A Quantitative Risk-based Model for Reasoning over Critical System Properties

Martin S. Feather
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr
Pasadena CA 91109-8099, USA
+1 818 354 1194
Martin.S.Feather@Jpl.Nasa.Gov

ABSTRACT
The workshop description lists *security, safety, survivability, fault-tolerant,* and *real-time* as included among the critical properties that may be required of high assurance systems. Each of these is an active area of ongoing research. It is a daunting challenge to develop systems that satisfy several of them at once, yet there is a pressing need to do so for many high assurance systems.

This position paper suggests the use of a quantitative risk-based model to help in such situations. The aim is to support reasoning and decision making that spans many of these critical properties. The potential benefits include prioritization of the challenges that high assurance system development efforts face, ability to identify solutions to match the challenges, identification of the areas of weakness where further research is warranted, and improved understanding of the strengths and weaknesses of a particular system.

Keywords
Risk, requirements, tradeoffs, prioritization, quantitative models.

1 INTRODUCTION

The purpose of this position paper is to advocate a quantitative risk-based model for use in planning the development of high assurance systems. The model has been developed to facilitate early-lifecycle reasoning. It has proven to be of value in the multi-disciplinary setting of spacecraft technology development, where knowledge and expertise from multiple experts must be pooled. High assurance systems development exhibits these same needs.

Section 2 describes the model. Section 3 uses as illustration an ongoing study of the challenges that impede the practical use of model checking technology, and what can be done to overcome those challenges. This example is selected to be of interest to RHAS workshop attendees. Section 4 suggests ways in which this model could be applied to the high assurance systems area.

2 THE QUANTITATIVE RISK-BASED MODEL

For several years a group of us at JPL have been developing and applying a quantitative risk-based model to the problem of assessing the viability of, and planning for, the development of novel technologies and systems [4]. We apply this model early in the lifecycle, when information is sparse, but the capability to influence the course of the development to follow is large. Our model tries to fill the niche between qualitative approaches and detailed design-centric analysis approaches.

Qualitative approaches, supported to varying degrees by a variety of methods, processes and tools, work well as an adjunct to human decision-making. They encourage the elicitation of issues, and excel in providing semi-formal means to record interrelationships among those issues. Formal analysis approaches, often supported by sophisticated tools founded upon deep theoretical underpinnings and significant development efforts, excel once given a detailed design and putative properties for which to analyze. Examples of qualitative approaches include Quality Function Deployment (QFD) [1], i* [9], fault mappings [10] and WinWin [3]. Examples of detailed design-centric approaches include the goal-decomposition approach of KAOS [2], probabilistic risk assessments methods (e.g., fault trees [12]), formal analysis techniques such as model-checking (e.g., Holzmann's SPIN [7]), constraint analysis (e.g., Jackson's Alloy language and analyzer [8]), specification analysis (e.g., consistency and completeness analyses within the SCR toolset [6]) and theorem proving (e.g., PVS [11]).

Steve Cornford at JPL conceived of the "Defect Detection and Prevention" (DDP) model. It was initially intended to facilitate planning of quality assurance activities. There are typically far more assurance activities (e.g., various analyses, tests, inspections, standards, policies, certifications, defensive measures) from which to choose than there are resources (e.g., time, budget, high fidelity testbeds, appropriately trained individuals) available to

perform those assurance activities. Their judicious selection, to make optimal use of the limited resources available, is essential.

The DDP model's key concepts are as follows:

- "**Requirements**" – the things that the system (in our case, spacecraft, or spacecraft's system or subsystem) is to achieve, and the limitations within which it must operate. Requirements can be given different "weights" to reflect their relative importance.
- "**Failure Modes**" – all the things that, should they occur, would lead to failure to attain Requirements.
- "**PACTs**" (**P**reventions, **A**nalyses, process **C**ontrols and **T**ests) – all the things that could be applied to reduce Failure Modes, by:
  - decreasing their likelihood of arising in the first place,
  - detecting them so that they can be repaired prior to use (for spacecraft, hardware repair must usually be done prior to launch, while repairs to software and changes to operating procedures can be done after launch), or
  - alleviating their severity should they occur.

