Application of a broad-spectrum quantitative requirements model to early-lifecycle decision making

Martin S. Feather*, Steven L. Cornford*, Kenneth A. Hicks*,
James D. Kiper** and Tim Menzies***

** Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA*
*** Miami University, Oxford, OH*
**** West Virginia University, Morgantown, WV*

{Martin.S.Feather, Steven.L.Cornford, Kenneth.A.Hicks}@jpl.nasa.gov
kiperjd@muohio.edu, tim@menzies.us

Abstract

During the early phases of projects' lifecycles, detailed information is scarce; yet there is frequent need to make key decisions, especially those concerning tradeoffs among quality requirements. Trading among competing concerns occurs in many fields, e.g. systems engineering, hardware engineering, and software engineering. Here, we report on our design and use of a model to help make requirements decisions that applies to all of these fields. To do so, we employ a model based upon coarse quantification of the factors involved. We illustrate our approach with fragments taken from some of our studies of software technologies.

Keywords: Decision Support, Requirements Analysis, Requirements Elicitation; Management: Cost Estimation, Risk Management; Information Visualization, Optimization, Data Mining.

Introduction

The Jet Propulsion Laboratory designs, builds and operates spacecraft for NASA. Our work applies to the early phases of spacecraft projects, and to the technology developments that will, it is hoped, mature successfully and be utilized on future spacecraft. Key decisions are made during these early phases, such as selection of which (out of many promising) technologies to pursue further, selection of the mission(s) on which to seek to introduce technologies, and planning the way forward to mature those technologies from research prototypes through to mission-ready capabilities that project managers will be willing to adopt. Our work supports decision making in these crucial early stages. These spacecraft utilize multiple technologies, software among them, and increasingly many of our studies are focused on software technologies.

The approach we follow is founded on the use of a model that represents the many factors involved in technology utilization, and how they interact. For each application we populate the model with information elicited from relevant spacecraft stakeholders during group sessions. Custom software that we have developed is used to store the model, compute information from it, and present that information (through cogent visualizations) to assist the stakeholders in their decision making and detailed planning.

Although our experience is on space mission applications, we believe there are obvious parallels in many terrestrial technology projects. There are often multiple potential applications of a new technology, prompting the decision of which application(s) to pursue first. There are commonly tradeoff decisions to be made about how adequately a technology will need to attain various quality requirements. These decisions will depend on the quality needs of the intended application(s) of that technology and how much it will cost to develop the technology to attain various quality levels. When the technology is part of a larger system, there will typically be options for how to allocate requirements between the technology itself and the surrounding system. Our approach also encompasses process requirements as well as product requirements, namely requirements on a technology's development as well as on the technology that will result from that development. For large and complex projects the process issues – manageability, conformity to accepted ways of doing business, availability of sufficiently skilled development personnel, etc. – are just as necessary as functional performance issues.

A risk-based requirements model

In this section we describe the background to our modeling approach, and its information content.

Steven Cornford invented our approach to aid hardware assurance [1]. He named it "Defect Detection and Prevention" (DDP), reflecting its purpose to help cost-effectively select assurance activities (tests, analyses, assembly procedures, etc.) to prevent introduction of hardware defects and/or detect their presence in time to correct them. However, the approach has since found its niche assisting early-lifecycle decision making, primarily to guide the advancement of promising novel technologies that have emerged from the research lab towards adoption by spacecraft projects [2]. It has also been applied to entire research or mission portfolios, where the choices are between which avenues of research to pursue and which missions to plan [3]. All these application areas make use of the same DDP model, even though the nomenclature (especially the "Defects" of DDP's name) may no longer be appropriate. The early-lifecycle applications to novel technologies are the focus of this paper, since requirements of all kinds feature prominently therein.

A DDP model is populated by instances of three kinds of concepts: *Requirements* – the needs (functional and quality – a.k.a. "non-functional") of the project, system or technology, and its development, *Risks* – what could occur to impede the attainment of Requirements, and *Mitigations* – what could be done to reduce Risks.

