

A QUANTITATIVE RISK MODEL FOR EARLY LIFECYCLE DECISION MAKING

Martin S. Feather, Steven L. Cornford, Julia Dunphy, Ken Hicks

Jet Propulsion Laboratory, California Institute of Technology

4800 Oak Grove Drive, Pasadena CA 91109, USA

{Martin.S.Feather, Steven.L.Cornford, Julia.Dunphy, Kenneth.Hicks}@Jpl.Nasa.Gov

ABSTRACT

Decisions made in the earliest phases of system development have most leverage to influence the success of the entire development effort, and yet must be made when information is incomplete and uncertain.

We have developed a scalable cost-benefit model to support this critical phase of early-lifecycle decision-making. We have focused on scalability in order to accommodate the many concerns that are relevant in planning complex development efforts. We use risk as the unifying concept from which both cost (sum of resources it takes to mitigate risk) and benefit (sum of requirements attained when risks are taken into account) are calculated.

The model is supported by custom-built software. We have used this to elicit and combine information from experts in the multiple disciplines. This is done in an on-the-fly manner, thus retaining the involvement of those experts. It has proven successful at pinpointing the most critical areas within a large space of concerns, and at guiding experts toward superior alternatives.

DEFECT DETECTION & PREVENTION (DDP) – A RISK-CENTRIC MODEL OF COST & BENEFIT

At NASA we have been developing and applying our risk management framework, “Defect Detection and Prevention” (DDP), for several years. DDP is a process for which we have custom-built software support. We have published several accounts of DDP – (Cornford et al., 2001) gives an overview, (Cornford et al., 2002) some recent directions, (Feather et al., 2000) the “look and feel” underpinning the software. In this paper we focus on the model underpinning DDP.

DDP deals with three key concepts: *requirements*, *risks* and *risk mitigations* (in some of the papers we have published, risks are referred to as “failure modes”, and mitigations as “PACTs”). Risks are quantitatively related to requirements, to indicate how much each risk, should it occur, *impacts* each requirement. Mitigations are quantitatively related to risks, to indicate how much of a risk-reducing *effect* the mitigation, should it be applied, has on the risk. The topology of this simple but scalable model is shown in Figure 1

In a DDP model, a set of mitigations *achieves benefits* (requirements are met because the risks that impact them are reduced by the selected mitigations), but *incurs costs* (the sum total cost of performing those mitigations). *The primary purpose of DDP is to facilitate the judicious selection of a set of mitigations, thus*

attaining requirements in a cost-effective manner.

The subsections that follow present more details of this model, and some of the ramifications of applying it to real-world problems.

Requirements

Requirements are what the system is to achieve. Requirements are assigned *weights*, representing their relative importance. Requirements are either “on”, or “off”. The DDP computations automatically apply to those and only those requirements that are “on”.

There can be a wide variety of requirements. They can be requirements imposed on the system to be developed, and/or requirements on the development process itself. We have seen instances of requirements on the functionality of the artifact (e.g., that it be able to control a spacecraft instrument), on the resources it consumes (e.g., memory), on its tolerance of its surrounding environment (e.g., be able to work around hardware memory errors) and on the development process by which it is constructed (e.g., that it be delivered on schedule and within budget). We have even seen a requirement stated as “no other technology is better”, when the purpose of the exercise was to assess the worthiness of, and approach to, maturing a novel technology into flight readiness. It is common to see a mixture of multiple such kinds of requirements within a single application of DDP. Requirements can be varied not only in area, but also in level of detail, with more detail provided for areas of especial concern.

Requirements can be numerous – in typical DDP applications experts have listed 30 – 100 requirements. Simple tree structures are used to taxonomize requirements, the advantages of which are:

- Grouping requirements into categories and subcategories provides a reminder to users of the range of issues they need to think about.
- Straightforward taxonomies help users locate where they placed a requirement. Navigation becomes challenging as the number of items grows, especially when they surpass the number that can be squeezed into view on a single screen.
- Allowing tree structures of requirements to be collapsed or expanded supports abstraction. A collapsed subtree aggregates all of its descendants, so serves as an abstraction of the detail beneath. For example, the weight of a non-leaf requirement is recursively computed as the sum of the weights of its children.

Requirements are candidates for trade.

Requirements are quantitatively related to the risks that impact them, and risks are in turn quantitatively related to the mitigations that serve to reduce those risks. Thus, the weights ascribed to requirements are at key to computing the relative “benefit” of a DDP model.

The existence of especially problematic requirements (impacted by risks which cannot readily be mitigated) quickly becomes apparent when using DDP. This enables users to identify those requirements whose removal would admit far less expensive solutions.