These are *quantitatively* related in the following manner:

- Requirements are quantitatively related to Failure Modes, to indicate the proportion of the Requirement that would be lost should the Failure Mode occur. In DDP terminology, we say that a Failure Mode has an "impact" on a Requirement.
- Failure Modes are quantitatively related to PACTs, to indicate the proportion by which each PACT reduces each Failure Mode ("effect" is the term used in DDP) should that PACT be applied. In DDP terminology, we say that a PACT has an "effect" on a Failure Mode.

The model further assumes that:

- Failure Modes' impacts on a Requirement combine by addition. For example, if two Failure Mode have impacts on the same Requirement of 0.1 and 0.2, then their combined impact is $0.1 + 0.2 = 0.3$).
- PACTs' effects on Failure Modes combine by, essentially, multiplication of their complements. For example, if two PACTs have effects on the same Failure Mode of 0.1 and 0.2, then their combined effect is $(1 - (1 - 0.1)*(1 - 0.2)) = 0.28$

Decision-making is supported through several risk-related measures of risk computed from this model:

- The extent to which a Requirement is "at risk": computed by summing the impacts of Failure Modes on that Requirement.
- Requirement's attainment: its weight multiplied by (1 – minimum(1, its "at-risk" measure)). The minimum calculation is there because a requirement's "at risk" measure may exceed 1, indicating it is more than totally eliminated. For calculation of requirements attainment, such a requirement contributes zero attainment. However, the "at risk" measure gives an indication of the amount of work yet to do to achieve that requirement, so both measures have their use.
- Overall requirements attainment: computed by the summing the attainment of the requirements.
- The extent to which a Failure Mode is contributing to total risk: computed by summing over all the Requirements the Failure Mode's impact on the Requirement multiplied by the Requirement's weight.

DDP's quantitative risk-based model is the foundation of these measures.

Most *qualitative* approaches *cannot* be used to meaningfully "add up" risks (see Figure 1 for an inventive use of addition illustrating this point).

On occasion, there is a mismatch between the case at hand and the combination rules assumed by the model. Often, such mismatches can be handled by manual workarounds. For example, if the combination of two PACTs is not as effective as the model's combination rule would calculate, enter a third PACT to represent that combination, manually score its effects accordingly, and thereafter be careful to select at most one of those three PACTs (either of the individual ones, or this manually scored combination one).

MARTIN FEATHER

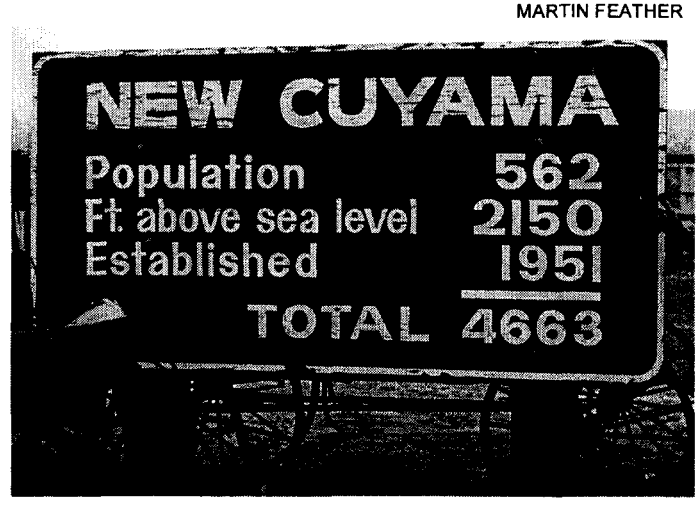

Fig. 1. Inventive use of addition on New Cuyama town sign

3 APPLICATION TO MODEL CHECKING CHALLENGES

For illustration, the model is shown being applied in an ongoing study of the challenges that impede the practical use of model checking technology, and what can be done to overcome those challenges. This example is selected to be of interest to RHAS workshop attendees.

The subsections that follow illustrate the kind of information used to populate the DDP model for this study, and the use of that model to begin investigation of its ramifications. This study is not yet complete, but the information is illustrative of the approach.

3.1 Population of the DDP model

The information required to populate the model was gathered in sessions involving model-checking experts who have experienced first-hand the excitement and the challenges of applying model checking to real-world problems.

3.1.1 Requirements

Two groups of Requirements were considered – the choices of artifacts to which model checking could be applied, and the choices of people who would apply model checking.

