Probabilistic Risk Reduction

Martin S. Feather, Steven L. Cornford, Leila Meshkat
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr., Pasadena CA 91109-8099
{Martin.S.Feather, Steven.L.Cornford,

James D. Kiper
Dept. of Computer Science &
Systems Analysis
Miami University
Oxford, OH 45056
kiperjd@muohio.edu

Abstract
We present an integrated approach to risk assessment and risk mitigation that is well suited to planning the development of complex software systems.

Our integrated approach is able to derive estimates of the costs and benefits (in terms of qualities of the developed product) at the time of planning a development. It accommodates both process knowledge (the efficacy of development practices) and product knowledge (the logical structure of the system under development). Functional and non-functional aspects of software can also be accommodated, and trades made among them. Optimization – selecting the best suite of process steps and design choices to maximize the expectation of success while remaining within budget – becomes possible.

The key to this is the integration of two complementary methods for reasoning about risks. One set of methods is that found in the area of Probabilistic Risk Assessment, specifically its methods for reasoning over logical fault trees. The other set of methods come from an early-lifecycle risk assessment and risk mitigation planning method that we have been developing and applying to spacecraft technology.

The integration of the two methods we call “Probabilistic Risk Reduction”, to draw attention to its probabilistic treatment of risk and explicit consideration of what can be done to reduce it.

1. Introduction
Risk is an important and recurring concern in system development. The field of probabilistic risk analysis (PRA) has developed methods to assess risks within complex systems. The key idea of PRA is to deduce the reliability of a system from knowledge of the system structure and knowledge of the reliability of the individual components from which the system is composed. Application of PRA techniques yields an overall assessment of a system’s reliability, confidence measures of that assessment, and insight into the key vulnerabilities of that system, thus indicating areas most in need of improvement. PRA is especially useful when a system is both expensive and safety-critical, rendering system testing impractical as a means to gain sufficient confidence in its reliability. The origins of these approaches lie in applications to assess risk in the nuclear power industry [NRC, 1975], with its need to estimate the probability of catastrophic failure (e.g., meltdown) from knowledge of the power system’s design, and reliability measures for the components used in that design. Fault Tree Analysis [Vesely et al, 1981] is now applied to a wide variety of systems, including some NASA missions and their hardware and software components [NASA PRA, 2002].

We have been developing a complementary approach to risk based planning, the key to which is the explicit representation and reasoning about the risk-reducing actions taken during development.

We show how our risk based planning approach can be combined with traditional PRA. This yields an integrated approach we call “Probabilistic Risk Reduction” well suited to planning the development of complex systems. The planning stage of a software development is a challenging time – information is sparse; few formal artifacts exist yet (e.g., code is unavailable to analyze, test, etc). Yet, the planning stage is the time of maximal influence on the course of the development to follow.

Our integrated approach is able to derive estimates of the costs and benefits (in terms of qualities of the developed product) at the time of planning a development. It accommodates both process knowledge (the efficacy of development practices) and product knowledge (the logical structure of the system under development). Functional and non-functional aspects of software can also be accommodated, and trades made among them. Optimization – selecting the best suite of process steps and design choices to maximize the expectation of success while remaining within budget – becomes possible.

The remainder of the paper is organized as follows:
Section 2 summarizes the salient points of our risk assessment process.
Section 3 motivates the need for a combination of approaches.
Section 4 gives details of our tight integration between PRA and our risk assessment process.
Section 5 provides a discussion, summary of related work, status and future work.
2. A complementary approach to risk-based planning: Defect Detection and Prevention

At JPL and NASA we have been developing and applying a risk-based approach to assist early-lifecycle planning of complex system developments. The approach is called “Defect Detection and Prevention” (DDP), the name reflecting its origins as a method intended for quality assurance planning of hardware systems [Cornford 1998].

Various aspects of DDP have been described in previously published papers: overviews of its status and application are in [Cornford et al, 2001], [Cornford et al, 2002]; the look and feel of the tool support in [Feather et al, 2000]. Here we provide a summary of DDP’s risk-based reasoning as a prelude to the main contribution of the paper, the integration of PRA and DDP.

