part, still relevant today when using PRA models. While it is not possible to cover all potential areas of PRA analysis that complicate metric calculations, several additional issues deserve mention.

1. Because only a finite number of CDF minimal cut sets can be found (and stored), PRA analysis requires the use of a frequency truncation level. This truncation level, where minimal cut sets below the level are discarded, affects all PRA metrics.

2. When evaluating a failure condition in a PRA, one or more relevant basic events are set to a failed condition. Because the PRA is a Boolean equation, one could evaluate the failure condition using either logic (i.e., TRUE) or probability (i.e., probability equal to one). Unfortunately, the use of the logic approach versus the probability approach could yield a difference in results.

3. The PRA is constructed assuming that components are in a "nominal" state. Using the PRA to represent non-nominal conditions raises the possibility that assumptions and context used to construct the logic models in the PRA will not be valid for a particular case. Examples of this limitation include assumptions to safety-system train alignment, standby status of components, and accessibility of components.

4. The correct calculation for the F-V measure relies on the ability to determine probabilities for sets of interest. For example, if one were interested in the F-V for event *X* from the Surry-1150 PRA, one must determine – from the set of minimal cut sets – which sets (in probability space) in which the event *X* reside. Note that the traditional calculation for F-V shown in Equation A-1 does not look at sets of events. Instead, this simplistic calculation approximates the exact calculation.

The calculation for the F-V measure, as shown in Equation A-1, is hindered because it focuses on minimal cut sets rather than sets of events. When the intersection (i.e., the “overlap”) of sets becomes large (larger than 0.1), the simplistic calculation embodied by Equation A-1 simply does not work. Instead, one must return to the sets in probability space – where a set represents a minimal cut set – to evaluate the percent contribution of a set (or sets) to the overall area of all sets representing failure. Unfortunately, the correct calculation is difficult, and, consequently, modern PRA software tools do not perform the calculation. We will demonstrate this calculation by way of a simple example.

Assume that the PRA result is given by two sets, **A** and **B**. These two sets represent two minimal cut sets, where set **A** contains basic event ***X*** (a failure event) and basic event ***Y*** (also a failure event). Thus, $\mathbf{A} \in \boldsymbol{X} \cap \boldsymbol{Y}$. Then, for the second set, it consists of events ***X*** and ***Z***, or $\mathbf{B} \in \boldsymbol{X} \cap \boldsymbol{Z}$. For the overall PRA results, failure occurs when either set **A** occurs or set **B** occurs. The resulting minimal cut sets from the PRA software would then look like

$$\text{PRA} = \boldsymbol{X}\ \boldsymbol{Y} + \boldsymbol{X}\ \boldsymbol{Z} \tag{A-11}$$

We can quantify the PRA probability by using any standard quantification method including the rare event approximation, the minimal cut set upper-bound approximation, or the exact probability quantification. If we choose to perform the quantification exactly, multiple methods are available to perform the quantification. For example, we could use the "inclusion-exclusion" technique, binary decision diagrams, or the structure function for the minimal cut sets. Let us utilize the third approach (structure function):

$$\begin{aligned} S(\text{PRA}) &= 1 - (1 - \boldsymbol{X}\,\boldsymbol{Y})(1 - \boldsymbol{X}\,\boldsymbol{Z}) \\ &= 1 - (1 - \boldsymbol{X}\,\boldsymbol{Z} - \boldsymbol{X}\,\boldsymbol{Y} + \boldsymbol{X}\,\boldsymbol{Y}\,\boldsymbol{X}\,\boldsymbol{Z}) \\ &= \boldsymbol{X}\,\boldsymbol{Z} + \boldsymbol{X}\,\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{Y}\,\boldsymbol{Z} \end{aligned} \tag{2-9}$$

where S indicates the structure function of the minimal cut sets. To find the probability of the PRA results once we have the structure function, one only needs to insert the event probabilities into the equation and solve. Let us assume that the probabilities used in the PRA are:

$$\begin{aligned} P(\boldsymbol{X}) &= 0.2 \\ P(\boldsymbol{Y}) &= 0.5 \\ P(\boldsymbol{Z}) &= 0.7 \end{aligned} \tag{A-12}$$

We can use the probabilities above to find:

$$P(\text{PRA}) = 0.17 \tag{A-13}$$

Now, we can determine the F-V measures for events X, Y, and Z using the traditional calculation shown in Equation A-2. Recall that the calculation for F-V is given by:

$$\text{F-V}_i = 1 - (Qj / Q_{total}) \qquad \text{(A-2)}$$

where Q_{total} is the summation of all of the minimal cut sets and Q*j* is the summation of the minimal cuts that do not contain event *i*. For example, the F-V for event ***Y*** would be given as:

$$\text{F-V}_Y = 1 - [P(\boldsymbol{X}\,\boldsymbol{Z}) / 0.17] = 1 - [0.2 \times 0.7 / 0.17] = 0.18 \qquad \text{(A-14)}$$

It can be shown that $\text{F-V}_X = 1.0$ and $\text{F-V}_Z = 0.41$. While at first glance these values for F-V may appear correct, they are really incorrect. The reason for the error lies in treatment of the Q*j* term. In general, Q*j* is supposed to represent either the risk or reliability portion that *does not* contain failure of event *j*. The PRA is really given in terms of sets, specifically sets **A** and **B**. These sets are illustrated in Figure A-3 through a Venn diagram.

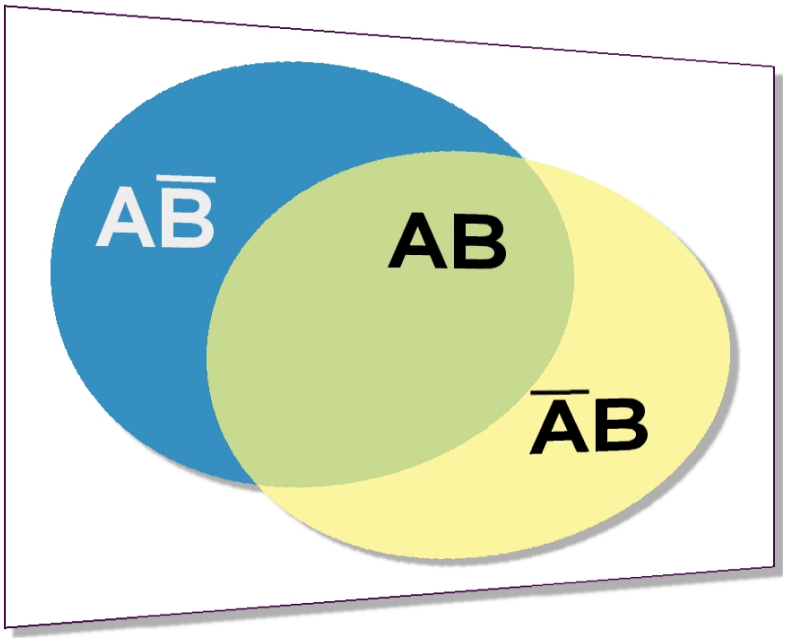

Figure A-3. Venn diagram illustrating events A and B.

Note that, in the Venn diagram, there are three areas of interest. The far left area represents the set **A** and not **B**. The middle set represents set **A** and set **B**. The set on the far right represents set not **A** and set **B**. In the incorrect F-V calculation, the term P(***X Z***) is analogous to the set **B**, but set **B** consists of two parts, **A B** or (not **A**) **B**. The area **A B** contains event Y. Consequently, the F-V calculation should discard this portion of **B** during the calculation, but it does not since most PRA tools deal with cut sets, not sets. Thus, the correct calculation of F-V can be found by evaluating the areas (i.e., probabilities) of the sets of interest. This calculation will be demonstrated using the sets and basic events from the example above.

$$
\begin{aligned}
\text{PRA(area)} &= \mathbf{A\ /B} + \mathbf{A\ B} + \mathbf{/A\ B} \\
&= \boldsymbol{X\ Y\ (/X + /Z) + X\ Y\ Z + (/X + /Y)\ X\ Z} \\
&= \boldsymbol{X\ Y\ /X + X\ Y\ /Z + X\ Y\ Z + /X\ X\ Z + /Y\ X\ Z} \qquad \text{(A-15)} \\
&= \boldsymbol{X\ Y\ /Z + X\ Y\ Z + X\ /Y\ Z}
\end{aligned}
$$

where an "/" indicates a complement

Now, the F-V measure for ***Y*** can be found by determining all sets containing failure of ***Y*** and then dividing this area by the total failure set area. Specifically, this calculation yields:

$$
\begin{aligned}
\text{F-V}_Y &= [\ \boldsymbol{X\ Y\ /Z + X\ Y\ Z}\ ] / [\ \boldsymbol{X\ Y\ /Z + X\ Y\ Z + X\ /Y\ Z}\ ] \\
&= [\ 0.2\times0.5\times0.3 + 0.2\times0.5\times0.7\ ] / [\ 0.2\times0.5\times0.3 + 0.2\times0.5\times0.7 + 0.2\times0.5\times0.7] \qquad \text{(A-16)} \\
&= [\ 0.1\ ] / [\ 0.17\ ] \\
&= 0.59
\end{aligned}
$$

Further, it can be shown that the correct F-V for events ***X*** and ***Z*** are:

$$
\begin{aligned}
\text{F-V}_X &= 1.0 \\
\text{F-V}_Z &= 0.82 \qquad \text{(A-17)}
\end{aligned}
$$

To summarize the issue with the F-V calculation being incorrect, we show the results in Table 1 for the simple example that was evaluated. Note that as the probability values for individual events *decrease* the incorrect F-V calculation approaches the exact calculation.

Table 1. Comparison of the correct F-V calculation with that used in typical PRA software.

Event	Correct F-V	Incorrect F-V
X	1.0	1.0
Y	0.59	0.18
Z	0.82	0.41

The other importance measures suffer from their own limitations also. To illustrate an issue related to RAW, let us look at a simple model for a two train low pressure injection system:. The equation representing system failure looks like:

$$SYS = A * B + CCF_AB \qquad \text{(A-18)}$$

where A is independent failure of the A pump, B is independent failure of the B pump, and CCF_AB is common-cause failure of both pumps.

The probabilities that we have are: P(A) = P(B) = 0.002; β = 0.10. We can evaluate the system failure probability to find:

P(SYS | nominal)= P(A) P(B) + P(CCF_AB) = 0.002*0.002 + 2E-4 = 2.04E-4

P(SYS | 1 pump failure, potential common-cause) = 1.0*0.002 + β = 0.002 + 0.10 = 0.102

P(SYS | 1 pump failure, independent failure) = 1.0*0.002 + 0.0 = 0.002

As can be seen, failures of redundant components affect the system failure probability. But, the extent to the change in probability is dependent on the type of component failure. For example, in the case where one component fails, the system failure probability varies from 0.102 (for a potential common-cause failure) to 0.002 (for an independent failure).

If we look at the importance measure given by the RAW, we see:

RAW(i'th component) = P(SYS | i'th component failed) / P(SYS | nominal)

Now, no one really calculates a RAW for the two types of failures we are talking about (potential common-cause and independent). Further, most PRA software packages do not adjust the common-cause basic event related to the i'th component event when calculating RAW. So, let us look at these calculations.

$$\begin{aligned} RAW(A)_{CCF} &= P(SYS \mid \text{A component failed due to common-cause}) / P(SYS \mid \text{nominal}) \\ &= 0.107 / 2.04E\text{-}4 \\ &= 525 \end{aligned}$$

$$\begin{aligned} RAW(A)_{IND} &= P(SYS \mid \text{A component failed due to independent}) / P(SYS \mid \text{nominal}) \\ &= 0.002 / 2.04E\text{-}4 \\ &= 9.8 \end{aligned}$$

$$\begin{aligned} RAW(A)_{\text{no CCF adjustment}} &= P(SYS \mid \text{A component failed}) / P(SYS \mid \text{nominal}) \\ &= 0.0022 / 2.04E\text{-}4 \\ &= 10.8 \end{aligned}$$

As can be seen from the results, the traditional (no adjustment) RAW calculation slightly overestimates (10.8 > 9.8) the RAW for independent failed components (in this example). But, the traditional RAW calculation can be a non-conservative estimate (10.8 << 525) for failure of component A if this failure is potentially of common-cause type. Other metrics such as a delta CDF will suffer from the "adjustment" problem.

A.3.3 Dynamic PRA for Accident Management

Accident management deals with a dynamic process: transient events in nuclear power plants involve complex interaction between the reactor core, primary loop, balance of the plant, emergency safeguards systems, and operator activities.. Traditional static PRA event trees are poorly suited to describe such a temporal evolution of events.

Methodologies based on dynamic PRA have been viewed as a natural step towards a more comprehensive way of evaluating Emergency Operating Procedures. The term "dynamic PRA" is used for a variety of methodologies, but essentially refers to modifications of the structure of the classical PRA to account for time-dependent effects (Hsueh and Mosleh, 1996). Dynamic effects can be grouped into two categories: long term effects (such as aging, environmental variations, plant design changes) and short time effects (such as time dependency of physical processes, time dependency of stochastic processes, operator response times). The events occurring during an accident pertains undoubtedly to the second category, short time effects.

Features of a dynamic PRA are the following:

1. Event sequences are represented by a "forward" branching tree where branching occurs in time, and therefore time is an explicit parameter of the model.

2. Branching times occur when an important characteristic of the system changes.

3. Event sequences are generated based on the rules describing the behavior of the various elements of the integrated model of the plant.

The event trees that result from such analyses are called Dynamic Event Trees (DETs). Depending on whether time is treated as a continuous or discrete variable, we will be dealing with continuous DETs or discrete DETs. An example of the application of discrete DETs is the

integrated safety assessment methodology which merges PRA and accident analysis techniques replacing the static event trees with dynamic ones (Sanchez and Melara, 1996).

Dynamic modeling of the plant behavior is not a simple task and adequate computer software must be developed. Nonetheless, the incident management prototype takes advantage of select facets of these general dynamic methods. For example, we utilize a simulation framework that has the ability to trace a sequence of events until an measurable outcome is realized.

A.4 References

Accorsi, R., G. Apostolakis, and E. Zio, 1999. "Prioritizing Stakeholder Concerns in Environmental Risk Management." *Journal of Risk Research* 2, 11-29.

Apostolakis, G., and S. Pickett, 1998. "Deliberation: Integrating Analytic Results into Environmental Decisions Involving Multiple Stakeholders," *Risk Analysis*, **18**, No. 5, 621-634.

Cheok, M. C., G. W. Parry, and R. R. Sherry, "Use of Importance Measures in Risk-Informed Regulatory Applications," *Reliability Engineering and System Safety*, **60**, 1998.

Clemen, R. T., 1996. *Making Hard Decisions*, Duxbury Press.

Cunningham, M. A. et al., "PRA Applications: Safety Goals and Acceptance Guidelines," *Proceedings of the International Topical Meeting on Probabilistic Safety Assessment PRA'99*, Editor: M. Modarres, American Nuclear Society, August 1999.

Ericson, D. M. et al., 1990. *Analysis of Core Damage Frequency: Internal Events Methodology*, NUREG/CR-4550, Vol. 1, Rev. 1.

Green, A. E., *Safety Systems Reliability*, John Wiley & Sons Ltd., 1983.

Henley, E. J. and H. Kumamoto, *Reliability Engineering and Risk Assessment*, Prentice-Hall, 1981.

Holahan, et al. "Current Developments in the Use of Probabilistic Risk Assessment in Nuclear Reactor Regulation," *Proceedings of the 4th International Conference on Probabilistic Safety Assessment and Management*, Editors: A. Mosleh and R. A. Bari, Springer, September 1998.

Hughes, W. R., 1986. "Deriving Utilities Using the Analytic Hierarchy Process." *Socio-Economic Planning Science*, **20,** No. 6, 393-395.

Keeney, R. L., and H. Raiffa, 1993. *Decisions with Multiple Objectives*, Cambridge University Press, New York, NY, USA.

Keeney, R.L., and von Winterfeldt, D., 1994. "Managing Nuclear Waste from Power Plants," *Risk Analysis*, **14**, No. 1, 107-130.

Lambert, H. E., "Measure of Importance of Events and Cut Sets in Fault Trees," *Reliability and Fault Tree Analysis*, SIAM, 1975.

Long, S. M., et al., 1998. "Current Status of the SAPHIRE Models for ASP Evaluations," *Proceedings of PRAM 4, Probabilistic Safety Assessment and Management*, pp. 1195-1202.

McCormick, N. J., *Reliability and Risk Analysis*, Academic Press, 1981.

Melnicoff, M. A., "PRA Support of Implementation of the Maintenance Rule at COMED," *Proceedings of the International Topical Meeting on Probabilistic Safety Assessment PRA'99*, Editor: M. Modarres, American Nuclear Society, August 1999.

Minarick, J. W., et al., 2000. *Precursors to Potential Severe Core Damage Accidents*, NUREG/CR-4674.

Modarres, M., 1993. *What Every Engineer Should Know About Reliability and Risk Analysis*, Marcel Dekker, Inc., New York, NY.

Russell, K. D, et al., 1998. *Systems Analysis Programs for Hands-On Integrated Reliability Evaluations (SAPHIRE) – System Overview Manual*, NUREG/CR-6532.

Smith, C. L., "Calculating Conditional Core Damage Probabilities for Nuclear Power Plants," *Reliability Engineering and System Safety*, 1998.

U.S. National Research Council, 1996. *Understanding Risk: Informing Decisions in a Democratic Society*, National Academy Press, Washington, DC.

U.S. Nuclear Regulatory Commission, 1975. *Reactor Safety Study, An Assessment of Accident Risk in U.S. Commercial Nuclear Power Plants*, WASH-1400.

U.S. Nuclear Regulatory Commission, 1988. *Individual Plant Examination For Severe Accident Vulnerabilities - 10 CFR 50.54(f)*, Generic Letter 88-20.

U.S. Nuclear Regulatory Commission, 1998. *An Approach for Using Probabilistic Risk Assessment in Risk-Informed Decisions on Plant-Specific Changes to the Licensing Basis*, RG 1.174.

B. XML Schema Definition for Probabilistic Risk Assessment

The purpose of this appendix is to provide an XML-based schema definition for the storage of PRA model information. The proposed definition captures the critical portions of a PRA model in a format following the XML specification defined by the World Wide Web Consortium (W3C). Specifically, we utilize the version 1.0 W3C format, as defined at:

http://www.w3.org/TR/REC-xml

XML is similar to the more familiar HTML in that tags – as defined by < > indicators – segregate information into individual groups. We utilize this feature to define the primary information related to PRA models. The XML structure is shown in Table B-1. While almost self-explanatory, comments are provide via the comment tag <!-- -->.

In general, our XML PRA specification addresses three entities, components, systems, and sequences. From the definition provided in Table B-1, it is possible (given the proper "back end" algorithm) to determine component failure probabilities related to various failure modes, plot graphical fault trees for the systems, plot graphical event trees for the accident sequences, generate minimal cut sets, and determine the epistemic uncertainy for the minimal cut sets. This XML PRA framework serves as a mechanism for information sharing between different analysis modules (for example, from one PRA tool to another) and as a means to import PRA information into the incident management prototype.

Within the XML scheme, some attributes can take on one of many options. For example, within the attribute tag, the "id" for the tag can identify different types of textual attributes, including the component train, failure mode, system, location, etc. When an attribute has mullitple options, they will be seperated by a "|" symbol. Also, when a particular tag has been defined once, subsequent use of the same tag will just show the abbreviated version of the tag.

Table B-1. The XML PRA schema definition and structure.

```
<?xml version="1.0"?>
<pra>
	<title>
		The project description.
	</title>
	<component id="Component_1">
		<title>
			Component 1 description
		</title>
		<text>
			Extra textual information.  This section could include HTML and
			RTF formatted text.
		</text>
		<attribute id="train | failure_mode | system | location | cost | weight">
			Attribute text
		</attribute>

		<failure_information>
			<attribute id="flow_rate">
				200 liters/minute
			</attribute>
		<!-- the rate tag indicates the failure information for failure modes
		such as operating, demand, and standby.  The 'aleatory' tag indicates
		the underlying stochastic model for the failure -->
			<rate id="operating | demand | standby">
				<aleatory>
					Poisson | Binomial
				</aleatory>
				<mean>
					1E-2
				</mean>
				<epistemic>
					<distribution>
						Lognormal
					</distribution>
					<parameter id="error factor">
						5
					</parameter>
					<correlated>
						Yes | No
					</correlated>
				</epistemic>
		<! -- if the failure is noticed at the time of the failure then the
		'revealed failure' tag should indicate yes, otherwise no -->
				<revealed_failure>
					Yes | No
				</revealed_failure>
			</rate>
		</failure_information>
		<! -- the 'restoration' section indicates whether or not a failure can
		be recovered.  The restoration 'rate' is the rate of repair -->
		<restoration_information>
			<spares>
				<mean></mean>
				<epistemic></epistemic>
			</spares>
			<rate>
				<aleatory></aleatory>
```

Table B-1. The XML PRA schema definition and structure.

```
                        <mean></mean>
                        <epistemic></epistemic>
                </rate>
        </restoration_information>
</component>
<system id="System_1">
        <title></title>
        <text></text>
        <!-- this system is defined like a normal fault tree where the failure
        criteria indicates the number of 'nodes' needed to fail and a node
        is define via the 'structure' tag -->
        <structure id="MainSystem">
                <failure_criteria>
                        2
                </failure_criteria>
                <structure>
                        Component_1
                </structure>
                <structure id="SubSystem">
                        <failure_criteria>
                                1
                        </failure_criteria>
                        <structure>
                                Component_2
                        </structure>
                        <structure>
                                Component_3
                        </structure>
                </structure>
        </structure>
        <!-- cut set postprocessing are actions that are to take place on the
        cut sets after they are generated. -->
        <cut_set_postprocessing>
                <search_criteria>
                        Component_1
                </search_criteria>
                <replace>
                        Component_1
                </replace>
                <replacement>
                        Component_4
                </replacement>
        </cut_set_postprocessing>

        <cut_set_postprocessing>
                <search_criteria>
                        Component_2
                </search_criteria>
                <append>
                        Component_4
                </append>
        </cut_set_postprocessing>
</system>
<system id="System_2">
        <title></title>
        <text></text>
        <!-- this system is defined by indicating the failure criteria based
        upon a component attribute.  For example, if a pumping system
        nominally supplies 400 liters per minute through two 200 liters per
        pump trains, and if 150 liters per minute is needed, then two pump
        trains would have to fail to fail the system.  In this case, the
```

Table B-1. The XML PRA schema definition and structure.

```
            failure criteria is 400 – 150 = 250 liters per minute.  In other words,
            as soon as 250 liters per minute (or more) of capacity is lost, then
            the system is failed. -->
            <structure id="MainSystem">
                  <failure_criteria>
                        250 liters per minute
                  </failure_criteria>
                  <structure>
                        Component_1
                  </structure>
                  <structure>
                        Component_2
                  </structure>
            </structure>
      </system>
            <!-- sequences are a series of events that start with an initiating
            event and lead to an end state (i.e., outcome) -->
      <sequence id="Sequence_1">
            <title></title>
            <text></text>
            <initiator id="Initiator_1">
                  <rate>
                        <aleatory></aleatory>
                        <mean></mean>
                        <epistemic></epistemic>
                  </rate>
            </initiator>
            <structure id="First_System">
                  <title></title>
                  <text></text>

                  <failed>
                        Yes | No
                  </failed>

                  <system>
                        System_1
                  </system>
            </structure>
            <structure id="Second_System">
                  <title></title>
                  <text></text>
                  <failed>
                        Yes | No
                  </failed>
                  <system>
                        System_2
                  </system>
            </structure>
            <end_state id="End_State_1">
                  <title>
                        End state name
                  </title>
                  <text></text>
            </end_state>
            <cut_set_postprocessing></cut_set_postprocessing>
      </sequence>
</pra>
```

C. Variability in Disutility Weight Determination.

Using our incident management decision makers, we ran an experiment to test utility independence between two attributes (X, the strength of containment, and Y, core damage probability). The outcome of this experiment was two utility curves, one for each attribute. In order for attribute X to be utility independent of Y, we require the condition $u(\{x \mid y0, y1, y2, \ldots\}) = u(x)$. In other words, the disutility of X must be constant for specific values of outcomes in Y.

Attribute X was defined as having three scales for containment performance: (1) robust, (2) moderate, and (3) weak. Attribute Y was defined as having one of two levels: unlikely (a probability of 1E-7) or likely (a probability of 0.2). The disutility elicitation process used the Analytic Hierarchy Process (AHP) for the pair-wise comparisons of preference on containment performance and then deliberation for finalization of the disutility weights. The AHP-deliberation step was repeated twice using our decision-makers, first under the assumption that core damage was unlikely and second under the assumption that core damage was likely. As part of this process, each of the four experts determined their individual disutility for containment performance, then a group disutility was obtained through consensus. The results of this experiment are listed in Table C-1.

Table C-1. Disutility values for containment performance conditional upon unlikely or likely core damage scenarios.

Containment performance	Disutility given probability core damage = 1E-7	Disutility given probability core damage = 0.2
Robust	0	0
Moderate	0.38	0.26
Weak	1	1

Notice that while the two disutilities are similar, they are not exactly the same. Indeed, it would have extremely fortuitous if the disutilities had been the same, considering that a two step process (AHP and deliberation) was used to determine the disutility. We postulate then that the decision maker must consider the variability in the disutility in order to determine utility independence.

We would like to consider the variability in the resulting disutility values, recalling that disutility is determined via subjective preference elicitation. While this concept is similar to that of epistemic uncertainty frequently encountered in decision analysis, variability, as used here, is really a special type of epistemic uncertainty. Since the disutility is a measure of preference, it is independent of probability. In other words, it is not appropriate to model preference variability by statements like "attribute X is absolutely more important than Y, with probability of 0.3." Note that in the decision-making framework, the impact of probabilities are felt in quantification of metrics such as expected weighted disutility. But, while disutility is not a probability, it may nonetheless display signs of variability due to the subjective nature of its development – variability due to the modeling uncertainty rather than parametric uncertainty.