Risks

Risks are all the things that, should they occur, lead to loss of requirements. Risks are assigned an *a-priori likelihood* (the chance of the risk occurring, if nothing is done to inhibit it). Generally, the a-priori likelihood is left at 1 (certainty). For example, the likelihood of some complex piece of software containing bugs (if nothing is done to inspect or test that software) is close to certainty. In instances where physical phenomena are involved, their likelihoods can be reasonably asserted to be somewhat less than certainty (e.g., the likelihood of a lightning during launch). Like requirements, risks are either “on” or “off”, and only those that are “on” contribute to DDP’s calculations. By default, risks start as “on”, forcing users to explicitly turn them “off” if they have reason to believe that they do not apply to the system at hand.

Risk assessment is calculated, not directly estimated. A key difference between DDP and most other approaches to risk assessment is that DDP calculates the magnitude of a risk by summing its impacts on requirements; other approaches ask directly for an estimate of the severity of the risk itself. DDP’s disadvantage is the extra overhead in gathering the quantitative data on which to base its calculations. DDP’s advantage is that its computation of risk is a more disciplined process. Furthermore, it allows for trades in requirements space, an option that is not so readily pursued in traditional risk assessment.

Risks are numerous. Like requirements, risks can be numerous. In typical DDP applications, experts have listed 30 – 200 risks. As with requirements, risks are organized into taxonomies for the purposes of navigation, reminders, and abstraction.

For example, in (Feather et al., 2001) we specialized DDP to software assurance planning. We pre-populated DDP with the Software Engineering Institute’s taxonomy of 64 software development risks (e.g., the “Product Engineering” category contains a “Requirements Risks” subcategory which in turn contains “Completeness: Incomplete requirements”). From this starting point, DDP allows users to discard risks that are irrelevant to their task, add risks that are not already included, and refine risks to greater levels of detail.

Mitigations

Mitigations are all the activities that could be done to

reduce the likelihood of risks and/or reduce their impact on requirements. Each mitigation is assigned *cost*, the costs of performing it. Mitigations are also assigned the time period within the development effort at which they would be performed (e.g., requirements, design). Like requirements and risks, mitigations either “on” or “off”, and only those that are “on” contribute to DDP’s calculations of risk reduction. One of the primary purposes of a DDP application is to choose which mitigations should be “on”.

Mitigations are numerous. Typical DDP applications have involved lists of 30 – 170 mitigations.

Mitigations are choices, not requirements. It is important to realize that during the use of DDP, mitigations are *choices*. At the conclusion of a DDP study, the chosen mitigations then may become requirements on the development to follow. In a similar vein, failure to turn a mitigation “on” is not a risk per se.

Mitigations can have multiple costs. Performing a mitigation can have multiple costs (e.g., schedule, budget). In spacecraft hardware designs we have tracked additional forms of cost (e.g., mass, power and volume).

Mitigations’ time periods. The sequence of time periods is set for the given application. It might be organized into calendar units (e.g., quarters of the financial year), or into development stages (e.g., requirements, design). This information provides insight into the progression of mitigation spending over the course of the planned development. For multi-year projects, there may be constraints on resources (e.g., budgets) expended by year. As discussed in (Cornford et al., 2002), the information also provides key insight into the “risk profile” – how risk diminishes over the course of the planned development. Plans that reduce risks early are, in general, preferred over plans that attain the same final risk level but do so by reducing risks late. The reason is that all of these plans contain considerable uncertainty (remember, DDP is applied early in the lifecycle where solid information is lacking). A plan that reduces risk early can slip and still have reduced risks to tolerable levels by the originally planned launch date (Plan A in Figure 2). The same tolerance to slippage is not true of a plan that reduces risk late (Plan B in Figure 2).

Impacts

For each Requirement x Risk pair, we assert how much of that Requirement will be lost should that risk occur. This value we call the “impact”. It is expressed as a number in the range 0 – 1, meaning the proportion of the requirement that would be lost. Thus 0 means no loss whatsoever, and 1 means total loss of the requirement.

Simple combination rule for impacts: impacts combine *additively*, e.g., if two different risks impact the same requirement, then their combined impact is the sum of their individual impacts. This may seem an overly simplistic combination rule, but in the early stages of risk assessment it suffices to capture the wide range of

problems that need to be considered. Recall that DDP is aimed at the early stages of planning, when detailed design information is absent. As designs mature, other more design-centric risk assessment methods become applicable (e.g., probabilistic risk assessment).

Impacts are numerous: A given risk may have impacts on multiple requirements, and those impacts need not be identical. Likewise, a given requirement may be impacted by multiple risks, whose impacts need not be identical. In typical DDP applications, the numbers of impacts (i.e., Requirement x Risk pairs for which the impact value is non-zero) range from the many hundreds to the low thousands.