The tree of requirements is shown in Figure 2. This, and the other trees that follow, are partial screenshots taken from the DDP tool loaded with this model. The artifacts group of Requirements is the subtree whose root is at the top ("1:artifacts"), while the users groups of Requirements is the subtree whose root is towards the bottom ("20:who uses the tool"). Alongside each requirement is a checkbox (each of which is shown as unchecked in this figure). DDP users select requirement(s) for consideration by clicking to set those requirements' checkboxes to checked status. For example, to explore the case of model checking used for requirements validation (e.g., to validate a system's requirements against safety properties) to be done by the developer of the system itself, the DDP user would check the "5:validation" checkbox, and the "21:developers" checkbox.

Note: the numbering of items in trees can be switched to the familiar "tree" style (e.g., 1.2.1) if so desired. The linear numbering scheme shown in these screenshots is advantageous when using DDP, because it can be used to label items very concisely, as will become apparent shortly.

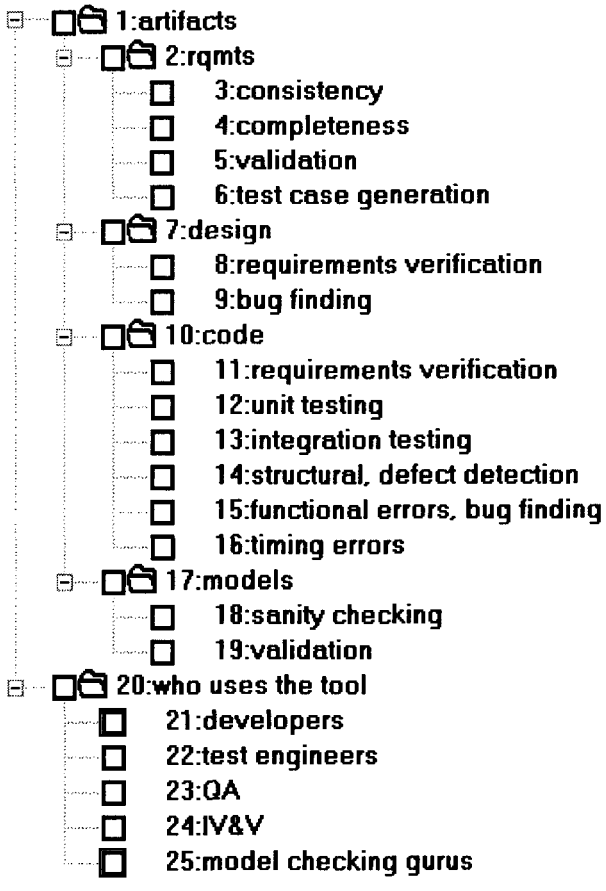

Fig. 2. Requirements that may drive the use model checking

3.1.2 *Failure Modes*

Two classes of Failure Modes were considered – the technical issues that impede use of model checking (e.g., state space explosion), and the social issues that impede use of model checking (e.g., resistance to learning new languages and tools, as would be required in most ways of using model checking). The categorization is used to spur thinking of the full range of problems that impede the use of model checking. It is not important that this be a perfect categorization – what does matter is that some serious impediment is not being overlooked. The tree of Failure Modes is shown in Figure 3.

3.1.3 *PACTs*

PACTs are the possible activities that could reduce the adverse impact of Failure Modes, and thereby lead to increased use of model checking. The ones considered are listed in Figure 4. These have not been arranged into any major categories, other than the small subtree of computing resources.

- 1:Technical issues
  - 2:state space explosion
  - 3:slow turnaround time
  - 4:notation that mc can't handle
    - 5:design notation incompatible with model checking
    - 6:property notation incompatible with model checking
  - 7:challenging generation of environment models
  - 8:unknown what applications domains are suitable
  - 9:unknown how much work it takes
  - 10:complexity of deciphering error traces
- 11:Social Issues
  - 12:resitance to learning new languages and tools
  - 13:need to have specification expertize
    - 14:modeling expertize (how to build the model)
    - 15:LTL etc expertize (how to specify the properties)
  - 16:well documented requirements are lacking
  - 17:beneficiaries not the ones who do it
  - 18:large effort of applying model checking
  - 19:knowledge of the application domain is required

