Risk-based System Refinement*

Victor L. Winter; Sandia National Laboratories; Albuquerque, NM

Ray S. Berg; Sandia National Laboratories; Albuquerque, NM

Larry J. Dalton; Sandia National Laboratories; Albuquerque, NM

Keywords: fault tree analysis, risk-based refinement, control theory

Abstract
When designing a high consequence system, considerable care should be taken to ensure that the system can not easily be placed into a high consequence failure state. A formal system design process should include a model that explicitly shows the complete state space of the system (including failure states) as well as those events (e.g., abnormal environmental conditions, component failures, etc.) that can cause a system to enter a failure state. In this paper we present such a model and formally develop a notion of risk-based refinement with respect to the model.

1 Overview
The goal of this paper is to develop a way to compare system designs and implementations from a risk-based perspective in a (predominantly) non-probabilistic manner. Our approach is essentially a fault tree analysis [1] [2] that is defined with respect to a specific modeling paradigm described in Section 2. Given a system design, the environments that the system can be exposed to are partitioned into three classes: normal, abnormal, and malevolent. In this context, system behavior is viewed as a (not necessarily alternating) two-player game between a properly functioning (i.e., correct) controller – player 1, and a set of postulated environmental conditions – player 2. In Section 5, a non-probabilistic notion of risk-based refinement is then developed for individual state transitions, which is then used in Section 6, as the basis for defining a system-level refinement-based comparison operator.

2 A Basic System Model
A system, $s$, can be abstractly modeled as a vector of monitored variables

$$m=(m_1, m_2, \ldots, m_n)$$

and a vector of controlled variables

$$c=(c_1, c_2, \ldots, c_k)$$

In our model, we require that the controlled variables be independent of one another in the sense that assigning a value to a variable $c_i$ should not restrict the value that can be assigned to a variable $c_j$ when $i \not< j$. Let $M$ and $C$ respectively denote the sets of all possible configurations of the monitored and controlled variables that are allowed by the environmental constraints. The set

$$S = \{(\vec{m}, \vec{c}) | \vec{m} \in M \land \vec{c} \in C\}$$

then describes the state space of the system.

**Observable State Space:** Given a system together with a behavior that we would like this system to satisfy, the **observable state space** of the system are those states that the controller must be able to distinguish and react to in order produce the desired system behavior. For example, if a controller needs to be able to distinguish distances on a robot arm to a resolution of 1 cm, then the sensors of the system must be able to directly (or indirectly) provide this level of resolution. Otherwise the capabilities of the system will not be sufficient to produce the desired behavior.

**Control State Space:** The **control state space** is the set of all possible values that the control vector (i.e., variables) can assume given the environmental constraints of the system.

---

1 This does not subsume the possibility of a safety violation.
Virtual Sensors: Let $spec$ be a problem specification. In general, $m$ will not directly provide enough information to distinguish all states in the observable state space of the system. In other words, in practice it is often the case that by using only the information provided by $m$ it will not be possible for a controller to achieve a behavior satisfying $spec$. However, by combining (1) knowledge of the system's initial state, (2) the information provided by $m$, together with (3) an historical trace of the system, the controller will have sufficient information to distinguish the states necessary to solve the problem described in $spec$.²

A more unified perspective of the observable state can be obtained by viewing the historical trace as a collection of virtual monitored variables, $(v_1, v_2, \ldots, v_j)$, which represent the cumulative information provided by the trace. Such a view extends the monitored variables from the definition in eq. 1 to

$\overrightarrow{m_v} = (m_1, m_2, \ldots, m_n, v_{n+1}, v_{n+2}, \ldots, v_{n+j}) \quad (4)$

This correspondingly modifies the definition of the system state space to:

$S_v = \{ (\overrightarrow{m_v}, \overrightarrow{c}) \mid \overrightarrow{m_v} \in M_v \land \overrightarrow{c} \in C \} \quad (5)$

where $M_v$ is observable state space of the system (i.e., $M_v = M$).