Much of the power of the DDP process stems from its ability to handle a large number of such cross-linkings. We eschew a complex model of combination in order to retain this scalability, notably the scalability of eliciting the information from the experts.

Requirements “at risk”: For each requirement, DDP computes the sum total impact on it. This indicates the extent to which the requirement is “at risk”. The effect of mitigations is to decrease risk, which in turn leads to increased requirements attainment.

One seemingly strange consequence of our combination rule for impacts is that requirements can be *more* than completely impacted (e.g., impacts of 0.8 and 0.7 add up to a total impact of 1.5)! We compute this metric as a guide to how much risk reduction is needed to attain a requirement. However, for assessing value, we compute another metric in which requirements that are more than completely impacted contribute zero.

Effects

For each Mitigation x Risk pair, we assert how much of the risk will be reduced if that mitigation is applied. This value we call the “effect”. It is expressed as a number in the range 0 – 1, meaning the proportion by which the risk would be reduced. Thus 0 means no reduction whatsoever, and 1 means total elimination of the risk.

Simple combination rule for effects: when several mitigations reduce the same risk, their total effect is computed as: $(1 - \text{the product, for each mitigation } M, \text{ of } (1 - M\text{'s effect}))$.

Intuitively, mitigations act as “filters” in series: each mitigation filters out its effect’s proportion of the risks that enter it. E.g., a mitigation with effect of 0.8 on some risk and another mitigation with effect of 0.3 on that same risk together have effect: $(1 - (1 - 0.8)(1 - 0.3)) = (1 - 0.2*0.7) = (1 - 0.14) = 0.86$ on that risk.

As was the case for impacts (Requirement x Risk pairs), each mitigation may effect multiple risks, and each risk may be “effected” by multiple mitigations.

Sum total risk mitigation: For each risk, DDP computes the combined effect of all the selected mitigations at reducing the risk. This reduced risk value is in turn used to calculate the requirements “at risk” figure, to reflect the positive contributions that stem from those

mitigations. To aid users in selecting mitigations, DDP also computes two metrics for each mitigation – its “solo” risk reducing effect (i.e., risk reduction it would accomplish if it were the *only* selected mitigation), and its “delta” risk reducing effect (i.e., the additional risk reduction, beyond that already achieved by the other selected mitigations, its selection would accomplish).

We have recently begun to explore automatic search for optimal solutions (discussed later). However, to date, DDP applications have relied upon the users to manually select mitigations. The “solo” and “delta” metrics have proven useful guidance in these cases.

ISSUES OF SCALE

We have indicated typical ranges of the number of objects in a DDP model. As a deliberately daunting demonstration of the scale problem, Figure 3 shows the topology of a real DDP model, drawn in the style of Figure 1, but containing the full number of objects.

The need for scale

The need for handling this many objects and links derives from the area of application. DDP is used on complex technologies, upon which many factors from multiple disciplines have a bearing. The primary purpose of DDP is to take this large number of such factors into account, so as to emerge with an understanding of which factors are most important. For example, determine which of the risks are truly the most damaging, which of the requirements are proving the most problematic to attain, and which of the mitigations are most appropriate to select to reduce risk (and thereby attain requirements).

Handling scale

The DDP software has several features that facilitate working with these fairly large, albeit simplistic models. These features have been incorporated in DDP software from the start (Feather et al., 2000), and so have been used in all of our DDP applications to date. Briefly, they are:

Multiple views – DDP offers multiple ways of viewing the information. For example, a tree viewer allows editing and viewing the tree structures of requirements, risks or mitigations. A matrix viewer (akin to simple spreadsheets) allows editing and viewing the impacts and effects. A bar chart viewer allows scrutiny of the computed values (e.g., requirements’ “at risk” levels).

Hierarchy – tree structures serve to organize information (e.g., requirements) hierarchically. The multiple views are kept coordinated with respect the current status of the hierarchy, e.g., if a subtree of requirements is currently “collapsed”, then the corresponding rows in the matrix of impacts (between requirements and risks) are aggregated into a single row. The values of the aggregated row’s cells are computed automatically by aggregating the values of the cells of which the aggregation was composed.

Compact views – impacts form a giant Requirement x Risk matrix, and effects form a giant Mitigation x Risk

matrix. These are generally rather sparse matrices (i.e., the majority of the cells are blank, corresponding to a zero impact or effect). We take advantage of this sparseness to portray the matrix information in a “list” style, in which only the non-zero impacts/effects are listed alongside each Requirement/Risk/Mitigation. (This feature derives from the work of JPLers D. Howard and C. Hartsough.)