To estimate the variability of disutility, we need to first return to the AHP process to understand how the disutility is determined. As used in this paper, we ask the decision-makers to rank pair-wise comparisons (between outcomes) on a 1-to-9 scale, where the scale represents:

1 Two outcomes are *equally* important
3 One outcome is *weakly* more important than the other outcome
5 One outcome is *strongly* more important than the other outcome
7 One outcome is *very strongly* more important than the other outcome
9 One outcome is *absolutely* more important than the other outcome

The even numbered scales are used to facilitate a compromise between slightly differing preferences. We will utilize this aspect of AHP to postulate an appropriate level of variability in the resulting disutility for containment performance. Specifically, we will assume that a decision maker precision on identifying his or her preference level is valid only to the next integer scale

(either high or lower, if applicable). For example, if the decision-maker indicates that a cost of $50 million is strongly more important than $10 million, his or her belief is that an AHP scale of 5 is applicable, but a scale of 4 or 6 could be possible. Further, we assume that jumps of two integer scales (e.g., from 5 to 7) are not realistic. Of course, these assumptions on variability in disutility are just those, assumptions, and have not been tested experimentally. But, for situations of interest with respect to incident management, asking the decision maker to constrain their subjective judgment locally to three scales out of nine is not unreasonable.

We revisited the original AHP weights provided to us by our decision makers to look at two variations. First, an "upper variation" was performed where we modified the three AHP scales up by a value of one (since there are three containment performance scales, we will have three pair-wise comparisons). We then recalculated the utility based upon the slight AHP scale modification. Second, a "lower variation" was performed where we modified the three AHP scales down by a value of one. Again, we recalculated the utility based upon this second modification. The results of these two calculations are shown in Figure C-1.

The maximum variation (from upper to lower sensitivity) in the disutility was about 38%. This implies that if we can determine preference on an AHP scale to a ±1 value, then the disutility may vary by a noticeable amount. Further, during the utility independence experiment discussed earlier, it was noticed that the containment performance disutility varied by a maximum of 31% for different outcomes of the core damage attribute. This deviation is within the variation that could be expected simply as a matter of performing the AHP process.

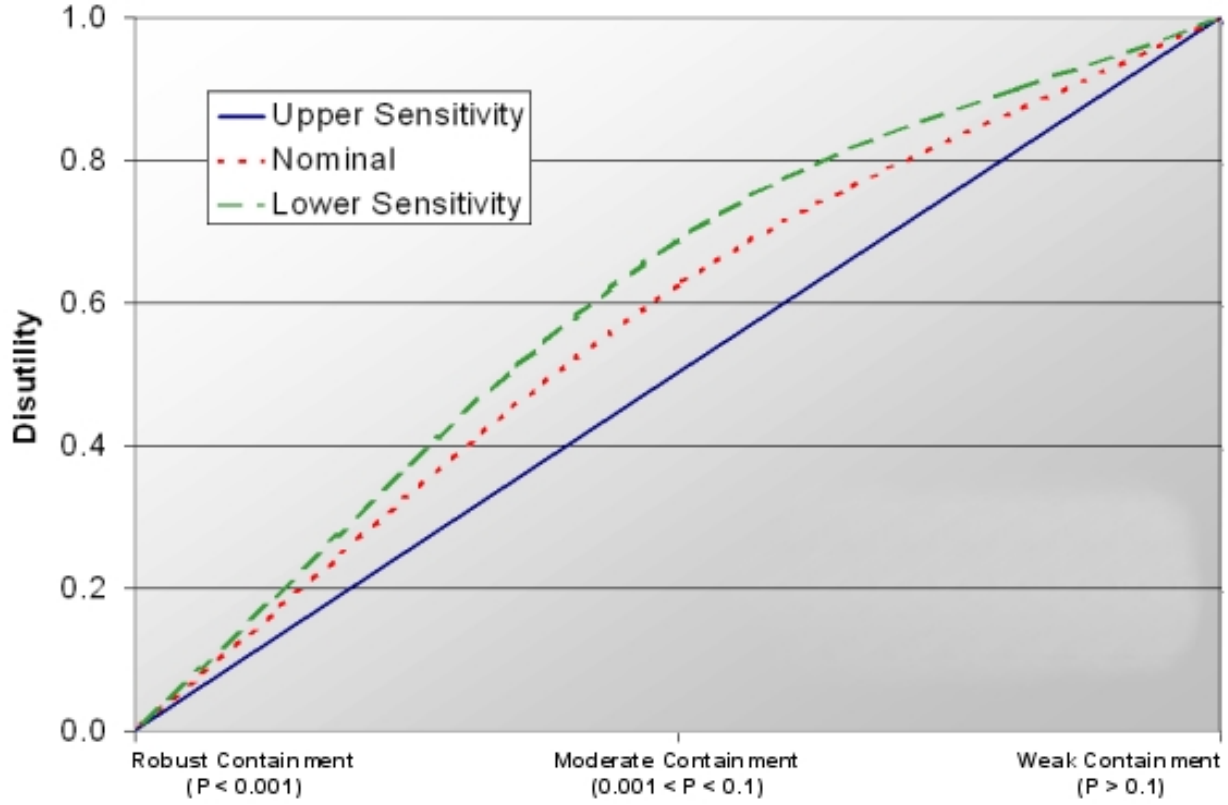

Figure C-1. Disutility results for the AHP scale sensitivity calculations.

It is interesting to note that for the "upper" sensitivity calculation, varying the decision-maker preference weights upward by a single unit changed the curve from "risk prone" to "risk neutral" (at least on the ordinal scale). It is somewhat surprising that a slight modification can have such an impact on a disutility curve, but this behavior may be due to the belief (by the decision-makers) that nuclear power plant containment provides one of multiple formidable engineering barriers to release. Consequently, the thinking might be that one may not need to "worry" (or have larger disutility) as the containment is weakened since ultimately the plant is likely to withstand accident events.

Lastly, we would like to note that AHP is intended for preference comparisons of items that are of relatively similar importance. For example, let us assume that we have two attributes, X and

Y, and wish to determine their respective weights. If Y is absolutely preferred over X, then we would assign an AHP value of 9 for the Y-to-X comparison (and, conversely, a 1/9 value for the X-to-Y comparison). Evaluating the eigenvector for the associated AHP matrix yields a weight of about 0.9 for Y and 0.1 for X. Consequently, with only two attributes, the largest resulting pair-wise comparison weights would be 0.9 and 0.1. If the decision maker intended that Y be weighted 0.99 and X be weighted 0.01, then AHP (with the 1 to 9 scale) would not be of use (in isolation of deliberation). Another numeric problem exists when the decision-maker is effectively indifferent between two attributes. If attribute Y is slightly more preferred over X, we could assign an AHP value of 2 for the Y-to-X comparison (and, conversely, a 1/2 value for the X-to-Y comparison). But, the resulting weights that come from the AHP process are for Y a value of 2/3 and for X a value of 1/3. Thus, even though the two attributes are almost equal in the mind of the decision-maker, the AHP weights places Y twice as important as X, which is probably not what the decision-maker desires. In summary, if one utilizes AHP, care should be exersized in determining the attribute weights due to numerical artifacts – artifacts that ultimately place additional burden on the deliberation process.

D. Source Listings for the Incident Management Prototype.

This appendix contains the PHP source code for Version 1 of the incident advisor prototype. We list the code needed to determine the decision analysis context and the subsequent solution of the decision model. Included in the source code listing are the following modules:

analyze_incident.php
analyze_incident_prototype.php
func_components.php
func_utility.php
get_resources.php
help.php
incident.php
module_decisions.php
module_do_decisions.php
module_do_simulation.php
module_edit_IE_impact.php
module_initiators.php
module_session_handler.php
module_state_information.php
module_step_0.php
module_step_1.php
module_step_2.php
module_step_2_components.php
module_step_2_initiators.php
module_step_3_1.php
module_step_3_2.php
module_step_3_3.php
module_step_3_4.php
module_step_3_edit.php
module_step_4_1.php
module_step_5_1.php
module_step_5_2.php
module_variables.php
page_footer.php
page_header.php
process_changes.php
utility_makeplot.php
utility_plot.php

< analyze_incident.php >

```
<?
// This page controls the flow of information for the incident analysis prototype.
// Page flow is controlled by two variables, $the_step and $sub_step.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
//
// Initialize and register session variables.
//
require("module_session_handler.php");

// Each page must have the variables:
//
// $page_name              Text to show for the title at top of page
// $page_title             Test to show for the page title
// $page_description       Test to show for the page description
// $the_step               Integer step number (1 to 5) corresponding to the level in the pyramid
// $sub_step               Integer indicating incremental step in process


// Check to see if we are starting.
if (!isset($the_step)) {$the_step = 0;}

// Check to see if we are on a sub step
if (!isset($sub_step)) {$sub_step = 1;}

if ($the_step == 0) {

        // Call the prototype module that contains this page HTML output.
        include("module_step_0.php");
}

if ($the_step == 1) {

        // Call the prototype module that contains this page HTML output.
        include("module_step_1.php");
}

if ($the_step == 2) {

        // Call the prototype module that contains this page HTML output.
        include("module_step_2.php");
}

if ($the_step == 3) {

        if ($sub_step == 1)
        {
                // Call the prototype module that contains this page HTML output.
                include("module_step_3_1.php");
        }
        elseif ($sub_step == 2)
        {
                // Call the prototype module that contains this sub step HTML output.
                include("module_step_3_2.php");
        }
        elseif ($sub_step == 3)
        {
                // Call the prototype module that contains this sub step HTML output.
                include("module_step_3_3.php");
        }
        elseif ($sub_step == 4)
        {
```

```
                // Call the prototype module that contains this sub step HTML output.
                include("module_step_3_4.php");
        }
}

if ($the_step == 4) {

        if ($sub_step == 1)
        {
                // Call the prototype module that contains this page HTML output.
                include("module_step_4_1.php");
        }
//      elseif ($sub_step == 2)
//      {
//              // Call the prototype module that contains this sub step HTML output.
//              include("module_step_4_2.php");
//      }

}

if ($the_step == 5) {

        if ($sub_step == 1)
        {
                // Call the prototype module that contains this page HTML output.
                include("module_step_5_1.php");
        }
}

// Call the prototype module that controls the HTML output.
require("analyze_incident_prototype.php");
?>
```

< analyze_incident_prototype.php >

```
<?
// This page controls HTML page output for the prototype.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Variables that are required to be passed to this module:
//
// $page_name              Text to show for the text at top of page
// $page_title             Text to show above description
// $page_description       Test to show for the page description
// $the_step               Integer step number (1 to 5)
// $the_step_state         Array with sub-state information related to the i'th step
// $page_table             HTML for the table containing the data collection.

// Defined needed variables.
//
$backcolor_dark = "\"#9999CD\"";

// Convert the step level to a Roman numeral.
if ($the_step == 5) {$the_step_roman = "V";} elseif ($the_step == 4) {$the_step_roman = "IV";} elseif ($the_step == 3) {$the_step_roman =
"III";} elseif ($the_step == 2) {$the_step_roman = "II";} else {$the_step_roman = "I";}

// Show HTML header information.
if ($the_step > 0)
{
        print ("<html><head><title>Incident Advisor - Step $the_step_roman </title><LINK REL=stylesheet TYPE=\"text/css\"
HREF=\"http://psa.mit.edu/PSA.css\"></head>");
} else
{
        print ("<html><head><title>Incident Advisor </title><LINK REL=stylesheet TYPE=\"text/css\"
HREF=\"http://psa.mit.edu/PSA.css\"></head>");
}


// Now show the page header text.
require("page_header.php");

// Continue with the HTML page.
print ("<div align=\"center\"><br>\n");

// Print the description table.
print ("<table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=600 align=center>");
print ("<tr bgcolor=$backcolor_dark align=\"center\" class=\"arial-med-light\"><td> <br>$page_title <br> <br></td></tr>");
print ("<tr align=\"left\" class=\"arial-med\"><td><br>$page_description<br><br></td></tr></table>");

// Print the table containing the data collection and information extraction routines.
print ($page_table);

// Finish the page.
print ("<br>");

require("page_footer.php");

print ("</div></body></html>");

?>
```

< func_components.php >

```
<?
// define needed functions
//
// This module contains the following functions:
//
// system_impact     returns the i'th system failure probability given j component probabilities
// ccdp_impact       returns the CCDP given j component probabilities
//
//
// system_impact returns a probability value
// inputs are: $system, $component, $component_probability
//
// where:
//        $system                   attribute
//        ----------------          ------------------------
//                ?                 i'th system identifier
//
//        $component                array of values
//        ---------------           ------------------------
//                ?                 j'th component identifier
//
//        $component_probability    array of values
//        ----------------------    ------------------------
//                ?                 actual failure probability
//

function system_impact ($system, $component, $component_probability)
{
// The system impact is evaluated via a binary decision diagram (BDD) or fault tree logic (FTL).


}


// ccdp_impact returns a probability value
// inputs are: $component, $component_probability
//
// where:
//
//        $component                array of values
//        ---------------           ------------------------
//                ?                 j'th component identifier
//
//        $component_probability    array of values
//        ----------------------    ------------------------
//                ?                 actual failure probability
//

function ccdp_impact ($component, $component_probability)
{
// The CCDP impact is evaluated via a binary decision diagram (BDD) or fault tree logic (FTL).


}

?>
```

< func_utility.php >

```
<?
// This page defines required disutility functions for the prototype.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
// This module contains the following functions:
//
// FUNCTIONS FOR CONTINUOUS DISUTILITY
//
// continuous_utility      returns continuous disutility given scale value (x)
// continuous_utility_inv  returns continuous scale value (x) given disutility
// continuous_utility_max  returns the maximim scale value (x) for a performance measure
//
// FUNCTIONS FOR DISCRETE DISUTILITY
//
// discrete_utility        returns continuous disutility given scale value (x)
// discrete_utility_inv    returns continuous scale value (x) given disutility
// discrete_utility_max    returns the maximim scale value (x) for a performance measure
//
// GENERAL FUNCTIONS
//
// measure_weight          returns the weight for a performance measure
// measure_name            returns the name for a performance measure
//
//
//
// continuous_utility returns a continuous disutility value between 0 and 1
// inputs are: $attribute_value, $x_value, $curve_type
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//
//        $x_value           value
//        ---------------    ---------
//                ?          actual outcome (euro, sV, etc.)
//
//        $curve_type        curve
//        ---------------    ---------
//                0          point to point from weights (raw data)

function continuous_utility ($attribute_value, $x_value, $curve_type="0")
{

// Cost
//
if ($attribute_value == 1) {

  $disutility = (($x_value / 5E8) - 2E-6);

  if ($disutility < 0) {
  $disutility = 0;
  }

  if ($disutility > 1) {
  $disutility = 1;
  }

  return $disutility;
```

```
}

// Dose
//
if ($attribute_value == 2) {

  $disutility = (($x_value / 875) - 1.1426E-7);

  if ($disutility < 0) {
  $disutility = 0;
  }

  if ($disutility > 1) {
  $disutility = 1;
  }

  return $disutility;
}

}

// continuous_utility_inv returns the inverse of continuous disutility value.
// inputs are: $attribute_value, $utility_value, $text, $curve_type
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//
//        $disutility        value
//        ---------------    ---------
//                ?          0 to 1
//
//        $text              value
//        ---------------    ---------
//               FALSE       raw value
//               TRUE        formatted text string is returned
//
//        $curve_type        curve
//        ---------------    ---------
//                0          point to point from weights (raw data)

function continuous_utility_inv ($attribute_value, $disutility, $curve_type="0")
{

// Cost
//
if ($attribute_value == 1) {

  if ($disutility < 0) {
  $disutility = 0;
  }

  if ($disutility > 1) {
  $disutility = 1;
  }

  if ($text) {
    $value = (($disutility + 2E-6)*5E8);
    $value = number_format($value/1000);

    if ($value < 1000) {
    $value = number_format($value*1000);
    return "$value euro";
    }
```

```
    else {
    return "$value x10<sup>3</sup> euro";
    }
  }
  else {return (($disutility + 2E-6)*5E8);}

}

// Dose
//
if ($attribute_value == 2) {

  if ($disutility < 0) {
  $disutility = 0;
  }

  if ($disutility > 1) {
  $disutility = 1;
  }

  if ($text) {
    $value = (($disutility + 1.1426E-7)*875);
    $value = number_format($value);
    return "$value Sv";
  }
  else {return (($disutility + 1.1426E-7)*875);}
}

}

// continuous_utility_max returns maximum scale (x) value.
// inputs are: $attribute_value
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//

function continuous_utility_max ($attribute_value)
{
// Cost
if ($attribute_value == 1) {

  return 5E8;

}

// Dose
if ($attribute_value == 2) {

  return 875;

}

}

//
// DISCRETE FUNCTIONS
//

//
// discrete_utility returns a disutility value between 0 and 1
// inputs are: $attribute_value, $x_value, $curve_type
//
```

```
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                3          industrial safety
//                4          core damage
//                5          external attention
//
//        $x_value           value
//        ---------------    ---------
//                ?          actual outcome (ordinal scale, 1, 2, 3, etc.)
//
//        $curve_type        curve
//        ---------------    ---------
//                0          point to point from weights (raw data)

function discrete_utility ($attribute_value, $x_value, $curve_type="0")
{

// Industrial safety
//
if ($attribute_value == 3) {

  $disutility_point[1] = 1.00E-05;
  $disutility_point[2] = 3.60E-05;
  $disutility_point[3] = 8.00E-04;
  $disutility_point[4] = 8.00E-03;
  $disutility_point[5] = 8.00E-02;
  $disutility_point[6] = 1.00E+00;

  return $disutility_point[$x_value];

}

// Core Damage
//
if ($attribute_value == 4) {

  $disutility_point[1] = 0;
  $disutility_point[2] = 1;

  return $disutility_point[$x_value];
}

// External Attention
//
if ($attribute_value == 5) {

  $disutility_point[1] = 1.00E-05;
  $disutility_point[2] = 8.20E-05;
  $disutility_point[3] = 1.71E-02;
  $disutility_point[4] = 8.00E-02;
  $disutility_point[5] = 5.20E-01;
  $disutility_point[6] = 1.00E+00;

  return $disutility_point[$x_value];
}

}

// discrete_utility_inv returns a inverse of continuous utility value.
// inputs are: $attribute_value, $utility_value, $curve_type
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                3          industrial safety
```

```
//                4          core damage
//                5          external attention
//
//        $utility_value     value
//        ---------------    ---------
//                ?          0 to 1
//
//        $text              value
//        ---------------    ---------
//               FALSE       raw value
//               TRUE        formatted text string is returned
//
//        $curve_type        curve
//        ---------------    ---------
//                0          point to point from weights (raw data)

function discrete_utility_inv ($attribute_value, $utility_value, $text, $curve_type="0")
{

// Industrial safety
//
if ($attribute_value == 3) {

  if ($utility_value < 3.60E-05) {if ($text){return "None";} else {return 1;}}
  elseif ($utility_value < 8.00E-04) {if ($text){return "Minor Injury";} else {return 2;}}
  elseif ($utility_value < 8.00E-03) {if ($text){return "Major Injury";} else {return 3;}}
  elseif ($utility_value < 8.00E-02) {if ($text){return "1 Fatality";} else {return 4;}}
  elseif ($utility_value < 1) {if ($text){return "10 Fatalities";} else {return 5;}}
  else {if ($text){return "125 Fatalities";} else {return 6;}}

}

// Core Damage
//
if ($attribute_value == 4) {

  if ($utility_value < 1) {if ($text){return "None";} else {return 1;}}
  else {if ($text){return "Core Damage";} else {return 2;}}

}

// External Attention
//
if ($attribute_value == 5) {

  if ($utility_value < 8.20E-05) {if ($text){return "None";} else {return 1;}}
  elseif ($utility_value < 1.71E-02) {if ($text){return "Report Event";} else {return 2;}}
  elseif ($utility_value < 8.00E-02) {if ($text){return "Inspection";} else {return 3;}}
  elseif ($utility_value < 5.20E-01) {if ($text){return "Regulatory Intervention";} else {return 4;}}
  elseif ($utility_value < 1) {if ($text){return "One Year Shutdown";} else {return 5;}}
  else {if ($text){return "Two Year Shutdown";} else {return 6;}}

}

}


// discrete_utility_max returns maximum scale (x) value.
// inputs are: $attribute_value
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                3          industrial safety
//                4          core damage
//                5          external attention
```

```
//

function discrete_utility_max ($attribute_value)
{
// Industrial safety
//
if ($attribute_value == 3) {

  return "125 Fatalities";

}

// Core Damage
//
if ($attribute_value == 4) {

  return "Core Damage";
}

// External Attention
//
if ($attribute_value == 5) {

  return "Two Year Shutdown";
}

}

//
// GENERAL FUNCTIONS
//

// measure_weight returns a performance measure weight between 0 and 1
// inputs are: $attribute_value
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//                3          industrial safety
//                4          core damage
//                5          external attention
//

function measure_weight ($attribute_value)
{
//cost
if ($attribute_value == 1) {
 return 0.32; }

// worker dose
if ($attribute_value == 2) {
 return 0.16; }

// worker accident
if ($attribute_value == 3) {
 return 0.16; }

// core damage
if ($attribute_value == 4) {
 return 0.21; }

// external influence
if ($attribute_value == 5) {
 return 0.15; }
```

```
}

// measure_scale returns the performance measure scale text.
// inputs are: $attribute_value, $utility_value
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//                3          industrial safety
//                4          core damage
//                5          external attention
//
//        $utility_value is between 0 and 1


// measure_name returns a performance measure name
// inputs are: $attribute_value
//
// where:
//        $attribute_value   attribute
//        ----------------   ---------
//                1          cost
//                2          radiological dose
//                3          industrial safety
//                4          core damage
//                5          external attention
//

function measure_name ($attribute_value)
{

if ($attribute_value == 1) {
 return "Cost"; }

if ($attribute_value == 2) {
 return "Radiological Dose"; }

if ($attribute_value == 3) {
 return "Industrial Safety"; }

if ($attribute_value == 4) {
 return "Core Damage"; }

if ($attribute_value == 5) {
 return "External Attention"; }

}

?>
```

< get_resources.php >

```
<?
// This page is used to get textual information that appears on-screen for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//

//
// First check to see if the user has changed the language option.
//

// If the language key is not set, default to English (value of 1)
if (!isset($_SESSION['l_key'])) {
        $lang_key = 1;
        $l_key = 1;
}

//
// The prototype uses the concept of 'resources' for screen displays.  Each
// resource is stored in the knowledge base and is duplicated for the languages
// represented in the prototype.
//
//
// Query for the page textual information, specific to the currently-selected
// language (French, English, Russian, etc.).
//

$connect = odbc_connect("incident_db", "", "");
// query the menu_text table for this page's resources
$query = "SELECT * FROM menu_text WHERE (Page_ID = " . $Page_ID . ") AND (Language = " . $_SESSION['l_key'] . ") ";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
        while (odbc_fetch_row($result)) {
        $resource_key = odbc_result($result, "ID_TextKey");
        $resource[$resource_key] = odbc_result($result, "Text");
        }
}
// close the dB connection
odbc_close($connect);

?>
```

< help.php >

```
<?
// This page is the help page for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
// This prototype uses sessions.  Sessions are global variables that, once set, are automatically
// stored by the server and are available on any subsequent page.
//
session_start();

if (!isset($_SESSION['l_key'])) {
    $_SESSION['l_key']=1;
}

//
// Define any needed variables
//
$Page_Title = "Incident Help Page";

//
// Determine the actual help text depending on the Page_ID variable.
//
// For each help page, a title for the page should be specified.  Three
// variables for help text are provided (it is not necessary to utilize
// all three).  These variables may contain HTML.
//

if ($Page_ID == 1) {
} elseif ($Page_ID == 2) {
        $page_title = "";
        $help_text_1 = "";
        $help_text_2 = "";
        $help_text_3 = "";
} else {
        $page_title = "";
        $help_text_1 = "";
        $help_text_2 = "";
        $help_text_3 = "";
}

?>

<html>
<head>
<title>Incident Advisor Help Page</title>
<LINK REL=stylesheet TYPE="text/css" HREF="http://psa.mit.edu/PSA.css">
</head>

<?
 // now show the header
 require("page_header.php");
?>

<div align="center">
<br>
<table border="0" cellpadding="0" cellspacing="0" width=600 align=center>
        <tr>
                <td>
                <span class=arial-large><? print $page_title ?></span><br>
                </td>
        </tr>
```

```
        <tr>
                <td>
                <span class=arial-med><? print $help_text_1 ?></span><br>
                </td>
        </tr>
        <tr>
                <td>
                <span class=arial-med><? print $help_text_2 ?></span><br>
                </td>
        </tr>
        <tr>
                <td>
                <span class=arial-med><? print $help_text_3 ?></span><br>
                </td>
        </tr>
        <tr>
                <td>
                <span class=arial-small>Click the 'Back' button to return to the previous page.</span><br>
                </td>
        </tr>
</table>

<br>

<?
//phpinfo();
require("page_footer.php")
 ?>

</div>
</body>
</html>
```