2.1. Risk calculation

Most risk assessment methods separate the expression of a risk’s impact (a.k.a. “severity” or “consequence”) from its likelihood of occurrence. It is usual to calculate the risk (a.k.a. risk “exposure”) by multiplying these two values. Even when purely qualitative measures are given (e.g., likelihood and impact values can be one of “low”, “medium” or “high”) the pairwise combinations of these qualitative values are typically grouped into a qualitative approximation of the product (e.g., the overall risk is deemed to be “high” for likelihood & impact pairs medium & high, high & high and high & medium). Whether qualitative or quantitative, the separate expression of those two kinds of values, followed by calculation of their composite affect, is used in order to:

- achieve a better risk assessment, by basing it on simple information (separate expressions of likelihoods and impacts) from which to calculate the risk (as compared to attempting to directly assess the composite risk exposure),
- allow the distinction between different cases that lead to the same exposure, e.g., low-probability high-impact risks, and high-probability, low-impact risks,
- suggest ways to reduce risk, e.g., if the likelihood is high and consequence low, then there may exist quick and easy ways to decrease the likelihood.

The hallmark of DDP is further separations, among the risks themselves, the objectives that those risks threaten, and the measures taken to reduce risk. The key aspects of DDP are summarized in the subsections that follow.

2.2. DDP’s Objectives, Risks and Mitigations

The DDP process deals with three key sets of data: Objectives, Risks and Mitigations.

Objectives (a.k.a. Requirements) are the things that the system is to achieve, and the limitations within which it must operate. Objectives are assigned different “weights” to reflect their relative importance.

Risks are all the kinds of things that, should they occur, would lead to failure to attain Objectives. In the software realm, “defects” and “bugs” are analogous terms. Risks are assigned an “a-priori” likelihood, namely the likelihood of that Risk occurring if nothing is done to prevent it. Risks are assigned a cost of “repair” (a.k.a. “correction”). This is the cost of repairing the problem – for example, the cost of fixing a coding bug, or the cost of adding a missed requirement. In the software engineering community it is widely understood that these costs escalate through the course of the software lifecycle (e.g., the cost of correcting a flawed requirement at requirements time vs. at design time vs. at coding time vs. at unit test time vs. at system test time vs. after release). In the DDP model Risks are assigned time-specific repair costs that capture this escalation.

Mitigations are the actions that could be applied to reduce Risks. These could be preventative measures, tests, analyses, inspections, reviews, redundant design elements, etc. Each Mitigation is assigned a cost, namely the resource costs of applying it. In our world of spacecraft development, there are typically several kinds of critical resources, e.g., budget ($), mass, volume, electrical power. Each Mitigation is also assigned a time, typically the “phase” in the development effort at which it is applied (e.g., requirements time, design time, coding time). It is possible to use other time scales (e.g., financial quarters or, for long duration developments, years).

2.3. Risk Basis

Most risk assessment processes take as starting point a design, and focus on the risks remaining in that design. The (potentially many) steps of the design process that assure that design’s quality are encoded within the high reliabilities assigned to its components. For example, under the assumption of programmers skilled in the domain of the project under development, the number of coding defects attributable to domain misunderstandings will be small.

In contrast, DDP takes as starting point a design in which the existence and benefits of its quality assuring steps are made explicit. The risks remaining in the design are calculated from the a-priori risks, and the risk reductions effects of those steps. The potential advantages that accrue from DDP’s approach are that it may:

- achieve a better risk assessment, by basing it on simple information (separate expressions of a-priori risk likelihoods and risk reducing effects) from which to calculate the residual risks (as compared to attempting to directly assess the residual risk)
- permit tradeoff decision making to help choose which (out of potentially many) risk reducing actions to employ
must be allocated, teams formed, etc. Furthermore, and when a product reliability engineering techniques based on measuring the setting is to help (1) plan a software development effort, and (2) assess the reliability of the software that results. Advance planning is obviously necessary – budgets must be allocated, teams formed, etc. Furthermore, improved decision making at planning time has the most scope to influence the entire development effort to follow. However, the obvious challenge is the lack of information available at planning time. For example, consider software number of defects discovered during testing to indicate remove and alleviate defects, i.e., into account process scope to influence the entire development effort to follow. The primary utility of this in the software engineering setting is to help (1) plan a software development effort, and (2) assess the reliability of the software that results.

However, the obvious challenge is the lack of information available at planning time. For example, consider software reliability engineering techniques based on measuring the number of defects discovered during testing to indicate when a product is ready to release. These are clearly useful at testing time, but by then it is too late, for example, to make a change to how much inspection of requirements is done, because that development phase is long passed.

Probabilistic Risk Assessment methods are not well suited to software reliability assessment, since software does not "fail" in the same way that hardware does. Rather, it has latent defects that may surface during operation. The DDP approach supports a focus on the efficacy of the development steps taken to prevent, remove and alleviate defects, i.e., into account process knowledge for reliability assessment purposes.