Sorting – we offer the option to sort risks into descending order (commonly referred to as a “Pareto” chart). Our model makes the usual distinction between likelihood and impact (a.k.a., severity), so we are able to produce 2-dimensional charts that sort risks along both these dimensions – Figure 4. The upper right corner is the maximum possible risk (maximum impact and maximum likelihood). The axes of this chart are *logarithmic* scale; hence the diagonal boundaries between the three different shades of background are lines of constant risk.

Calculations – DDP calculates a variety of metrics from the user-provided information. For example, for each requirement it computes the “at risk” metric as the sum of risk impacts on that requirement. The number of impact values (between Requirement x Risk pairs) and effect values (between Mitigation x Risk pairs) determines the complexity of these calculations. In practice, DDP models are “sparse” (approximately 10% of all possible these pairs have non-zero values). Nevertheless, a complete recalculation running DDP on a 1 GHz Pentium® can take two or three seconds, and some displays of information take a noticeable time to be updated. For key operations (adding/changing/removing an impact or effect value, and selecting/unselecting a mitigation) DDP is programmed to *incrementally* recompute metrics. Exploiting the hierarchical structure of the information (e.g., use the root of a subtree of requirements as an aggregate in place of its many descendents) is a further option, not currently used. The extensions to the DDP model (described later) complicate the situation. It would be very desirable to *derive* efficient code from a lucid specification, in the style of the program transformation research community.

Elicitation

The bulk of the time for a DDP application goes into *eliciting* models from users. We typically decompose DDP applications into four half-day sessions. In each of these, we have on hand experts who represent all of the aspects of the system being studied – mission scientists, engineers from multiple disciplines, quality assurance personnel, etc. The first three sessions are devoted to populating DDP with the model information (requirements, risks, mitigations, impacts and effects), and the last session to decision making.

Since many of the DDP applications to date have been studies of widely different technologies, there has been little opportunity for reuse between these studies. There have been efforts to pre-populate DDP with information specific to certain disciplines, e.g., the

software assurance planning mentioned before, however at this point we have little experience with their use. As open question is the degree to which such pre-populated databases save time – will users spend as much time going through them as they would to build them from scratch?

We have found that in eliciting information from a group of experts, we can use disagreement to drive the need for refining the information. For example, if there is disagreement about the impact of a risk on a requirement, this almost always stems from those experts thinking of different cases (e.g., the impact in the “nominal” scenario, vs. the impact in a high-criticality scenario). Subdividing the risk and/or the requirement into multiple subcases, and assigning appropriately different impact values to each, resolves these disagreements. Similarly, agreement indicates the lack of need to subdivide into greater depth. We do not always recognize the latter in advance, resulting in subcases that we find are being assigned the same values. When this occurs, we simply delete the myriad of subcases, and make do with the parent.

Decision Making

The primary purpose of DDP applications is to result in the selection of a set of mitigations that reduce risk (and thereby lead to attainment of requirements) in a cost-effective manner. On some occasions DDP applications have led to the discovery of problematic requirements – ones whose attainment is proving particularly expensive to achieve.

We have also used DDP to compare alternatives. DDP allows for the turning on and off of individual elements (requirements, risks and mitigations), of subtrees of them, and indeed of arbitrary sets of them.

In applications to date, we have relied on the human experts to make the decisions of which mitigations to select (and/or of which requirements to discard). This is clearly a challenging task, given the interconnectedness of DDP’s elements. In response, we have begun investigating techniques that automate search for (near) optimal solutions. In one approach, we adapted genetic algorithms to this purpose. The preliminary results are quite promising, and we intend to pursue this further. In another approach, we collaborated with Tim Menzies, who has a machine-learning based approach (Menzies and Hu, 2001) to identifying critical decisions to make (and which way to make them!). Again, the preliminary results are quite promising. A snapshot of a recently completed pilot study (Feather and Menzies, 2002) is shown in Figure 5. This shows a chart whose two dimensions are cost (the sum total cost of selected mitigations) and benefit (the sum total value of attained requirements, taking into account the beneficial risk-reducing effect of the selected mitigations). Each point represents a selection of mitigations. Some selections are very wasteful – they cost a lot, yet attain little benefit. These are the points towards the bottom right corner of the diagram. Conversely, some selections are very effective – they

attain near-maximal benefit, at significant cost savings as compared to many of the solutions. These are the points towards the upper left corner. The black points are those generated by random selection of mitigations, so their distribution illustrates the wide range of possibilities. The white region consists of a large number of individual white points, each of which is a solution recommended by Menzies' machine learning based technique.

VALIDITY OF THE MODEL

The validity of a DDP model is often called into question, given that it is based on a large number of estimated values combined in a simplistic manner. We respond to this concern in three ways: reflection on the intended purposes of the model, mechanisms to explore the sensitivity of the model to its data, and population of the model with data based on past experience:

Purpose of a DDP model

DDP is best applied early in the lifecycle of a development. Its purpose is to guide developers to focus on the issues that are of most importance. Thus, the model need only have sufficient fidelity to be able to distinguish between alternatives, and need not (indeed, we argue *should* not) be used to compute absolute measures (e.g., probability of success).