< incident.php >

```
<?
// This page is the main starting page for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
//
// Initialize and register session variables.
//
session_start();
session_destroy();

require("module_session_handler.php");
// If it is not already a session variable, initialize the language key.
if (!isset($_SESSION['l_key'])) {$_SESSION['l_key']=1; $lang_key = 1;}
//
// Define any needed variables
//
$Page_Title = "Incident Home Page";
$the_step = 0;
$Page_ID = 1;
//
// Obtain information specific to this page ($Page_ID must be set).
//
require("get_resources.php");

?>
<html>
<head>
<title>Incident Home Page</title>
<LINK REL=stylesheet TYPE="text/css" HREF="http://psa.mit.edu/PSA.css">
</head>

<?
 // now show the header
 require("page_header.php");
?>

<div align="center">
<br>
<table border="0" cellpadding="0" cellspacing="0" width=600 align=center>
        <tr>
                <td>
                <span class=arial-large><? print $resource[1] ?></span><br>
                </td>
        </tr>
        <tr>
                <td>
                <span class=arial-med><a href="analyze_incident.php"><? print $resource[2] ?></a></span><br><br>
                </td>
        </tr>
</table>

<br>
<?
        require("page_footer.php")
?>
</div>
</body>
</html>
```

< module_decisions.php >

```
<?
// This page queries the knowledge base for decision alternatives.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables:
//
// $decision[$the_key], where
//     $the_key is the decision ID_key
//     $decision contains the decision description
//

// Query the knowledge base to obtain the relevant DECISIONS.
$connect = odbc_connect("incident_db", "", "");
// query the node table (type 1)
$query = "SELECT * FROM Node WHERE (Type_Text = 1)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $d_description = odbc_result($result, "Description");
    $d_key = odbc_result($result, "ID_Key");
    $decision[$d_key] = $d_description;
  }
}
else {
  $decision[1]="none";
}
// close the dB connection
odbc_close($connect);

?>
```

< module_do_decisions.php >

```
<?
// This page calculates the PI for the i'th decision alternative.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//


// Assume the following modules have already been called.
//  require("func_utility.php");
//  require("module_initiators.php");
//  require("module_variables.php");
//  require("module_decisions.php");


function state_value ($state_key)
{
        if ($state_key == 724 OR $state_key==723) {return (1);}
        if ($state_key == 726 OR $state_key==727 OR $state_key==728 OR $state_key==729) {return (2);}
        if ($state_key == 730 OR $state_key==731 OR $state_key==732) {return (3);}
        if ($state_key == 733 OR $state_key==734) {return (4);}
}

function worker_outcome ($hours_of_work)
{
        $incident = 1 - exp(-$hours_of_work * $V[8][$u]);
        $minor_probability = $incident * $V[9][$u];
        $severe_probability = $incident * $V[10][$u];
        $fatality_probability = $incident * $V[11][$u];
        $tenfatality_probability = $incident * $V[11][$u]/10;
        $onetwentyfivefatality_probability = $incident * $V[11][$u]/125;

        $out = $minor_probability * discrete_utility(3, 2);
        $out = $out + $severe_probability * discrete_utility(3, 3);
        $out = $out + $fatality_probability * discrete_utility(3, 4);
        $out = $out + $tenfatality_probability * discrete_utility(3, 5);
        $out = $out + $onetwentyfivefatality_probability * discrete_utility(3, 6);

        return ($out);
}

function CD_disutility_outcome ($u)
{
        $out[1] = continuous_utility(1, $V[7][$u]);  // CD cost + replacement power + special costs
        $out[2] = continuous_utility(2, 7500 * 0.001);      // Assume 7500 hours of cleanup in 0.001 Sv/hr field
        $out[3] = worker_outcome(15000);             // Assume 15000 hours of cleanup
        $out[4] = 1;                                 // Core damage
        $out[5] = discrete_utility(5, 6);            // CD level of attention

        return $out;
}

function IE_disutility_outcome ($init_key, $u)
{
        if ($init_key == 714) { //   $IE[714] = "APRP - LOCAs"
                $out[1] = continuous_utility(1, $V[12][$u]*$V[4][$u] + $V[21][$u]); // replacement power + special costs
                $out[2] = continuous_utility(2, 750 * 0.001);       // Assume 750 hours of cleanup in 0.001 Sv/hr field
                $out[3] = worker_outcome(1500);                     // Assume 1500 hours of cleanup
                $out[4] = 0;                                 // No core damage
                $out[5] = discrete_utility(5, $V[30][$u]);   // LOCA level of attention
        }
        elseif ($init_key == 715) { //   $IE[715] = "ATWS - Anticipated trans. w/o scram"
```

```
		$out[1] = continuous_utility(1, $V[16][$u]*$V[4][$u] + $V[25][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 250 * 0.001);       // Assume 250 hours of cleanup in 0.001 Sv/hr field
		$out[3] = worker_outcome(500);                      // Assume 500 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[34][$u]);  // ATWS level of attention
	}
	elseif ($init_key == 716) { //   $IE[716] = "PSF - Loss of heat sink"
		$out[1] = continuous_utility(1, $V[17][$u]*$V[4][$u] + $V[26][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 250 * 0.0001);      // Assume 250 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(500);                      // Assume 500 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[35][$u]);  // HS level of attention
	}
	elseif ($init_key == 717) { //   $IE[717] = "PSL - Loss of power"
		$out[1] = continuous_utility(1, $V[18][$u]*$V[4][$u] + $V[27][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 50 * 0.0001);       // Assume 50 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(100);                      // Assume 100 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[36][$u]);  // LOP level of attention
	}
	elseif ($init_key == 718) { //   $IE[718] = "RTE - Secondary system rupture"
		$out[1] = continuous_utility(1, $V[13][$u]*$V[4][$u] + $V[22][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 750 * 0.0001);      // Assume 750 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(1500);                     // Assume 1500 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[31][$u]);  // Sec level of attention
	}
	elseif ($init_key == 719) { //   $IE[719] = "RTGV - Steam generator tube rupture"
		$out[1] = continuous_utility(1, $V[14][$u]*$V[4][$u] + $V[23][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 750 * 0.001);       // Assume 750 hours of cleanup in 0.001 Sv/hr field
		$out[3] = worker_outcome(1500);                     // Assume 1500 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[32][$u]);  // SGTR level of attention
	}
	elseif ($init_key == 720) { //   $IE[720] = "RTV - Steam line break"
		$out[1] = continuous_utility(1, $V[15][$u]*$V[4][$u] + $V[24][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 750 * 0.0001);      // Assume 750 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(1500);                     // Assume 1500 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[33][$u]);  // Steam level of attention
	}
	elseif ($init_key == 721) { //   $IE[721] = "TGTA - Secondary side transient"
		$out[1] = continuous_utility(1, $V[19][$u]*$V[4][$u] + $V[28][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 12 * 0.0001);       // Assume 12 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(24);               // Assume 24 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[37][$u]);  // SecTrans level of attention
	}
	elseif ($init_key == 722) { //   $IE[722] = "TRCP - Primary side transient"
		$out[1] = continuous_utility(1, $V[20][$u]*$V[4][$u] + $V[29][$u]); // replacement power + special costs
		$out[2] = continuous_utility(2, 12 * 0.0001);       // Assume 12 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome(24);               // Assume 24 hours of cleanup
		$out[4] = 0;                                // No core damage
		$out[5] = discrete_utility(5, $V[38][$u]);  // SecTrans level of attention
	}

	return $out;
}

function calc_PI ($theoutcome, $u)
{
	$PI_cost = $probability * $theoutcome[1] * $V[39][$u];
	$PI_dose = $probability * $theoutcome[2] * $V[40][$u];
	$PI_work = $probability * $theoutcome[3] * $V[41][$u];
	$PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
	$PI_attn = $probability * $theoutcome[5] * $V[43][$u];
```

```
        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        return $PI_total;
}

//
// function weighted_PI (i'th decision key, index for j'th variable, index for decision number)
//

function weighted_PI ($key, $u, $jth_decision)
{
        // Declare any global variables.
        global $IE;
        global $IE_rate;
        global $V;

        // $key = Decision key (1 = as is, 7 = repair at power, 9 = isolate, 10 = reduce power, 702 = hot shutdown, etc.
        //
        // $u   = Index into the array for a variable.  If doing nominal case and sensitivity
        //        calculations (1=nominal, 2=low value, 3=high value).  If doing the uncertainty
        //        anaysis, then $u = iteration number.
        //
        // $jth_decision = the numerical order of the decisions (0, 1, 2, etc.)
        //

        include("module_state_information.php");

        if ($key == 1) // as is
        {
                // Only one state (1) is used.
                $j = 1;

                // Repair can not trip the plant since there is no repair.
                // Repair can not be successful since there is no repair.

                $state_number = state_value($plant_state_key[$j]);

                $sequence_number = 1;

                $cumulative_prob = 0;
                $cumulative_PI = 0;

                // Loop through the IE categories while in this state

                foreach($IE as $init_key => $init_name)
                {
                        if (isset($IE_modification[$init_key][$state_number])) { // State has an impact to the rate.
                                $rt = $IE_modification[$init_key][$state_number];
                        } else { $rt = $IE_rate[$init_key][$state_number];}

                        $probability = (1-exp(- $rt * $V[1][$u]/8760)) * (1-$V[46][$u]);   // P(trip) * P(no CD)

                        // Determine outcomes[DT seq. #][Performance Measure]
                        // Cost = 1  Dose = 2  Worker = 3  CD = 4  Attention = 5
                        //
                        // First do the non-CD branch following a IE, but we need to
                        // get the right impacts for each IE.

                        $theoutcome = IE_disutility_outcome($init_key, $u);

                        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];
```

```
                $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                $cumulative_prob = $cumulative_prob + $probability;
                $cumulative_PI = $cumulative_PI + $PI_total;
                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

                // Now do the CD branch following a IE

                $sequence_number = $sequence_number + 1;

                $probability = (1 - exp(- $rt * $V[1][$u]/8760)) * $V[46][$u];      // P(trip) * P(CD)

                $theoutcome = CD_disutility_outcome($u);

                $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

                $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                $cumulative_prob = $cumulative_prob + $probability;
                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
                $cumulative_PI = $cumulative_PI + $PI_total;

                //$out_text = $out_text . "Seq. #: " . $sequence_number . "  probability = " . sci_notation($probability,2) . "  PI
= " . sci_notation($PI_total,4) . "<br>\n";

                $sequence_number = $sequence_number + 1;
        }

        // Now do the non-trip branch.

        $sequence_number = $sequence_number + 1;

        $probability = 1 - $cumulative_prob;

        // For now, model no impact on the 'continue as-is, no trip' branch.  In reality, there may be
        // impacts.  For example, if a steam generator tube is leaking, then we should have an increased
        // level of radioactivity in the secondary system, which gets worse as the plant operates.  Thus,
        // one should have a model for time-based impacts, potentially on four of the performance measures
        // (core damage is ignored here since no trip was seen).

        $theoutcome[1] = 0;    // No costs
        $theoutcome[2] = 0;    // No extra dose
        $theoutcome[3] = 0;    // No cleanup
        $theoutcome[4] = 0;    // No core damage
        $theoutcome[5] = 0;    // Minimal attention

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;
```

```
                //$out_text = $out_text . "Seq. #: " . $sequence_number . "  probability = " . sci_notation($probability,2) . "  PI = " .
sci_notation($PI_total,4) . "<br>\n";

                $total_sequences = $sequence_number;
                $cumulative_PI = $cumulative_PI + $PI_total;
                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

        }

        if ($key == 7) // repair at power
        {
                // Two states are used...first the repair state and then following repair.

                // There are three major parts to the generic tree structure.
                // 1=P(trip)  2=P(repair failed|no trip)  3=P(repair works|no trip)

                $sequence_number = 1;
                $cumulative_PI = 0;

                for ($k=1; $k<=3; $k++)
                {
                        $cumulative_prob = 0;

                        // Repair tripped the plant (if user indicates so).
                        // 1 Fix the problem after tripping the plant...$j=2
                        // 2 no trip, but no repair either...$j=1
                        // 3 no trip, repair works...$j=1 (initially) ,$j=2 (long term)
                        if ($k == 1) {$j = 2; $prob = $V[44][$u];}
                        elseif ($k == 2) {$j = 1; $prob = (1 - $V[44][$u]) * $V[45][$u];}
                        elseif ($k == 3) {$j = 2; $prob = (1 - $V[44][$u]) * (1 - $V[45][$u]);}

                        $state_number = state_value($plant_state_key[$j]);

                        // Loop through the IE categories while in this state

                        foreach($IE as $init_key => $init_name)
                        {
                                if (isset($IE_modification[$init_key][$state_number]) AND $j==1) { // State has an impact to the rate.
                                        $rt = $IE_modification[$init_key][$state_number];
                                } else { $rt = $IE_rate[$init_key][$state_number];}

                                $probability = $prob * (1-exp(- $rt * $V[1][$u]/8760)) * (1-$V[46][$u]);   // P() * P(trip) * P(no CD)

                                // Determine outcomes[DT seq. #][Performance Measure]
                                // First do the non-CD branch following a IE, but we need to get the right impacts for each IE.

                                $theoutcome = IE_disutility_outcome($init_key, $u);

                                $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                                $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                                $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                                $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                                $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

                                $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                                $cumulative_prob = $cumulative_prob + $probability;
                                $cumulative_PI = $cumulative_PI + $PI_total;
                                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
```

```
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

        // Now do the CD branch following a IE

        $sequence_number = $sequence_number + 1;

        $probability = $prob * (1 - exp(- $rt * $V[1][$u]/8760)) * $V[46][$u];     // P() * P(trip) * P(CD)

        $theoutcome = CD_disutility_outcome($u);

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $cumulative_prob = $cumulative_prob + $probability;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
        $cumulative_PI = $cumulative_PI + $PI_total;

        $sequence_number = $sequence_number + 1;
}

// Now do the non-trip branch.

$sequence_number = $sequence_number + 1;

$probability = 1 - $cumulative_prob;

if ($k == 1) {  // still have a trip
        $theoutcome[1] = continuous_utility(1, 24*$V[4][$u]);       // trip
        $theoutcome[2] = continuous_utility(2, 12 * 0.0001); // Assume 12 hours of cleanup in 0.0001 Sv/hr field
        $theoutcome[3] = worker_outcome(12);                 // Assume 12 hours of cleanup
        $theoutcome[4] = 0;                                  // No core damage
        $theoutcome[5] = discrete_utility(5, 1);             // Minimal attention
}
elseif ($k == 2) { // no trip
        $theoutcome[1] = continuous_utility(1, 24*$V[2][$u]);       // no trip, just hourly worker cost
        $theoutcome[2] = continuous_utility(2, 12 * 0.001); // Assume 2 hours of in 0.001 Sv/hr field
        $theoutcome[3] = worker_outcome(2);                  // Assume 2 hours of worker time
        $theoutcome[4] = 0;                                  // No core damage
        $theoutcome[5] = discrete_utility(5, 1);             // Minimal attention
}
elseif ($k == 3) {
        $theoutcome[1] = continuous_utility(1, 24*$V[2][$u]);       // no trip, just hourly worker cost
        $theoutcome[2] = continuous_utility(2, 12 * 0.001); // Assume 2 hours of in 0.001 Sv/hr field
        $theoutcome[3] = worker_outcome(2);                  // Assume 2 hours of worker time
        $theoutcome[4] = 0;                                  // No core damage
        $theoutcome[5] = 0;                                  // No attention
}

$PI_cost = $probability * $theoutcome[1] * $V[39][$u];
$PI_dose = $probability * $theoutcome[2] * $V[40][$u];
$PI_work = $probability * $theoutcome[3] * $V[41][$u];
$PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
$PI_attn = $probability * $theoutcome[5] * $V[43][$u];

$PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;
```

```
                $total_sequences = $sequence_number;
                $cumulative_PI = $cumulative_PI + $PI_total;
                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
        }
}

if ($key == 702)      // hot shutdown and repair
{
        // Two states are used...first the shutdown/repair state and then following repair.

        // There are two major parts to the generic tree structure.
        // 1=P(repair failed)  2=P(repair works)

        $sequence_number = 1;
        $cumulative_PI = 0;

        for ($k=1; $k<=2; $k++)
        {
                $cumulative_prob = 0;

                // Plant is shutdown for the repair.
                // Repair unsuccessful...$j=1
                // Repair works...$j=2
                if ($k == 1) {$j = 1; $prob = $V[45][$u];}
                elseif ($k == 2) {$j = 2; $prob = (1 - $V[45][$u]);}

                $state_number = state_value($plant_state_key[$j]);

                // Loop through the IE categories while in this state

                foreach($IE as $init_key => $init_name)
                {
                        if (isset($IE_modification[$init_key][$state_number]) AND $j==1) { // State has an impact to the rate.
                                $rt = $IE_modification[$init_key][$state_number];
                        } else { $rt = $IE_rate[$init_key][$state_number];}

                        $probability = $prob * (1-exp(- $rt * $V[1][$u]/8760)) * (1-$V[46][$u]);   // P() * P(trip) * P(no CD)

                        // Determine outcomes[DT seq. #][Performance Measure]
                        // First do the non-CD branch following a IE, but we need to get the right impacts for each IE.

                        $theoutcome = IE_disutility_outcome($init_key, $u);

                        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

                        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                        $cumulative_prob = $cumulative_prob + $probability;
                        $cumulative_PI = $cumulative_PI + $PI_total;
                        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

                        // Now do the CD branch following a IE

                        $sequence_number = $sequence_number + 1;
```

```
                $probability = $prob * (1 - exp(- $rt * $V[1][$u]/8760)) * $V[46][$u];     // P() * P(trip) * P(CD)

                $theoutcome = CD_disutility_outcome($u);

                $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

                $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                $cumulative_prob = $cumulative_prob + $probability;
                $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
                $cumulative_PI = $cumulative_PI + $PI_total;

                $sequence_number = $sequence_number + 1;
        }

        // Now do the non-trip branch.

        $sequence_number = $sequence_number + 1;

        $probability = 1 - $cumulative_prob;

        if ($k == 1) {  // repair unsuccessful
                $theoutcome[1] = continuous_utility(1, 24*$V[4][$u]);       // no trip, just hourly shutdown/repair cost
                $theoutcome[2] = continuous_utility(2, 12 * 0.001); // Assume 2 hours of in 0.001 Sv/hr field
                $theoutcome[3] = worker_outcome(12);                // Assume 2 hours of worker time
                $theoutcome[4] = 0;                                 // No core damage
                $theoutcome[5] = discrete_utility(5, 1);            // Minimal attention
        }
        elseif ($k == 2) { // repair ok
                $theoutcome[1] = continuous_utility(1, 24*$V[4][$u]);       // no trip, just hourly shutdown/repair cost
                $theoutcome[2] = continuous_utility(2, 12 * 0.001); // Assume 2 hours of in 0.001 Sv/hr field
                $theoutcome[3] = worker_outcome(12);                // Assume 2 hours of worker time
                $theoutcome[4] = 0;                                 // No core damage
                $theoutcome[5] = 0;                                 // Minimal attention
        }

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $total_sequences = $sequence_number;
        $cumulative_PI = $cumulative_PI + $PI_total;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
    }
}

if ($key == 10)       // reduce power
{
    // Only one state (1) is used.
```

```
$j = 1;

// Repair can not trip the plant since there is no repair.
// Repair can not be successful since there is no repair.

$state_number = state_value($plant_state_key[$j]);

$sequence_number = 1;

$cumulative_prob = 0;
$cumulative_PI = 0;

// Loop through the IE categories while in this state

foreach($IE as $init_key => $init_name)
{
        if (isset($IE_modification[$init_key][$state_number])) { // State has an impact to the rate.
                $rt = $IE_modification[$init_key][$state_number];
        } else { $rt = $IE_rate[$init_key][$state_number];}
        $probability = (1-exp(- $rt * $V[1][$u]/8760)) * (1-$V[46][$u]);   // P(trip) * P(no CD)

        // Determine outcomes[DT seq. #][Performance Measure]
        // Cost = 1  Dose = 2  Worker = 3  CD = 4  Attention = 5
        //
        // First do the non-CD branch following a IE, but we need to
        // get the right impacts for each IE.

        $theoutcome = IE_disutility_outcome($init_key, $u);
        $extra_cost = continuous_utility(1, 0.1*$V[1][$u]*$V[4][$u]); // 10% lost power generation.

        $PI_cost = $probability * ($theoutcome[1]+$extra_cost) * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $cumulative_prob = $cumulative_prob + $probability;
        $cumulative_PI = $cumulative_PI + $PI_total;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

        // Now do the CD branch following a IE

        $sequence_number = $sequence_number + 1;

        $probability = (1 - exp(- $rt * $V[1][$u]/8760)) * $V[46][$u];      // P(trip) * P(CD)

        $theoutcome = CD_disutility_outcome($u);
        $extra_cost = continuous_utility(1, 0.1*$V[1][$u]*$V[4][$u]); // 10% lost power generation.

        $PI_cost = $probability * ($theoutcome[1]+$extra_cost) * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $cumulative_prob = $cumulative_prob + $probability;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
```

```
                $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
                $cumulative_PI = $cumulative_PI + $PI_total;

                $sequence_number = $sequence_number + 1;
        }

        // Now do the non-trip branch.

        $sequence_number = $sequence_number + 1;

        $probability = 1 - $cumulative_prob;

        // For now, model no impact on the 'continue as-is, no trip' branch.  In reality, there may be
        // impacts.  For example, if a steam generator tube is leaking, then we should have an increased
        // level of radioactivity in the secondary system, which gets worse as the plant operates.  Thus,
        // one should have a model for time-based impacts, potentially on four of the performance measures
        // (core damage is ignored here since no trip was seen).

        $extra_cost = continuous_utility(1, 0.1*$V[1][$u]*$V[4][$u]); // 10% lost power generation.

        $theoutcome[1] = $extra_cost;         // lost power production
        $theoutcome[2] = 0;   // No extra dose
        $theoutcome[3] = 0;   // No cleanup
        $theoutcome[4] = 0;   // No core damage
        $theoutcome[5] = 0;   // Minimal attention

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $total_sequences = $sequence_number;
        $cumulative_PI = $cumulative_PI + $PI_total;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

}

if ($key == 711)      // go to D2
{
        // Two states are used...first the shutdown state and then following repair.

        // There are two major parts to the generic tree structure.
        // 1=P(repair failed)  2=P(repair works)

        $sequence_number = 1;
        $cumulative_PI = 0;

        for ($k=1; $k<=2; $k++)
        {
                $cumulative_prob = 0;

                // Plant is shutdown for the repair.
                // Repair unsuccessful...$j=1
                // Repair works...$j=2
                if ($k == 1) {$j = 1; $prob = $V[45][$u];}
                elseif ($k == 2) {$j = 2; $prob = (1 - $V[45][$u]);}
```

```
$state_number = state_value($plant_state_key[$j]);

// Loop through the IE categories while in this state

foreach($IE as $init_key => $init_name)
{
        if (isset($IE_modification[$init_key][$state_number]) AND $j==1) { // State has an impact to the rate.
                $rt = $IE_modification[$init_key][$state_number];
        } else { $rt = $IE_rate[$init_key][$state_number];}

        $probability = $prob * (1-exp(- $rt * $V[1][$u]/8760)) * (1-$V[46][$u]);   // P() * P(trip) * P(no CD)

        // Determine outcomes[DT seq. #][Performance Measure]
        // First do the non-CD branch following a IE, but we need to get the right impacts for each IE.

        $theoutcome = IE_disutility_outcome($init_key, $u);

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $cumulative_prob = $cumulative_prob + $probability;
        $cumulative_PI = $cumulative_PI + $PI_total;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;

        // Now do the CD branch following a IE

        $sequence_number = $sequence_number + 1;

        $probability = $prob * (1 - exp(- $rt * $V[1][$u]/8760)) * $V[46][$u];     // P() * P(trip) * P(CD)

        $theoutcome = CD_disutility_outcome($u);

        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        $cumulative_prob = $cumulative_prob + $probability;
        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
        $cumulative_PI = $cumulative_PI + $PI_total;

        $sequence_number = $sequence_number + 1;
}

// Now do the non-trip branch.

$sequence_number = $sequence_number + 1;

$probability = 1 - $cumulative_prob;
```

```
                        if ($k == 1) {  // repair unsuccessful
                                $theoutcome[1] = continuous_utility(1, 48*$V[4][$u]);       // no trip, just hourly shutdown/repair cost
                                $theoutcome[2] = continuous_utility(2, 24 * 0.0001); // Assume 2 hours of in 0.0001 Sv/hr field
                                $theoutcome[3] = worker_outcome(12);                 // Assume 2 hours of worker time
                                $theoutcome[4] = 0;                                  // No core damage
                                $theoutcome[5] = discrete_utility(5, 1);             // Minimal attention
                        }
                        elseif ($k == 2) { // repair ok
                                $theoutcome[1] = continuous_utility(1, 48*$V[4][$u]);       // no trip, just hourly shutdown/repair cost
                                $theoutcome[2] = continuous_utility(2, 24 * 0.0001); // Assume 2 hours of in 0.0001 Sv/hr field
                                $theoutcome[3] = worker_outcome(12);                 // Assume 2 hours of worker time
                                $theoutcome[4] = 0;                                  // No core damage
                                $theoutcome[5] = 0;                                  // Minimal attention
                        }

                        $PI_cost = $probability * $theoutcome[1] * $V[39][$u];
                        $PI_dose = $probability * $theoutcome[2] * $V[40][$u];
                        $PI_work = $probability * $theoutcome[3] * $V[41][$u];
                        $PI_cd   = $probability * $theoutcome[4] * $V[42][$u];
                        $PI_attn = $probability * $theoutcome[5] * $V[43][$u];

                        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

                        $total_sequences = $sequence_number;
                        $cumulative_PI = $cumulative_PI + $PI_total;
                        $cumulative_PI_cost = $cumulative_PI_cost + $PI_cost;
                        $cumulative_PI_dose = $cumulative_PI_dose + $PI_dose;
                        $cumulative_PI_work = $cumulative_PI_work + $PI_work;
                        $cumulative_PI_cd   = $cumulative_PI_cd + $PI_cd;
                        $cumulative_PI_attn = $cumulative_PI_attn + $PI_attn;
                }
        }
        return array($cumulative_PI, $cumulative_PI_cost, $cumulative_PI_dose, $cumulative_PI_work, $cumulative_PI_cd,
$cumulative_PI_attn);

}

?>
```