Instances of these in a DDP model are assigned values for the following attributes:

- Each Requirement has a *Weight*, its relative importance: Weight(Requirement):float $\in$ [0,maxfloat] Usually we pick values from an easy-to-remember scale such as 0-100. The purpose of assigning weights to Requirements is to guide tradeoff and descoping decisions when, as is often the case, not all of them can be attained (or it would be overly expensive to do so). Attainment of higher weighted Requirements will be valued more than of lower weighted Requirements.

- Each Risk has an *A-Priori Likelihood*, its probability of occurrence were no Mitigations reducing it: APL(Risk):float $\in$ [0,1]

- Each Mitigation has a *Cost*, the cost of performing it – usually a financial cost, but other resources can also be considered, such as schedule, memory utilization, and bandwidth: Cost(Mitigation):float $\in$ [0,maxfloat], and a Boolean indicating whether or not it is selected: Selected(Mitigation):boolean

There are also the following relationships among a DDP model's instances:

- Risks are quantitatively related to Requirements to indicate how much each Risk, should it occur, *Impacts* (i.e., detracts from the attainment of) each Requirement. Each such value is expressed as a proportion of the Requirement's attainment that will be lost were that Risk to occur (e.g., 0.1 means one-tenth of the Requirement's attainment will be lost): Impact(Risk,Requirement):float $\in$ [0,1]

- Mitigations are quantitatively related to Risks, to indicate how much of a Risk-reducing *Effect* a Mitigation, should it be selected, has on reducing each Risk. Each such value is expressed as a proportion of the reduction that Mitigation will achieve (e.g., 0.1 means it reduces by one-tenth): Effect(Mitigation,Risk):float $\in$ [0,1]

Overall, selected Mitigations incur costs and reduce Risks, leading to increased attainment of Requirements. The following formulae define the calculations of these for a given selection of Mitigations:

- A Risk's *Likelihood* is its a-priori likelihood, diminished by the selected Mitigations. For a Risk K:

  Likelihood(K : Risk) = APL(K) * $\prod$(M $\in$ Mitigations) : If(Selected(M), 1 – Effect(M,K), 1)

  Thus, multiple Mitigations effects on a single Risk act like a series of filters, each removing some fraction of the Risk. For example, if a Risk has an a-priori likelihood of 1.0, then two Mitigations with effect proportions of 0.1 and 0.7 respectively act as follows: the first filters out 0.1 of the Risk's likelihood, reducing it from 1.0 to 0.9; the second filters out 0.7 of what remains, reducing it from 0.9 to 0.27. Note: we omit here the details of phenomena such as Mitigations that reduce Risk *impacts* rather than likelihoods, and Mitigations that *increase* Risks – for these details, see [4].

- A Requirement's *Attainment* is its weight, diminished by the Risks that impact it. For a Requirement R:

Attainment(R : Requirement) = Weight(R) * (1 – Min(1, (∑(K ∈ Risks) : Likelihood(K) * Impact(K,R)))
Thus, multiple Risks' impacts on the same Requirement simply add up.

- The *TotalCost*:

TotalCost = ∑(M ∈ Mitigations) : If(Selected(M), Cost(M), 0)

- The *TotalAttainment*:

TotalAttainment = ∑(R ∈ Requirements) : Attainment(R)

One of the primary outcomes of a DDP study is the determination of which Mitigations to select. In most situations the total cost of all the postulated Mitigations far exceeds the available budget (in whatever kinds of resources are being modeled), so arriving at a cost-effective selection of Mitigations is crucial.

An example

One of our studies concerned a GUI-based environment for prototyping of control systems. The study examined the idea of machine-generating from the prototype a stand-along executable to serve as the actual spacecraft control system, rather than recoding it from scratch. We illustrate detail of the DDP model and its quantitative calculations on a fragment from this study.