2.4. Impacts and Effects

The DDP process deals with quantitative relationships that link Objectives, Risks and Mitigations, as follows:

**Impacts** are the quantitative relationships between Objectives and Risks, namely the proportion of the objective attainment that would be lost should the Risk occur. A risk can impact multiple requirements to different extents, and similarly a requirement can be impacted by multiple risks, again to different extents.

**Effects** are the quantitative relationships between Mitigations and Risks, namely the proportion by which a Mitigation reduces a Risk should that Mitigation be applied. A Mitigation can effect multiple Risks to different extents, and similarly a Risk can be effected by multiple Mitigations, again to different extents.

2.5. Categories of Mitigations

Mitigations are subdivided into three important categories:

**Preventions** – these prevent Risks from arising in the first place. In the DDP model, they act to decrease the Risk likelihoods. Software examples are: training of programmers, establishment of coding standards.

**Alleviations** – these decrease the severity of Risks should they occur. In the DDP model, they decrease the Impacts that Risks have on Objectives. A software example would be to have a module check that its inputs are indeed in the valid range expected – the check does nothing to decrease the likelihood of invalid input, but it allows the module to recognize the situation and respond gracefully rather than fail catastrophically.

**Detections** – these prevent Risks from occurring (i.e., having their detrimental impact on Requirements). As the name suggests, they work by detecting the presence of a Risk, with the assumption that Risks so detected are then "repaired". In the DDP model, they decrease the likelihood that the Risk will occur. A wide range of software practices fall into this category, for example, testing activities (e.g., unit testing, system testing, stress testing), analyses (e.g., the many forms of static analysis, model checking, theorem proving) and inspections (e.g., peer reviews, code walkthroughs, formal inspections (Fagan, etc)).

The DDP model also captures the phenomenon of a Mitigation that can make some things worse. For example, using an elaborate algorithm in order to achieve a performance speedup may increase the risk of coding errors. Adding code to measure timing might inadvertently change the behavior. Correcting one problem may introduce others. In DDP, such situations are represented by assigning a negative value to the Effect link between a Mitigation and a Risk. This is further subdivided into one of two kinds – effects that "induce" a Risk (increase the likelihood of that Risk) and effects that "aggravate" a Risk (increase the impact of the Risk).

2.6. DDP's quantitative calculations

The "topology" of a DDP model is shown in Figure 1. Benefits are the sum of attainment of objectives, and Costs are the sum of mitigations and risk (defect) repairs.

![Figure 1. Topology of DDP model](image)

Mitigations reduce Risk likelihoods and/or impacts, and thereby lead to increased attainment of Objectives. In the limited space available in this paper we have room only to
convey a feel for these quantitative calculations. by describing one of them:

Multiple Mitigations’ effects on the same Risk combine essentially like filters arranged in series, so that their combined effectiveness is the complement of the product of the complements of their individual effectiveness. For example, if two Mitigations have effects on the same Risk of 0.1 and 0.2, then their combined effect is \((1 - (1 - 0.1)(1 - 0.2)) = 0.28\). In other words, if the first one removes 10% of its incoming risks (leaving 90%), and the second 20% of those 90%, what remains is 72% - a net combined effectiveness of 28%.

A thorough discussion of DDP’s quantitative nature may be found in [Feather & Cornford, 2003].

2.7. Use of DDP in practice

DDP has been used to assess risks and plan their mitigation for a variety of spacecraft-related technology developments, software, hardware and combinations of both. The key steps of the DDP process are:

1. Gather the DDP information (Objectives, Risks and Mitigations, and the Impacts and Effects that link them). Information may come from previously assembled knowledge bases [Kurtz & Feather, 2000], and/or on-the-fly elicitation from experts.

2. Perform the quantitative calculations, and present the information to the experts for scrutiny and revision. The DDP software automates the calculations, and offers a variety of cogent visualizations of the results.

3. Select mitigations that cost-effectively reduce risk. The quantitative representation makes possible the treatment of risk mitigation as an optimization problem (e.g., select the mitigations that lead to maximal attainment of requirements while costing no more than some given limit) [Cornford et al, 2003]. Another outcome of the DDP process may be the revision of the Objectives when they prove overly costly to attain [Feather et al, 2002].

The amount of information taken into account in these DDP-based risk studies is typically detailed and highly coupled. This accounts for the need for an appropriate process and accompanying tool support. A sense of the detail can be seen from Figure 2, which shows the topology of the data in an actual DDP application.

3. Combining PRA and DDP

Over the past year we separately applied PRA and DDP to the same spacecraft design. This gave us insights into the relative strengths and weaknesses of each approach and motivated us to seek their combination [Cornford et al, 2003].