3 Risk

At this point we have briefly described a modeling paradigm in which systems are described by state machines. An important aspect of this paradigm is that control values belong strictly to the transition space and sensor values belong strictly to the state space. At all times, a system is in a particular observable state and its control variables are effecting a transition to another (possibly the same) observable state.

One thing that is worth noting is that ultimately, it is not the values of control variables themselves that bring about state changes but rather the set of physical components, $S_C$, that are (indirectly) controlled by these variables. By realizing this indirect relationship between the controller and system behavior a reasonable question to ask is: “Can a system be controlled via other means?”

For example, to what extent can the system's components be indirectly controlled by environmental factors that lie outside of the system's normal operating environment (e.g., power failure, noisy communications, etc.)? Furthermore, can direct external forces cause the system to enter into a failure state before it can be safely shut-down? A general question is: “What transitions can be brought about, either directly or indirectly, through environmental factors?”

A standard approach to answering this question is to postulate³ a set of environments, $S$, that one believes the system could be exposed to during its operation, and then to analyze observable states in the (highest level) system model with respect to $S$. In particular we are looking for elements in $S$ (e.g., environmental factors) that can (1) create a transition from one observable state to another that lies outside of the scope of the system’s control, or (2) affect the

---
² Note that we are assuming that a solution is possible given the controller’s capabilities.
³ We say “postulate” to emphasize the fact that environments are viewed as falling along a risk-based continuum and that we will ignore environments having sufficiently low risk.
computation of a control vector in such a way so as to cause the system to transition (under its own erroneous control) to an undesired state.

In this paper, we do not focus on how such a set is postulated, instead we focus on how to formally compare system implementations, in a qualitative manner, with respect to a given set $S_e$.

3.1 A Two Player Game

In the following sections we present a paradigm that allows one to model system behavior in the context of a postulated set of environments $S_e$. In particular, the transitions in a system model are annotated (in a hierarchical fashion) with fault tree-like conditions that need to hold in order for the transition to be taken.

Our goal is to specifically focus on modeling and reasoning about how $S_e$ can disrupt system behavior. A natural framework for doing this is to view the system as a “board” in a two player game. Given this perspective, the (system) controller and the environment $S_e$ can be viewed as players who are vying for control of the board. A conservative assumption is that $S_e$, when given a chance, will make the most direct moves (transitions) leading to a high-consequence failure state, while the controller will make correct moves to avoid entering hazardous regions of the state space.

Given our focus on $S_e$ we make the assumption that a controller will make the correct moves when $S_e$ permits it (i.e., when $S_e$ doesn’t interfere with the controller). We would like to point out that this assumption has far reaching implications. In particular, it implies that the system is designed and implemented correctly (including the software), and that $S_e$ is the only source of “errors”.

4 Annotated Models

Our first step towards understanding and analyzing the effect of a set $S_e$ on a system design is to annotate the conditions or events (referred to as hazards [2]) needed to cause the system to take a specific transition within our system model. The resulting expression is a boolean expression on events and is called an event initiator. In a system model, a particular transition will be taken iff (1) the system is in the state from which the transition originates, and (2) environmental conditions exist which cause the event initiator that is bound to the transition to evaluate to true.

As we mentioned earlier, transitions that can be controlled by the system through proper assignment to the control vector are brought about by components that enable these transitions. In turn, (controllable) events within the model of a system component correspond to states of sub-components, and so on. The resulting hierarchy is essentially the same that would be produced by a fault tree analysis of the particular transition (refs. 1 and 2). The edge or bottom of this hierarchy bounds the scope of our analysis and edge events are treated as environmental conditions belonging to $S_e$.

We would like to point out that while fault trees are often used in hazard analysis to provide a quantitative assessment, our refinement-based approach towards risk provides a framework to qualitatively compare one system design against another. Our reason for doing this is because we want to avoid the uncertainty (false sense of security) that is often introduced when one assigns (poorly understood) probabilities to system or environmental factors.