Anecdotal evidence culled from DDP applications is supportive of the value of DDP. Initially skeptical participants typically emerge convinced that DDP has helped. (Cornford et al., 2001 reports benefits of:

- Clarification of a customer requirement leading to considerable savings in work not required.
- Rejuvenation of a technology by identification of opportunities for its utilization.
- Support for adoption of a commercial software development environment (balancing the pros and cons of making this switch from current practice)

Explorations of the sensitivity of a DDP model

We recently added a capability to study the sensitivity of the model to variations in the model's effect numbers (the quantitative estimates of how effective each mitigation is at reducing each risk).

This capability offers a menu of ways of making changes to effect values. For the user's selection, the tool then applies the change one-by-one to each of the non-zero effect values, recomputes benefit (i.e., requirements attainment), and builds a table listing in descending order variations of that benefit figure with respect to each effect value. Figure 6 shows the top portion of this table for an actual DDP model. This shows that the mitigation "Select/make laser" (in the "PACT" column) on the risk "Insufficient power" (in the "Failure Mode" column) has the greatest change on requirements attainment. Information such as this allows users to know which of the hundreds of such values to scrutinize most closely.

In another approach to compute sensitivities, we applied Menzies' machine learning (discussed earlier) to

search for the impact and effect values most critical to changing the computed costs and benefits of an optimized solution. Application to an actual DDP dataset suggested that the solution was relatively robust.

Population with experience-based data

The ideal answer would be to populate the model with data based on experience, and use combination rules that yield answers in agreement with experience. In the software realm, we look to groups such as CeBASE consortium <http://www.cebase.org> to gather such data.

When DDP is applied to plan the development of novel technologies, data may be available for some aspects of the development, but lacking for the more novel factors. In such cases, we must continue to rely on at least some of the data being experts' best estimates.

EXTENSIONS OF THE MODEL

In this section we describe some extensions to the core DDP model. Most of these have been incorporated within the DDP software, but we have not yet had chance to employ them in full-scale DDP applications. Generally, these are conservative extensions of the standard DDP model. They take effect if and only if optional additional information is provided when building a DDP model.

The most significant of these concerns the partitioning of mitigations into three categories: "preventions", "detections" and "alleviations", and the ramifications of making this distinction. We have also allowed for the possibility that mitigations may increase certain risks. These extensions are described next.

Categories of mitigations, and repair costs

- *Preventions* – mitigations that reduce the likelihood of risks occurring, e.g., training of programmers reduces the number of mistakes they make.
- *Detections* – mitigations that detect risks, with the assumption that detected risks will be repaired, e.g., unit testing detects coding errors internal to the unit, which are then corrected. The net effect of detection and repair is a reduction in the likelihood of risks present prior to detection remaining afterwards.
- *Alleviations* – mitigations that decrease the impact (severity) of risks should they occur, e.g., programming a module to be tolerant of out-of-bound values input to it from another module.

These extensions give rise to differences from the "standard" DDP model's calculation of risks and costs:

- Risks: Alleviations reduce *impact* of risks, while preventions and detections reduce *likelihood* of risks. These effects can be viewed via the risk region chart shown in Figure 4. Users may find it more palatable to accept high likelihood but low impact risks than low likelihood but high impact risks. Calculation of risk as the product of impact and likelihood would be unable to differentiate between the two. It is common to make this distinction in traditional reasoning about risk. The difference here is that we have incorporated

the distinction into our existing framework, thus retaining its advantages of scalability and applicability to early-lifecycle planning.

- **Costs:** Mitigations in all three categories continue to have costs associated with them. However, detections also incur a cost of repair of the risks they detect. This repair cost is the product of the quantity of risk detected (computed as the reduction in likelihood attributed to the detection) and the basic repair cost associated with the risk itself (an additional attribute of risks). We allow for the repair cost associated with a risk to depend upon the time period in which the repair is performed. Recall that mitigations are associated with the time period in which they are performed. Hence, a repair triggered by a detection occurs in the time period of the mitigation.

Using this capability, we can represent the escalation of costs of repair as time progresses. For example, the cost of repairing a requirements flaw may be tiny at requirements time, larger at design time, larger still at coding time, etc. Once this information is provided, the DDP calculations of cost then reflect the escalation of repair costs when risks are allowed to linger to later stages of development.

It is well understood that early-lifecycle activities such as inspections can increase benefit (requirements attainment through reduction of risk) and decrease cost (fewer risks to be corrected at later phases in the lifecycle, when repair costs have escalated), e.g., (Kaner, 1996). In (Feather et al., 2001) we use the extended DDP model to recreate quantitatively such reasoning.