One of the quality concerns was about how this approach would affect the speed of the resultant code, specifically, its speed when running real-time control loops. To represent the needs in this respect, two DDP Requirements were created as follows: one to represent the speed requirement when there are few and/or slow control loops, the other when there are many and/or fast control loops. In the interests of conciseness, we refer to the former as "few/slow", and the latter as "many/fast". Among the Risks identified as potentially threatening attainment of these requirements was that of heap fragmentation. Among the Mitigations identified as helping address this was that of built-in garbage collection. The DDP equations for these were as follows:

Requirements:

- "few/slow" – Weight("few/slow") = 100
- "many/fast" – Weight("many/fast") = 100

Risk:

- "heap frag" – APL("heap frag") = 1

Mitigation:

- "built-in GC" – Cost("built-in garbage collection") = $10,000

Impacts:

- Impact("heap frag", "few/slow") = 0.1
- Impact("heap frag", "many/fast") = 0.99

Effect:

- Effect("built-in garbage collection", "heap frag") = 0.9

From these, the following indicates calculation of the Attainment for the requirements for whether or not "built-in GC" is selected:

- "built-in garbage collection" is *not* selected:

Likelihood("heap frag") = APL("heap frag") = 1

Attainment("few/slow")
= Weight("few/slow") * (1 - Min(1, Likelihood("heap frag") * Impact("heap frag", "few/slow")))

= 100 * (1 - Min(1, 1*0.1)) = 100 * (1 – 0.1) = 100 * 0.9 = 90

Attainment(“many/fast”)
= Weight(“many/fast”)*(1 - Min(1, Likelihood(“heap frag”) * Impact(“heap frag”, “many/fast”)))
= 100 * (1 - Min(1, 1*0.99)) = 100 * (1 – 0.99) = 100*0.01 = 1

- “built-in garbage collection” *is* selected:

Likelihood(“heap frag”) = APL(“heap frag”) * (1- Effect(“built-in GC”, “heap frag”))
= 1 * (1 - 0.9) = 0.1

Attainment(“few/slow”)
= Weight(“few/slow”) * (1-Min(1, Likelihood(“heap frag”) * Impact(“heap frag”, “few/slow”)))
= 100 * (1 - Min(1, 0.1*0.1)) = 100 * (1 – 0.01) = 100*0.99 = 99

Attainment(“many/fast”)
= Weight(“many/fast”) * (1 - Min(1, Likelihood(“heap frag”) * Impact(“heap frag”, “many/fast”)))
= 100 * (1 - Min(1, 0.1*0.99)) = 100 * (1 – 0.099) = 100*0.901 = 90.1

These calculations show that if the target application needs the machine-generated software to operate many and/or fast control loops, then adoption of the “built-in garbage collection” mitigation is warranted.

From the above it is apparent that the DDP model calculations are relatively straightforward. The complexity arises from the volume and intertwinedness of data that goes into a typical DDP model. To illustrate this, Figure 1 shows the entire Requirements-Risks-Mitigations structure constructed for this study. There are 35 Requirements (tiny circles along the top), over 100 each of Risks (tiny circles in the middle row) and Mitigations (tiny circles along the bottom row), and over 600 each of Impacts (quantitatively scored red links) and Effects (quantitatively scored green links).