Fig. 3. Failure Modes that may impede the use of model checking

- 1:tools for abstraction
- 2:tools for translation into mc Ls
- 3:hiring PhDs
- 4:training application engineers
- 5:increase computing resources
  - 6:chips get faster
  - 7:more memory
  - 8:parallel h/w
- 9:short training course for LTL el al
- 10:emphasize the unique role that m/c can play
- 11:cost of failure a driver (rqmt?)
- 12:specification patterns for properties
- 13:model checking provided as a "service"
- 14:Develop cost models
- 15:Case studies
- 16:Baselining & benchmarking
- 17:(Funded) partnerships with projects
- 18:Search heuristics
- 19:custom model checkers for programming Ls
- 20:compositional m/c
- 21:marketing
- 22:design for verification
- 23:tools for visualizing results
- 24:OUT OF SCOPE correcting flawed models
- 25:include mc into existing toolset
- 26:pick customers

Fig. 4. PACTs that may help use of model checking

3.1.4 *Quantitative relationships ("effects" and "impacts")*

A portion of the quantitative relationships between PACTs and Failure Modes is shown in Figure 5. These relationships are captured in a matrix whose rows are PACTs, and columns are Failure Modes. A cell entry indicates the strength of the effect of the row PACT at reducing the column Failure Mode. This is usually expressed as a number in the range [0, 1] where the extreme of 0 means no effect whatsoever, and the extreme of 1 means completely effective (i.e., eliminates the Failure Mode). A blank cell is equivalent to an entry of 0. An intermediate value, k say, means it reduces the Failure Mode by the proportion k. For example, the highlighted cell is at the intersection of the PACT row "short training course for LTL et al", and the Failure Mode column "property notation incompatible with model checking". The cell value, 0.99, means the application of that PACT will reduce by 99% that Failure Mode. The assumption underpinning this high value is that the aspects of LTL or similar formal property notations needed for model checking can be very effectively taught to model checking practitioners.

PACT x FM Col = property notation incompatible with model checking
Row = short training course for LTL el al

		FMs	[-]Technical								[-]Socie	
		FMs	state spec expl	slow turna time	[illegible]		chall gene of envir	unkn what appli dom	unkn how much work	comp of deci error	resit to learn new	[- h s e:
		FMs			desi	[illegible]						m
PACTs	PACTs	FoM\R	1	1	1	1	1	1	1	1	1	1
tools for		3.11	0.7	0.7			0.7	0.1		-0.3		*(
tools for		7.66		0.9	0.95			0.1	0.5	0.7	0.99	0.
hiring		9.9	0.8	0.1	0.9	0.9	0.8	0.6	0.3	0.99	0.9	0.
training		11.1	0.3	0.7	0.9	0.99	0.9	0.9	0.3	0.8	*	0.
[-]increase	chips	1.14	0.1	0.7						0.1		
computing	more	1.2	0.6	0.3								
resources	paralle	1.82	0.7	0.7								
short		2.03				0.99						
emphasize		1.39						0.7			0.3	
cost of		1.39						0.7			0.3	
specificatic		2.05		0.1		0.7					0.3	

Fig. 5. The effects of PACTs on reducing Failure Modes

When unsure of what value to enter into a cell, certain non-numeric strings are allowed as cell values (for example, the "*" in the fourth white row down and rightmost entirely visible column). Such non-numeric entries play no role in the quantitative calculations, but do serve as visible placeholders for work yet to be done.

On occasion, a PACT may actually make the situation *worse*. This is indicated by giving a negative number, in the range [-1 0), as the strength of the effect. The magnitude of this negative number indicates the likelihood of inducing the Failure Mode. For example, the topmost white row (whose PACT is "tools for abstraction") has a value of –0.3 in the cell for the Failure Mode "complexity of deciphering error traces", on the grounds that abstraction moves a specification further from the system, rendering deciphering of error traces that result from model checking somewhat more problematic.

The kinds of numbers visible in the fragment of the "effect" matrix of Figure 5 are representative of those we see entered for assessments of advanced technologies. The nature of these assessments precludes high precision entries. Instead, we make do with these coarse estimates. Nevertheless, the aggregation of these coarse estimates can point to clear decisions.

The information that connects Requirements to Failure Modes is captured in a similar manner in the DDP "impacts" matrix (in the interest of brevity, not shown here).