< module_do_simulation.php >

```
<?
// This page provides the functions which calculate the PI for the i'th decision alternative based upon simulation.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
//
// function weighted_PI ($jth_decision-->j'th decision from 0 to n-1 decisions)
//
// $jth_decision      The numerical order of the decisions (0, 1, 2, etc.)
//
// To call this function, the following global variables must be set.
//
// $V][1]             An array that stores ALL of the variable/parameters used in the analysis.
//
//
//

function weighted_PI ($jth_decision)
{
        // Declare any global variables.
        global $V;                    // ALL the parameters
        global $use_decision;         // ALL the applicable decisions, indexed by ID_key
        global $IE;
        global $TIME_TO_SHUTDOWN;
        global $INIT_PLANT_STATE;
        global $trip_array;
        global $repair_array;
        global $worker_array;
        global $dose_array;
        global $state1;
        global $state2;
        global $state3;

        // First, determine the decision ID_key since some of the data arrays use this key as an index (rather than 0, 1, 2, etc.)
        //
        // $decision_key = Decision key (1 = as is, 7 = repair at power, 9 = isolate, 10 = reduce power, 702 = hot shutdown, etc.

        $decision_key = $use_decision[$jth_decision];

        //
        // We begin the simulation at, of course, t = 0.  Also, the interval number, k, begins at 1.  And, the initial
        // plant state is indicated by the user.
        //

        $t = 0;
        $sum_costs = 0;                                                 // Cumulative sum of the cost we encounter during the simulation.
        $sum_dose = 0;                                          // Cumulative sum of the worker dose we encounter during the simulation.
        $sum_workers = 1;                                       // Cumulative sum of the worker impacts we encounter during the simulation.
        $sum_cd = 1;                                            // Core damage event (max. of one) we encounter during the simulation.
        $sum_attention = 1;                                     // Cumulative sum of external attention we encounter during the simulation.
        $IE_trip = FALSE;
        $core_damage = FALSE;
        $incident_fixed = FALSE;

        $current_state = state_value($INIT_PLANT_STATE);     // Stored as state ID_key, so convert to state 1, 2, 3, or 4.
        $current_state_duration = $V[(118 + ($jth_decision * 7))][1];

        //
        // We only allow a maximum of three interval
        //

        for ($k = 1; $k <= 3; $k++)
```

```
{

if (($current_state < 1) OR ($current_state > 4)) {die(" State problem in func_simulate_decision |". $current_state . "|");}

$current_state_duration = $V[(118 + ($k - 1) + ($jth_decision * 7))][1];

if ($current_state_duration <= 0) {break;}

//
// Determine the worker impacts as indicated by the user (if any)
//
$theindx = 121 + ($k - 1) + ($jth_decision * 7);
$sum_costs = $sum_costs + ($V[$theindx][1] * $V[2][1]);                    // Worker hours * cost/hour

$workerdose = $dose_array[$k][$jth_decision] * $V[$theindx][1];
$sum_dose = $sum_dose + $workerdose;

//
// The discrete disutilities are a little tricky since we are keeping track of where
// we are with respect to an ordinal scale (1, 2, 3, etc.)
//
$worker = worker_outcome($V[$theindx][1]);
if ($worker > $sum_workers) {$sum_workers = $worker;}

//
// First, determine if an initiating event will cause a transition prior to the end of the k'th interval.
//

foreach($IE as $key => $Value)
{
        $rate = IE_rate_lookup ($key, $current_state, $incident_fixed);

        //
        // From Equation (40) of my thesis.
        //
        if ($rate <= 0) { $sim_time = 1E31;} else {$sim_time = -log(RN(0)) / $rate;}

        if ($sim_time > $current_state_duration)
        {
                // The new state arrival is AFTER the end of the current state, so no IE upset this time through.
                //
                // If the user indicated, then we still may have an upset due to the operators.


                if ($trip_array[$k][$jth_decision])
                {
                        //
                        // Yes, we have the potential for a trip.
                        //

                        if (RN(0) < $V[44][1])
                        {
                                //
                                // Worker-caused trip, so determine the impact.
                                //

                                $theoutcome = IE_outcome(722);                        // Assume least impact IE

                                $sum_costs = $sum_costs + $theoutcome[1];
                                $sum_dose = $sum_dose + $theoutcome[2];
                                //
                                // The discrete disutilities are a little tricky since we are keeping track of where
                                // we are with respect to an ordinal scale (1, 2, 3, etc.)
                                //
                                if ($theoutcome[3] > $sum_workers) {$sum_workers = $theoutcome[3];}
                                if ($theoutcome[4] > $sum_cd) {$sum_cd = $theoutcome[4];}
```

```
                if ($theoutcome[5] > $sum_attention) {$sum_attention = $theoutcome[5];}
                $IE_trip = TRUE;
                $incident_fixed = TRUE;                     // Assume after an IE, the problem will be fixed.
                break;                                      // Assume only one trip per interval.
        }
}

//
// No, we do not have the potential of a worker caused trip.
//
// But, as long as we did not trip due to a worker, we need to worry about
// a successful repair (if user indicated to model this).  We also need to
// worry about what we are actually doing in this state, which is a function
// of the particular decision.  For example, if we shut down to fix the problem,
// then this impacts the cost.
//

if ($repair_array[$k][$jth_decision])
{
        if (RN(0) < $V[45][1]) {$incident_fixed = FALSE;} else {$incident_fixed = TRUE;}
}


if ($decision_key == 1)              // Decision to stay as-is, which could be at power or not depending on state.
{
        // Determine the actual outcome based upon the current state.
        //
        if ($current_state == 1)     // Assume that are (or will be shortly) producing power in this state.
        {
                //
                // No additional impacts...we already accounted for the worker hourly
                // costs, the worker risk, and the worker dose.  Further, since there
                // was no trip in this interval, there is no core damage.
                //
        } else                       // Assume that we are not producing power
        {
                $sum_costs = $sum_costs + $current_state_duration*$V[4][1]; // replacement power costs
        }

} elseif ($decision_key == 7) // repair at power
{
        // Determine the actual outcome based upon the current state.
        //
        if ($current_state == 1)     // Assume that are (or will be shortly) producing power in this state.
        {
                //
                // No additional impacts...
                //
        } else                       // Assume that we are not producing power
        {
                $sum_costs = $sum_costs + ($current_state_duration * $V[4][1]);            // replacement power costs
        }
} elseif ($decision_key == 10)       // reduce power
{
        // Determine the actual outcome based upon the current state.
        //
        if ($current_state == 1)     // Assume that are producing power at 80% of normal.
        {
                $sum_costs = $sum_costs + (0.15 * $current_state_duration*$V[4][1]);       // replacement power costs
        } else                       // Assume that we are not producing power
        {
                $sum_costs = $sum_costs + ($current_state_duration * $V[4][1]);            // replacement power costs
        }

} elseif ($decision_key == 9) // Isolate the problem
{
```

```
                        // Isolation does not impact power production (if at power).
                        //
                        // Determine the actual outcome based upon the current state.
                        //
                        if ($current_state == 1)      // Assume that are producing power in this state.
                        {
                                // No additional impacts...
                        } else                        // Assume that we are not producing power
                        {
                                $sum_costs = $sum_costs + ($current_state_duration * $V[4][1]);            // replacement power
}

                } elseif (($decision_key > 700) AND ($decision_key < 714) AND ($incident_fixed == FALSE)) // One of shutdown states.
                {
                        $sum_costs = $sum_costs + ($current_state_duration * $V[4][1]);            // replacement power costs

                }

        } else
        {
                // The new state arrival is BEFORE the end of the current state, so an upset condition will take place.
                //
                // Since we had an initiating event, we need to determine its impact.
                //
                // First, see if the IE leads to a core damage event via the CCDP calculation.
                //
                $ccdp = IE_ccdp_lookup($key, $current_state, $incident_fixed);
                if (RN(0) < $ccdp)
                {

                        // Core damage
                        $sum_costs = $sum_costs + $V[7][1];          // CD cost + replacement power + special costs
                        $sum_dose = $sum_dose + ($V[189][1]/2) * 0.001;     // Assume 7500 hours of cleanup in 0.001 Sv/hr field
                        $sum_workers = worker_outcome($V[189][1]);   // Assume 15000 hours of cleanup
                        $sum_cd = 2;                                 // Core damage
                        $sum_attention = 6;                          // CD level of attention
                        $core_damage = TRUE;
                } else
                {
                        // No core damage
                        $theoutcome = IE_outcome($key);

                        $sum_costs = $sum_costs + $theoutcome[1];
                        $sum_dose = $sum_dose + $theoutcome[2];
                        //
                        // The discrete disutilities are a little tricky since we are keeping track of where
                        // we are with respect to an ordinal scale (1, 2, 3, etc.)
                        //
                        if ($theoutcome[3] > $sum_workers) {$sum_workers = $theoutcome[3];}
                        if ($theoutcome[4] > $sum_cd) {$sum_cd = $theoutcome[4];}
                        if ($theoutcome[5] > $sum_attention) {$sum_attention = $theoutcome[5];}
                        $IE_trip = TRUE;
                        $incident_fixed = TRUE;                              // Assume after an IE, the problem will be fixed.
                }
                break;  // This ends the IE foreach loop...since we only need to have one trip per interval.
        }
} // End IE loop

$t = $t + $current_state_duration;

if ($core_damage == TRUE) { $t = ($TIME_TO_SHUTDOWN + 1)*24; break;}       // Jump to the end if we have a core damage.

if ($k == 1)
{
        $current_state = state_value($state2[$jth_decision]);              // Stored state ID_key, so convert state 1, 2, 3, or 4.
```

```
        } elseif ($k == 2)
        {
                $current_state = state_value($state3[$jth_decision]);              // Stored state ID_key, so convert state 1, 2, 3, or 4.
        }

                if ($current_state == 0)      // User has not identified any further states, so assume that we return to initial state.
                {
                        $current_state_duration = ($TIME_TO_SHUTDOWN * 24) - $t;
                        if ($current_state_duration <= 0)                    // Past the end of mission time so jump out of loop.
                        {
                                break;
                        }
                } else
                {
                        $current_state_duration = $V[(118 + $k + ($jth_decision * 7))][1];
                }


        } // End $k loop

        //
        // Once we finish the user-defined intervals, we need to make sure that we have reached the
        // end of the mission time (as defined by the time to shutdown).  Of course, if we have a
        // core damage we are done, but otherwise continue the simulation.
        //
        // Determine the length of time remaining.
        //

        $current_state_duration = ($TIME_TO_SHUTDOWN*24) - $t;

        if (($incident_fixed == TRUE) AND ($current_state_duration > 0))
        {
                // Back to the nominal state.

        } elseif (($incident_fixed == FALSE) AND ($current_state_duration > 0))
        {
                // Back to same state as t=0
        }


        // Determine the outcomes.

        $PI_cost = continuous_utility(1, $sum_costs) * $V[41][1];   // Cost Disutility * Cost Weight
        $PI_dose = continuous_utility(2, $sum_dose) * $V[42][1];    // Dose Disutility * Dose Weight
        $PI_work = discrete_utility(3, $sum_workers) * $V[41][1];
        $PI_cd   = discrete_utility(4, $sum_cd) * $V[42][1];
        $PI_attn = discrete_utility(5, $sum_attention) * $V[43][1];

        $PI_total = $PI_cost + $PI_dose + $PI_work + $PI_cd + $PI_attn;

        return array($PI_total, $PI_cost, $PI_dose, $PI_work, $PI_cd, $PI_attn);
}

function state_value ($state_key)
{
        if ($state_key == 724 OR $state_key==723 OR $state_key==725) {return (1);}
        if ($state_key == 726 OR $state_key==727 OR $state_key==728 OR $state_key==729) {return (2);}
        if ($state_key == 730 OR $state_key==731 OR $state_key==732) {return (3);}
        if ($state_key == 733 OR $state_key==734) {return (4);}
}

function IE_rate_lookup ($IE_key, $state, $incident_fixed)
{
        global $V;                    // ALL the parameters

        //   $IE[714] = "APRP - LOCAs";
```

```
        //   $IE[715] = "ATWS - Anticipated trans. w/o scram";
        //   $IE[716] = "PSF - Loss of heat sink";
        //   $IE[717] = "PSL - Loss of power";
        //   $IE[718] = "RTE - Secondary system rupture";
        //   $IE[719] = "RTGV - Steam generator tube rupture";
        //   $IE[720] = "RTV - Steam line break";
        //   $IE[721] = "TGTA - Secondary side transient";
        //   $IE[722] = "TRCP - Primary side transient";

        if ($IE_key == 714) {$theindex = 192; $nominalindex = 46;}
        elseif ($IE_key == 715) {$modindex = 196; $nominalindex = 50;}
        elseif ($IE_key == 716) {$modindex = 200; $nominalindex = 54;}
        elseif ($IE_key == 717) {$modindex = 204; $nominalindex = 58;}
        elseif ($IE_key == 718) {$modindex = 208; $nominalindex = 62;}
        elseif ($IE_key == 719) {$modindex = 212; $nominalindex = 66;}
        elseif ($IE_key == 720) {$modindex = 216; $nominalindex = 70;}
        elseif ($IE_key == 721) {$modindex = 220; $nominalindex = 74;}
        elseif ($IE_key == 722) {$modindex = 224; $nominalindex = 78;}
        else { die($IE_key . "| Error since we should not have any decision keys other than those in func_simulate_decision");}

        //
        // If there is a IE modification for this IE and state, AND the problem has not been fixed, then
        //  return the modified value.  Otherwise, return the nominal value.
        //
        $theidx = $modindex + ($state - 1);

        if (($V[$theidx][1] > 0) AND ($incident_fixed == FALSE))
        {
                return ($V[($modindex + ($state - 1))][1]/8760);     // return rate in per hour units
        } else
        {
                return ($V[($nominalindex + ($state - 1))][1]/8760); // return rate in per hour units
        }
}

function IE_ccdp_lookup ($IE_key, $state, $incident_fixed)
{
        global $V;                      // ALL the parameters

        //   $IE[714] = "APRP - LOCAs";
        //   $IE[715] = "ATWS - Anticipated trans. w/o scram";
        //   $IE[716] = "PSF - Loss of heat sink";
        //   $IE[717] = "PSL - Loss of power";
        //   $IE[718] = "RTE - Secondary system rupture";
        //   $IE[719] = "RTGV - Steam generator tube rupture";
        //   $IE[720] = "RTV - Steam line break";
        //   $IE[721] = "TGTA - Secondary side transient";
        //   $IE[722] = "TRCP - Primary side transient";
        //
        // Still need to add the 'modified' ccdp variables into $V[][].
        // For now, just use the nominal CCDP.

        if ($IE_key == 714) {$theindex = 82; $nominalindex = 82;}
        elseif ($IE_key == 715) {$modindex = 86; $nominalindex = 86;}
        elseif ($IE_key == 716) {$modindex = 90; $nominalindex = 90;}
        elseif ($IE_key == 717) {$modindex = 94; $nominalindex = 94;}
        elseif ($IE_key == 718) {$modindex = 98; $nominalindex = 98;}
        elseif ($IE_key == 719) {$modindex = 102; $nominalindex = 102;}
        elseif ($IE_key == 720) {$modindex = 106; $nominalindex = 106;}
        elseif ($IE_key == 721) {$modindex = 110; $nominalindex = 110;}
        elseif ($IE_key == 722) {$modindex = 114; $nominalindex = 114;}
        else { die($IE_key . "| Error we should not decision keys other than those in func_simulate_decision");}

        //
        // If there is a IE modification for this IE and state, AND the problem has not been fixed, then
        //  return the modified value.  Otherwise, return the nominal value.
```

```
        //
        $theidx = $modindex + ($state - 1);

        if (($V[$theidx][1] > 0) AND ($incident_fixed == FALSE))
        {
                return ($V[($modindex + ($state - 1))][1]);
        } else
        {
                return ($V[($nominalindex + ($state - 1))][1]);
        }
}

function IE_outcome ($init_key)
{
        global $V;                      // ALL the parameters

        if ($init_key == 714) { //   $IE[714] = "APRP - LOCAs"
                $out[1] = $V[12][1] * $V[4][1] + $V[21][1];  // replacement power + special costs
                $out[2] = ($V[180][1]/2) * 0.001;            // Assume 750 hours of cleanup in 0.001 Sv/hr field
                $out[3] = worker_outcome($V[180][1]);        // Assume 1500 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[30][1];                         // LOCA level of attention
        }
        elseif ($init_key == 715) { //   $IE[715] = "ATWS - Anticipated trans. w/o scram"
                $out[1] = $V[16][1] * $V[4][1] + $V[25][1];  // replacement power + special costs
                $out[2] = ($V[181][1]/2) * 0.001;            // Assume 250 hours of cleanup in 0.001 Sv/hr field
                $out[3] = worker_outcome($V[181][1]);        // Assume 500 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[34][1];                         // ATWS level of attention
        }
        elseif ($init_key == 716) { //   $IE[716] = "PSF - Loss of heat sink"
                $out[1] = $V[17][1] * $V[4][1] + $V[26][1];  // replacement power + special costs
                $out[2] = ($V[182][1]/2) * 0.0001;           // Assume 250 hours of cleanup in 0.0001 Sv/hr field
                $out[3] = worker_outcome($V[182][1]);                                // Assume 500 hours of cleanup
                $out[4] = 1;                                                         // No core damage
                $out[5] = $V[35][1];                         // HS level of attention
        }
        elseif ($init_key == 717) { //   $IE[717] = "PSL - Loss of power"
                $out[1] = $V[18][1] * $V[4][1] + $V[27][1];  // replacement power + special costs
                $out[2] = ($V[183][1]/2) * 0.0001;           // Assume 50 hours of cleanup in 0.0001 Sv/hr field
                $out[3] = worker_outcome($V[183][1]);        // Assume 100 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[36][1];                         // LOP level of attention
        }
        elseif ($init_key == 718) { //   $IE[718] = "RTE - Secondary system rupture"
                $out[1] = $V[13][1] * $V[4][1] + $V[22][1];  // replacement power + special costs
                $out[2] = ($V[184][1]/2) * 0.0001;           // Assume 750 hours of cleanup in 0.0001 Sv/hr field
                $out[3] = worker_outcome($V[184][1]);        // Assume 1500 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[31][1];                         // Sec level of attention
        }
        elseif ($init_key == 719) { //   $IE[719] = "RTGV - Steam generator tube rupture"
                $out[1] = $V[14][1] * $V[4][1] + $V[23][1];  // replacement power + special costs
                $out[2] = ($V[185][1]/2) * 0.001;            // Assume 750 hours of cleanup in 0.001 Sv/hr field
                $out[3] = worker_outcome($V[185][1]);        // Assume 1500 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[32][1];                         // SGTR level of attention
        }
        elseif ($init_key == 720) { //   $IE[720] = "RTV - Steam line break"
                $out[1] = $V[15][1] * $V[4][1] + $V[24][1];  // replacement power + special costs
                $out[2] = ($V[186][1]/2) * 0.0001;           // Assume 750 hours of cleanup in 0.0001 Sv/hr field
                $out[3] = worker_outcome($V[186][1]);        // Assume 1500 hours of cleanup
                $out[4] = 1;                                 // No core damage
                $out[5] = $V[33][1];                         // Steam level of attention
        }
        elseif ($init_key == 721) { //   $IE[721] = "TGTA - Secondary side transient"
```

```
		$out[1] = $V[19][1] * $V[4][1] + $V[28][1];  // replacement power + special costs
		$out[2] = ($V[187][1]/2) * 0.0001;           // Assume 12 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome($V[187][1]);        // Assume 24 hours of cleanup
		$out[4] = 1;                                 // No core damage
		$out[5] = $V[37][1];                         // SecTrans level of attention
	}
	elseif ($init_key == 722) { //   $IE[722] = "TRCP - Primary side transient"
		$out[1] = $V[20][1] * $V[4][1] + $V[29][1];  // replacement power + special costs
		$out[2] = ($V[188][1]/2) * 0.0001;           // Assume 12 hours of cleanup in 0.0001 Sv/hr field
		$out[3] = worker_outcome($V[188][1]);        // Assume 24 hours of cleanup
		$out[4] = 1;                                 // No core damage
		$out[5] = $V[38][1];                         // SecTrans level of attention
	}

	return $out;
}
function worker_outcome ($hours_of_work)
{
	global $V;                     // ALL the parameters
	//
	// This function determines the outcome for the worker performance measure using simulation.
	//
	// The input to the function is the total number of hours, and returned from the function is
	// an integer (scale value) from 1 to 6.  1 = no impact, 2 = minor injury, ..., 6 = 125 fatalities
	//
	if ($V[8][1] <= 0) {$accident_time = 1E31;} else {$accident_time = -log(RN(0)) / $V[8][1];}

	if ($accident_time > $hours_of_work)
	{
		// No worker incident
		return (1);
	} else
	{
		//print ($V[9][1] . "  A TIME= " . $accident_time . "<br>");
		// A worker incident, so do further simulation to see how bad it is...

		if (RN(0) < $V[9][1])
		{
			// Minor injury
			return (2);
		} elseif (RN(0) < $V[10][1])
		{
			// Major injury
			return (3);
		} elseif (RN(0) < $V[11][1])
		{
			// Fatality
			return (4);
		} elseif (RN(0) < ($V[11][1]/50))
		{
			// 10 Fatalities
			return (5);
		} elseif (RN(0) < ($V[11][1]/1000))
		{
			// 125 Fatalities
			return (6);
		} else
		{
			// No injury/fatality
			return (1);
		}
	}
}
?>
```

< module_edit_IE_impact.php >

```
<?
// This page allows the user to edit incident impacts.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// This page is typically called by the URL looking like:
//
//  "<a href='module_edit_IE_impact.php?key=$key" . "&state=1&edit=$edit'>" . $IE_rate[$key][1] . "</a>";
//
// Page variables
//
//   $key        = the ID_key for the initiator being modified
//
//   $edit       = the component key (if > 0) that caused the impact
//               = -1 if general IE impact
//               = -2 if general system impact (we should not call this page if edit = -2)
//
//   $state      = 1, 2, 3, or 4 (states A through D, respectively)
//
session_start();

$plant_state[1] = "A";
$plant_state[2] = "B";
$plant_state[3] = "C";
$plant_state[4] = "D";

if ($edit > 0)
{
        // Query the knowledge base to obtain the relevant COMPONENT.
        $connect = odbc_connect("incident_db", "", "");
        // query the node table for name
        $query = "SELECT * FROM Node WHERE (ID_Key = $edit)";
        // perform the query
        $result = odbc_exec($connect, $query);
        if ($result) {
          while (odbc_fetch_row($result)) {
            $component_description = odbc_result($result, "Description");
            $component_key = odbc_result($result, "ID_Key");
            $component[$component_key] = $component_description;
          }
        }
        else {
          $component[1]="none";
        }
        // close the dB connection
        odbc_close($connect);
}

// Call the prototype module that controls the initiator information.
require("module_initiators.php");


$page_name = "Incident Advisor - Initiator Impact";

$page_title = "Determine impacts to specific initiator rate";

$page_description = "Please indicate the impact on the '<b>" . ucwords($IE[$key]) . "</b>' initiating event for plant State
$plant_state[$state].  ";
$page_description = $page_description. "Recall that this impact is due to '<i>";

if ($edit > 0) {
        $page_description = $page_description . $component[$edit] . "</i>'.<br><br>\n";
```

```
}

if ($edit == -1) {
        $page_description = $page_description . "a general initiator impact</i>'.<br><br>\n";
}

$page_description = $page_description. "The nominal frequency (per year) of this initiator is " . $IE_rate[$key][$state] . "<br><br>\n";

$page_table = <<< START_HTML

<div align="center">
<form method=post action="process_changes.php">
<table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>
                <td align=center class=arial-large COLSPAN=5>
                        The NEW frequency (per year) is: <input type="text" name="the_new_value" size=15>
                </td>
        </tr>
        <tr>
                <td align=center class=arial-large COLSPAN=5>
                         
                </td>
        </tr>
        <tr>
                <td align=center class=arial-small COLSPAN=5>
                        <input type="hidden" name="key" value="$key">
                        <input type="hidden" name="state" value="$state">
                        <input type="hidden" name="edit" value="$edit">
                        <input type="hidden" name="return_URL" value="$return_URL">
                        <input type="hidden" name="change_type" value="1">
                        <input type="submit" value="Process"><br><br>
                        <i>(click the BACK button to cancel)</i>
                </td>
        </tr>
   </table>
   </form>
</div>
START_HTML;

// Call the prototype module that controls the HTML output.
require("analyze_incident_prototype.php");

?>
```