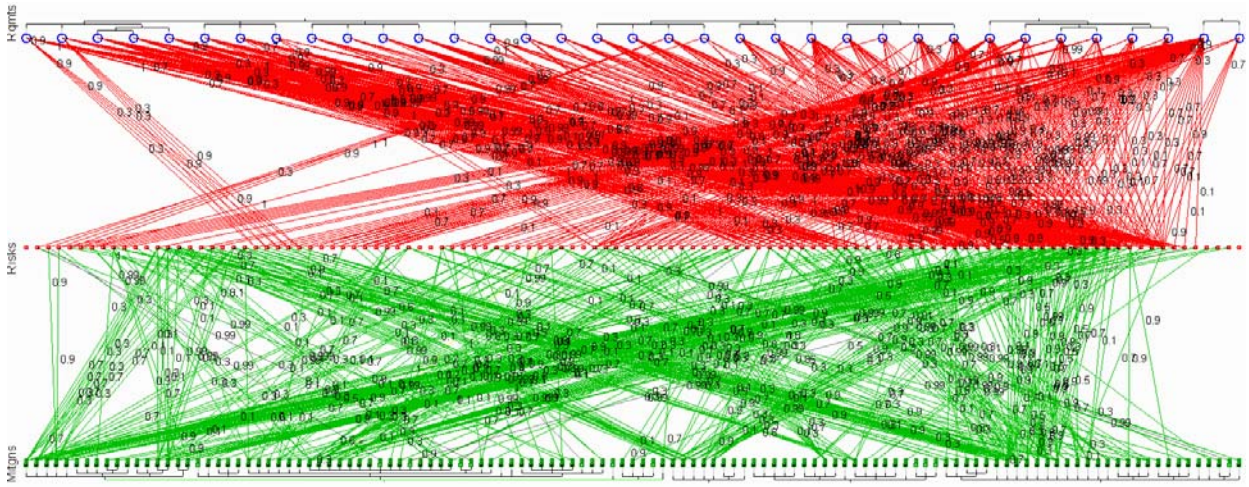

Figure 1 - information of an actual DDP model

The art of using DDP to represent Requirements

When utilizing DDP for technology studies, a wide range of requirements need to be simultaneously considered – *properties of the technology itself*, both functional (e.g., “computes distance to ground”) and non-functional (e.g., “computes in 10 seconds”), *properties of the environment surrounding the technology*, both functional (e.g., “provides radar data to the technology”) and non-functional (“provides at least 10% of CPU cycles to the technology”), and *properties of the development process* (e.g., “predictability of cost and schedule”; “ability to track development progress”) that will be followed to yield the instance of the technology to be used.

DDP’s capacity to encompass properties of the technology *and* of its environment (in the terminology of [5], the former are “optative” requirements, the latter “indicative” requirements) is useful because it allows tradeoff

decisions to be made between them. This illustrates accommodation of the intertwining between specification and implementation reported in [6]. The need to consider process requirements is useful because for large and complex projects the process issues – manageability, conformity to accepted ways of doing business, availability of sufficiently skilled development personnel, etc. – are just as necessary as functional performance issues.

DDP's concept of Requirements requires only that each Requirement be described in such a way that all stakeholders have a common understanding of what it means, and be given a "weight" representing its relative importance. This allows it to coarsely represent all the aforementioned variety of properties, quality requirements included. However, there are some implications of this coarse representation:

- DDP Requirements are discrete, not continuous. In the previous section's fragmentary example we used a DDP Requirement to represent "the speed requirements when there are few and/or slow control loops" – during the actual DDP session we would ensure a common understanding of the precise meaning (e.g., number of control loops divided by seconds-per-loop is no more than…). Note, however, that our DDP Requirement models a single point in the potentially continuous space of number of loops divided by seconds-per-loop. In our actual study we modeled just two points, "few/slow" and "many/fast" – chosen to be distinct capability regimes of particular interest for the postulated spacecraft applications. This is typical of our studies: for a given quality measure, a handful (at most) of individual DDP Requirements are used to represent discrete points in some continuous range.

- DDP Requirements lack an attribute of cost (e.g., if a requirement were being used to represent a potential "feature", one would expect there to be a cost associated with providing that feature). Other requirements tradeoff methods commonly ascribe costs directly to requirements, for example as seen in the cost/benefit approach of [7]. In DDP we achieve this same effect indirectly – we assume that a DDP Requirement will be met unless there are DDP Risks that detract from its attainment; to decrease those Risks we select DDP Mitigations, which do have costs associated with them. At first glance this seems convoluted, but in fact allows us to represent a variety of approaches. For example, if some level of computational performance is required, we are able to represent alternate solutions such as hosting the processing on a more powerful computer or developing faster algorithms.