Risks induced by mitigations

Mitigations generally reduce risks, but some can also increase certain risks. For example, a vibration test of hardware used to detect flaws can potentially damage that hardware further. In the software world, additional code whose purpose is to make a design more fault tolerant (e.g., a software voting algorithm) can introduce risk if it is itself incorrectly implemented. Finally, repairs can introduce risks, e.g., bug fixes may introduce new bugs.

We incorporate these phenomena within the DDP model by allowing for a mitigation to increase risks in addition to decreasing other risks (presumably it decreases *some* risks, otherwise it would be pointless to apply it!).

Ongoing and future extensions

We are in the process of incorporating various further extensions to the DDP model.

- Logical structure to risks, for example, and/or nodes of fault trees in probabilistic risk assessment.
- More sophisticated means to calculate requirements attainment than simply the sum of child requirements' attainment (e.g., maximum, root-mean-square).
- Expression and use of additional relationships among DDP elements. For example, the concept of one mitigation being a "necessary precursor" to another.

In the absence of these capabilities in the current version of DDP, we rely on manual workarounds. For example,

when we know that the combination of two mitigations M1 and M2 does not match that predicted by our formulae, we manually add a third mitigation, M1&M2. We assign to this the combined effectiveness and cost values that we believe hold for the combination of the two. When selecting mitigations, we are careful to select at most one of { M1, M2, M1&M2 }. Such workarounds allow us to proceed with DDP applications, at the expense of a small amount of additional effort.

We use actual DDP applications to gauge which manual workarounds are recurring and tiresome, thus motivating our choice of which extensions to work on next. We also try to be proactive in predicting new features (for which simple workarounds do not exist) of benefit for future applications. An example of this is the need to support simultaneous contributions to a shared DDP model. At present, we serialize DDP sessions to input information to the one and only model. This results in either a waste of the cumulative time of the experts present, and/or a failure to capture all the valuable information that emerges during the session.

RELATED WORK AND CONCLUSIONS

The DDP model has some similarity with a number of other models of systems. We briefly discuss several:

Estimation models

COCOMO and COQUALMO models predict factors such as cost and quality based on inputs that characterize the development at hand (Boehm et al., 2000).

Generally, estimation models such as these are "closed" – they are not intended to be extended with new factors (although they do encourage tuning the models to a given organization). In contrast, the DDP model is "open", relying on expert users to input and link the factors that are relevant to the development at hand. (Kurtz and Feather, 2000) describes our work to mix of these approaches, linking DDP to NASA's Ask Pete tool. The latter does estimation and planning of software assurance activities. In combination, the Ask Pete tool is used to build a first-cut model, and the DDP tool can then be used to tailor this to the development at hand

Goal models

The software engineering research community has shown increasing interest in models of "goals" (roughly speaking, precursors to requirements). See the mini-tutorial (van Lamsweerde, 2001) for an overview of this area. We discuss two of these kinds of models:

The KAOS framework for goals, requirements, etc. (Bertrand et al., 1998) is used to build a logical structure of how system-wide requirements decompose to, ultimately, requirements on the individual components in a system. Models built in this framework seem well suited to exploring the functional behavior, and to some extent, non-functional aspects. DDP models are weaker in that they lack the logical structure of KAOS models, but conversely have emphasized more the quantitative aspects

that predominate in imperfect solutions.

The i* framework (Chung et al., 1999), (Mylopoulos et al., 2001) combines logical structures with *qualitative* models. Their framework's combination rules support tradeoff analysis between a few major design alternatives. DDP models seem more appropriate when there are a large number of small alternatives.

Bayesian Nets / Influence diagrams

Influence diagrams (a form of Bayesian nets) offer a general framework in which factors can be combined to assess designs and study alternatives. (Burgess et al., 2001) uses them to compute the utility of requirements that are candidates for inclusion in the next release of a piece of software. In principle it would seem that a DDP model could be represented as an influence diagram, with a relatively "flat" topology (Figure 1). However, as seen in Figure 3, typical DDP applications give rise to rather voluminous such models. DDP is better tuned for decision making when a multitude of factors must be considered.

Requirements Prioritization

Requirements prioritization has also emerged as a topic of interest.

(Karlsson and Ryan, 1997) developed a "cost-value" approach to prioritizing requirements. They use a cost-value diagram to plot each requirement's relative value and implementation cost, facilitating the selection of an appropriate subset of requirements.

WinWin (Boehm et al., 1994) and its custom tool (In et al., 2001) supports multiple stakeholders to identify conflicts between their respective evaluations of requirements, and locate feasible solutions that are mutually satisfactory combinations of requirements.

