Assurance Optimization

Martin S. Feather
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr
Pasadena CA 91109-8099
Martin.S.Feather@Jpl.Nasa.Gov

Tim Menzies
Lane Department of Computer Science & Electrical Engineering,
West Virginia University
PO Box 6109
Morgantown WV 26506-6109
tim@menzies.com

The purpose of assurance activities is to reduce risk, thereby ensuring requirements. However, assurance activities incur costs such as budget, schedule, mass (e.g., radiation shielding), etc. The selection of assurance activities to perform is thus an **assurance optimization** problem. For example, for a given budget, selection of the set of assurance activities that will minimize risk (i.e., maximize requirements). Alternately, for a given level of requirements, selection of the minimal cost set of assurance activities that will achieve that level of requirements.

Our work demonstrates a novel technique to assurance optimization. Users indicate their preferences by assigning relative weights to solution classes (e.g., weighting highly a solution class that is low risk and at or below the users' target cost threshold, and weighing less highly a solution class that is low risk but slightly above the users' target cost threshold). The technique uses machine learning to identify the critical choices that lead to contrastingly different classifications. The net result is near-optimal solutions to assurance optimization problems, even in huge search spaces. Furthermore, the technique reveals which of the many decisions are the most crucial to achieving those optimal results.

The technique is realized in an operational computer program. Experiments on assurance datasets of considerable size show promising empirical results. For example, we experimented on an assurance model that arose from a study of an advanced spacecraft technology. This assurance model contained 99 options of risk mitigation actions, i.e. $2^{99}(\approx 10^{30})$ possible combinations of these actions. Our technique was successful at determining the 16 actions most crucial to perform, the 14 actions most crucial to *not* perform, and the remaining 66 actions whose influence was the least on the quality of the solution.

A literature review [7], a mathematical analysis [6] and experiments on numerous case studies suggest this technique has broad applicability.

1. The technique

Our technique follows an iterative cycle of execution, learning and decision-making. This is shown in Figure 1. The assurance model is executed to generate a random set of examples. Our implementation, TAR2, is the "contrast set learner" that identifies the critical decision alternatives. The human experts then scrutinize these, and select the decision set of their choice. This selection is from among the several leading candidates that the learner has identified. This gives the human experts the opportunity to inject strategic guidance into this overall decision process. After several cycles, the result is a set of near-optimal assurance actions that the experts feel personally committed to.

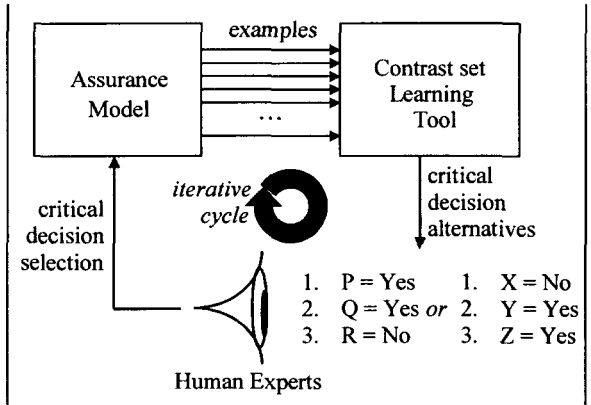


Figure 1. Execution/Learning/Decision Cycle

Alternate approaches (e.g. standard decision tree learning or genetic algorithms) are non-iterative and, to a user, the resulting conclusions are somewhat opaque in the sense that they don't know where they came from. Also,

- Experiments with decision tree learners resulted in theories too large to read.
- Experiments with genetic algorithms in this domain proposed far more actions than our method.

2. Experiments

We report briefly upon two sizable experiments with this technique. The first experiment dealt with a large assurance model built using NASA's Defect Detection and Prevention (DDP), the second with COCOMO-II - a public domain software cost/risk model.

2.1 DDP Experiment

NASA's Defect Detection and Prevention (DDP) process uses risk as a "currency" to enable trades between cost of assurance and attainment of requirements [1,2].

A representative real-life DDP dataset was used for the assurance optimization experiment [3]. This dataset contains 99 different assurance activities, each of which can be chosen (or not) independently of the others. Thus in principle the decision space is $2^{99}(\approx 10^{30})$ possible combinations. The figure on the next page shows the before-and-after results of the experiment.

- Before: Each of the tiny black points indicates a random selection out of the 2^{99} possible combinations of assurance activities. The wide spread of these points indicates that it is possible to waste a lot of resources and/or fail to maximize requirements.
- After: TAR2, the contrast set learner tool, was able to identify the 30 most critical of those 99 decisions, namely the 16 actions most crucial to perform, and the 14 actions most crucial to *not* perform. Making the decisions reduced to the much smaller white area towards the high-benefit low-cost quadrant of the chart. The variability remaining within the white region results from the other 66 as yet unconstrained decisions. Picking a near-optimal solution from within this region is straightforward.