3.2 Using the DDP Model

Once the DDP model has been populated, it can be used to reveal the ramifications of the combined set of information, and ultimately to make decisions. A brief example scenario follows.

Suppose that the objective is to understand the use of model checking for a system's requirements validation, to be done by the developers of the system itself. This is input to the DDP model by checking the check boxes in the requirements tree for items 5 (validation, within rqmts, within artifacts) and 21 (developers within who uses the tool) of the requirements tree shown earlier in Figure 2.

The DDP tool automatically computes the various risk measures, and offers several ways to scrutinize the results. A key one is shown in Figure 6, where each bar corresponds to a Failure Mode, and the height corresponds to that Failure Mode's contribution to risk (the vertical axis is a log scale). These are shown sorted in descending order. We see that Failure Mode number 9 ("unknown how much work it takes") is the highest ranked one, while Failure Mode number 19 ("knowledge of the application domain is required") is the lowest ranked one. It is low because the system developer, who already has that knowledge, is doing the application.

We can explore ways to reduce risks by selecting various PACTs. After each change, the DDP tool automatically recomputes its various risk measures, and redisplays the results. For a model such as this, with a few dozen at most of each kind of item, recomputation is very rapid.

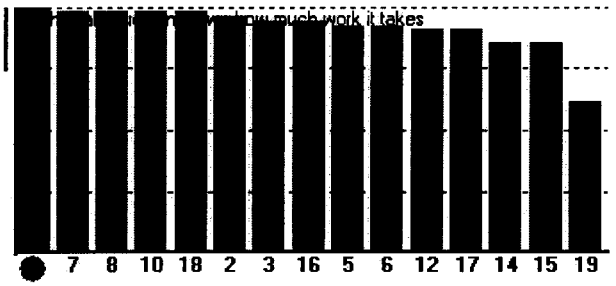

Fig. 6. Failure Modes in Decreasing Order

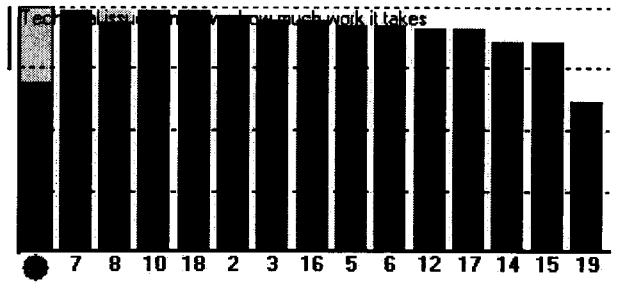

Fig. 7. Failure Modes after PACT 14 selection

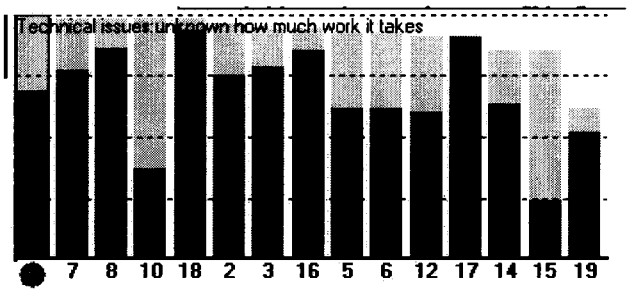

Fig. 8. Failure Modes after PACT 13 selection

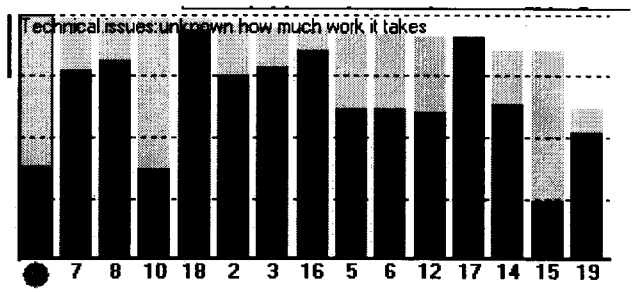

Fig. 9. Failure Modes after PACT selections 13 & 14

Figures 6, 7, 8 and 9 show several displays of the risk bar chart corresponding to different selections of PACTs.