< module_initiators.php >

```
<?
//
// This module contains information related to use of the initiating events.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//

// Query the knowledge base to obtain all the relevant INITIATORS
$connect = odbc_connect("incident_db", "", "");
// query the table for name
$query = "SELECT * FROM Node WHERE (Type_Text = 15)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $IE_name = odbc_result($result, "Description");
    $IE_key = odbc_result($result, "ID_Key");
    $IE[$IE_key] = $IE_name;
  }
}
else {
  $IE[1]="none";
}
// close the dB connection
odbc_close($connect);


//
// Do the IEs (if the database ID_Key is modified then change array index below)
//
//   $IE[714] = "APRP - LOCAs";
//   $IE[715] = "ATWS - Anticipated trans. w/o scram";
//   $IE[716] = "PSF - Loss of heat sink";
//   $IE[717] = "PSL - Loss of power";
//   $IE[718] = "RTE - Secondary system rupture";
//   $IE[719] = "RTGV - Steam generator tube rupture";
//   $IE[720] = "RTV - Steam line break";
//   $IE[721] = "TGTA - Secondary side transient";
//   $IE[722] = "TRCP - Primary side transient";

//
// IE_rate[category][state, 1=A, 2=B, 3=C, 4=D]
//
// (units in per year)
//
$IE_rate[714][1] = "2.9E-3";
$IE_rate[714][2] = "2.4E-2";
$IE_rate[714][3] = "4.5E-1";
$IE_rate[714][4] = "3.7E-3";

$IE_rate[715][1] = "1.4E-6";
$IE_rate[715][2] = "0";
$IE_rate[715][3] = "0";
$IE_rate[715][4] = "0";

$IE_rate[716][1] = "1.3E-4";
$IE_rate[716][2] = "3.0E-2";
$IE_rate[716][3] = "3.7E-4";
$IE_rate[716][4] = "1.1E-4";

$IE_rate[717][1] = "6.7E-1";
$IE_rate[717][2] = "6.7E-1";
$IE_rate[717][3] = "6.7E-1";
```

```
$IE_rate[717][4] = "6.7E-1";

$IE_rate[718][1] = "1.3E-3";
$IE_rate[718][2] = "1.3E-3";
$IE_rate[718][3] = "0";
$IE_rate[718][4] = "0";

$IE_rate[719][1] = "1.6E-2";
$IE_rate[719][2] = "1.4E-2";
$IE_rate[719][3] = "6.2E-3";
$IE_rate[719][4] = "0";

$IE_rate[720][1] = "4.2E-3";
$IE_rate[720][2] = "1.0E-2";
$IE_rate[720][3] = "0";
$IE_rate[720][4] = "0";

$IE_rate[721][1] = "7.0E-1";
$IE_rate[721][2] = "7.0E-1";
$IE_rate[721][3] = "0";
$IE_rate[721][4] = "0";

$IE_rate[722][1] = "3.9E-2";
$IE_rate[722][2] = "3.6E+0";
$IE_rate[722][3] = "5.7E-2";
$IE_rate[722][4] = "1.5E-1";


// define needed functions
//
// This module contains the following functions:
//
// IE_impact         returns the frequency for the i'th IE given j component probabilities
//
//
// IE_impact returns a frequency value
// inputs are: $system, $plant_state, $component_probability, $duration
//
// where:
//        $initiator                attribute
//        ----------------          ------------------------------------
//                ?                 i'th IE identifier
//
//        $plant_state              values
//        ----------------------    ------------------------------------
//                1-4               Substate of the reactor (1=A, 2=B, 3=C, 4=D)
//
//        $component_probability    array of values
//        ----------------------    ------------------------------------
//                ?                 actual failure probability
//
//        NOTE:  The key of component_probability array is the component ID_key
//
//        $duration                 value
//        ----------------------    ------------------------------------
//                ?                 period of time of interest, in years
//                                  DEFAULT = 1
//

function IE_impact ($initiator, $plant_state, $component_probability='', $duration='1')
{
// The frequency impact is evaluated via a binary decision diagram (BDD) or fault tree logic (FTL).

        if ($initiator == 722)
        {
                if ($plant_state == 1)
```

```
                {
                // Primary transients are a function of several items.
                // Currently just model the pressure transducers (key=737, 738, 739, 740)

                // Determine the nominal probabibilities for each modeled component
                // PT failure rate = 1E-6/hr
                $c[737]= 1 - exp(- 1E-6 * ($duration * 8760));
                $c[738]= 1 - exp(- 1E-6 * ($duration * 8760));
                $c[739]= 1 - exp(- 1E-6 * ($duration * 8760));
                $c[740]= 1 - exp(- 1E-6 * ($duration * 8760));

                // See if the user defined a probability for a component (generally P=1 indicating failure)
                foreach ($component_probability as $key => $value) {$c[$key] = $value;}

                // Do the calculation

                $pt_ind = 1 - ((1 - $c[737]*$c[738]) * (1 - $c[737]*$c[739]) * (1 - $c[737]*$c[740]) * (1 - $c[738]*$c[739]) * (1 -
$c[738]*$c[740]) * (1 - $c[739]*$c[740]));

                $pt_ccf = 1 - ((1 - max($c[737], $c[738]) * $beta) * (1 - max($c[737], $c[739]) * $beta) * (1 - max($c[737], $c[740]) *
$beta) * (1 - max($c[738], $c[739]) * $beta) * (1 - max($c[738], $c[740]) * $beta) * (1 - max($c[739], $c[740]) * $beta));

                $pt_freq = $pt_ind + $pt_ccf - ($pt_ind * $pt_ccf);

                $non_pt_freq = 3.39E-2 * $duration;

                $frequency = $non_pt_freq + $pt_freq;

                return ($frequency);

                } else
                {
                        // Originally, I was going to return the default, but just return nothing if no change is made.
                        return ('');
                }
        }
        else
        {
                return ('');
        }


}

?>
```

< module_session_handler.php >

```
<?
// This page initializes and manages the session variables for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
//
// This prototype uses sessions.  Sessions are global variables that, once set, are automatically
// stored by the server and are available on any subsequent page.
//
session_start();

// The language key.
//if (!isset($_SESSION['l_key'])) {
//      $_SESSION['l_key']=1;
//      $lang_key = 1;
//}

// Component impacts (TRUE/FALSE).
//if (!isset($_SESSION['COMPONENT_IMPACT'])) {
//      $_SESSION['COMPONENT_IMPACT'] = FALSE;
//}

// GENERAL USE OF SESSION VARIABLES
// if ($variable from page or form submittal) {set session variable (could be different name) via $_SESSION[]}

// Languages
if ($lang_key) {$_SESSION['l_key'] = $lang_key;}

// Determine the page state vector.
// If variable CI is set (via FORM), then user indicated that incident involves component failures/degradations.
// If variable II is set, then user indicated that incident involves an actual initiating event.  Note
// that in this version of the prototype, actual initiating event cases are not addressed.
//
if ($CI) {$_SESSION['COMPONENT_IMPACT'] = TRUE;}
if ($II) {$_SESSION['INITIATOR_IMPACT'] = TRUE;}
if ($PS) {$_SESSION['INIT_PLANT_STATE'] = $PS;}
if ($TTS) {$_SESSION['TIME_TO_SHUTDOWN'] = $TTS;}
if ($AN) {$_SESSION['ANALYSIS_NAME'] = $AN;}
if ($AD) {$_SESSION['ANALYSIS_DESCRIPTION'] = $AD;}

//
// Initiator impacts (TRUE/FALSE).
//
//if (!isset($_SESSION['INITIATOR_IMPACT'])) {
//      $_SESSION['INITIATOR_IMPACT']=FALSE;
//}

// Form variable
if ($C) {$_SESSION['COMPONENT_STATE'] = $C; $COMPONENT_STATE = $C;}

// Form variable
if ($OTHER_C_IMPACTS_IE) {$_SESSION['OTHER_COMPONENT_IMPACTS_IE'] = TRUE; $OTHER_COMPONENT_IMPACTS_IE = TRUE;}

// Form variable
if ($OTHER_C_IMPACTS_SYSTEMS) {$_SESSION['OTHER_COMPONENT_IMPACTS_SYSTEMS'] = $OTHER_C_IMPACTS_SYSTEMS;}

// Form variable
if ($use_dec) {$_SESSION['use_decision'] = $use_dec;}

// Form variable
if ($s1) {$_SESSION['state1'] = $s1; $state1 = $s1;}

// Form variable
```

```
if ($d1) {$_SESSION['duration1'] = $d1; $duration1 = $d1;}

// Form variable
if ($s2) {$_SESSION['state2'] = $s2; $state2 = $s2;}

// Form variable
if ($d2) {$_SESSION['duration2'] = $d2; $duration2 = $d2;}

// Form variable
if ($s3) {$_SESSION['state3'] = $s3; $state3 = $s3;}

// Form variable
if ($d3) {$_SESSION['duration3'] = $d3; $duration3 = $d3;}

// Form variable
if ($trip1) {$_SESSION['allow_trip1'] = $trip1; $allow_trip1 = $trip1;}

// Form variable
if ($repair1) {$_SESSION['repair_failed1'] = $repair1; $repair_failed1 = $repair1;}

// Form variable
if ($worker_key1) {$_SESSION['worker_time_key1'] = $worker_key1; $worker_time_key1 = $worker_key1;}

// Form variable
if ($worker1) {$_SESSION['worker_time1'] = $worker1; $worker_time1 = $worker1;}

// Form variable
if ($workerdose1) {$_SESSION['worker_rad_dose1'] = $workerdose1; $worker_rad_dose1 = $workerdose1;}

// Form variable
if ($trip2) {$_SESSION['allow_trip2'] = $trip2; $allow_trip2 = $trip2;}

// Form variable
if ($repair2) {$_SESSION['repair_failed2'] = $repair2; $repair_failed2 = $repair2;}

// Form variable
if ($worker_key2) {$_SESSION['worker_time_key2'] = $worker_key2; $worker_time_key2 = $worker_key2;}

// Form variable
if ($worker2) {$_SESSION['worker_time2'] = $worker2; $worker_time2 = $worker2;}

// Form variable
if ($workerdose2) {$_SESSION['worker_rad_dose2'] = $workerdose2; $worker_rad_dose2 = $workerdose2;}

// Form variable
if ($trip3) {$_SESSION['allow_trip3'] = $trip3; $allow_trip3 = $trip3;}

// Form variable
if ($repair3) {$_SESSION['repair_failed3'] = $repair3; $repair_failed3 = $repair3;}

// Form variable
if ($worker_key3) {$_SESSION['worker_time_key3'] = $worker_key3; $worker_time_key3 = $worker_key3;}

// Form variable
if ($worker3) {$_SESSION['worker_time3'] = $worker3; $worker_time3 = $worker3;}

// Form variable
if ($workerdose3) {$_SESSION['worker_rad_dose3'] = $workerdose3; $worker_rad_dose3 = $workerdose3;}

?>
```

< module_state_information.php >

```
<?
// Declare the global variables.

global $duration1;
global $state1;
global $allow_trip1;
global $repair_failed1;
global $worker_time_key1;
global $worker_time1;

global $duration2;
global $state2;
global $allow_trip2;
global $repair_failed2;
global $worker_time_key2;
global $worker_time2;

global $duration3;
global $state3;
global $allow_trip3;
global $repair_failed3;
global $worker_time_key3;
global $worker_time3;

// First state
$time_in_state[1] = $duration1[$jth_decision];
$plant_state_key[1] = $state1[$jth_decision];

// See if we allow inadvertant trips.
if (isset($allow_trip1)){
        if (in_array($key, $allow_trip1)) {$allow_trip[1] = TRUE;} else {$allow_trip[1] = FALSE;}
} else {$allow_trip[1] = FALSE;}

// See if we allow failed repairs.
if (in_array($key, $repair_failed1)) {$allow_fail_repair[1] = TRUE;} else {$allow_fail_repair[1] = FALSE;}

// See if worker time is accounted for...
if (in_array($key, $worker_time_key1))
{
        $dec_key = current(array_keys($worker_time_key1, $key));
        $worker_time[1] = $worker_time1[$dec_key];
} else
{
        $worker_time[1] = 0;
}

$states_used[1] = TRUE;


if ($state2[$jth_decision] > 0)
{
        // Second state
        $time_in_state[2] = $duration2[$jth_decision];
        $plant_state_key[2] = $state2[$jth_decision];

        // See if we allow inadvertant trips.
        if (isset($allow_trip2)){
                if (in_array($key, $allow_trip2)) {$allow_trip[2] = TRUE;} else {$allow_trip[2] = FALSE;}
        } else {$allow_trip[2] = FALSE;}

        // See if we allow failed repairs.
        if (in_array($key, $repair_failed2)) {$allow_fail_repair[2] = TRUE;} else {$allow_fail_repair[2] = FALSE;}
        // See if worker time is accounted for...
```

```
        if (in_array($key, $worker_time_key2))
        {
                $dec_key = current(array_keys($worker_time_key2, $key));
                $worker_time[2] = $worker_time2[$dec_key];
        } else
        {
                $worker_time[2] = 0;
        }
        $states_used[2] = TRUE;
}

if ($state3[$jth_decision] > 0)
{
        // Third state
        $time_in_state[3] = $duration3[$jth_decision];
        $plant_state_key[3] = $state3[$jth_decision];

        // See if we allow inadvertant trips.
        if (isset($allow_trip3)){
                if (in_array($key, $allow_trip3)) {$allow_trip[3] = TRUE;} else {$allow_trip[3] = FALSE;}
        } else {$allow_trip[3] = FALSE;}

        // See if we allow failed repairs.
        if (in_array($key, $repair_failed3)) {$allow_fail_repair[3] = TRUE;} else {$allow_fail_repair[3] = FALSE;}
        // See if worker time is accounted for...
        if (in_array($key, $worker_time_key3))
        {
                $dec_key = current(array_keys($worker_time_key3, $key));
                $worker_time[3] = $worker_time3[$dec_key];
        } else
        {
                $worker_time[3] = 0;
        }
        $states_used[3] = TRUE;
}

?>
```

< module_step_0.php >

```
<?
//
// This module contains the HTML page information required if:
//
// $the_step = 0
//
$Page_ID = 2;
//
// Obtain textual information specific to this page ($Page_ID must be set).
//
require("get_resources.php");

$page_name = $resource[1];
$page_title = $resource[2];
$page_description = $resource[3];
$page_table = <<< START_HTML
   <table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>           <td colspan=9></td>   </tr>
        <tr>
                <td width=10%> <br></td><td width=10%> <br></td><td width=10%> <br></td>
                <td width=10%> <br></td><td bgcolor="#7777CD" align=center class=arial-med-light> $resource[4] </td>
                <td width=10%> <br></td><td width=10%> <br></td>
                <td width=10%> <br></td><td width=10% class=arial-very-small><form method=post action="analyze_incident.php"><input
type="hidden" name="the_step" value="1"><input type="submit" value="$resource[9]"></td></tr></form>
        <tr>
                <td colspan=9></td>
        </tr>
        <tr>
                <td width=10%> <br></td><td width=10%> <br></td>
                <td width=7%> <br></td><td bgcolor="#7777CD" align=center class=arial-med-light colspan=3>
                $resource[5] </td><td width=7%> <br></td><td width=10%> <br></td>
                <td width=10%> <br></td>
        </tr>
        <tr>   <td colspan=9></td>   </tr>
        <tr>
                <td width=10%> <br></td>
                <td width=10%> <br></td><td bgcolor="#7777CD" align=center class=arial-med-light colspan=5>
                $resource[6] </td>
                <td width=10%> <br></td><td width=10%> <br></td>
        </tr>
        <tr>
                <td colspan=9></td>
        </tr>
        <tr>
                <td width=10%> <br></td><td bgcolor="#7777CD" align=center class=arial-med-light colspan=7>
                $resource[7] </td><td width=10%> <br></td>
        </tr>
        <tr>
                <td colspan=9></td>
        </tr>
        <tr>
                <td bgcolor="#7777CD" align=center class=arial-med-light colspan=9>
                $resource[8] </td>
        </tr>
        <tr>   <td colspan=9></td>   </tr>
        <tr>   <td align=center colspan=9>   </td>
        </tr>
   </table>
START_HTML;
?>
```

< module_step_1.php >

```
<?
//
// This module contains the HTML page information required if:
//
// $the_step = 1
//

 $page_name = "Incident Advisor - Step I <br> ";
 $page_title = "Problem initialization (value tree and disutilities)";
 $page_description = "The decision analysis preference information for incident management has been predetermined.  This information is
embodied within the value tree structure, associated attribute weights, and attribute disutility functions.";
 unset($the_step_state);   // Not used in step 1

 $page_table = <<< START_HTML

   <table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr align=center class=arial-small>
                <td>
                        <OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0" WIDTH=400 HEIGHT=350>
                        <PARAM NAME=movie VALUE="value_tree_final.swf"><PARAM NAME=quality VALUE=high><PARAM NAME=bgcolor VALUE=#FFFFFF>
                        <EMBED src="value_tree_final.swf" quality=high bgcolor=#FFFFFF  WIDTH=400 HEIGHT=350 TYPE="application/x-shockwave-
flash" PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash">
                        </EMBED>
                        </OBJECT>

                </td>
        </tr>
        <tr align=center class=arial-med>
                <td>
                        <form method=post action="analyze_incident.php">
                        <input type="hidden" name="the_step" value="2">
                        <input type="submit" value="Next Step">
                        </form>
                </td>
        </tr>
   </table>
START_HTML;

?>
```

< module_step_2.php >

```
<?
//
// This module contains the HTML page information required if:
//
// $the_step = 2
//

// Query the knowledge base to obtain the relevant PLANT STATES.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 16)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $state_name = odbc_result($result, "Name");
    $state_key = odbc_result($result, "ID_Key");
    $plant_state[$state_key] = $state_name;
  }
}
else {
  $plant_state[1]="none";
}
// close the dB connection
odbc_close($connect);

 $page_name = "Incident Advisor - Step II <br> Incident Facts";

 if ($_SESSION['COMPONENT_IMPACT']) {

       // Call the prototype module that contains this page HTML output.
       include("module_step_2_components.php");

 } elseif ($_SESSION['INITIATOR_IMPACT']) {

       // Call the prototype module that contains this page HTML output.
       include("module_step_2_initiators.php");

 } else {
   $page_title = "Determine incident facts and boundary conditions";
   $page_description = "Now, we need to determine facts associated with the incident.  During this step, we will collect the preliminary
information related to the incident.  Note that additional information specific to the incident may need to be provided in a later step,
but this information will be collected as required.";
   $page_table = <<< START_HTML
   <form method=post action="analyze_incident.php">
   <table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
       <tr>
               <td align=left class=arial-small>
                       General Incident Identification<br><HR SIZE=1 WIDTH=200 ALIGN=LEFT NOSHADE>Analysis Name<br>
                       <INPUT TYPE="TEXT" VALUE="Analysis Name" NAME="AN" size="30" MAXLENGTH="60"><br>
                       Analysis Description<br>
                       <TEXTAREA NAME="AD" ROWS="4" COLS="30" WRAP>Description goes here!</TEXTAREA><br>
               </td>

               <td valign=top align=left class=arial-small>
                       General Incident Type<br><HR SIZE=1 WIDTH=200 ALIGN=LEFT NOSHADE>
                       <INPUT TYPE="CHECKBOX" NAME="CI" CHECKED>
                       Incident involves component failures or degradations? <br><br>
                       <INPUT TYPE="CHECKBOX" NAME="II">
                       Incident involves an initiating event? <br>
               </td>
       </tr>
```

```
        <tr>
                <td align=left class=arial-small COLSPAN=2>
                        Current Plant Status<br><HR SIZE=1 WIDTH=500 ALIGN=LEFT NOSHADE>State
                        <SELECT NAME="PS" size="1">
START_HTML;

        foreach ($plant_state as $key => $value)
        {
                $page_table = $page_table . "<option value='" . $key ."'>" . $plant_state[$key];
        }

   $page_table = $page_table . <<< START_HTML2

                        </SELECT>
                           Days until the next shutdown
                        <INPUT TYPE="TEXT" VALUE="100" NAME="TTS" size="6" MAXLENGTH="6"><br> 
                </td>
        </tr>

        <tr>
                <td align=center class=arial-small COLSPAN=2>
                        <input type="hidden" name="the_step" value="2">
                        <input type="submit" value="Next step">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;
 }

?>
```

< module_step_2_components.php >

```
<?
// This page handles component-related upsets for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// This module contains the HTML page information required if:
//
// $the_step = 2
//
//  AND
//
// User indicated incident involves components failures/degradations
//

// Query the knowledge base to obtain all the relevant COMPONENTS.
$connect = odbc_connect("incident_db", "", "");
// query the users table for name and surname
$query = "SELECT * FROM Node WHERE (Type_Text = 13) ORDER BY Description ";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $component_description = odbc_result($result, "Description");
    $component_key = odbc_result($result, "ID_Key");
    $component[$component_key] = $component_description;
  }
}
else {
  $component[1]="none";
}
// close the dB connection
odbc_close($connect);


$page_title = "Specify the component impacts";

$page_description = "We now need to collect information related to the component failures or degradations.";

$page_table = "<form method=post action=\"process_changes.php\"><table border=0 cellpadding=4 cellspacing=0 width=700 align=center><tr>";
$page_table = $page_table . "<td align=left class=arial-small COLSPAN=2>Select the failed or degraded component(s)<br>";
$page_table = $page_table . "<SELECT NAME=\"C[]\" MULTIPLE size=\"15\"><OPTION Value=\"0\">None";

foreach ( $component as $key => $desc_text) {
  $page_table = $page_table . "<OPTION Value=\"$key\"> $component[$key]";
}

$page_table = $page_table . <<< START_HTML
</SELECT>

                </td>
        </tr>

        <tr>
                <td valign=top align=left class=arial-small COLSPAN=2>
                         <br>
                        Incident includes components not on list above?<br><HR SIZE=1 WIDTH=400 ALIGN=LEFT NOSHADE>
                        <INPUT TYPE="CHECKBOX" NAME="OTHER_C_IMPACTS_IE">
                        Component(s) impacts initiating events?   
                        <INPUT TYPE="CHECKBOX" NAME="OTHER_C_IMPACTS_SYSTEMS">
                        Component(s) impacts safety systems? <br><br> 
                </td>
        </tr>
```

```
        <tr>
                <td align=center class=arial-small COLSPAN=2>
START_HTML;

//
// Finish the HTML, but check to see if we need to process initiator information.
//

if ($the_page_state["initiator"] == TRUE) {
// this needs to be updated to process initiators
$page_table2 = <<< START_HTML2
                        <input type="hidden" name="the_step" value="2">
                        <input type="submit" value="Next step">
                </td>
        </tr>
</table>
</form>
START_HTML2;

} else {
$page_table2 = <<< START_HTML2
                        <input type="hidden" name="change_type" value="2">
                        <input type="hidden" name="the_step" value="3">
                        <input type="hidden" name="sub_step" value="1">
                        <input type="submit" value="Next step"><br><br>
                </td>
        </tr>
</table>
</form>
START_HTML2;

}
$page_table = $page_table . $page_table2;

?>
```

< module_step_2_initiators.php >

```
//
// This module contains the HTML page information required if:
//
// $the_step = 2
// User indicated incident involves initiators
//

$page_title = "Specify the initiator impacts";
$page_description = "We now need to collect information related to the initiating events.";
$page_table = <<< START_HTML

<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>
                <td align=left class=arial-small>
                        General Incident Identification<br><HR SIZE=1 WIDTH=200 ALIGN=LEFT NOSHADE>Analysis Name<br>
                        <INPUT TYPE="TEXT" VALUE="" NAME="ANALYSIS_NAME" size="30" MAXLENGTH="60"><br>
                        Analysis Description<br>
                        <TEXTAREA NAME="ANALYSIS_DESCRIPTION" ROWS="4" COLS="30" WRAP></TEXTAREA><br>
                </td>

                <td valign=top align=left class=arial-small>
                        General Incident Type<br><HR SIZE=1 WIDTH=200 ALIGN=LEFT NOSHADE>
                        <INPUT TYPE="CHECKBOX" NAME="COMPONENT_IMPACT" CHECKED>
                        Incident involves component failures or degradations? <br><br>
                        <INPUT TYPE="CHECKBOX" NAME="INITIATOR_IMPACT">
                        Incident involves an initiating event? <br>
                </td>
        </tr>

        <tr>
                <td align=center class=arial-small COLSPAN=2>
                        <input type="hidden" name="the_step" value="3">
                        <input type="submit" value="Next step">
                </td>
        </tr>
</table>
</form>
START_HTML;
```

< module_step_3_1.php >

```
<?
// This page allows the user to specify component (or other) impacts for the incident decision advisor.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// This module contains the HTML page information required if:
//
// $the_step = 3
//

// Query the knowledge base to obtain all the relevant COMPONENTS.
$connect = odbc_connect("incident_db", "", "");
// query the users table for name and surname
$query = "SELECT * FROM Node WHERE (Type_Text = 13) ORDER BY Description ";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $component_description = odbc_result($result, "Description");
    $component_key = odbc_result($result, "ID_Key");
    $component[$component_key] = $component_description;
  }
}
else {
  $component[1]="none";
}
// close the dB connection
odbc_close($connect);


$page_name = "Incident Advisor - Step III <br> Map Incident Specifics";

$page_title = "Determine attributes that contribute to the incident";

$page_description = "Now, we need to determine impacts associated with the incident.  During this step, we will collect information that
will tie the current incident state to the decision analysis model.";
$page_description = $page_description . " Please specify all impacts to the items listed below prior to continuing to the next step.";

$page_table = "<form method=post action=\"analyze_incident.php\"><table border=0 cellpadding=4 cellspacing=0 width=650 align=center>";
$page_table = $page_table . "<tr><td align=left class=arial-small>";

//
// Show components that play a part of the analysis.
//

$temp_string = "";
if ($_SESSION['COMPONENT_STATE']) {
        $temp_string= "Edit the current state impact for:<br><br>";

        foreach($_SESSION['COMPONENT_STATE'] as $value) {
                $temp_string= $temp_string . "<a href='module_step_3_edit.php?edit=" . $value ."'><img src=\"images/edit_button.gif\"
width=21 height=17 border=0 alt=edit align=texttop>  ";
                $temp_string= $temp_string . $component[$value] . "</a><br>";
        }
}
if ($_SESSION['OTHER_COMPONENT_IMPACTS_IE']) {
        $temp_string= $temp_string . "<br><br>";
        $temp_string= $temp_string . "<a href='module_step_3_edit.php?edit=-1'><img src=\"images/edit_button.gif\" width=21 height=17
border=0 alt=edit align=texttop>";
        $temp_string= $temp_string . "  'Other component IE impacts.'</a><br>";
}

if ($_SESSION['OTHER_COMPONENT_IMPACTS_SYSTEMS']) {
```

```
        $temp_string= $temp_string . "<br><br>";
        $temp_string= $temp_string . "<a href='module_step_3_edit.php?edit=-2'><img src=\"images/edit_button.gif\" width=21 height=17
border=0 alt=edit align=texttop>";
        $temp_string= $temp_string . "  'Other component SYSTEM impacts.'</a><br>";
}

if (isset($IE_modification))
{
        if (count($IE_modfication) > 1)
        {
                $temp_string= $temp_string . "<br> A total of " . count($IE_modification).  " changes have been specified.";
        }
        else
        {
                $temp_string= $temp_string . "<br> A total of " . count($IE_modification).  " change has been specified.";
        }
}

//
// Check to make sure that at least one impact was indicated.
//
if ($temp_string == "") {
        $temp_string= "You must indicate at least one impact.  Please press the back button to modify selection.<br>";
}

//
// Continue the HTML.
//
$page_table = $page_table . $temp_string . <<< START_HTML2
                </td>

        </tr>

        <tr>
                <td align=center class=arial-small COLSPAN=2>
                        <input type="hidden" name="the_step" value="3">
                        <input type="hidden" name="sub_step" value="2">
                        <input type="submit" value="Next step">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;

?>
```