3.1. Comparison of PRA and DDP

In brief, the comparison showed DDP’s relative strengths to be the ability to capture the wide range of risks that threaten a development, and to plan risk mitigation accordingly, and showed PRA’s relative strengths to be the ability to faithfully represent the interplay of faults in combination, and to pinpoint areas of vulnerability in such combinations.

From that study we identified a loosely coupled way to integrate PRA and DDP. The essence of our vision was iteration between the two techniques. We could see that starting with DDP would allow the rapid pinpointing of the riskier areas of the project plan. This would suggest the areas to which PRA could then be applied to study in more detail. The improved likelihood and consequence estimates would be fed back into DDP, and the risks re-ranked. Continuing this iteration would further refine the accuracy of the risk assessment in the areas that mattered the most.

Such a loosely coupled integration would do better than either technique alone, but would fall short of a true combination of the strengths of each technique. It is the goal for such a combination through the tight integration of the two techniques that is the focus of this paper. From the topology of an actual DDP project shown in Figure 2 it is obvious that DDP differs from the event diagrams and fault trees common to PRA approaches in two key ways:

- DDP, by explicitly representing objectives, has a finer-grained notion of success, and by explicitly representing mitigations allows for reasoning to encompass choice from among those mitigations.

![Figure 2. Topology of the data in a DDP application](image)
Conversely, PRA, by explicitly representing the logical structure of faults, has a more faithful representation of how faults combine. DDP can only approximate this, or rely on users to manually encode different combinations of faults as distinct "Risks", and manually score each of them. (While possible, this is prohibitively tedious in all but the simplest of cases).

A tight integration of the two would combine (1) the strengths of DDP at representing multiple criteria (Objectives) and at facilitating cost/benefit reasoning to guide selection of Mitigations, and (2) the strengths of PRA at representing the influence of the design on the structure of the possible failures of that design.

3.2. Approach to tightly integrating PRA and DDP

Our approach to the tight integration of PRA and DDP is to embed PRA fault trees, and all the PRA reasoning that goes with them, into the center of the DDP topology. This is sketched in Figure 3. The fault tree structures of PRA replace the simple list of risks in DDP (the standard DDP program does use a tree hierarchy to organize risks, but it is merely a taxonomy). To make the sketch simple, the fault trees are drawn as if there are no shared events ("common cause" events of fault trees) within a tree or between trees. In practice, there may be, and the combined approach must be able to accommodate them.

\[
\text{Cost} = \Sigma \text{cost of Mitigations & Repairs}
\]

\[
\text{Benefit} = \Sigma \text{attainment of Objectives}
\]

Figure 3. Topology of combined PRA and DDP

Within our integrated approach, the usual gamut of PRA techniques are available for calculating likelihoods of faults trees, identifying cut sets, performing sensitivity analysis, etc. DDP relationships are used to capture how those fault trees are related to Objectives, and on how the Mitigation actions effect the likelihoods of the leaf nodes of those fault trees, allowing the cost-benefit based reasoning of DDP to apply to the whole. The next section presents this in more detail.

4. Tight integration of PRA and DDP

4.1. Objectives and Fault Trees

Standard DDP "Impact" links connect Risks to the Objectives they threaten, using each link's quantitative measure to represent how much of the Objective would be lost were the Risk to occur. In the tight integration of PRA and DDP these same Impact links now connect the root nodes ("top events" in PRA terminology) of logical fault trees to Objectives. The probabilities of occurrence of these root nodes are calculated by means of the PRA techniques from the logical structure of the fault trees and the likelihoods of the leaf nodes of those fault trees. As we shall see in the next subsection, DDP's "Effect" links come into play to determine the likelihoods of those leaf nodes.

On occasion it is also necessary to relate Objectives to interior root nodes of fault trees. For example, a space mission could have multiple science Objectives, one being a science experiment that needs battery power, another being the demonstration of a novel battery technology. The interesting case is when there is a standard battery to support the experiment, but the spacecraft design allows for the experiment to make use of the novel battery technology in the event that the standard battery fails. Thus the fault tree of risks to the science experiment would contain within it the subtree of risks to the novel battery. The root node of the no-power-available fault tree would be linked to the science experiment Objective, and the interior subtree of risks to the novel battery would be linked to the novel battery demonstration Objective.

4.2. Fault Trees and Mitigations

Standard DDP "Effect" links connect Mitigations to the nodes within the fault trees that they reduce (by decreasing the likelihood or decreasing the impact) or increase (for Mitigations that make some risks worse). The kind of Mitigation – prevention, alleviation or detection – determines the nature of the reduction, and the way the reduction combines with the fault tree, as discussed in the subsections that follow.