These examples typify approaches in which users are asked to directly estimate the costs and benefits of individual requirements. Significant interactions among requirements (e.g., if two requirements can be achieved by sharing the same solutions to sub-problems) complicate this. DDP's approach is to explicitly relate requirements to risks, and risks to mitigations.

Risk estimation approaches (e.g., fault tree analysis, bayesian methods) appear well suited to the assessment of a single design. However, our application is the planning of mitigations, where the driving concern is the cost-benefit-guided selection from among a large set of them.

Conclusions

We have outlined the DDP model, designed to fill the early-life cycle niche in risk-based estimation and planning. DDP thus complements a number of other modeling techniques.

DDP's key elements are requirements, risks and mitigations, are linked to one another in a quantitative manner. Custom tool support facilitates use of this model when relatively large numbers of items are involved.

DDP has been successfully applied in early lifecycle decision-making. It appears well suited to applications where a multitude of factors must be considered

simultaneously. We fully expect use of DDP to continue. We also anticipate that recent extensions to the DDP model will further its ability to compute costs and benefits associated with risk mitigations.

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. 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. Contributions from, and discussions with, Burton Sigal (JPL), Patrick Hutchinson (Wofford College, Spartanburg SC), Peter Hoh In (Texas A&M), John Kelly (JPL), Tim Kurtz (NASA Glenn), James Kiper (Miami Univ., Ohio) and Tim Menzies (U. British Columbia) have been most useful in helping us formulate our ideas.

REFERENCES

- Bertrand, P., Darimont, R., Delor, E., Massonet, P., and van Lamsweerde, A., 1998, "GRAIL/KAOS: an environment for goal driven requirements engineering", *30th Int. Conference on Software Engineering*.
- Boehm, B., et al., "Software Cost Estimation with COCOMO II" Prentice Hall.
- Boehm, B., Bose, P., Horowitz, E., and Lee, M., 1994, "Software Requirements as Negotiated Win Conditions", *Proceedings 1st International Conference on Requirements Engineering*, pp. 74-83.
- Burgess, C.J., Dattani, I., Hughes, G., May, J.H.R., and Rees, K., 2001, "Using Influence Diagrams to Aid the Management of Software Change", *Requirements Engineering* 6(3), pp. 173-182.
- Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J., 1999, "Non-Functional Requirements in Software Engineering" Kluwer Academic Publishers.
- Cornford, S.L., Feather, M.S., and Hicks, K.A., 2001, "DDP - A tool for life-cycle risk management", *IEEE Aerospace Conference*, pp. 441-451.
- Cornford, S.L., Dunphy, J., and Feather, M.S., 2002, "Optimizing the Design of end-to-end Spacecraft Systems using risk as a currency", *IEEE Aerospace Conference*.
- Feather, M.S., Cornford, S.L., and Gibbel, M., 2000, "Scalable Mechanisms for Requirements Interaction Management", *IEEE Int. Conference on Requirements Engineering*, pp. 119-129.
- Feather, M.S., Sigal, B., Cornford, S.L., and Hutchinson, P., 2001, "Incorporating Cost-Benefit Analyses into Software Assurance Planning", to appear in *Proceedings, 26th IEEE/NASA Software Engineering Workshop, Greenbelt, Maryland November 27-29*.
- Feather, M.S., and Menzies, T., 2002, "Converging

on the Optimal Attainment of Requirements”, in submission.

In, H., Boehm, B., Rodgers, T., and Deutsch, M., 2001, “Applying WinWin to Quality Requirements: A Case Study”, *Proceedings 23rd International Conference on Software Engineering*, pp. 555-564.

Kaner, C., 1996, “Quality Cost Analysis: Benefits and Risks”, *Software QA* Vol 3, #1, pp. 23.

Karlsson, J., and Ryan, K., 1997, A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, Sept./Oct. pp. 67-74.

Kurtz, T., and Feather, M.S., 2000, “Putting it All Together: Software Planning, Estimating and Assessment for a Successful Project”, *Proceedings 4th International Software & Internet Quality Week Conference*, Belgium.

Menzies, T., and Hu, Y., 2001, “Constraining Discussions in Requirements Engineering via Models”, *1st International Workshop on Model Based Requirements Engineering*, San Diego, California.

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

van Lamsweerde, A., 2001, “Goal-Oriented Requirements Engineering: A Guided Tour”, *Proceedings 5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, August, pp. 249-263.