- We do not model directly a set of Requirements being mutually incompatible. Rather, we typically find the limiting factor to be the cost of what it would take to address all the Risks that threaten those Requirements. For example, "security" and "response time" may be perceived as mutually incompatible (think of virus checkers hogging CPU cycles), but both can be had with a sufficiently powerful computer.

- It is important that the DDP notion of "Risk" be used to encompass all kinds of potential impediments to attainment of Requirements, not just of mechanism breakage and software bugs. Again, DDP's nomenclature may be misleading – other work has used the term "obstacles" [8]. Indeed, as a guide to identifying Risks, we often begin by thinking of a Requirement, and imagining what could cause it to fail to be attained. We share with risk assessment approaches the idea that risks have likelihoods, which can be reduced by mitigations. However, although project risk management approaches often advocate scoring risks against just three factors: "cost", "schedule" and "performance", we find these too abstract to serve as Requirements for the kind of decision making we wish to support. We decompose these further (e.g., consider separately bounds on run-time memory size, and storage size of the executable itself). Note, however, that we do not descend to the level of detailed "shall" statements that characterize a well-worked out design; those emerge later in the lifecycle, past the point where DDP is typically applied.

Overall, DDP favors *breadth* over *fidelity* of representation. Breadth has proven useful to allow us to take into account a wide range of relevant aspects in early lifecycle decision making, of which there are surprisingly many.

Experience using DDP

To populate a DDP model we use a series of facilitated group sessions in which all key stakeholders participate. As participants proffer information (e.g., Requirements and their relative weights), we enter it on-the-fly into the DDP model, and project the aggregation of this information on a screen visible to all. The DDP software that we have developed offers several visualizations suited to displaying this information. This allows the group to see how they are progressing, encourages them to identify mistakes and suggest corrections, and triggers them to be wide-ranging

in their thinking (e.g., the mention of a Risk to one kind of Requirement might serve to trigger another participant to think of an analogous Risk to a different Requirement). It also allows for the discovery and resolution of disagreements among participants' views. For example, if participants disagree about a Risk's impact on a Requirement, we encourage them to raise the issue immediately. We often find that disagreement stems from trying to assess at too high level of abstraction (e.g., assess a Risk's impact on a very general Requirement), and is resolved by decomposing into finer details (e.g., decomposing a general Requirement into its constituent pieces). Thus, we use disagreement constructively, as a guide to where we need to delve into more detail to adequately capture important distinctions. We also vary the depth to match the problem at hand – for example, when assessing a novel technology, we delve into more detail on the Risks pertaining to the most novel aspects of that technology as compared with the Risks pertaining to well-understood aspects.

It is apparent that populating a DDP model can involve a non-trivial amount of effort – is it worth it? We have been conducting DDP studies since 1999, in which time we have performed several per year. Based on these experiences, we make the following observations:

The *cost* of performing a DDP study is the time of the stakeholders' participation, to build the model, and to perform decision making informed by the model. When our studies are of novel technologies there is little historical data to draw from, so model-population – identification of Requirements, Risks and Mitigations, and of the Impact and Effect links among them – is a human-intensive elicitation process. In particular, elicitation of Impacts requires consideration of all pairs of Risks and Requirements (and similarly elicitation of Effects requires consideration of all pairs of Mitigations and Risks), so these steps are major time-sinks. While there are some shortcuts we can often take, such as ruling out sub-areas (where a whole subset of Risks can be identified as not impacting a subset of Requirements, without requiring their individual consideration), and decomposing the effort into different areas of expertise (so that different stakeholders can in parallel provide inputs for different areas), DDP model population remains a relatively labor-intensive process. A typical technology study involves from ten to twenty participants, all of them participating in several (two to four) half-day sessions. Thus, total labor time can be as much as several hundred hours.