4.2.1 Prevention Mitigations and Fault Trees

"Prevention" type Mitigations can only be connected to leaf nodes ("basic events" in PRA terminology) of fault trees. Intuitively, this is because a non-leaf node of a fault tree correspond to logical combinations of that node's children. Hence the only way to effect its occurrence is to effect the occurrence of those children; applying this line of reasoning recursively, we see that this leads to affecting the occurrence of the leaf nodes (basic events) of the fault trees. Prevention mitigations serve to reduce the likelihoods, i.e., in PRA terminology, they decrease the "likelihood" half of the equation: risk = likelihood x severity. See Figure 4 for a sketch of where they fit in to the picture of DDP with fault trees.
4.2.2 Alleviation Mitigations and Fault Trees

"Alleviation" type Mitigations are generally connected to the root nodes of fault trees, because it is the occurrence of these faults that detract from objectives attainment via the "Impact" links. Alleviation mitigations serve to reduce the impacts, i.e., in PRA terminology, they decrease the "severity" half of the equation: $\text{risk} = \text{likelihood} \times \text{severity}$. See Figure 5 for a sketch of where they fit in to the picture of DDP with fault trees.

The case of a non-root node of a fault tree linking to an Objective (recall "novel battery technology" example) would be an exception to this rule – it would make sense to link an Alleviation-type Mitigation to that non-leaf node. (Note: in an even more elaborate model, we might link Alleviations to individual Impact links, allowing for the possibility that an Alleviation's Effectiveness differs from Impact to Impact).

4.2.3 Detection Mitigations and Fault Trees

"Detection" type mitigations (e.g., tests, analyses, inspections) detect the presence of faults. They can thus apply to any of the nodes in a fault tree: root node (top event), leaf node (basic event) or interior node! However, detection type mitigations achieve their risk-reducing benefits by leading to the repair of the faults they detect.

In the case of detection of a leaf node fault, the situation is straightforward – repair decreases the likelihood of that leaf node. When the standard PRA techniques are used to calculate fault tree likelihoods, they base their calculations on the decreased likelihoods that result from such repairs. A simple example is shown in Fig. 6, where a “Detection” type Mitigation $M_1$ is applied to the left leaf node of a fault tree with a single “And” gate. If the mitigation is not applied, the likelihood of the root node of the fault tree is calculated as shown in the left side of the figure; since it is an “And” gate, the likelihood of the root node is the product of the likelihoods of its children, i.e., $0.8 \times 0.6 = 0.48$. If the mitigation is applied, its effect is to reduce by half the likelihood of the left child, from 0.8 to 0.4. Hence, the likelihood of the root node now becomes $0.4 \times 0.6 = 0.24$.

Figure 6. Detection Mitigation effect on a leaf node of a fault tree

In the case of detection of a non-leaf-node fault (e.g., system test applied to a system composed of several units) the situation is more interesting – repair equates to tracing to the cause(s) of that fault, namely the leaf nodes of the subtree, and repairing them (i.e., decreasing their likelihoods of occurrence). However, the allocation of causes to leaf nodes, and hence the repairs, is not uniquely determined. For example, suppose a detection mitigation with effectiveness of 0.5 applies to the root node of Figure 6. One solution would be to halve the likelihood of the left of the leaf nodes; another would be to halve the likelihood of the right of the leaf nodes; others would be to apportion the likelihood reductions between the two. To handle this case, we make the key assumption that:

The effect of a detection type mitigation on a fault tree’s parent node has the effect of decreasing in the same proportion the likelihoods of its children.
For example, suppose the mitigation is a system test, and the system is composed of two components, both of which must function correctly if the system is to function correctly—i.e., its fault tree would use an “or” node (a fault in either component would produce a fault in the system). Faults discovered by a system test must result from faults in one or both of its units. If one of the units is more error prone than the other, then presumably the system test will reveal more errors attributable to that unit. Our assumption of proportionality captures this phenomenon—the proportion of errors of a more error prone unit will be a larger number of errors than the same proportion but of a less error prone unit. In the extreme case of one of the two units being perfect (error free), then any proportion of its errors will be zero errors.

Several ramifications stem from this key assumption, considered in the subsections that follow.