4.1 A Notation for Annotating Individual Transitions

Given a model of an implementation, $I$, we can annotate each transition with a postulated event initiator $\varepsilon$. Alternatively we can express the same kind of information in terms of the following logical formula:

$$s_1 \land \varepsilon_1 \Rightarrow s_2$$

This expression states that if a system is in state $s_1$ and the event initiator $\varepsilon_1$ becomes true, then the system will transition to state $s_2$.

Consider a system that is in a state, $s_1$, in which an electromagnet holds a metal object. Furthermore, let us assume that, in this state, if the electromagnet releases the object a high-consequence failure will result. Let $s_2$ denote the corresponding high-consequence failure state.
Suppose that for the set $S_j$ that is under consideration, the only event that can cause the system to transition from $s_1$ to $s_2$ is “shutting the electromagnet off”. Then the expression
\[ s_1 \land \text{electromagnet}(off) \Rightarrow s_2 \] (7)
describes this transition with respect to $S_j$. What this expression is saying is that, at the highest level of abstraction, in order for environment factors to cause the system to transition from $s_1$ to $s_2$ they would need to be able to cause the electromagnet to enter its “off” state. Proceeding down the fault-tree like hierarchy, we can examine a formal model of the electromagnet and try to determine what factors could cause the electromagnet system to enter an “off” state. This could lead to yet further examination of models of the components of the electromagnet system, and so on. The elements, components, etc., that lie at the bottom of our hierarchy fall, by definition, outside of our modeling and are considered directly with respect to $S_j$.

4.2 A Notation for Annotating all Transitions to a Specific State

To enable a formal approach to risk-based system design, we require that (sub)technologies within the scope of the analysis be given a complete event initiator semantics with respect to their individual state spaces as well as the postulated set $S_j$. Whenever possible, we require that each state of a concrete technology be defined in terms of a postulated event initiator that will put the technology into that state.

In the previous section, we described a notation for binding event initiators to individual transitions. We now extend that notation to allow us to describe all of the transitions that lead to a specific state. Let $\{s_1, s_2, \ldots, s_n\}$ denote the states of a system and let $\{e_1, e_2, \ldots, e_n\}$ denote the corresponding event initiators that will cause the system to enter an arbitrary state $s_j$. This gives rise to the following equations:
\[
\begin{align*}
    s_1 \land e_1 & \Rightarrow s_j \\
    s_2 \land e_2 & \Rightarrow s_j \\
    \vdots \\
    s_n \land e_n & \Rightarrow s_j
\end{align*}
\] (8)

Which in turn, produces the equality:
\[
(s_1 \land e_1) \lor (s_2 \land e_2) \lor \ldots \lor (s_n \land e_n) = s_j
\] (9)

Thus, when we write an equality we mean a complete coverage of the state space, and when we write an implication we mean coverage with respect to a specific state(s).

Consider the following example. Suppose that we have postulated that an electromagnet system can transition to an off state only by (1) switching it off, or by (2) loss of power to the magnet. Thus the (complete) event initiator semantics for the electromagnet(off) state is:
\[
\text{electromagnet(on)} \land (\text{switch(on)} \lor \text{power(loss)}) = \text{electromagnet(off)}
\] (10)

We can continue with this process by postulating an event initiator predicate for switch(off). If we assume that a switch is a technology having an on/off state space. The event initiator semantics for switch off might be:
\[
\text{switch(on)} \land \\
\text{controller(command(switch_off))} = \text{switch(off)}
\] (11)

Let us further suppose that in the proposed system implementation that we are considering the controller function is realized by a computer. At this point we can define the following higher order (i.e., functional) application specific equivalence:
\[
\text{controller} = \text{computer}
\] (12)

We can then continue and define:
\[
\text{computer}(x) \land (\text{sw}(y) \lor \text{hw}(y)) = \text{computer}(y)
\] (13)

Note that this is a semantic definition that (somewhat imprecisely) states, in general terms, that in order for a computer to be in a state $y$ (e.g., $y = \text{issue command(switch}\_\text{off})$), the transition to $y$ must be done by either the software (sw) or the hardware (hw).