< module_step_3_2.php >

```
<?
// This page allows the user to delinate decision alternatives for incident impacts.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $the_step   = 3
//   $sub_step   = 2
//

$the_step = 3;
$sub_step = 2;

// Call the module that will extract decision alternatives from knowledge base.
include("module_decisions.php");

$page_name = "Incident Advisor - Step III <br> Map Incident Specifics";
$page_title = "Identify application decision alternatives";

$page_description = "We need to determine appropriate decision alternatives for the incident.  Alternatives already checked below indicate
a potential alternative based upon your earlier data inputs.";

$page_table = <<< START_HTML
<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=700 align=center>
        <tr><td align=left class=arial-large COLSPAN=2></td></tr>
START_HTML;
$temp_string = $temp_string . "<tr bgcolor='#DDDDDD'><td align=left class=arial-small>Use?</td>";
$temp_string = $temp_string . "<td align=center class=arial-small>Decision Alternative</td>";
$temp_string = $temp_string . "</tr>";
// Loop through the decisions
foreach($decision as $key => $value)
{
        $temp_string = $temp_string . "<tr> <td align=left class=arial-small>";
        if (isset($_SESSION['use_decision']))
        {
                if (in_array($key, $_SESSION['use_decision'])) {$checked = " CHECKED ";} else {$checked = "";}
        } else {
                $checked = "";
        }
        $temp_string = $temp_string . "<input type='checkbox' name='use_dec[]' value='" . $key . "'" . $checked . ">";
        $temp_string = $temp_string . "</td>";
        $temp_string = $temp_string . "<td align=left class=arial-small>" . $decision[$key] . "</td>";
        $temp_string = $temp_string . "</tr>\n";
}
// put a space in the table...
$temp_string = $temp_string . "<tr><td COLSPAN=2> </td></tr>";
// Continue the HTML.
$page_table = $page_table . $temp_string . <<< START_HTML2

        <tr>
                <td align=center class=arial-small COLSPAN=2>
                        <input type="hidden" name="the_step" value="3">
                        <input type="hidden" name="sub_step" value="3">
                        <input type="submit" value="Next Step"></td></tr></table></form>
START_HTML2;
?>
```

< module_step_3_3.php >

```
<?
// This page allows the user to delinate decision alternatives for incident impacts.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $the_step   = 3
//
//   $sub_step   = 3
//
//

$the_step = 3;
$sub_step = 3;


// Query the knowledge base to obtain the relevant DECISIONS.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 1)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $d_description = odbc_result($result, "Description");
    $d_key = odbc_result($result, "ID_Key");
    $decision[$d_key] = $d_description;
  }
}
else {
  $decision[1]="none";
}
// close the dB connection
odbc_close($connect);


// Query the knowledge base to obtain the relevant PLANT STATES.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 16)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $state_name = odbc_result($result, "Name");
    $state_key = odbc_result($result, "ID_Key");
    $plant_state[$state_key] = $state_name;
  }
}
else {
  $plant_state[1]="none";
}
// close the dB connection
odbc_close($connect);

// Call the prototype module that controls the initiator information.
//require("module_initiators.php");


$page_name = "Incident Advisor - Step III <br> Map Incident Specifics";

$page_title = "Indicate decision alternative details (1 of 2)";
```

```
$page_description = "We need to determine information for each decision alternative.  For each alternative, indicate the plant states and
durations that will be experienced if the decision is implemented.";
$page_description = $page_description . "<br><br><span class=arial-small>Note:  The advisor will assume (unless otherwise specified) that
any time following the decision durations below will be spent at power in a nominal state.</span>";

$page_table = <<< START_HTML
<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>
                <td align=left class=arial-large COLSPAN=6></td></tr>
START_HTML;

$temp_string = $temp_string . "<tr bgcolor='#DDDDDD'><td align=left class=arial-small>Applicable?</td>";
$temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=5>Decision Alternative</td>";
$temp_string = $temp_string . "</tr>";

//
// Loop through the identified decisions
//
foreach($_SESSION['use_decision'] as $key)
{
        // NOTE: the decisions are stored in the knowledge base.  If the keys incidated below change, then this page
        // may need to be modified accordingly.
        //
        // if $key = 1   --> continue as is
        // if $key = 7   --> repair at current power state
        // if $key = 702 --> A3 hot shutdown (node 725)
        if ($key == 7)
        {
                $first_duration = sprintf("%5d", 2*24);
                $first_state_key = $INIT_PLANT_STATE;
                $second_duration = sprintf("%5d", ($TIME_TO_SHUTDOWN*24)-$first_duration);
                $second_state_key = $INIT_PLANT_STATE;
                $third_duration = 0;
                $third_state_key = 0;
        } elseif ($key == 702)
        {
                $first_duration = sprintf("%5d", 2);
                $first_state_key = 725;
                $second_duration = sprintf("%5d", ($TIME_TO_SHUTDOWN*24)-$first_duration);
                $second_state_key = $INIT_PLANT_STATE;
                $third_duration = 0;
                $third_state_key = 0;
        } else
        {
                $first_duration = sprintf("%5d", $TIME_TO_SHUTDOWN*24);
                $first_state_key = $INIT_PLANT_STATE;
                $second_duration = 0;
                $second_state_key = 0;
                $third_duration = 0;
                $third_state_key = 0;
        }

        $temp_string = $temp_string . "<tr> <td align=left class=arial-small><input type='checkbox' name='use_dec[]' value='$key'
CHECKED></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=5>" . $decision[$key] . "</td>";
        $temp_string = $temp_string . "</tr>\n";

        $temp_string = $temp_string . "<tr>\n";

        // First state and duration (maximum of three)
        $temp_string = $temp_string . "<td align=left class=arial-small>State <select name='s1[]' size='1'>";
        foreach ($plant_state as $key => $value)
        {
                if ($key == $first_state_key)
```

```
                {
                        $temp_string = $temp_string . "<option value='" . $key ."' SELECTED>" . $plant_state[$key];
                } else
                {
                        $temp_string = $temp_string . "<option value='" . $key ."'>" . $plant_state[$key];
                }
        }
        $temp_string = $temp_string . "</select></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small>Duration (hours) <input type='text' name='d1[]' size='5' value='" .
$first_duration . "'></td>";

        // Second state and duration (maximum of three)
        $temp_string = $temp_string . "<td align=left class=arial-small>State <select name='s2[]' size='1'>";
        $temp_string = $temp_string . "<option value='0'>None";
        foreach ($plant_state as $key => $value)
        {
                if ($key == $second_state_key)
                {
                        $temp_string = $temp_string . "<option value='" . $key ."' SELECTED>" . $plant_state[$key];
                } else
                {
                        $temp_string = $temp_string . "<option value='" . $key ."'>" . $plant_state[$key];
                }
        }
        $temp_string = $temp_string . "</select></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small>Duration (hours) <input type='text' name='d2[]' size='5' value='" .
$second_duration . "'></td>";

        // Third state and duration (maximum of three)
        $temp_string = $temp_string . "<td align=left class=arial-small>State <select name='s3[]' size='1'>";
        $temp_string = $temp_string . "<option value='0'>None";
        foreach ($plant_state as $key => $value)
        {
                $temp_string = $temp_string . "<option value='" . $key ."'>" . $plant_state[$key];
        }
        $temp_string = $temp_string . "</select></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small>Duration (hours) <input type='text' name='d3[]' size='5' value='" .
$third_duration . "'></td>";

        $temp_string = $temp_string . "</tr>\n";
        $temp_string = $temp_string . "<tr><td COLSPAN=6><hr></td></tr>\n";
}

// put a space in the table...
$temp_string = $temp_string . "<tr><td COLSPAN=6> </td></tr>";


//
// Continue the HTML.
//
$page_table = $page_table . $temp_string . <<< START_HTML2
        <tr>
                <td align=center class=arial-small COLSPAN=6>
                        <input type="hidden" name="the_step" value="3">
                        <input type="hidden" name="sub_step" value="4">
                        <input type="submit" value="Next Step">
                </td>
        </tr>
   </table></form>
START_HTML2;
?>
```

< module_step_3_4.php >

```
<?
// This page allows the user to indicate decision-specific information.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $the_step   = 3
//
//   $sub_step   = 4
//
//

$the_step = 3;
$sub_step = 4;


// Query the knowledge base to obtain the relevant DECISIONS.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 1)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $d_description = odbc_result($result, "Description");
    $d_key = odbc_result($result, "ID_Key");
    $decision[$d_key] = $d_description;
  }
}
else {
  $decision[1]="none";
}
// close the dB connection
odbc_close($connect);


// Query the knowledge base to obtain the relevant PLANT STATES.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 16)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $state_name = odbc_result($result, "Name");
    $state_key = odbc_result($result, "ID_Key");
    $plant_state[$state_key] = $state_name;
  }
}
else {
  $plant_state[1]="none";
}
// close the dB connection
odbc_close($connect);

// Call the prototype module that controls the initiator information.
//require("module_initiators.php");


$page_name = "Incident Advisor - Step III <br> Map Incident Specifics";

$page_title = "Indicate decision alternative details (2 of 2)";
```

```
$page_description = "We need to determine additional information for each decision alternative state.  For each state, indicate the plant
and worker impacts (if necessary).";
//$page_description = $page_description . "<br><br><span class=arial-small>Note:  The advisor will assume (unless otherwise specified)
that any time following the decision durations below will be spent at power in a nominal state.</span>";

$page_table = <<< START_HTML
<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=700 align=center>
        <tr>
                <td align=left class=arial-large COLSPAN=6></td></tr>
START_HTML;

//
// Loop through the identified decisions
//
$jth_decision = 0;

foreach($use_decision as $key)
{
        // NOTE: the decisions are stored in the knowledge base.  If the keys indicated below change, then this page
        // may need to be modified accordingly.
        //
        // if $key = 1   --> continue as is
        // if $key = 7   --> repair at current power state
        // if $key = 9   --> isolate problem
        // if $key = 10  --> reduce power
        // if $key = 702 --> A3 hot shutdown (node 725)
        if ($key == 7)
        {
                $include_trip_potential = TRUE;
                $include_worker_time = TRUE;
                $include_repair_success = TRUE;

        } elseif ($key == 1)
        {
                $include_trip_potential = FALSE;
                $include_worker_time = FALSE;
                $include_repair_success = FALSE;

        } elseif ($key == 9)
        {
                $include_trip_potential = TRUE;
                $include_worker_time = TRUE;
                $include_repair_success = TRUE;

        } elseif ($key == 10)
        {
                $include_trip_potential = FALSE;
                $include_worker_time = FALSE;
                $include_repair_success = FALSE;

        } else
        {
                $include_trip_potential = FALSE;
                $include_worker_time = TRUE;
                $include_repair_success = TRUE;
        }

        $temp_string = $temp_string . "<tr bgcolor='#DDDDDD'>";
        $temp_string = $temp_string . "<td align=left class=arial-med COLSPAN=3>D" . ($jth_decision + 1) . ": " . $decision[$key] .
"</td>";
        $temp_string = $temp_string . "</tr>\n";

        $temp_string = $temp_string . "<tr>\n";
```

```
	// First state
	$temp_string = $temp_string . "<td align=left class=arial-med>First State<br><span class=arial-small>";
	$temp_string = $temp_string . "(" . $duration1[$jth_decision] . " hours, " . $plant_state[$state1[$jth_decision]] .
")</span></td>";
	$temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";

	// Check to see if any additional information is needed
	if (!$include_trip_potential AND !$include_repair_success AND !$include_worker_time)
	{
		$temp_string = $temp_string . "No other inputs are needed.</td>";

	} else
	{

	if ($include_trip_potential)
	{
		$temp_string = $temp_string . "<input type='checkbox' name='trip1[]' value='" . $key . "' CHECKED>";
		$temp_string = $temp_string . "Include potential for decision to cause inadvertant trip?<br>";
	}
	if ($include_repair_success)
	{
		$temp_string = $temp_string . "<input type='checkbox' name='repair1[]' value='" . $key . "' CHECKED>";
		$temp_string = $temp_string . "Include potential for repair/modification to be unsuccessful?<br>";
	}
	if ($include_worker_time)
	{
		$temp_string = $temp_string . "<input type='hidden' name='worker_key1[]' value='" . $key . "'>";
		$temp_string = $temp_string . "<input type='text' name='worker1[]' size='5' value='0'>";
		$temp_string = $temp_string . "Worker hours involved in incident?   ";
		$temp_string = $temp_string . "<input type='text' name='workerdose1[]' size='5' value='0'>";
		$temp_string = $temp_string . "Dose rate seen by workers (mSv/hr)?<br>";
	}
	$temp_string = $temp_string . "</td></tr>";

	// Check to see if there is a second state for this decision.
	if ($state2[$jth_decision] > 0)
	{
		$temp_string = $temp_string . "<tr><td COLSPAN=3><hr></td></tr>\n";

		// Second state
		$temp_string = $temp_string . "<tr>\n";

		$temp_string = $temp_string . "<td align=left class=arial-med>Second State<br><span class=arial-small>";
		$temp_string = $temp_string . "(" . $duration2[$jth_decision] . " hours, " . $plant_state[$state2[$jth_decision]] .
")</span></td>";
		$temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";
		if ($include_trip_potential)
		{
			$temp_string = $temp_string . "<input type='checkbox' name='trip2[]' value='" . $key . "' CHECKED>";
			$temp_string = $temp_string . "Include potential for decision to cause inadvertant trip?<br>";
		}
		if ($include_repair_success)
		{
			$temp_string = $temp_string . "<input type='checkbox' name='repair2[]' value='" . $key . "' CHECKED>";
			$temp_string = $temp_string . "Include potential for repair/modification to be unsuccessful?<br>";
		}
		if ($include_worker_time)
		{
			$temp_string = $temp_string . "<input type='hidden' name='worker_key2[]' value='" . $key . "'>";
			$temp_string = $temp_string . "<input type='text' name='worker2[]' size='5' value='0'>";
			$temp_string = $temp_string . "Worker hours involved in incident?   ";
			$temp_string = $temp_string . "<input type='text' name='workerdose2[]' size='5' value='0'>";
			$temp_string = $temp_string . "Dose rate seen by workers (mSv/hr)?<br>";
		}
		$temp_string = $temp_string . "</td></tr>";
	}
```

```
        // Check to see if there is a third state for this decision.
        if ($_SESSION['state3[$jth_decision]'] > 0)
        {
                $temp_string = $temp_string . "<tr><td COLSPAN=3><hr></td></tr>\n";

                // Third state
                $temp_string = $temp_string . "<tr>\n";

                $temp_string = $temp_string . "<td align=left class=arial-med>Third State<br><span class=arial-small>";
                $temp_string = $temp_string . "(" . $duration3[$jth_decision] . " hours, " . $plant_state[$state3[$jth_decision]] .
")</span></td>";
                $temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";
                if ($include_trip_potential)
                {
                        $temp_string = $temp_string . "<input type='checkbox' name='trip3[]' value='" . $key . "' CHECKED>";
                        $temp_string = $temp_string . "Include potential for decision to cause inadvertant trip?<br>";
                }
                if ($include_repair_success)
                {
                        $temp_string = $temp_string . "<input type='checkbox' name='repair3[]' value='" . $key . "' CHECKED>";
                        $temp_string = $temp_string . "Include potential for repair/modification to be unsuccessful?<br>";
                }
                if ($include_worker_time)
                {
                        $temp_string = $temp_string . "<input type='hidden' name='worker_key3[]' value='" . $key . "'>";
                        $temp_string = $temp_string . "<input type='text' name='worker3[]' size='5' value='0'>";
                        $temp_string = $temp_string . "Worker hours involved in incident?   ";
                        $temp_string = $temp_string . "<input type='text' name='workerdose3[]' size='5' value='0'>";
                        $temp_string = $temp_string . "Dose rate seen by workers (mSv/hr)?<br>";
                }
                $temp_string = $temp_string . "</td></tr>";
                $temp_string = $temp_string . "<tr><td COLSPAN=3><hr></td></tr>\n";
        }

        $temp_string = $temp_string . "</tr>\n";

        }

        $jth_decision = $jth_decision + 1;

}

// put a space in the table...
$temp_string = $temp_string . "<tr><td COLSPAN=3> </td></tr>";
//
// Continue the HTML.
//
$page_table = $page_table . $temp_string . <<< START_HTML2

        <tr>
                <td align=center class=arial-small COLSPAN=3>
                        <input type="hidden" name="the_step" value="4">
                        <input type="hidden" name="sub_step" value="1">
                        <input type="submit" value="Review Data Inputs">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;

?>
```

< module_step_3_edit.php >

```
<?
// This page allows the user to edit incident impacts.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $edit       = the component key (if > 0) that caused the impact
//               = -1 if general IE impact
//               = -2 if general system impact
//
//   $the_step   = 3
//
// Initialize and register session variables.
//
require("module_session_handler.php");

$the_step = 3;

if ($edit > 0)
{
        // Query the knowledge base to obtain the relevant COMPONENT.
        $connect = odbc_connect("incident_db", "", "");
        // query the node table for name
        $query = "SELECT * FROM Node WHERE (ID_Key = $edit)";
        // perform the query
        $result = odbc_exec($connect, $query);
        if ($result) {
          while (odbc_fetch_row($result)) {
            $component_description = odbc_result($result, "Description");
            $component_key = odbc_result($result, "ID_Key");
            $component[$component_key] = $component_description;
          }
        }
        else {
          $component[1]="none";
        }
        // close the dB connection
        odbc_close($connect);
}

// Query the knowledge base to obtain all the relevant SYSTEMS.
$connect = odbc_connect("incident_db", "", "");
// query the users table for name
$query = "SELECT * FROM Node WHERE (Type_Text = 14)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $system_description = odbc_result($result, "Description");
    $system_key = odbc_result($result, "ID_Key");
    $system[$system_key] = $system_description;
  }
}
else {
  $component[1]="none";
}
// close the dB connection
odbc_close($connect);


// Call the prototype module that controls the initiator information.
require("module_initiators.php");
```

```
$page_name = "Incident Advisor - Step III <br> Map Incident Specifics";

$page_title = "Determine impacts to initiator rates or system unavailability";

$page_description = "We need to determine impacts associated with the incident for identified components.";

$page_table = <<< START_HTML
<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>
                <td align=left class=arial-large COLSPAN=5>
START_HTML;

//
// Show component.
//

$temp_string = "";
if ($edit > 0) {
        $temp_string= $component[$edit] . "<br>";
}

if ($edit == -1) {
        $temp_string= "Other component IE impacts<br>";
}

if ($edit == -2) {
        $temp_string= "Other component system impacts<br>";
}

//
// Check to make sure that at least one impact was indicated.
//
if ($temp_string == "") {
        $temp_string= "You must indicate at least one impact.  Please press the back button to modify selection.<br>";
}


$temp_string = $temp_string . "</td></tr> <tr bgcolor='#DDDDDD'><td align=left class=arial-small>IE Category</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>State A frequency</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>State B frequency</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>State C frequency</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>State D frequency</td></tr>";

function sci_notation($x, $d=-1) {
    $minus=($x<0)?"-":"";
    $x=abs($x);
    $e=floor(($x!=0)?log10($x):0);
    $x*=pow(10,-$e);
    $fmt=($d>=0)?".".$d:"";
    $e=($e>=0)?"+".sprintf("%02d",$e):"-".sprintf("%02d",-$e);
    return sprintf("$minus%".$fmt."fE%s",$x,$e);
}


//
// Loop through the IEs
//
foreach($IE as $key => $value)
{

        $temp_string = $temp_string . "<tr> <td align=left class=arial-small>" . $IE[$key] . "</td>";

        // Step through the four plant states for each IE
```

```
        for ($i = 1; $i <= 4; $i++)
        {
                // See if the IE rate has been modified...
                if (isset($IE_modification[$key][$i]))
                {
                        $temp_string = $temp_string . "<td align=left class=arial-small><a href='module_edit_IE_impact.php?key=$key" .
"&state=" . $i . "&edit=$edit&return_URL=module_step_3_edit.php'>" . sci_notation($IE_modification[$key][$i], 1) . "</a><br><span
class='arial-verysmall'>(" . sci_notation($IE_rate[$key][$i],1) . ")</span></td>";
                } else
                {
                        $temp_string = $temp_string . "<td align=left class=arial-small><a href='module_edit_IE_impact.php?key=$key" .
"&state=" . $i . "&edit=$edit&return_URL=module_step_3_edit.php'>" . sci_notation($IE_rate[$key][$i],1) . "</a></td>";
                }
        }

        $temp_string = $temp_string . "</tr>\n";
}

// put a space in the table...
$temp_string = $temp_string . "<tr><td> </td></tr>";


$temp_string = $temp_string . "<tr bgcolor='#DDDDDD'><td align=left class=arial-small>IE Category</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>CCDP (State A)</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>CCDP (State B)</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>CCDP (State C)</td>";
$temp_string = $temp_string . "<td align=left class=arial-small>CCDP (State D)</td></tr>";

//
// Loop through the IEs
//
$temp = "tbd";
foreach($IE as $key => $value)
{

        $temp_string = $temp_string . "<tr> <td align=left class=arial-small>" . $IE[$key] . "</td>";
        $temp_string = $temp_string . "<td align=left class=arial-small><a href=''>" . $temp . "</a></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small><a href=''>" . $temp . "</a></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small><a href=''>" . $temp . "</a></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small><a href=''>" . $temp . "</a></td></tr>";

}
//
// Continue the HTML.
//
$page_table = $page_table . $temp_string . <<< START_HTML2

        <tr>
                <td align=center class=arial-small COLSPAN=5>
                        <input type="hidden" name="the_step" value="3">
                        <input type="hidden" name="sub_step" value="1">
                        <input type="submit" value="<< Back">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;

// Call the prototype module that controls the HTML output.
require("analyze_incident_prototype.php");
?>
```