2.2 COCOMO Experiment

Menzies&Sinsel used an early version of TAR2 to explore a space of 54 million options and find the two key variables that could most control the system [5].

In that application, a COCOMO-based tool [4] was used to evaluate the risk that a NASA software project would suffer from develop-time overrun. The tool used in that study required a guesstimate of the source lines of code (SLOC) in the system and certain internal tuning parameters, which ideally would be learnt from historical data. Lacking such data, Menzies & Sinsel used three guesses for SLOC and three sets of tunings that they took from the literature. Competing stakeholders proposed 11 changes to a project; i.e. $2^{11} = 2048$ options. Some of the project features were unclear and, for those features, project managers could only offer ranges for the required inputs to the COCOMO-based tool. These ranges offered 2930 possible combinations for the inputs. When combined with the other uncertainties, this generated a space of 54 million possibilities:

54 million=
*(2930*2^{11}*three guesses for SLOC * three tunings).*

Faced with this overdose of possibilities, Menzies & Sinsel performed 50,000 Monte Carlo simulations where the inputs were taken from the 54 million possibilities. TAR2 found that, of the 11 proposed changes, seven had little impact. Of the remaining four proposed changes, two were found by TAR2 to be clearly superior.

3. 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, and at the University of British Columbia, Vancouver, Canada. We thank Steve Cornford for granting us access to experiment on the DDP dataset.

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.

4. References

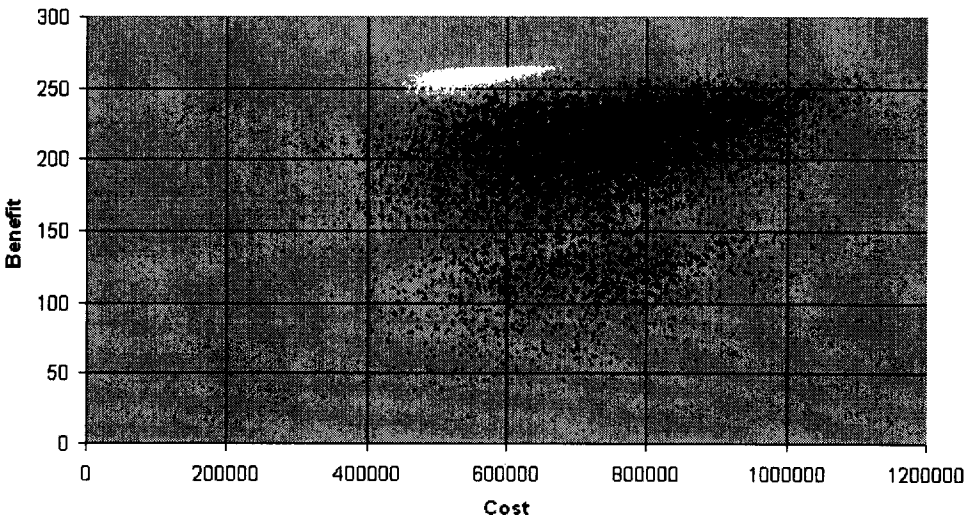
[1] S.L. Cornford, M.S. Feather & K.A. Hicks. "DDP – A tool for life-cycle risk management", IEEE *Aerospace Conference*, Big Sky, Montana, Mar 2001, pp. 441-451.

[2] S.L. Cornford, J. Dunphy, & M.S. Feather. "Optimizing the Design of end-to-end Spacecraft Systems using risk as a currency", *IEEE Aerospace Conference.*, Big Sky, Montana, Mar 2002.

[3] M.S. Feather & T. Menzies. "Converging on the Optimal Attainment of Requirements", in submission.

[4] [Madachy 1997] R. Madachy "Heuristic Risk Assessment Using Cost Factors" , IEEE Software, 1997, 14, 3, pages 51-59, May

[5] [Menzies2000] T. Menzies and E. Sinsel, "Practical Large Scale What-if Queries: Case Studies with Software Risk Assessment" Proceedings ASE 2000

[6] [Menzies 2001] T. Menzies and H. Singh. "Many Maybes Mean (Mostly) the Same Thing", 2nd International Workshop on Soft Computing applied to Software Engineering (Netherlands), February, 2001

[7] [Menzies & Cukic, 2000] Menzies & Cukic. "Adequacy of Limited Testing for Knowledge Based Systems", International Journal on Artificial Intelligence Tools (IJAIT), June, 20