4.2.4 Detection Mitigations and “And” gates

If a Detection Mitigation with effect $E$ is applied to an “And” gate consisting of $m$ children, then by our key assumption, each of the gates’ children’s likelihoods are decreased in the same proportion. In fact, their likelihoods become multiplied by the same factor $k$, such that $k$ is the $m$th root of $(1-E)$, i.e., $k = (1-E)^{1/m}$.

$$0.8 \times 0.6 = 0.48$$

Then: $(k \times 0.8) \times (k \times 0.6) = 0.24$

So: $k \times k = 0.24 / 0.48$

Hence: $k = \sqrt[2]{0.5}$, $p_1 = 0.8 \times \sqrt[2]{0.5}$, $p_2 = 0.6 \times \sqrt[2]{0.5}$

Figure 7. Detection Mitigation effect on a root node of a fault tree, percolating to its leaf nodes

If a Detection Mitigation with effect $0.5$ is applied to the root node. In order to achieve a 50% reduction in the root node likelihood (from 0.48 to 0.24), we can compute that $k = 0.38$ (approximately).

$$(1 - (1 - 0.8) \times (1 - 0.6)) = 1 - 0.08 = 0.92$$

Effect = 0.5

$$(1 - (1 - 0.8k) \times (1 - 0.6k)) = 0.92 \times (1 - 0.5) = 0.46$$

Figure 8 shows a simple example. The left of the figure shows the likelihood calculation for a simple “Or” gate. Now suppose that a Detection Mitigation with effect 0.5 is applied to the root node. In order to achieve a 50% reduction in the root node likelihood (from 0.48 to 0.24), we can compute that $k = 0.38$ (approximately).

$$0.8 \times 0.6 = 0.512$$

Effect = 0.5

Figure 9. Risk Mitigation through one layer

The lower half of the figure shows the risk reductions following application of a detection type Mitigation that cuts the system fault likelihood by half. Following our scheme, this is achieved by the same proportional reduction of the likelihoods of its two children. The left one’s likelihood decreases from 0.8 to 0.57, the right one from 0.64 to 0.45.

Suppose that the right hand unit (the one with initial fault likelihood of 0.64) is in fact composed of two units of its own, each with initial fault likelihood 0.8. (Figure 10, top half) Then, the reduction of the fault likelihood of that unit, from 0.64 to 0.54, is in turn achieved by
proportionally reducing the fault likelihoods of its children. (Figure 10, bottom half).

\[
0.8 \times 0.64 = 0.512
\]

\[
\begin{array}{c}
0.8 \\
0.64
\end{array}
\]

\[
\text{Detection} \\
\text{Effect} = 0.5
\]

\[
0.256
\]

\[
0.57 \\
0.67 \\
0.67
\]

Figure 10. Risk Mitigation through a two-layer tree

Now consider a logically equivalent, but structurally distinct, configuration of the same three leaf nodes, as three children of a single “and” node. (Figure 11, top half). In this situation, the effect of a detection Mitigation propagates evenly among the three children. (Figure 11, bottom half). Observe that all three leaf nodes began with likelihoods of 0.8, and end with likelihoods of 0.63. In contrast, in the structure shown in Figure 10, while all three leaf nodes began there with likelihoods of 0.8, they end with likelihoods of 0.57, 0.67 and 0.67.

\[
0.8 \times 0.8 \times 0.8 = 0.512
\]

\[
\begin{array}{c}
0.8 \\
0.8
\end{array}
\]

\[
0.8
\]

\[
0.8k \times 0.8k \times 0.8k = 0.512 \times 0.5 = 0.256
\]

\[
\begin{array}{c}
0.63 \\
0.63
\end{array}
\]

\[
\text{Detection} \\
\text{Effect} = 0.5
\]

\[
0.63
\]

Figure 11. Risk Mitigation among three children

The intuition we attach to this phenomenon is that the system structure influences how the testing of that system is likely to reveal faults in its units. For example, a system level test is more likely to reveal faults in units that are close to that system level than units that are many levels deep. In Figure 10, the one unit closer to the top has more of its faults detected than either of the two units deeper within the subsidiary “And” gate.

It follows that it is important that the fault tree structure reflect the architecture of the application. By contrast, traditional analyses of fault trees are insensitive to logically equivalent reorganizations.

### 4.3. Repair costs of “Detection” Mitigations applied to fault trees

In the DDP model, detection type mitigations incur both a cost of performing the mitigation (e.g., running the test) and a cost of repairing the flaws they detect (e.g., correcting the bugs). This carries over smoothly into the integration with PRA's fault trees. As discussed in the previous subsections, the risk-reducing effect of applying a detection type mitigation is percolated down to the leaf nodes of the fault tree, where reductions take place on the basic event likelihoods. These reductions are, in fact, the results of repairs, and so the cost incurred in performing them is the product of the unit repair cost and the number of repairs made (note that the number of repairs is proportional to the drop in likelihood – e.g., if as a result of detection and repair the individual fault likelihood dropped from 0.7 to 0.3, then the number of repairs must be proportional to that drop of 0.4).