- The status when no PACTs have been applied is shown in Figure 6. The risks are sorted in descending order (the circle obscuring the number 9 below the leftmost bar is the tool's mechanism for drawing attention to a particular item).
- Selecting the PACT 14 "develop cost models", leads to the risk profile shown in Figure 7. The dark portion of each bar shows its reduced level, while the light gray portion shows the cumulative risk reduction. (In the tool we prefer to use vivid colors over shades of gray). Note that Failure Mode number 9 has been substantially reduced, and also Failure Mode number 8 has been slightly reduced. This is because the selected PACT has an effect on both those Failure Modes. Indeed, it is common for PACTs to have multiple such effects.
- The alternative of selecting PACT 13 "model checking provided as a "service"" has the effect shown in Figure 8. This one PACT reduces every one of the Failure Modes to some extent!
- The combined effect of selecting both PACTs 13 and 14 is shown in Figure 9. We can see their combined effect on Failure Mode number 9, and to a lesser extent, on Failure Mode number 8.

Of course, different PACTs will have different costs, so in selecting them, users must balance their beneficial affects against those costs.

A populated DDP model can be used to explore the cost and benefit ramifications of these choices.

The DDP tool provides assistance to users during this selection process via further visualizations. A key one is shown in Figure 10, where each row corresponds to a Failure Mode, listed here in descending order of magnitude from top to bottom. The dark bar at the left of each row indicates the magnitude of the Failure Mode. Alongside it are smaller numbered rectangles, each corresponding to a PACT that has some non-zero effect on that Failure Mode. For example, the sixth Failure Mode down, shown highlighted within the prominent border, has listed alongside it PACTs numbered 1, 3, 4, 13, 17 and 19. Each PACT's rectangle is shaded to indicate the magnitude of its effect on the Failure Mode. For example, PACT number 19 has no visible shading, so its effect must be very small (but non-zero, since otherwise it would not have been listed), while PACT number 17 has a sizeable gray shaded area, indicating its correspondingly large effect. Checkboxes indicate the selected status for each PACT. Finally, positioning the cursor over a PACT causes *all* instances of that PACT to be highlighted by a surrounding border. In the figure we see this for PACT number 4, which is highlighted in many places. In the tool, judicious use of color and dynamic effects (e.g., blinking) renders these more effectively than is conveyed by this static grayscale figure.

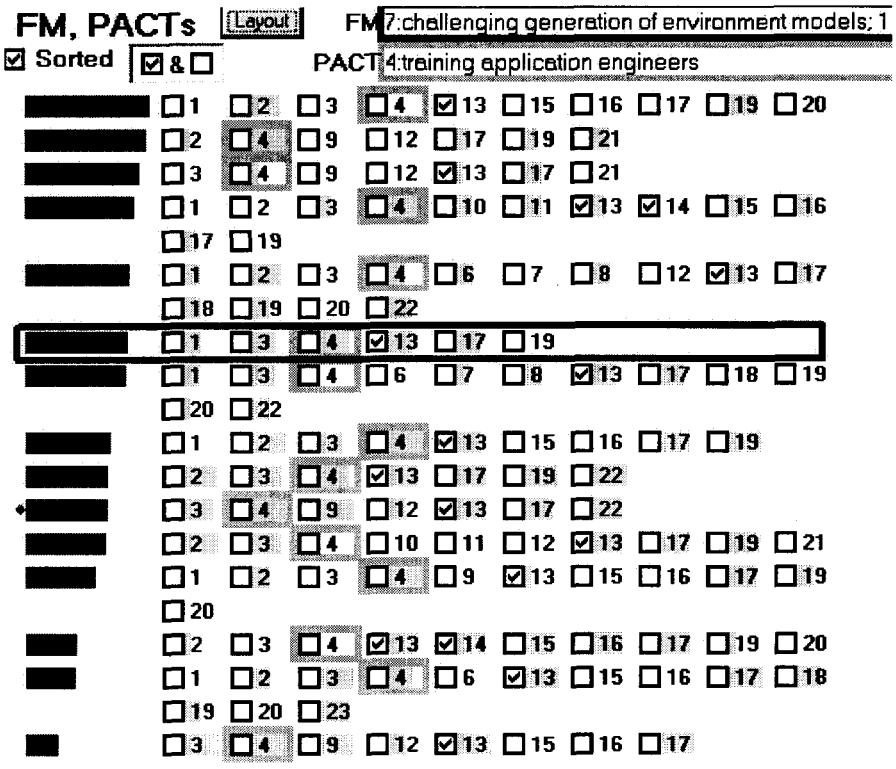

Fig. 10. An alternative visualization of Failure Modes and the PACTs that address them

If users have populated the model with cost information (what each PACT would cost to perform), it is possible to treat the PACT selection as an optimization problem. For example, for a given amount of resources, select PACTs to maximize requirements attainment while remaining within that resource limit [5].