The *benefit* of performing a DDP study is harder to quantify, but our experiences to date suggest that in most instances it substantially outweighs the cost. The nature of the benefit has varied from study to study, and we have seen several instances of each of the following:

- Realization of a mismatch between the technology developers' belief of what the application requirements are, and the would-be customers' actual needs; since DDP model building usually starts with Requirements, this is realized early on, allowing the mismatch to be corrected and the study to proceed.

- More subtly, a study may reveal mismatch between different stakeholders' perceptions of the cost of achieving certain requirements. There is the tendency for customers to declare all their requirements to be essential; however as they come to realize that some requirements may cost much more than others to attain, they become motivated to reconsider their stance.

- Another recurring phenomena is that going in to the study there may already be a plan for technology's development, but the study reveals that it is "unbalanced" – it spends too much to mitigate some risks, too little to mitigate others, as a result of which requirements attainment is sub-optimal (e.g., the same level of attainment could be had for less cost). Because these studies focus on novel technologies, whose gamut of concerns cut across multiple discipline areas, it is hard for an individual to have the knowledge it takes to develop a "balanced" plan. DDP's *quantitative* treatment (albeit coarse) is key to addressing this.

- The most common benefit of a DDP technology study comes from its broad ranging consideration of potential impediments to success. By combining inputs from all relevant stakeholders (e.g., project managers, engineers, quality assurance personnel, technologists whose (often novel) technologies are the subject of the study, and scientists whose investigations drive the nature of the mission), it allows the early identification of problems and what to do about them. This yields cost savings (prevention and early correction is often far cheaper than late correction, with attendant rework, redesign, etc) and/or improves the targeting of the technology to well-suited applications.

The next section describes how we arrive at these insights from a DDP model.

Gleaning decision-making information from DDP models

Figure 1 displayed a dauntingly convoluted picture of the information in a typical DDP model. This raises the question of how can we glean decision-making information from such models? We use a combination of calculation and visualization – calculation to derive useful information from the raw data in a model, and cogent visualization to present it to the decision makers for their perusal.

The DDP equations define how to calculate the total cost and total Requirements attainment of a populated DDP model given a selection of Mitigations. The custom software we built to support application of DDP performs these calculations automatically – for a typical DDP model, they execute in less than a second on a modest current-day laptop. This allows the decision-makers to explore different Mitigation selections in real time.

In addition to the overall measures of total cost and total Requirements attainment, the calculations also reveal the following specifics:

- the extent to which individual Requirements are being met,
- the amount by which each Risk detracts from the attainment of Requirements (measured in terms of Requirements' weights)
- the contribution of each Mitigation at reducing Risks, and so leading to increased attainment of Requirements (again, measured in terms of Requirements' weights)

We use bar charts to display these specifics in a straightforward manner, so as to help guide the decision-makers. For example, from our bar-chart display of Requirements, it is apparent which Requirements are being attained, and which are not, thus helping when selecting the applications for the technology being studied, or helping prompt the search for Mitigations to reduce the Risks impacting the key unattained Requirements.

In addition to helping decision-makers understand the details of a specific selection of Mitigations, DDP also helps them understand the overall decision space of alternate Mitigation selections. There is a tradeoff between maximizing Requirements' total attainment (by selecting Mitigations to reduce Risks) and minimizing total cost of the selected Mitigations. The DDP software explores this trade space, and generates a scatter plot indicating the two key measures of total cost and total Requirements attainment for a wide variety of Mitigation selections.