Clearly, the process of defining the semantics of sub-technologies can go on for a long long time.
In particular, it can go on longer than is practical or even realistically possible. For these reasons we allow the event initiator semantics of sub-technologies to be terminated at any time. The resulting semantic framework then defines the extent (scope) and granularity of the risk analysis capabilities.

5 Refinements on Transitions

Our notion of risk-based refinement is based on event initiators. Let $\varepsilon_1$ and $\varepsilon_2$ be two event initiators that can be bound to a specific transition in our system model. That is, suppose in our system design we have two implementation choices that result in the semantic expressions:

\[
\begin{align*}
s_1 \land \varepsilon_1 &\Rightarrow s_2 \\
s_1 \land \varepsilon_2 &\Rightarrow s_2
\end{align*}
\]  

(14)

If we let $\leq_R$ denote risk-based refinement, then given a risk analysis function, $f : \text{event initiator} \rightarrow \text{real}$, we say that:

Definition 1:

\[
(\varepsilon_1 \Rightarrow \varepsilon_1 \Rightarrow s_2) \leq_R (\varepsilon_1 \land \varepsilon_2 \Rightarrow s_2) \text{ iff } f(\varepsilon_2) \leq f(\varepsilon_1)
\]

(15)

Due to the complexity of environments and systems, a risk analysis function will, in practice, be an approximation of the actual risk. This means that we should realistically assume a margin of error in the value produced by a given risk analysis function. In the presence of this margin of error we must be careful when concluding that one system design is a refinement of another. A standard approach to take here is to conclude that one design is better than another only if its associated risk value (obtained from $f$) is substantially lower (e.g., an order of magnitude) than the risk value of the initial design. The assumption being that the error in the risk analysis function will "certainly be less than an order of magnitude".

Given the objective of this paper to avoid probabilistic evaluations, the following interesting variation exists to the risk analysis function perspective of refinement. First partition the states in a system model into states to be avoided (i.e., hazardous states) and those states that we expect to occur during normal correct operation of the system. Let $S_H$ denote the set of hazardous states and let $S_X$ denote the expected (normal) states. Consider the following semantic expressions:

\[
\begin{align*}
 s_1 \land \varepsilon_1 &\Rightarrow s_2 \\
 s_1 \land \varepsilon_2 &\Rightarrow s_2
\end{align*}
\]

(16)

where $s_1 \in S_X$ and $s_2 \in S_H$. We now define risk-based refinement as:

Definition 2:

\[
(\varepsilon_1 \Rightarrow \varepsilon_1 \Rightarrow s_2) \leq_R (\varepsilon_1 \land \varepsilon_2 \Rightarrow s_2) \text{ if } f(\varepsilon_2) \leq f(\varepsilon_1)
\]

(17)

Informally, this definition states that $\varepsilon' \varepsilon$ enables or induces the transition from $s_1$ to $s_2$ no more than $\varepsilon_1 \varepsilon$. Clearly every accurate risk analysis function, $f$, must be consistent with this definition of refinement. That is, $\forall f : f(\varepsilon_2) \leq f(\varepsilon_1)$ when $\varepsilon_2 \rightarrow \varepsilon_1$. Also note that, in some sense, this is the reverse of correctness preservation-based refinement, but this is to be expected since we are after all trying to avoid this transition.

Further pursuit of this view of refinement can lead one to consider

\[
(\varepsilon_1 \Rightarrow \varepsilon_1 \Rightarrow s_2) \leq_R (\varepsilon_1 \land \varepsilon_2 \Rightarrow s_2) \text{ if } f(\varepsilon_2) \leq f(\varepsilon_1)
\]

(18)

where $s_1, s_2 \in S_X$, as a definition of refinement on $S_X$, but here one needs to be very careful to make sure that risk-based refinements preserve other system properties (e.g., liveness). In particular, the above expression (incorrectly) states that one transition is a refinement of another if it is more likely to occur. In the limit, the likelihood would be 1.0 meaning that this transition will be preferred over other possibilities in all cases. Among other things, such preferences can lead to

---

*In general, an expression of the form $f : D_1 \rightarrow D_2$ is a function signature stating that $f$ is a function taking elements from $D_1$ as input and producing elements in $D_2$ as its output.*
starvation and deadlock. Extension of refinement to transitions in the set $S_X$ is a topic of current research.