4 POTENTIAL FOR APPLICATION TO HIGH ASSURANCE SYSTEMS

The purpose of this position paper is to suggest that this same quantitative risk-based approach be considered for application to High Assurance Systems.

Our experience in applying this model to advanced technologies and systems suggests that it has the following benefits:

- Encourages the elicitation of risks, and therefore diminishes the danger of overlooking risks.
- Prioritizes risks so that resources can be directed to the most significant risks.
- Permits the exploration of alternative risk mitigation strategies (selections of PACTs)
- Reveals the purpose of risk mitigations (PACTs), namely the reduction of Failure Modes, particularly the more risky of these.
- Gives insight into the requirements, indicating which of them are proving the most problematic to attain (because they are at risk due to Failure Modes whose reduction is expensive or impossible).
- Point the way to research areas in need of further investigation.

All of these benefits are important to High Assurance Systems. The area lies at the intersection of several disciplines, so the pooling of experts' knowledge and best practices is both challenging and essential.

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. Special thanks are due to researchers both at JPL and at the NASA Ames Research Center Automated Software Engineering Group for sharing their expertise with model checking.

Further information on the DDP process and tool can be found at: http://ddptool.jpl.nasa.gov

REFERENCES

[1] Akao, Y. 1990 "*Quality Function Deployment*", Productivity Press, Cambridge, Massachusetts.

[2] Bertrand, P., Darimont, R., Delor, E., Massonet, P., and van Lamsweerde, A., 1998, "GRAIL/KAOS: an environment for goal driven requirements engineering", 20th Int. Conference on Software Engineering.

[3] Boehm, B., Bose, P., Horowitz, E., and Lee, M., 1994, "Software Requirements as Negotiated Win Conditions", Proceedings 1st International Conference on Requirements Engineering, Colorado Springs, Colorado, pp. 74-83.

[4] Cornford, S.L. , Feather, M.S., & Hicks, K.A. "DDP – A tool for life-cycle risk management", IEEE *Aerospace Conference*, Big Sky, Montana, Mar 2001, pp. 441-451.

[5] Feather, M.S. & Menzies, T. "Converging on the Optimal Attainment of Requirements", to appear in the Proceedings of the IEEE Joint International Conference on Requirements Engineering, Essen, Germany, Sept. 2002.

[6] Heitmeyer, C., Labaw, B., & Kiskis, D., "Consistency Checking of SCR-Style Requirements Specifications", Proceedings 2nd IEEE International Symposium on Requirements Engineering, York England, 1995, pp. 56-63.

[7] Holzmann, G.J., "The Spin Model Checker SPIN", IEEE Trans. on Software Engineering, Vol. 23, No. 5, May 1997, pp. 279-295.

[8] Jackson, D., Schechter, I. & Shlyakhter, I., "Alcoa: the Alloy Constraint Analyzer", Proceedings of the 2000 International Conference on Software Engineering, Limerick, Ireland, 2000, pp. 739-733.

[9] Mylopoulos, J., Chung, L., Liao, S., Wang, H., and Yu, E., 2001, "Exploring Alternatives during Requirements Analysis", IEEE Software 18(1), pp. 92-96.

[10] Raz, O. & Shaw, M. "An Approach to Preserving Sufficient Correctness in Open Resource Coalitions", Proceedings of the 10th International Workshop on Software Specification and Design, San Diego, California, Nov. 2000, pp. 159-170.

[11] Shankar, N., Owre, S. & Rushby, J.M., "The PVS Proof Checker: A Reference Manual." Computer Science Laboratory, SRI International, Menlo Park, CA, February 1993.

[12] Vesely, W.E., Goldberg, F.F., Roberts, N.H. & Haasl, D.F., "Fault Tree Handbook", U.S. Nuclear Regulatory Commission NUREG-0492, 1981.