### 4.4. Common cause failures (shared nodes)

Common causes of failure – shared events in fault trees – significantly complicate the standard calculation of fault tree likelihoods. In interests of efficiency, many PRA tools perform approximate calculations that ignore second order effects from these – such approximations are often acceptable because the high reliability systems they study are built from high reliability components, so products of their tiny likelihoods are extremely small. For our purposes of calculating likelihoods from first principles (i.e., from a-priori risk likelihoods, then reduced by Mitigations), the likelihood numbers we deal with need not be tiny, so the same kind of approximations are inappropriate, and the more computationally complex, but more accurate, likelihood calculation is required. Recently, some PRA tools have started to make use of Binary Decision Diagram (BDD) representations to perform the calculations in an efficient (or in most cases efficient) manner without resorting to approximations, e.g., the Galileo tool of [Manian et al, 1998].

For fault trees with common cause failures, we are currently experimenting with an approach in which the reductions are percolated down using the same algorithm as described earlier. The calculations may yield differing likelihood reductions for the multiple instances of the shared event, in which case we choose the greatest of these reductions as the one that applies to all its instances. Our reasoning is that this is the greatest likelihood reduction corresponds to having identified the most faults in that unit, so, on the assumption that we repair all those faults, the benefit of the repair (i.e., the decreased fault likelihood) is available wherever it occurs in the fault tree.
5. Discussion

5.1. Significance

Our work represents a fundamental first step towards the merging of elements drawn from two (currently somewhat disparate) themes of software engineering: (1) process-centric approaches (e.g., CMM, ISO, TSP), and (2) artifact-centric approaches.

In the former, the emphasis is on populating and organizing the development effort so as to promote the attainment of a quality result in a predictable and timely fashion. In the latter, the emphasis is on means to assess the quality of the software artifacts themselves, regardless of the process by which they were produced. Our in-house DDP approach falls into the former theme, with its explicit treatment of the effect of quality steps on preventing, removing and alleviating defects. PRA techniques fall into the latter theme, with their emphasis on deriving reliabilities and insights into system vulnerabilities through calculation based on the structure of the artifact under consideration.

The integration of PRA and DDP that we have described, while still in its infancy, indicates that there is a rich area of study in seeking a blend of strengths drawn from both the process-centric and artifact-centric endeavors.

5.2. Summary and Status

We have described an integration of the logical fault trees of traditional Probabilistic Risk Assessment methods and the explicit treatment of Objectives and Mitigations of our own risk assessment method. The combination achieves the strengths of both approaches, namely that it (1) permits the simultaneous consideration of a wide variety of risks and objectives, (2) enables knowledge of the system development steps effects on risk to be taken into account, and (3) allows for cost/benefit based reasoning about risk reduction.

The status of our work is that all of the probabilistic risk reduction calculations described in this paper have been incorporated within the DDP software. Requests for DDP may be made via the site http://ddptool.jpl.nasa.gov

5.3. Related work

The work that comes closest in spirit is that of Fenton et al. We share the motivation that they express in [Fenton & Neil, 1999] for improved techniques that are able to take into consideration defect prevention, detection and correction. The have made use of Bayesian Belief Networks to combine knowledge of causal structure with evidence (expert judgments and/or historical records) of the error rates and efficacy of testing, etc. [Fenton et al, 2003]. Our approach differs from theirs in adding the explicit treatment of multiple Objectives, and in representing the logical fault structures common to PRA techniques (whereas Fenton et al appear to focus exclusively on the structure of the development process).

There are other approaches that decompose Objectives (a.k.a. “Requirements” or “Goals”) so as to assess and plan suitable developments. Notable among these is the goal-tree based “KAOS” work of van Lamsweerde, in which goals are refined towards specifications of component behaviors that together achieve those goals. He has investigated “Obstacles”, the ways in which his goals can fail to be achieved [van Lamsweerde, 1998]. Obstacles have a close parallel with what we would term risks. Key differences are that we emphasize a quantitative treatment, while van Lamsweerde’s emphasis is primarily logical, and the focus is primarily on the artifact, not the process by which it is to be developed and its quality assured.

A somewhat similar treatment of goals is to be found in the Non Functional Requirements work of Mylopoulos et al [Chung et al, 2000]. This has been used to qualitatively compare the relative strengths and weaknesses of alternative designs [Mylopoulos et a, 2001]. Like van Lamsweerde’s approach, goal decompositions and refinements are captured in tree structures. Impediments are linked to those goals and their refinements, reminiscent of our “Impact” and “Effect” links. However, we do not see a corresponding emphasis on the efficacy of process steps, notably of the kind we would term “Detection type Mitigations” within their treatment. Also, to date their approach has been primarily qualitative in nature.