6 System Model Refinements

At this point, we have a system model in which transitions are annotated with event initiators that are defined with respect to a postulated set of environments. We also have a formal definition of risk-based refinement at the transition level. In this section, we describe how one might use the ability to qualitatively compare single transitions as the basis for qualitatively comparing entire system models.

There are many ways that one can go about doing this. For example, one can partition the set of system states into hazardous, $S_H$, and expected, $S_X$, and try to take into account the percentage of transitions that lead from $S_X$ to $S_H$. When considering percentages, it becomes difficult perhaps even unrealistic to expect that the assignment of probabilities to events can be avoided. Additional comparison difficulties are encountered when $S_H$ for one system design has “buffer” states that lie between expected states and high-consequence failure states while $S'_H$ for a second system design contains only high-consequence failure states. Here it is tempting to pursue a path that requires one to make “failure independence” assumptions in order to conclude that one system is better than another.

Our goal is to avoid as many of these difficulties (uncertainties) as we can. One way to do this is to define a comparison function that is “weaker” in the sense that not all system designs will be comparable. Such a function creates a partial order on system designs rather than the total order that could be achieved by assigning probabilities to events whenever needed. However, the advantage of this weaker function is that it needs to make significantly fewer probabilistic assumptions. The weakness of our comparison is not as bad as one might think because we are interested in using this operator to provide guidance in how to refine our system design from a risk-based perspective. That is, we aren’t interested in comparing completely unrelated system designs. Given this objective, we want to limit our consideration to small incremental system design refinements.

In our approach we strive to avoid making probabilistic evaluations by making the following assumptions:

- We assume that we are dealing with high-consequence systems. In such systems, the safety related behavior is not usually overly complex.

- Under normal operating conditions the system works correctly (e.g., the components function as intended, and the software is correct). In particular, we are not taking system complexity (e.g., the likelihood that the system will actually be implemented correctly) into account when comparing system designs.

- Abnormal conditions have a short duration (e.g., they can effect $n$ system transitions) and are followed by normal operating conditions during which the controller regains control of the system and can undertake defensive actions (e.g., controlled shutdown). Abnormal conditions occur at a certain frequency (a macro-level probability) and for that reason defensive actions should be simple and require as few transitions (ideally 1) as possible. A system can emerge from an abnormal environment and remain operable.

- Malevolent conditions occur when abnormal conditions exceed expected durations, frequencies, or intensities (e.g., extreme heat). Under such conditions the controller will not be able to regain control of the system’s behavior and a high-consequence failure can result unless first principle system properties can assure that such failure states cannot be reached. In this case, the system will become safely inoperable at best and will reach a high-consequence failure state at worst.

Let $S_i$ denote a system model having and expected region of states, $S_X$, and hazardous regions $\{S_{H_1}, S_{H_2}, \ldots, S_{H_n}\}$ corresponding to high-consequence failure states $\{s_{h_1}, s_{h_2}, \ldots, s_{h_n}\}$. Recall that in Section 5 we defined a hazardous region as a subset of the observable state space that a correctly
functioning system/controller will avoid. Thus if the system enters a hazardous region, then it follows that (abnormal or malevolent) environmental conditions exist. The diagram below shows a portion of a system with transitions leading into a hazardous region.

In $S_1$, let us consider all paths to a specific high-consequence failure state, $s_{h_1}$. Ideally, we would like to (incrementally) produce a new system $S'_1$ by changing one edge in a single such path while leaving all other edges and paths in $S_1$ unchanged. If this can be accomplished we can apply the definition of an edge refinement given in Section 5 to determine if the new (modified) design is a risk-based refinement.