< module_step_4_1.php >

```
<?
// This page allows the user to review the incident data that was entered.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $the_step   = 4
//
//   $sub_step   = 1
//
//

$the_step = 4;
$sub_step = 1;

// Query the knowledge base to obtain the relevant DECISIONS.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 1)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $d_description = odbc_result($result, "Description");
    $d_key = odbc_result($result, "ID_Key");
    $decision[$d_key] = $d_description;
  }
}
else {
  $decision[1]="none";
}
// close the dB connection
odbc_close($connect);


// Query the knowledge base to obtain the relevant PLANT STATES.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 16)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $state_name = odbc_result($result, "Name");
    $state_key = odbc_result($result, "ID_Key");
    $plant_state[$state_key] = $state_name;
  }
}
else {
  $plant_state[1]="none";
}
// close the dB connection
odbc_close($connect);

// Query the knowledge base to obtain all the relevant COMPONENTS.
$connect = odbc_connect("incident_db", "", "");
// query the users table for name and surname
$query = "SELECT * FROM Node WHERE (Type_Text = 13) ORDER BY Description ";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
```

```
    $component_description = odbc_result($result, "Description");
    $component_key = odbc_result($result, "ID_Key");
    $component[$component_key] = $component_description;
  }
}
else {
  $component[1]="none";
}
// close the dB connection
odbc_close($connect);


// Call the prototype module that controls the initiator information.
//require("module_initiators.php");


$page_name = "Incident Advisor - Step IV <br> Make Decision Model";

$page_title = "Construct the decision model for the incident";

$page_description = "Prior to the automated construction of the decision model, the input data (shown below) should be reviewed.";
$page_description = $page_description . "<br><br>If a portion of the data needs to be modified, either use the 'Back' button or click on
the edit button below for the applicable section.";

$page_table = <<< START_HTML
<form method=post action="analyze_incident.php">
<table border=0 cellpadding=4 cellspacing=0 width=650 align=center>
        <tr>
                <td align=left class=arial-large COLSPAN=3></td></tr>
START_HTML;

$temp_string= "";

//
// Show the incident type designations  (components, general IE, or general system)
//

$temp_string= $temp_string ."<tr><td COLSPAN=3>The incident involves impacts due to...<br><b>";

if ($COMPONENT_STATE) {
        foreach($COMPONENT_STATE as $value) {
                $temp_string= $temp_string . $component[$value] . "<br>\n";
        }
}
if ($OTHER_COMPONENT_IMPACTS_IE) {
        $temp_string= $temp_string . "  'Other component IE impacts.'<br>\n";
}

if ($OTHER_COMPONENT_IMPACTS_SYSTEMS) {
        $temp_string= $temp_string . "  'Other component SYSTEM impacts.'<br>\n";
}

$temp_string= $temp_string ."</b><hr><br></td></tr>";

//
// Loop through the identified decisions
//
$jth_decision = 0;

foreach($use_decision as $key)
{
        // NOTE: the decisions are stored in the knowledge base.  If the keys indicated below change, then this page
        // may need to be modified accordingly.
        //
        // if $key = 1   --> continue as is
        // if $key = 7   --> repair at current power state
```

```
        // if $key = 9   --> isolate problem
        // if $key = 10  --> reduce power
        // if $key = 702 --> A3 hot shutdown (node 725)
        $temp_string = $temp_string . "<tr bgcolor='#DDDDDD'>";
        $temp_string = $temp_string . "<td align=left class=arial-med COLSPAN=3>D" . ($jth_decision + 1) . ": " . $decision[$key] .
"</td>";
        $temp_string = $temp_string . "</tr>\n";

        $temp_string = $temp_string . "<tr>\n";

        // First state
        $temp_string = $temp_string . "<td align=left class=arial-med>First State<br><span class=arial-small>";
        $temp_string = $temp_string . "(" . $duration1[$jth_decision] . " hours, " . $plant_state[$state1[$jth_decision]] .
")</span></td>";
        $temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";

        // Check to see if any additional information is needed
        if (isset($allow_trip1)){
                if (in_array($key, $allow_trip1))
                {
                        $temp_string = $temp_string . "The potential for an inadvertant trip is included.<br>";
                }
        }
        if (in_array($key, $repair_failed1))
        {
                $temp_string = $temp_string . "The potential for unsuccessful repair/modification is included.<br>";
        }
        if (in_array($key, $worker_time_key1))
        {
                $dec_key = current(array_keys($worker_time_key1, $key));
                $temp_string = $temp_string . "Worker hours involved in incident = " . $worker_time1[$dec_key] . "<br>";
        }

        $temp_string = $temp_string . "</td></tr>";

        // Check to see if there is a second state for this decision.
        if ($state2[$jth_decision] > 0)
        {
                $temp_string = $temp_string . "<tr><td COLSPAN=3><hr></td></tr>\n";

                // Second state
                $temp_string = $temp_string . "<tr>\n";

                $temp_string = $temp_string . "<td align=left class=arial-med>Second State<br><span class=arial-small>";
                $temp_string = $temp_string . "(" . $duration2[$jth_decision] . " hours, " . $plant_state[$state2[$jth_decision]] .
")</span></td>";
                $temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";

                // Check to see if any additional information is needed

                if (isset($allow_trip2)){
                        if (in_array($key, $allow_trip2))
                        {
                                $temp_string = $temp_string . "The potential for an inadvertant trip is included.<br>";
                        }
                }
                if (in_array($key, $repair_failed2))
                {
                        $temp_string = $temp_string . "The potential for unsuccessful repair/modification is included.<br>";
                }
                if (in_array($key, $worker_time_key2))
                {
                        $dec_key = current(array_keys($worker_time_key2, $key));
                        $temp_string = $temp_string . "Worker hours involved in incident = " . $worker_time2[$dec_key] . "<br>";
                }
```

```
                $temp_string = $temp_string . "</td></tr>";
        }

        // Check to see if there is a third state for this decision.
        if ($state3[$jth_decision] > 0)
        {
                $temp_string = $temp_string . "<tr><td COLSPAN=3><hr></td></tr>\n";

                // Third state
                $temp_string = $temp_string . "<tr>\n";

                $temp_string = $temp_string . "<td align=left class=arial-med>Third State<br><span class=arial-small>";
                $temp_string = $temp_string . "(" . $duration3[$jth_decision] . " hours, " . $plant_state[$state3[$jth_decision]] .
")</span></td>";
                $temp_string = $temp_string . "<td align=left class=arial-small COLSPAN=2>";

                // Check to see if any additional information is needed

                if (isset($allow_trip3)){
                        if (in_array($key, $allow_trip3))
                        {
                                        $temp_string = $temp_string . "The potential for an inadvertant trip is included.<br>";
                        }
                }
                if (in_array($key, $repair_failed3))
                {
                        $temp_string = $temp_string . "The potential for unsuccessful repair/modification is included.<br>";
                }
                if (in_array($key, $worker_time_key3))
                {
                        $dec_key = current(array_keys($worker_time_key3, $key));
                        $temp_string = $temp_string . "Worker hours involved in incident = " . $worker_time3[$dec_key] . "<br>";
                }

                $temp_string = $temp_string . "</td></tr>";

        }

        $jth_decision = $jth_decision + 1;

}

// put a space in the table...
$temp_string = $temp_string . "<tr><td COLSPAN=3> </td></tr>";

//
// Continue the HTML.
//
$page_table = $page_table . $temp_string . <<< START_HTML2

        <tr>
                <td align=center class=arial-small COLSPAN=3>
                        <input type="hidden" name="the_step" value="5">
                        <input type="hidden" name="sub_step" value="1">
                        <input type="submit" value="Solve the Decision Model">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;
?>
```

< module_step_5_1.php >

```
<?
// This page makes the decision model and solves it.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Page variables
//
//   $the_step   = 5
//   $sub_step   = 1
//

$the_step = 4;
$sub_step = 1;

// Call the prototype module that are necessary for the page.
require("func_utility.php");
require("module_initiators.php");
require("module_variables.php");
require("module_decisions.php");
require("module_do_decisions.php");

function sci_notation($x, $d=-1) {
    $minus=($x<0)?"-":"";
    $x=abs($x);
    $e=floor(($x!=0)?log10($x):0);
    $x*=pow(10,-$e);
    $fmt=($d>=0)?".".$d:"";
    $e=($e>=0)?"+".sprintf("%02d",$e):"-".sprintf("%02d",-$e);
    return sprintf("$minus%".$fmt."fE%s",$x,$e);
}

//
// Loop through z number of calculations.  For the roll-back, the first iteration provides just
// the point estimate.  Following iterations provide sensitivity calculations for EACH of the
// important variables.
//
// This same approach will be used in a later page to perform the uncertainty analysis, where
// the variable vectors (indexed by z) will be filled with epistemic uncertainty values instead
// of the point estimate values.
//

// Total calculations are one nominal case and then two sensitivity calculations for each variable.
//
// First do just the nominal case.
//
// Set $u = 1 to indicate that we are going to use the nominal case of each variable.

$u = 1;
$jth_decision = 0;

//
// Loop through the identified decisions
//

foreach($use_decision as $key)
   {
       // Call the function that does all the work.  Store the results in $decision_PI, where:
       //
       // $decision_PI ==> Array[$I], where $I=0 ==> total weighted PI for this decision option
       //                                  $I=1 ==> cost weighted PI for this decision option
       //                                  $I=2 ==> dose weighted PI for this decision option
       //                                  $I=3 ==> worker safety weighted PI for this decision option
       //                                  $I=4 ==> core damage weighted PI for this decision option
```

```
        //                                  $I=5 ==> external attention weighted PI for this decision option
        //
        $decision_PI = weighted_PI($key, $u, $jth_decision);

        // Store the total results
//      if ($decision_PI[0]) {$results_total[$key] = $decision_PI[0];} else {$results_total[$key] = 1;}
        $results_total[$key] = $decision_PI[0];

        $jth_decision = $jth_decision + 1;

}

// Sort the decision PI from lowest (most preferred) to highest.
asort($results_total);

$pref = array_keys($results_total);
reset($pref);
$pref_dec = current($pref);

//print $pref_dec;
//exit;
//[use_decision] => Array ( [0] => 1 [1] => 10 [2] => 711 )

// Find the decision number for array lookup.
$j = 0;
foreach($use_decision as $key)
{
        If ($key == $pref_dec) {$jth_decision = $j;}
        $j = $j + 1;
}
// Calculate LOW sensitivity for each variable.
for ($z=1; $z <= $number_sens; $z++)
{
        $u = 1;
        // Switch to the low value of the z'th variable (store nominal in temp variable so it can be restored).
        $the_temp = $V[$z][1];
        $V[$z][1] = $V[$z][2]*.5;

        // Call the function that does all the work.  Store the results in $decision_PI, where:

        $decision_PI = weighted_PI($pref_dec, $u, $jth_decision);

        // Store the total results
        $results_low[$z] = $decision_PI[0];

        // Switch nominal back from the temp variable so it can be restored.
        $V[$z][1] = $the_temp;
}
// Store the sensitivity calcs.
$_SESSION['results_low'] = $results_low;
// Calculate HIGH sensitivity for each variable, but just for the preferential decision.
for ($z=1; $z <= $number_sens; $z++)
{
        $u = 1;
        // Switch to the high value of the z'th variable (store nominal in temp variable so it can be restored).
        $the_temp = $V[$z][1];
        $V[$z][1] = $V[$z][3]*1.5;

        $decision_PI = weighted_PI($pref_dec, $u, $jth_decision);

        // Store the total results
        $results_high[$z] = $decision_PI[0];

        // Switch nominal back from the temp variable so it can be restored.
        $V[$z][1] = $the_temp;
```

```
}
// Store the sensitivity calcs.
$_SESSION['results_high'] = $results_high;

// Show results.

$page_name = "Incident Advisor - Step V <br> Decision Results";

$page_title = "Display the results of the decision analysis";

$page_description = "The point estimate decision analysis results are shown below, sorted from most preferential to least preferential.";

$page_table = <<< START_HTML
<table border=0 cellpadding=4 cellspacing=0 width=700 align=center>
        <tr>
                <td align=left class=arial-large COLSPAN=3></td></tr>
START_HTML;
//
// Continue the HTML.
//
$out_text = $out_text . "<table border=0 cellpadding=4 cellspacing=0 width=700 align=center>";
$out_text = $out_text . "<tr bgcolor='#DDDDDD'><td align=center class=arial-med>Decision Alternative</td><td align=center class=arial-
med>PI</td></tr>\n";

foreach($results_total as $key => $value)
{
        $out_text = $out_text . "<tr><td>" . $decision[$key] . "</td>";
        $out_text = $out_text . "<td>" . sci_notation($value,1) . "</td></tr>\n";
}

$out_text = $out_text . "</table>";

$page_table = $page_table . "<tr><td align=left class=arial-large COLSPAN=3>" . $out_text . "</td></tr>";

$page_table = $page_table . <<< START_HTML2
        <tr>
                <td COLSPAN=3>
                 
                </td>
        </tr>
        <tr>
                <td align=center class=arial-small>
                        <form method=post action="module_step_5_2.php">
                        <input type="hidden" name="the_step" value="5">
                        <input type="hidden" name="sub_step" value="2">
                        <input type="submit" value="Sensitivity Analysis">
                </td>
                <td align=center class=arial-small>
                &lt;-- Choose an option --&gt;
                </td>
                <td align=center class=arial-small>
                        <form method=post action="module_step_5_3.php">
                        <input type="hidden" name="the_step" value="5">
                        <input type="hidden" name="sub_step" value="3">
                        <input type="submit" value="Uncertainty Analysis">
                </td>
        </tr>
   </table>
   </form>
START_HTML2;
?>
```

< module_step_5_2.php >

```
<?
// This page does the sensitivity output for the prototype.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Variables that are required to be passed to this module:
//
// $page_name              Text to show above description
// $results_low[]          Sensitivity Results (low)
// $results_high[]         Sensitivity Results (high)
session_start();

include("func_utility.php");
include("module_variables.php");

function sci_notation($x, $d=-1) {
    $minus=($x<0)?"-":"";
    $x=abs($x);
    $e=floor(($x!=0)?log10($x):0);
    $x*=pow(10,-$e);
    $fmt=($d>=0)?".".$d:"";
    $e=($e>=0)?"+".sprintf("%02d",$e):"-".sprintf("%02d",-$e);
    return sprintf("$minus%".$fmt."fE%s",$x,$e);
}

$the_step = 5;

// Defined needed variables.
//

$page_name = "Tornado Diagram";

?>

<html>
<head>
<title>Incident Tornado Plot</title>
<LINK REL=stylesheet TYPE="text/css" HREF="http://psa.mit.edu/PSA.css">
</head>

<?
// now show the header (after /head)
require("page_header.php");
?>

<div align="center">

<br>

<table border="0" cellpadding="3" cellspacing="0" width=800 align=center>
        <tr bgcolor="#9999CD" align="center" class="arial-med-bold">
                <td>
                </td>
        </tr>
</table>

<br>

<table border="0" cellpadding="3" cellspacing="0" width=800 align=center>
        <tr align="center" class="arial-med-bold">
                <td>
                        <?
                        // Determine the 10 variables that have the most 'variation' from low to high.
```

```
                        $num_vars = count($results_low);

                        foreach ($results_low as $key => $value)
                        {
                                $diff[$key] = $results_high[$key] - $value;
                        }

                        // Sort the difference array from largest to smallest.
                        arsort($diff);
                        reset($diff);

                        $out_text = $out_text . "<table border=0 cellpadding=4 cellspacing=0 width=700 align=center>";
                        $out_text = $out_text . "<tr bgcolor='#DDDDDD'><td align=center class=arial-med>Variable</td><td align=center
class=arial-med>Low PI</td><td align=center class=arial-med>High PI</td></tr>\n";

                        for ($i=1; $i <= 10; $i++)
                        {
                                $out_text = $out_text . "<tr><td>" . $V_name[key($diff)] . "</td>";
                                $out_text = $out_text . "<td>" . sci_notation($results_low[key($diff)],1) . "</td>\n";
                                $out_text = $out_text . "<td>" . sci_notation($results_high[key($diff)],1) . "</td></tr>\n";
                                next($diff);
                        }

                        $out_text = $out_text . "</table>\n";

                        print $out_text;
                        ?>

                        <!-- <img src="tornado_makeplot.php" alt="tornado plot" width="700" height="600" border="0"> -->
                </td>
        </tr>
</table>

<br>

<? require("page_footer.php") ?>

</div>
</body>
</html>
```

< module_variables.php >

```
<?
// This page sets up the variables for the point estimate roll-back calculations.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//

// Query the knowledge base to obtain the relevant DECISIONS.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 1)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $d_description = odbc_result($result, "Description");
    $d_key = odbc_result($result, "ID_Key");
    $decision[$d_key] = $d_description;
  }
}
else {
  $decision[1]="none";
}
// close the dB connection
odbc_close($connect);


// Query the knowledge base to obtain the relevant PLANT STATES.
$connect = odbc_connect("incident_db", "", "");
// query the node table
$query = "SELECT * FROM Node WHERE (Type_Text = 16)";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $state_name = odbc_result($result, "Name");
    $state_key = odbc_result($result, "ID_Key");
    $plant_state[$state_key] = $state_name;
  }
}
else {
  $plant_state[1]="none";
}
// close the dB connection
odbc_close($connect);

// Query the knowledge base to obtain all the relevant COMPONENTS.
$connect = odbc_connect("incident_db", "", "");
// query the users table for name and surname
$query = "SELECT * FROM Node WHERE (Type_Text = 13) ORDER BY Description ";
// perform the query
$result = odbc_exec($connect, $query);
if ($result) {
  while (odbc_fetch_row($result)) {
    $component_description = odbc_result($result, "Description");
    $component_key = odbc_result($result, "ID_Key");
    $component[$component_key] = $component_description;
  }
}
else {
  $component[1]="none";
}
// close the dB connection
odbc_close($connect);
```

```
// Fixed Variable storage
//   z           Value
// -----   ----------------
//   1      nominal results
//   2      low value
//   3      high value
//
// $number_sens stores the total number of fixed variables that we will use for the sensitivity calculations.
//

$number_sens = 46;

for ($i = 1; $i <= $number_sens; $i++)
{
        if ($i==1) {
                $V_name[$i] = "time to next shutdown"; $V_units[$i] = "hours";
                $V[$i][1] = $TIME_TO_SHUTDOWN * 24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==2) {
                $V_name[$i] = "hourly labor costs"; $V_units[$i] = "euro/hour";
                $V[$i][1] = 1900; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==3) {
                $V_name[$i] = "cost multiplier for high dose rates"; $V_units[$i] = "x";
                $V[$i][1] = 2; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==4) {
                $V_name[$i] = "replacement power costs"; $V_units[$i] = "euro/hour";
                $V[$i][1] = 333000/24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==5) {
                $V_name[$i] = "fatality cost"; $V_units[$i] = "euro/fatality";
                $V[$i][1] = 2E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==6) {
                $V_name[$i] = "injury cost"; $V_units[$i] = "euro/injury";
                $V[$i][1] = 2E5; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==7) {
                $V_name[$i] = "core damage cost"; $V_units[$i] = "euro/event";
                $V[$i][1] = 7.5E8; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==8) {
                $V_name[$i] = "worker incident rate"; $V_units[$i] = "incident/hour";
                $V[$i][1] = 1.6E-4; $V[$i][2] = $V[$i][1] * 0.1; $V[$i][3] = $V[$i][1] * 5; }
        if ($i==9) {
                $V_name[$i] = "worker minor injury probability"; $V_units[$i] = "prob/incident";
                $V[$i][1] = 2.57E-1; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==10) {
                $V_name[$i] = "worker severe injury probability"; $V_units[$i] = "prob/incident";
                $V[$i][1] = 2.09E-1; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==11) {
                $V_name[$i] = "worker fatality probability"; $V_units[$i] = "prob/incident";
                $V[$i][1] = 5E-4; $V[$i][2] = $V[$i][1] * 0.1; $V[$i][3] = $V[$i][1] * 5; }
        if ($i==12) {
                $V_name[$i] = "LOCA plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 90*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==13) {
                $V_name[$i] = "secondary pipe break plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 90*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==14) {
                $V_name[$i] = "SGTR plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 90*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==15) {
                $V_name[$i] = "steam pipe break plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 90*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==16) {
                $V_name[$i] = "ATWS plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 30*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
        if ($i==17) {
                $V_name[$i] = "loss heat sink plant down time"; $V_units[$i] = "hours";
                $V[$i][1] = 30*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
```

```
if ($i==18) {
        $V_name[$i] = "loss electric power plant down time"; $V_units[$i] = "hours";
        $V[$i][1] = 8*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==19) {
        $V_name[$i] = "secondary transient plant down time"; $V_units[$i] = "hours";
        $V[$i][1] = 1*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==20) {
        $V_name[$i] = "primary transient plant down time"; $V_units[$i] = "hours";
        $V[$i][1] = 1*24; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==21) {
        $V_name[$i] = "LOCA special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 2E7; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==22) {
        $V_name[$i] = "secondary pipe break special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 5E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==23) {
        $V_name[$i] = "SGTR special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 5E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==24) {
        $V_name[$i] = "steam pipe break special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 5E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==25) {
        $V_name[$i] = "ATWS special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 2E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==26) {
        $V_name[$i] = "loss heat sink special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 2E6; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = $V[$i][1] * 1.5; }
if ($i==27) {
        $V_name[$i] = "loss electric power special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 0; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = 1E5; }
if ($i==28) {
        $V_name[$i] = "secondary transient special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 0; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = 1E5; }
if ($i==29) {
        $V_name[$i] = "primary transient special costs"; $V_units[$i] = "euro/incident";
        $V[$i][1] = 0; $V[$i][2] = $V[$i][1] * 0.5; $V[$i][3] = 1E5; }

if ($i==30) {
        $V_name[$i] = "LOCA external attention"; $V_units[$i] = "attention/incident";
        // The attention variables are ordinal scale based (1, 2, 3, etc.)
        // Assume 1=none, 2=report, 3=inspection, 4=intervention, 5=1yr shutdown, and 6=2yr shutdown
        $V[$i][1] = 5; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==31) {
        $V_name[$i] = "secondary pipe break external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 5; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==32) {
        $V_name[$i] = "SGTR external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 4; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==33) {
        $V_name[$i] = "steam pipe break external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 5; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==34) {
        $V_name[$i] = "ATWS external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 4; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==35) {
        $V_name[$i] = "loss heat sink external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 4; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==36) {
        $V_name[$i] = "loss electric power external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 3; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==37) {
        $V_name[$i] = "secondary transient external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 2; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
if ($i==38) {
        $V_name[$i] = "primary transient external attention"; $V_units[$i] = "attention/incident";
        $V[$i][1] = 2; $V[$i][2] = $V[$i][1] - 1; $V[$i][3] = $V[$i][1] + 1; }
```

```
        if ($i==39) {
                $V_name[$i] = measure_name(1) . " weight"; $V_units[$i] = "";
                $V[$i][1] = measure_weight(1); $V[$i][2] = $V[$i][1] * 0.95; $V[$i][3] = $V[$i][1] * 1.05; }
        if ($i==40) {
                $V_name[$i] = measure_name(2) . " weight"; $V_units[$i] = "";
                $V[$i][1] = measure_weight(2); $V[$i][2] = $V[$i][1] * 0.95; $V[$i][3] = $V[$i][1] * 1.05; }
        if ($i==41) {
                $V_name[$i] = measure_name(3) . " weight"; $V_units[$i] = "";
                $V[$i][1] = measure_weight(3); $V[$i][2] = $V[$i][1] * 0.95; $V[$i][3] = $V[$i][1] * 1.05; }
        if ($i==42) {
                $V_name[$i] = measure_name(4) . " weight"; $V_units[$i] = "";
                $V[$i][1] = measure_weight(4); $V[$i][2] = $V[$i][1] * 0.95; $V[$i][3] = $V[$i][1] * 1.05; }
        if ($i==43) {
                $V_name[$i] = measure_name(5) . " weight"; $V_units[$i] = "";
                // The fifth performance measure weight is found by 1-w(1)-w(2)-w(3)-w(4)
                $V[$i][1] = measure_weight(5); $V[$i][2] = 1 - $V[39][2] - $V[40][2] - $V[41][2] - $V[42][2]; $V[$i][3] = 1 - $V[39][3] -
$V[40][3] - $V[41][3] - $V[42][3]; }

        if ($i==44) {
                $V_name[$i] = "probability trip due to repair"; $V_units[$i] = "prob/repair";
                $V[$i][1] = 2E-3; $V[$i][2] = $V[$i][1] * 0.1; $V[$i][3] = $V[$i][1] * 5; }
        if ($i==45) {
                $V_name[$i] = "probability repair is fails"; $V_units[$i] = "prob/repair";
                $V[$i][1] = 1E-3; $V[$i][2] = $V[$i][1] * 0.1; $V[$i][3] = $V[$i][1] * 5; }
        if ($i==46) {
                $V_name[$i] = "ccdp"; $V_units[$i] = "prob/incident";
                $V[$i][1] = 1E-3; $V[$i][2] = $V[$i][1] * 0.1; $V[$i][3] = $V[$i][1] * 5; }

}

?>
```

< page_footer.php >

```
<?
//
//  This module controls what appears at the bottom of each page.
//

print "<center><span class='arial-small'>";

//
// Customize the bottom depending on the selected language.
//

if ($_SESSION['l_key'] == 1) {
        print "<a href='incident.php'>Incident home page</a>    ";
} elseif ($_SESSION['l_key'] ==2) {
        print "<a href='incident.php'>d'incident Home Page</a>    ";
}

// --- output the current date ---
print(date("F j, Y"));

print "</span></center>";
?>
```

< page_header.php >

```
<?
// Define site colors
$back_color = "#FFFFFF";
$TH_color = "#EFEFFF";
$TR_color = "#DFDFFF";
$T_color = "#CCCCFF";
?>
<BASE TARGET="_top">
<body bgcolor="#FAFAFA" text="#000000" link="#0033CC" vlink="#666633" alink="#CC3333" background="images/background_light.jpg">
<table border="0" cellpadding="0" cellspacing="0" width=600 align=center>
<tr><td width="20%">
   <?
     if ($the_step < 1){
       // Show the MIT logo if we are not going through the five steps of the advisor (i.e., performing an actual analysis)
       print ("<a href='incident.php'><img src='images/MIT_Logo_1.gif' width='56' height='48' border='0' align='left' alt='MIT'></a>");
     } else {
       print ("<img src='images/pyramid_" . $the_step . ".png' width='56' height='48' border='0' align='left' alt='Step " . $the_step .
"'>");
     }
       print "</td><td width='60%' align='left'><span class=arial-large>";
       print ($page_name);
       print "</span></td><td width='20%' align='right'><span class=arial-small>";

       if ($_SESSION['l_key'] == 1) {
              print "<a href='help.php?Page_ID=" . $Page_ID . "'>help</a> (ID=" . $Page_ID. ")</span></td></tr>";
       } elseif ($_SESSION['l_key'] ==2) {
              print "<a href='help.php?Page_ID=" . $Page_ID . "'>aide</a> (ID=" . $Page_ID. ")</span></td></tr>";
       }
   print ("<tr><td></td><td></td><td align=\"right\">");
   print "<form name='pulldown' action='" . $PHP_SELF . "'>";
   print "<input type='hidden' name='the_step' value='" . $the_step . "'>";
   print "<input type='hidden' name='sub_step' value='" . $sub_step . "'>";
   print "<span class=arial-verysmall>";
   print "<select name=\"lang_key\" onChange =\"javascript:document.pulldown.submit()\">";
   // Query for the language (French, English, Russian, etc.).
   $connect = odbc_connect("incident_db", "", "");
   // query the users table for name and surname
   $query = "SELECT * FROM Language";
   // perform the query
   $result = odbc_exec($connect, $query);
   if ($result) {
     while (odbc_fetch_row($result)) {
       $db_lang_name=odbc_result($result, "Name");
       $db_lang_key=odbc_result($result, "ID_Key");
       if ($_SESSION['l_key'] == $db_lang_key) {
         print "<option selected value=\"$db_lang_key\">$db_lang_name </option>";
       } else {
         print "<option value=\"$db_lang_key\">$db_lang_name </option>";
       }
     }
   }
   else {
     print "<option value=\"1\">English</option>";
   }
   // close the dB connection
   odbc_close($connect);
   print("</select></span></td></form></tr>");
   ?>
</table>
```