An example is seen in Figure 2, derived from a DDP model constructed for another one of our software technology studies. The red line, the so-called "Pareto frontier", demarks the optimal that can be attained (greatest attainment for least cost). Points away from the Pareto frontier represent sub-optimal selections of Mitigations (there are many more than the density of the cloud of black points would suggest, since the heuristic search we employ tends to

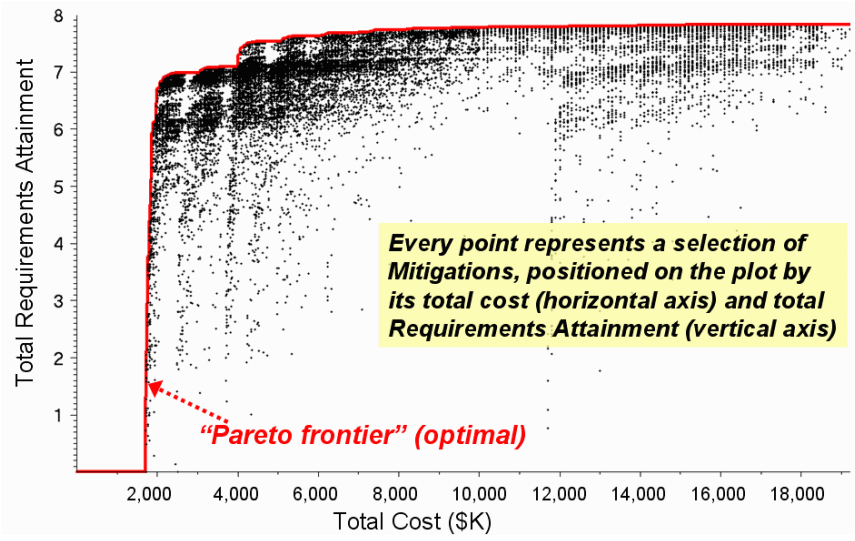

Figure 2 – Scatter plot of many mitigation selections

explore more thoroughly the neighborhood of the Pareto frontier).

That there is a tradeoff between cost and benefit is no surprise; the value of this visualization is in revealing *where* the Pareto frontier of this tradeoff space lies. Figure 2 reveals the following insights for this specific study:

- It is necessary to spend at least $1,800K to get anything, and at $2,000K the first significant plateau is reached (Requirements attainment of 7).
- A significant improvement – a step up of Requirements Attainment from 7 to 7.5 – is evident at approximately $4,000K.
- Beyond that, additional spending more leads to only incremental improvements.

Armed with this kind of information, the decision-makers can arrive at *defensible decisions*. Almost every project manager will argue the need for a larger budget, more time, etc.; the information revealed by DDP gives them (and the *recipients* of their requests) the ability to gauge just how much of an improvement is expected to be had for that extra.

Identifying Critical Factors

Any approach applied to assist in early lifecycle decision making for situations with elements of novelty will of necessity have to work from estimates, not certainties. DDP's estimates take the form of the stakeholders' indications of impacts (of Risks on Requirements) and effects (of Mitigations on Risks). Also, as noted earlier, a DDP model is a coarse representation, eschewing fidelity of representation in order to favor ease of construction. Thus, we do not restrict our attention to solutions on the Pareto frontier, but use a DDP model to offer guidance on locating *neighborhoods* of acceptable decisions (ones that have similar cost and benefit). Decision makers explore alternatives within those neighborhoods in order to arrive at a specific selection. In studying this issue, we have found the following phenomena recur:

- Within a neighborhood of acceptable decisions *there can be radically distinct alternatives*. For example, in one of our studies we examined the neighborhood within 5% of the maximal Requirements attainment that could be had for the available budget. We found several strikingly distinct alternatives; in one, its selection of 30 Mitigations included two that together cost more than half the cost of the other 28, while another alternative avoided use of either of those two expensive Mitigations!

  To find these we adopt data mining techniques that use distance from nearest neighbors as a measure of unusualness [9] in order to identify outliers in datasets. We do not seek outliers per se, merely interestingly distinct alternatives. Alternately, we use the same metric as the basis for *clustering* to group together similar solutions. For a description of applying these metric-based techniques to a DDP models see [10].