In a more general setting a system modification may change a number of paths simultaneously. Consider the hazardous region, $S_{H_1}$, associated with $s_{h_1}$. Let $P$ denote the set of all paths, $p$, that: (1) originate from a state in $S_{X_1}$, (2) have a first transition that leads to a state in $S_{H_1}$, (3) have the property that all states, other than the first, belong to $S_{H_1}$, and (4) terminate at $s_{h_1}$.

For each $p \in P$ construct an expression consisting of the conjunction of all event initiators associated with the edges in $p$. Place the resulting expressions in a bag, $B$. Now change the system model to reflect the proposed system modification. Construct a corresponding bag, $B'$, for the new model. If $|B'| < |B|$ then we may add the boolean expression $\text{false}$ to $B'$ until $|B'| = |B|$. (If $|B'| < |B|$ then we consider the systems incomparable.) We now can define risk-based refinement on bags as follows:

$$B \subseteq_R B' \iff (\exists h: B' = h(B'))$$

Essentially what we have captured here is a modest system modification where a collection of single refinements are realized in a single design step.

### 6.1 Nontrivial Comparisons

Up to this point we have defined a comparison operator having a limited scope. In order to be able to compare two systems they needed to be quite similar. In this section we would like to briefly touch on the issues associated with extending the scope of risk-based comparisons a bit further. Consider the two systems shown in Figure 3. Note that in this diagram, the labels on transitions denote event initiators.

What can be said in a non-probabilistic qualitative manner about the design of system 2 when compared against the design of system 1? Here there is an almost overwhelming desire to assign probabilities to system states (e.g., the likelihood that a system is in a particular state) and events that make up event initiators. However, if we resist this urge, what can be said about the two systems? In particular, what are the minimum number of assumptions that need to be made in order to compare these systems in some meaningful fashion? For example, system 2 might be considered “safer” than system 1 because, after entering its hazardous region, it
takes a sequence of two event initiators, \(<A,D>\), in order to reach the high-consequence failure state – this assumes that A and D are independent. On the other hand, it might be the case that system 1 is rarely in state 1 (the only state from which \(<A,D>\) leads to a high-consequence failure) and therefore is highly resistant to a failure caused by the event initiator sequence \(<A,D>\).

In spite of the variables that exist among probabilistic interpretations, there are certain absolutes that hold over all interpretations. For example, the event initiator sequence \(<B,A,D>\) will not be more likely than \(<A,D>\) for all probabilistic interpretations. It is a qualitative constant. Extending the scope of non-probabilistic risk-based comparison is difficult and is a subject of current research.

7 Summary and Conclusions

This paper presents some preliminary results on how system models can be compared in a non-probabilistic fashion. Our comparison function is based on the notion of refinement on event initiators. In order for this function to be non-probabilistic it has been weakened, resulting in its inability to compare arbitrary systems. Instead, only systems that are extremely similar can be compared. To date, our research leads us to believe that our comparison function can be extended, allowing the comparison of more dissimilar systems. However, the information that can be provided by such comparisons is, at present, a little unclear.

References


* This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy.
**Biography**

Victor L. Winter, PhD, High Integrity Software Systems Engineering Department, MS 0535, Sandia National Laboratories, Albuquerque, NM 87185-0535, telephone – (505) 284-2696, facsimile – (505) 844-9478, e-mail – vlwinte@sandia.gov.

Dr. Winter is project lead for the High Integrity Software project at Sandia National Laboratories.

Ray S. Berg, PhD, High Integrity Software Systems Engineering Department, MS 0535, Sandia National Laboratories, Albuquerque, NM 87185-0535, telephone – (505) 845-8090, facsimile – (505) 844-9478, e-mail – rsberg@sandia.gov.

Dr. Berg is a Principal Member of Technical Staff in the High Integrity Software Systems Engineering Department, Sandia National Laboratories.

Larry J. Dalton, High Integrity Software Systems Engineering Department, MS 0535, Sandia National Laboratories, Albuquerque, NM 87185-0535, telephone – (505) 844-2520, facsimile – (505) 844-9478, e-mail – ljdalto@sandia.gov.

Mr. Dalton is Manager of the High Integrity Software Systems Engineering Department, Sandia National Laboratories.