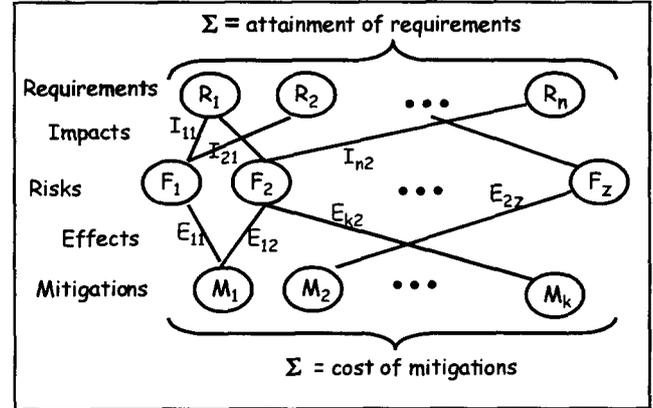


Fig. 1 Topology of DDP's risk-centric cost-benefit model

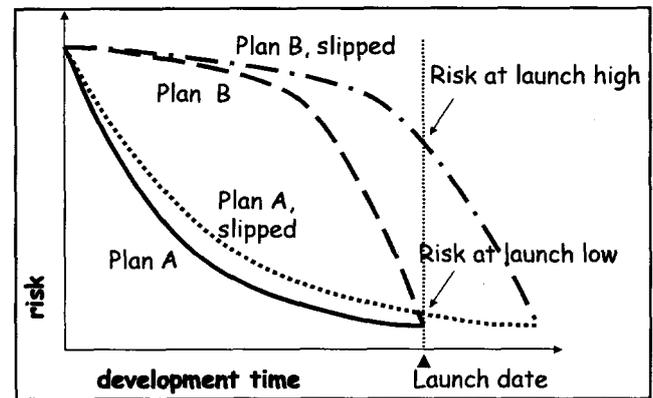


Fig. 2 Risk profiles over time

FIGURES

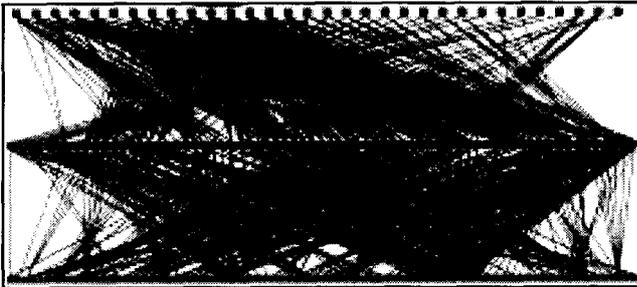


Fig. 3 Topology of an actual DDP model

Change	% Change	FACT	Failure Mode
2.02	0.76	Select/Make Issue	Insufficient power
0.834	0.314	CCD Qualification	CCD degradation
0.6	0.226	System Study	Other technologies are better
0.325	0.124	Harmful packaging	non Harmful
0.246	0.0925	Fibre qualification	Fibre degradation

Fig. 6 Sensitivity analysis results table

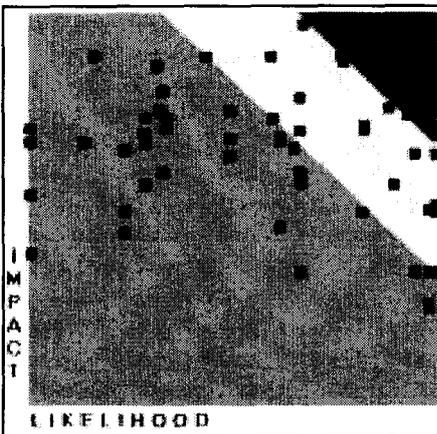


Fig. 4 Risk region chart

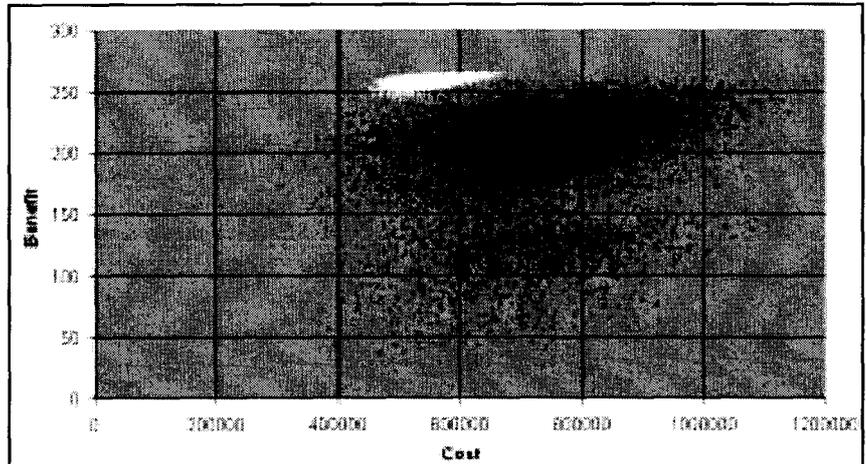


Fig. 5. Converging on optimal attainment of requirements