- In deciding which Mitigations to select and which to not select, *only a relatively small fraction of those decisions are crucial*. For example, in one of our larger studies, we had almost 100 Mitigations. In the neighborhood of interest only one-third of those select/not-select decisions were crucial – the other two-thirds of the Mitigations had very little effect on the overall cost and benefit. For identifying subsets of critical decisions, we use a form of machine learning that one of our authors, Tim Menzies, has pioneered and applied widely [11]. For details on its application to DDP models, see [12].

Both of these phenomena can be helpful. If critical decisions can be identified, decision-makers can better focus their attention. If the radically distinct alternatives can be revealed, decision-makers can pick and choose, taking into account preferences that have not been encoded in the model. We are motivated in part by the so-called "design by shopping" paradigm [13], which has an emphasis on revealing the space of options available from which to choose, without presuming that all selection criteria have previously been elicited.

Closing Remarks

DDP might be thought of as akin to the mainstream decision support approach Quality Function Deployment (QFD) [14], but with a quantitative, probabilistic basis inspired by risk assessment techniques. This novel combination places it in it a sparsely populated niche in decision making techniques. We believe this is why it continues to be

useful in studying the requirements needs of a wide variety of technologies, software, hardware and combinations of the two.

Acknowledgments

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.

References

[1] Cornford, S.L. "Managing Risk as a Resource using the Defect Detection and Prevention process", *4th International Conference on Probabilistic Safety Assessment and Management*, New York City, NY, September 13-18. 1998.

[2] Feather, M.S., Cornford, S.L., Hicks K.L. and Johnson, K.H., "Applications of tool support for risk-informed requirements reasoning" *Computer Systems Science and Engineering* (CRL Publishing Ltd); 20(1): January 2005 pp. 5-17.

[3] Tralli, D.M. "Programmatic Risk Balancing" *IEEE Aerospace Conference* (2003): pp. 2_775 – 2_784.

[4] Feather, M.S. and Cornford, S.L. "Quantitative Risk-Based Requirements Reasoning" *Requirements Engineering* (2003) 8: pp. 248-263.

[5] Zave, P. and Jackson, M. "Four dark corners of requirements engineering" *ACM Transactions on Software Engineering and Methodology* (1997) 6(1): pp. 1-30.

[6] Swartout, W. and Balzer, R. "On the inevitable intertwining of specification and implementation" *Communications of the ACM* (July 1982) 25(7) pp. 438-440.

[7] Karlsson, J. and Ryan, K. "A Cost-Value Approach for Prioritizing Requirements" *IEEE Software* September/October 1997, pp. 67-74.

[8] Van Lamsweerde, A. and Letier, E. "Integrating Obstacles in Goal-Driven Requirements Engineering"*20th International Conference on Software Engineering*, Kyoto, April 1998.

[9] Knorr, E.M. and Ng., R.T. "Finding intensional knowledge of distance-based outliers", *25th VLDB Conference*, 1999.

[10] Feather, M.S., Kiper, J. and Kalafat, S. "Combining Heuristic Search, Visualization and Data Mining for Exploration of System Design Spaces", *14th Annual International Symposium Proceedings of INCOSE,* Toulouse, France, June 20-24 2004.

[11] Menzies, T. and Singh, H. "Many Maybes Mean Mostly the Same Thing", *Soft Computing in Software Engineering*, Springer-Verlag, 2003, pp. 125-150.

[12] Feather, M.S. and Menzies, T. "Converging on the Optimal Attainment of Requirements" *IEEE International Conference on Requirements Engineering*; Essen, Germany, September 9-13, 2002, IEEE Computer Society, pp. 263-270.

[13] Balling, R. "Design by Shopping: A New Paradigm," *Third World Congress of Structural and Multidisciplinary Optimization (WCMSO-3)*, Buffalo, NY., May 1999, pp. 295-297.

[14] Akao, Y. "Quality Function Deployment", Productivity Press, Cambridge, Massachusetts (1990).