< process_changes.php >

```
<?
// This page allows manages the storage and modification of user variables.
// It can be called by any page, but does not display any information.  Once
// the processing is finished on this page, it transfers to a user specified
// page (given by $return_URL).
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// This page is typically called by the URL looking like:
//
//  "<a href='process_changes.php?variables_go_here...</a>";
//
// Page variables
//
//   $change_type = # indicating the routine to be called (see below).
//
//   $return_URL  = the page to go back to...
//
//
// Initialize and register session variables.
//
require("module_session_handler.php");

// 1 indicates user specified IE rate changes
if ($change_type == 1)
{
        //   $key        = the ID_key for the initiator being modified
        //
        //   $edit       = the component key (if > 0) that caused the impact
        //               = -1 if general IE impact
        //               = -2 if general system impact (we should not call this page if edit = -2)
        //
        //   $state      = 1, 2, 3, or 4 (states A through D, respectively)
        //
        // Store new frequency for IE
        header("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']."/incident/" . $return_URL . "?edit=" . $edit);
        session_start();
        $IE_modification[$key][$state] = $the_new_value;
        // Join the new data to the existing array
        $_SESSION['IE_modification'] = $IE_modification;
        exit;
}

// 2 indicates that we need to calculate IE rate changes due to specific COMPONENT(s) failure
if ($change_type == 2)
{
        // Decision = 1   --> continue as is
        // Decision = 7   --> repair at current power state
        // Decision = 9   --> isolate problem
        // Decision = 10  --> reduce power
        // Decision = 702 --> A3 hot shutdown (state node 725)
        //
        // Set default decisions to use based upon user input.
        if ($COMPONENT_STATE)
        {
                foreach($COMPONENT_STATE as $value)
                {
                        if ($value == 737 OR $value == 738 OR $value == 739 OR $value == 740) {$default_decisions = array(1,7,702);}
                }

                session_start();
                // Store the decision in the session variable
                $_SESSION['use_decision'] = $default_decisions;
```

```
        }

        // Calculate new frequencies based upon component failures (if needed).
        if ($COMPONENT_STATE)
        {
                include ("module_initiators.php");

                // Build up an array of the failed components (P=1)

                foreach($COMPONENT_STATE as $value)
                {
                        $component_array[$value] = 1;
                }

                // Loop through the IEs
                foreach($IE as $key => $value)
                {
                        // Step through the four plant states for each IE
                        for ($i = 1; $i <= 4; $i++)
                        {
                                //function IE_impact ($initiator, $plant_state, $component_probability, $duration)
                                $new_rate = IE_impact ($key, $i, $component_array, 1);
                                if ($new_rate)
                                {
                                        $IE_modification[$key][$i] = $new_rate;
                                }
                        }
                }

                // Store new frequencies for IE (if any)
                header("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']."/incident/analyze_incident.php?the_step=" . $the_step .
"&sub_step=" . $sub_step);
                if ($IE_modification)
                {
                        session_start();
                        // Store the new data in the session variable
                        $_SESSION['IE_modification'] = $IE_modification;
                }
                exit;

        }
        else
        {
                // No modifications are needed.
                header("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']."/incident/analyze_incident.php?the_step=" . $the_step .
"&sub_step=" . $sub_step);
                exit;
        }
}

// Jump to the appropriate page.
if ($return_URL == "")
{
        header("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']."/incident/analyze_incident.php");
        exit;
}
else
{
        header("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']."/incident/" . $return_URL);
        exit;
}
?>
```

< utility_makeplot.php >

```
<?
if ($attribute < 1)
{$attribute = 1;}

// load the utility functions
include("func_utility.php");
include("jpgraph.php");

// initialize variables
$plot_name = "Disutility Plot for " . measure_name($attribute);

if ($attribute < 3)
{
        include("jpgraph_log.php");
        include("jpgraph_scatter.php");
        // make array of the continuous disutility data
        $num_steps = 10;
        $max_scale = continuous_utility_max($attribute);
        for ($j = 0; $j < $num_steps; $j++) {
         $utility_x_data[$j] = continuous_utility_inv($attribute, $j/$num_steps, FALSE, 0);
         $utility_y_data[$j] = $j/$num_steps;
        }
} else
{
        include("jpgraph_line.php");
        // make array of the discrete disutility data
        if ($attribute == 3) {$num_steps = 6;} elseif ($attribute == 4) {$num_steps = 2;} else {$num_steps = 6;}

        for ($j = 0; $j < $num_steps; $j++) {
          $utility_x_data[$j] = discrete_utility_inv($attribute, discrete_utility($attribute, $j+1, 0), TRUE, 0);
          $utility_y_data[$j] = discrete_utility($attribute, $j+1, 0);
        }
}

// make the graph
$graph = new Graph(700,600,"utility_plot.jpg");
$graph->img->SetMargin(60,60,60,60);
$graph->img->SetAntiAliasing();

$graph->SetShadow();
$graph->title->Set($plot_name);
$graph->title->SetFont(FF_FONT1,FS_BOLD);

if ($attribute < 3)
{
        $graph->SetTickDensity(TICKD_VERYSPARSE, TICKD_VERYSPARSE);
        if ($log_x == 1) {
        $graph->SetScale("loglin");  // log scale on x axis
        $notused = array_shift($utility_x_data);  // remove the 'zero' point
        $notused = array_shift($utility_y_data);  // remove the 'zero' point
        }
        else {
        $graph->SetScale("linlin");  // linear scale on x axis
        }

        $graph->yaxis->title->Set("Disutility");
        $graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
        $graph->xaxis->title->Set(measure_name($attribute));
        $graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

        $plot = new ScatterPlot($utility_y_data,$utility_x_data);
        $plot->SetLinkPoints(true,"red",2);
}
```

```
else
{
        $graph->SetScale("textlin");  // Text scale on x axis
        $graph->xaxis->SetTickLabels($utility_x_data);
        $graph->yaxis->title->Set("Disutility");
        $graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
        $graph->xaxis->title->Set(measure_name($attribute));
        $graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

        $plot = new LinePlot($utility_y_data);
}

$plot->mark->SetType(MARK_FILLEDCIRCLE);
$plot->mark->SetFillColor("navy");
$plot->mark->SetWidth(3);

$graph->Add($plot);

$graph->Stroke();


//include ("../jpgraph.php");
//include ("../jpgraph_line.php");

//$labels = array("Oct 2000","Nov 2000","Dec 2000","Jan 2001","Feb 2001","Mar 2001","Apr 2001","May 2001");
//$datay = array(1.23,1.9,1.6,3.1,3.4,2.8,2.1,1.9);
//$graph = new Graph(300,250,"auto");
//$graph->img->SetMargin(40,40,40,80);
//$graph->img->SetAntiAliasing();
//$graph->SetScale("textlin");
//$graph->SetShadow();
//$graph->title->Set("Example slanted X-labels");
//$graph->title->SetFont(FF_VERDANA,FS_NORMAL,18);

//$graph->xaxis->SetFont(FF_ARIAL,FS_NORMAL,11);
//$graph->xaxis->SetTickLabels($labels);
//$graph->xaxis->SetLabelAngle(45);

//$p1 = new LinePlot($datay);
//$p1->mark->SetType(MARK_FILLEDCIRCLE);
//$p1->mark->SetFillColor("red");
//$p1->mark->SetWidth(4);
//$p1->SetColor("blue");
//$p1->SetCenter();
//$graph->Add($p1);

//$graph->Stroke();

?>
```

< utility_plot.php >

```
<?
// This page prints the disutility function specified by $attribute.
//
// (C) 2002 Curtis L. Smith.  All rights reserved.
//
// Variables that are required to be passed to this module:
//
// $page_name              Text to show for the text at top of page
// $page_title             Text to show above description
// $page_description       Test to show for the page description
// $the_step               Integer step number (1 to 5)
// $page_table             HTML for the table containing the data collection.
// $attribute              The disutility function to plot (1 = cost, 2 = dose, etc.)

// load the disutility functions
require("func_utility.php");

// Defined needed variables.
//
$Page_Title = "Utility Plot";
$the_step = 0;


?>

<html>
<head>
<title>Incident Utility Plot</title>
<LINK REL=stylesheet TYPE="text/css" HREF="http://psa.mit.edu/PSA.css">
</head>

<?
// now show the header (after /head)
require("page_header.php");
?>


<center>
 <span class="arial-large">
 <a href="value_tree_consistency.php"><img src="images/back_button.gif" width="25" height="25" border="0"
alt="back"> Back </a>   
 <a href="utility_show_data.php?attribute=<? print($attribute); ?>&log_x=<? print($log_x); ?>"><img src="images/forward_button.gif"
width="25" height="25" border="0" alt="back"> Show Data </a>
 </span>
</center>


<div align="center">

<br>

<table border="0" cellpadding="3" cellspacing="0" width=800 align=center>
 <tr bgcolor="#9999CD" align="center" class="arial-med-bold">
  <td>
    <form action="utility_plot.php" method="post">

    Utility =
    <select name="attribute" size="1">
    <?
    for ($j = 1; $j <= 6; $j++) {
      print "<option ";
      if ($attribute == $j) { print("SELECTED "); }
      print "value = $j>";
```

```
      print measure_name($j)."\n";
    }

    ?>
    </select>

       Logarithmic X?
    <select name="log_x" size="1">
    <option <? if ($log_x == 1) { print("SELECTED "); } ?>value="1">Yes
    <option <? if ($log_x == 0) { print("SELECTED "); } ?>value="0">No
    </select>

    <input type="submit" value="Update plot">

    </form>
  </td>
 </tr>
</table>

<br>

<table border="0" cellpadding="3" cellspacing="0" width=800 align=center>
 <tr align="center" class="arial-med-bold">
  <td>
    <img src="utility_makeplot.php?attribute=<? print($attribute); ?>&log_x=<? print($log_x); ?>" alt="utility plot" width="700"
height="600" border="0">
  </td>
 </tr>
</table>
<br>
<? require("page_footer.php") ?>
</div>
</body>
</html>
```

E. Source Code for Discrete Event Simulation of System Performance.

Below is the simulation code for a 1-of-3 system, with a component failure rate of 2×10^{-5}/hr, a mission time of 2400 hours. Two cases are modeled, the first without repair (no spares) and the second is with the potential for repair. The code is written in Microsoft's Visual Basic for Applications (VBA) and was developed within the Excel 97 environment. Following the code are numerical calculations comparing simulation against traditional static type logic models.

```
'
' System simulation (c) 2002 Curtis L. Smith.  All rights reserved.
'
Dim I As Integer, J As Integer, K As Integer, L As Integer
Dim M As Integer, NumberOfTrips As Integer
Dim Spares As Integer, Current_Spares As Integer
Dim Iterations As Integer, Time_Steps As Integer, NumPT As Integer
Dim Num_Failed As Integer, Failure_Criteria As Integer
Dim install_time As Double, PT_Train As Double, lambda As Double
Dim Sys_Un As Double, PT_Unavailability As Double
Dim PT_Pt_Unavailability As Double, E_Trips As Double
Dim failure_state As Boolean, Use_Spares As Boolean
Dim temp_string As String

Iterations = 20000  ' Number of iterations
Time_Steps = 2400   ' Check at one hour intervals
NumPT = 4           ' Total number of pressure transducers


ReDim PT_failure(NumPT, Time_Steps) As Boolean
ReDim Point_PT(NumPT, Iterations) As Boolean
ReDim PT(NumPT, Iterations) As Double
ReDim System(Time_Steps) As Boolean
ReDim Sys_Unavailability(Iterations) As Double
ReDim Shutdown_Times(Iterations) As Integer
ReDim Dead_Time(NumPT) As Integer

' Shuffle the random number generator
Randomize

' Initialize the time to install a spare
install_time = Range("B2").Value
' Initialize the failure rate for a pressure transducer
lambda = Range("B3").Value
' Initialize the system failure criteria for the system
Failure_Criteria = Range("B4").Value

' When      J = 1 --> No spares
'           J = 2 --> Unlimited spares
```

```
For J = 1 To 2      ' Different boundary condition cases.
    Randomize
    For L = 1 To Iterations
        ' Print out which iteration we are on...
        Cells(11, 1) = L
        DoEvents

        If J = 1 Then
            Spares = 0
            Use_Spares = False
        End If
        If J = 2 Then
            Spares = 1000
            Use_Spares = True
        End If
        Current_Spares = Spares

        ' Initialize loop variables
        For K = 1 To NumPT
            Dead_Time(K) = 0
        Next
        ' At t=0, transducer #4 is inoperable (but can be fixed if
        ' another one fails)
        Dead_Time(4) = Time_Steps + 1

        For I = 1 To Time_Steps
            Num_Failed = 0
            ' Check each PT
            For K = 1 To NumPT
                If I <= Dead_Time(K) Then
                    ' Check to see if we are in a dead time (under
                    ' spare, failed waiting for spare, etc.)
                    failure_state = True
                Else
                    If Rnd() <= lambda Then
                        failure_state = True
                        ' Check to see if spares are allowed, and if
                        ' any are available...
                        If Use_Spares = True Then
                            If Current_Spares >= 1 Then
                                ' A spare is available, so use it
                                Current_Spares = Current_Spares - 1
                                ' Figure in time to install spare
                                Dead_Time(K) = I + install_time
                            Else
                                ' Fail the component until the end of
                                ' mission since all spares are gone
                                Dead_Time(K) = Time_Steps + 1
                            End If
                        Else
                            ' Fail component until end of mission
                            ' since no spares to use.
                            Dead_Time(K) = Time_Steps + 1
                        End If
                    Else
                        failure_state = False
                    End If
```

```
        End If

        ' Set the PT to either up (FALSE) or down (TRUE)
        PT_failure(K, I) = failure_state
        If failure_state = True Then
            Num_Failed = Num_Failed + 1
        End If
    Next
    ' Determine if system is failed (via failure criteria)
        If Num_Failed >= Failure_Criteria Then
            System(I) = True
            ' If the system just failed, then plant is shut
            ' down and all components fixed
           If (System(I-1)=False) And (Use_Spares=True) Then
                For K = 1 To NumPT
                    Dead_Time(K) = I + install_time
                Next
            End If
        End If
Next I ' End time step loop for L'th iteration

' Now determine statistics
' Check each PT
For K = 1 To NumPT
    Point_PT(K, L) = PT_failure(K, Time_Steps)
    ' Store the end point (for the pt. wise calc)
    PT_Train = 0
    For I = 1 To Time_Steps
        If PT_failure(K, I) = True Then
            PT_Train = PT_Train + 1
        End If
        PT_failure(K, I) = False  ' Reset after counting
    Next
    PT(K, L) = PT_Train / Time_Steps
Next K

' Now determine system statistics
    Sys_Un = 0
    NumberOfTrips = 0
    For I = 1 To Time_Steps
        If System(I) = True Then
            Sys_Un = Sys_Un + 1
            ' Check to see if we are at the "end" of the
            ' failure time
            If I < Time_Steps Then
                If System(I + 1) = False Then
                    NumberOfTrips = NumberOfTrips + 1
                End If
            Else
                If System(I) = True Then
                    NumberOfTrips = NumberOfTrips + 1
                End If
            End If
        End If
        System(I) = False ' Reset after counting variable
    Next I
    Sys_Unavailability(L) = Sys_Un / Time_Steps
```

```
            Shutdown_Times(L) = NumberOfTrips
    Next L ' End iteration loop

    ' Now determine pressure transducer statistics
    For K = 1 To NumPT
        PT_Unavailability = 0
        PT_Pt_Unavailability = 0
        For L = 1 To Iterations
            PT_Unavailability = PT_Unavailability + PT(K, L)
            If Point_PT(K, L) = True Then
                PT_Pt_Unavailability = PT_Pt_Unavailability + 1
            End If
        Next
        PT_Unavailability = PT_Unavailability / Iterations
        PT_Pt_Unavailability = PT_Pt_Unavailability / Iterations
        ' Print out info for the PT trains
        Cells(4 + K, 3 + J) = PT_Unavailability
        Cells(9 + K, 3 + J) = PT_Pt_Unavailability
    Next
    ' Now determine system statistics
    Sys_Un = 0
    NumberOfTrips = 0
    For L = 1 To Iterations
        Sys_Un = Sys_Un + Sys_Unavailability(L)
        NumberOfTrips = NumberOfTrips + Shutdown_Times(L)
        ' Reset after counting
        Shutdown_Times(L) = 0
    Next
    Sys_Un = Sys_Un / Iterations
    E_Trips = NumberOfTrips / Iterations
    ' Print out info for the system
    Cells(15, 3 + J) = Sys_Un
    ' Print out expected number of trip (shutdowns)
    Cells(17, 3 + J) = E_Trips
Next
```

We ran several test cases (for a three train system) to determine the expected unreliability of the system over a mission time of 2400 hours. For these calculations, we varied the component failure rate to see the affect, if any, on the resulting system failure probability. Our results are shown in Figures E-1 through E-3. In these plots, we show four curves. The first curve is labeled "MCUB" and represents the minimal cut set upperbound from a static fault tree analysis. One should notice that in every case, the MCUB is a conservative calculation for the average unreliability. The second curve is labeled "exact" and represents the analytic evaluation via integration. The third curve is labeled "simulation" and represents the simulation approach described earlier in this Appendix. The fourth curve is labeled "DT simulation" and represents an improved simulation technique where the delta time (hence the DT) is varied slightly as a function of lambda. This variation in the delta time step allows us to reduce the calculation time and, at the same time, avoid numeric precision problems seen when the probability per time step becomes small. Both simulation approaches match the exact calculation very well, but the "DT simulation" is closer to the exact value in almost every case.

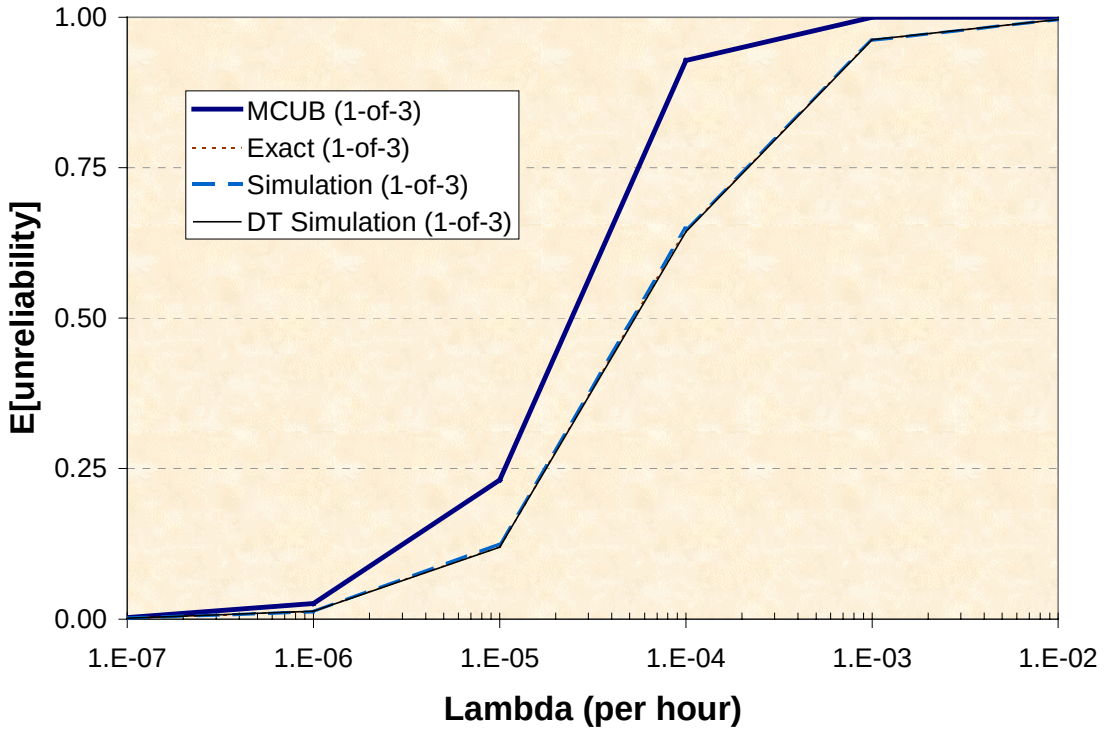

Figure E-1. Simulation results for a 1-of-3 system as a function of the failure rate lambda.

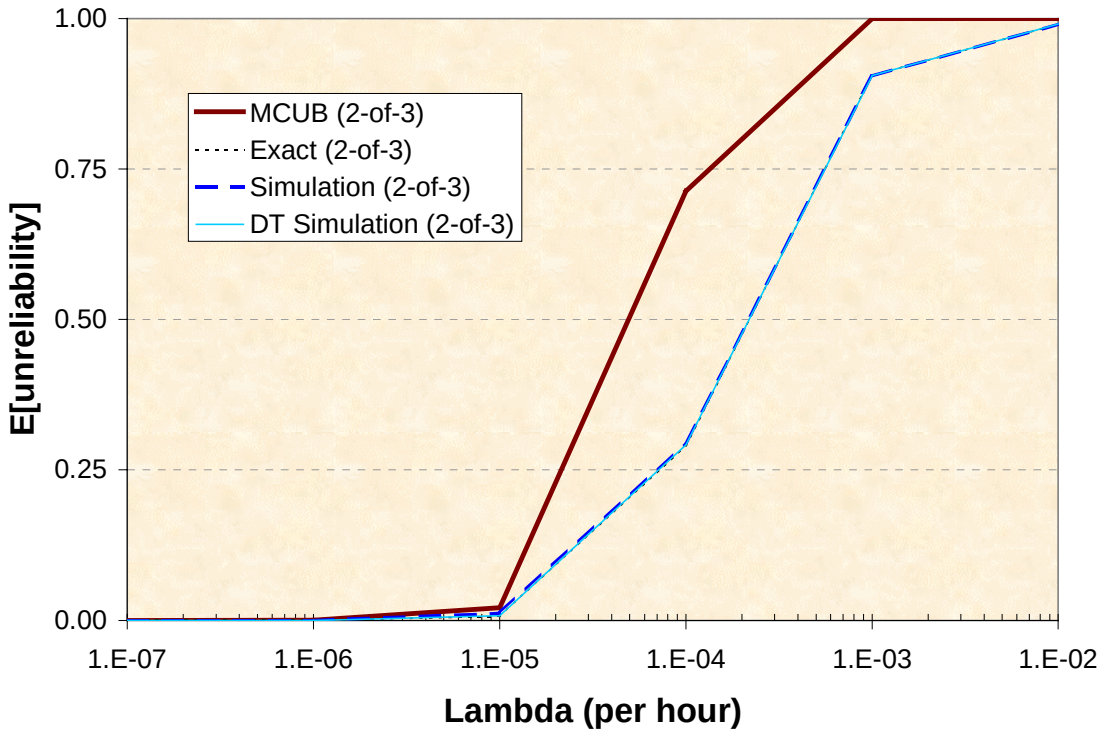

Figure E-2. Simulation results for a 2-of-3 system as a function of the failure rate lambda.

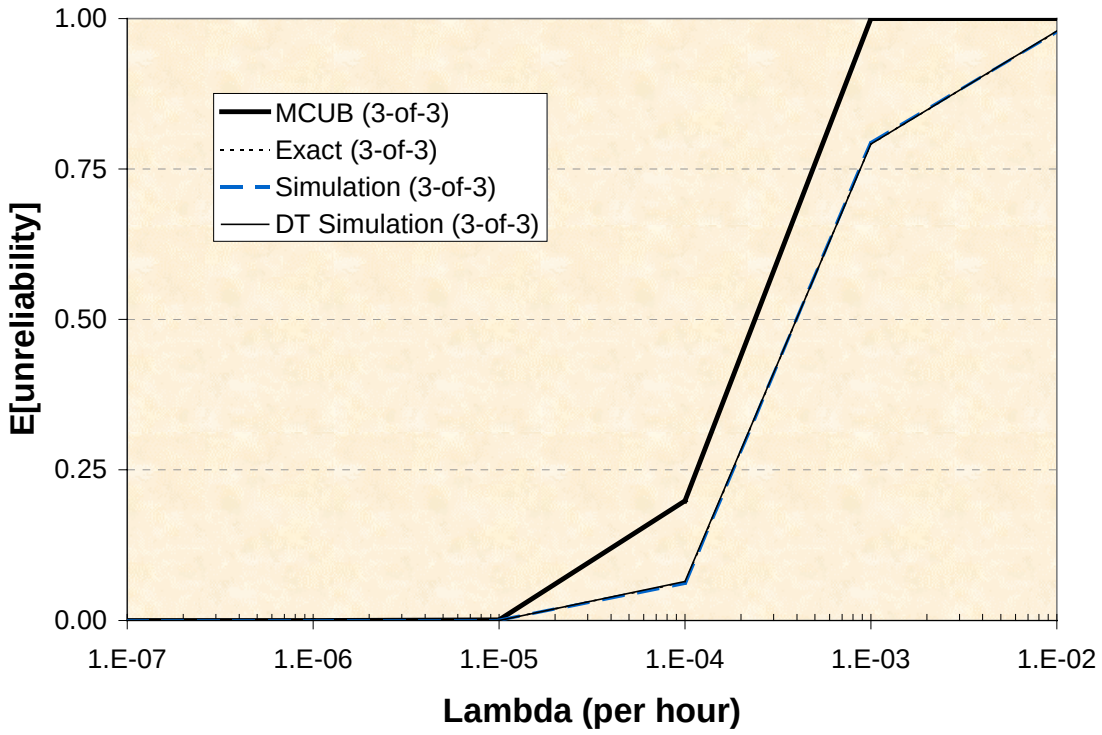

Figure E-3. Simulation results for a 3-of-3 system as a function of the failure rate lambda.