5.4. Future Work

DDP without logical fault trees has seen extensive usage. Now that we have built the logical fault tree capability into DDP, we are ready to seek applications that make use of it. We feel that a software development, where there is at least a preliminary notion of the design or architecture (thus allowing for a fault tree like treatment) is a promising area for application. We see software fault tree analysis [Leveson, 1995] being applied to a wider variety of software problems (e.g., intrusion detection [Helmer et al, 2002]), so opportunities abound.

We built into DDP a simple logical fault tree capability (“And” and “Or” gates) to allow us to explore the ramifications of combining DDP-like mitigations with PRA-like logical fault trees. However, we have not built in the other capabilities of PRA tools, e.g., distributions. The obvious next step is to link DDP directly with an existing PRA tool. In this direction we have performed some very preliminary experiments at passing a DDP-generated fault tree to the PRA tool Galileo and having it compute the likelihood.

PRA approaches use more complex gates than simply “And” and “Or” (e.g., ordered “And” gates, where there is importance to the order in which its constituent events occur). In order to extend our integration to encompass these additional kind of gates, we will need to study and
extend as appropriate the way in which mitigations’ effectiveness distributes over those gates, in a similar manner to our treatment of detection type mitigations with logical fault trees.

6. Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

Numerous individuals have helped with the development and application of DDP. The blending of DDP with elements of traditional PRA techniques, and its application to software development, have benefited from discussions with John Kelly (JPL), Tim Kurtz (NASA Glenn), Tim Menzies (WWU & NASA IV&V), Todd Paulos (JPL) and Burton Sigal (JPL).

7. References


This page provides screenshots taken from our DDP implementation shown operating on a small hypothetical example. This information is provided as a courtesy to reviewers to indicate that we do indeed have a working implementation of these ideas.

### Risks

<table>
<thead>
<tr>
<th>Likelihoods</th>
<th>Mitigations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1: Fault protection system fails</td>
</tr>
<tr>
<td>1</td>
<td>2: Performs the wrong response</td>
</tr>
<tr>
<td>1</td>
<td>3: FP itself gets into an infinite loop</td>
</tr>
<tr>
<td>1</td>
<td>4: Fails to respond to a fault</td>
</tr>
<tr>
<td>1</td>
<td>5: Lacking a response for a given fault</td>
</tr>
<tr>
<td>1</td>
<td>6: Internal queues get full</td>
</tr>
<tr>
<td>1</td>
<td>7: Flooded with duplicate fault reports</td>
</tr>
<tr>
<td>1</td>
<td>8: Can’t recognize duplicate fault reports</td>
</tr>
</tbody>
</table>

**The risk tree. A-Priori likelihoods are all set to 1**

### Mitigations

<table>
<thead>
<tr>
<th>Likelihoods</th>
<th>Mitigations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1: System testing (detection)</td>
</tr>
<tr>
<td>1</td>
<td>2: Build a table mapping faults to responses (prevention)</td>
</tr>
<tr>
<td>1</td>
<td>3: Model check the FP design (detection)</td>
</tr>
</tbody>
</table>

The three available mitigations

### The effectiveness table. Mitigations are the rows, risks the columns. A numerical entry indicates the effectiveness of the mitigation on the risk.

<table>
<thead>
<tr>
<th>System testing (detection)</th>
<th>0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build a table mapping faults to responses (prevention)</td>
<td>0.8</td>
</tr>
<tr>
<td>Model check the FP design (detection)</td>
<td>0.9</td>
</tr>
</tbody>
</table>

### Calculated risk likelihoods when the “Model check the FP design” mitigation is applied.

<table>
<thead>
<tr>
<th>Likelihoods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
</tbody>
</table>

### Calculated risk likelihoods when the “Build a table mapping faults” mitigation is applied.

<table>
<thead>
<tr>
<th>Likelihoods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
</tr>
</tbody>
</table>

### Calculated risk likelihoods when all three mitigations are applied.

<table>
<thead>
<tr>
<th>Likelihoods</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.155</td>
</tr>
<tr>
<td>0.0246</td>
</tr>
<tr>
<td>0.0123</td>
</tr>
<tr>
<td>0.0123</td>
</tr>
<tr>
<td>0.0113</td>
</tr>
<tr>
<td>0.0113</td>
</tr>
<tr>
<td>0.036</td>
</tr>
<tr>
<td>0.036</td>
</tr>
</tbody>
</table>

Probabilistic Risk Reduction Feather, Cornford, Meshkat & Kiper