Fusing Quantitative Requirements Analysis with Model-based Systems Engineering

Steven L. Cornford Martin S. Feather Vance A. Heron J. Steven Jenkins
Jet Propulsion Laboratory, California Institute of Technology
{Martin.S.Feather, Steven.L.Cornford, Vance.A.Heron, J.S.Jenkins}@jpl.nasa.gov

Abstract

A vision is presented for fusing quantitative requirements analysis with model-based systems engineering. This vision draws upon and combines emergent themes in the engineering milieu. "Requirements engineering" provides means to explicitly represent requirements (both functional and non-functional) as constraints and preferences on acceptable solutions, and emphasizes early-lifecycle review, analysis and verification of design and development plans. "Design by shopping" emphasizes revealing the space of options available from which to choose (without presuming that all selection criteria have previously been elicited), and provides means to make understandable the range of choices and their ramifications. "Model-based engineering" emphasizes the goal of utilizing a formal representation of all aspects of system design, from development through operations, and provides powerful tool suites that support the practical application of these principles.

A first step prototype towards this vision is described, embodying the key capabilities. Illustrations, implications, further challenges and opportunities are outlined.

1. Introduction

This paper is structured as follows: we begin by presenting the existing ideas we draw upon, taken from three mainstream areas, and thereafter introduce our visionary approach towards their fusion (Section 2). We have confirmed the soundness of our vision by taking a significant first step – construction of a prototype that realizes the essential aspects of our vision, and application of this prototype on a representative example. This served to clarify our own understanding of the vision, yield a demonstration to show to our sponsors and hoped-for customers, and help us identify future problems and opportunities. Our prototype is outlined in Section 3. Succinct illustrations of its operation, highlighting its novel capabilities, are shown in Section 4. The revolutionary implications if this vision were adopted are discussed in Section 5, along with some challenges and opportunities that lie in its advancement. Finally the references in Section 6, although constrained in length, are notable for drawing from a diversity of sources. This diversity reflects the fact that attainment of our vision will require a fusion of ideas taken from disparate areas of engineering.

2. A fusion of ideas and approaches

2.1. Requirements engineering

Requirements engineering emphasizes the explicit representation and treatment of requirements. Herein the kinds of requirements we explicitly deal with span both functional and non-functional ones, and include constraints ("the system shall...") and preferences ("the more ... the better..."). We also draw from requirements engineering the emphasis on early-lifecycle activities. Proponents of the various kinds of "-ilities" (e.g., affordability, reliability) often claim that these cannot readily be built into a design as an afterthought. Rather, these "-ilities" should be used from the very beginning to guide the selection and refinement of designs. This requires that it be possible to gauge (by whatever early-phase process is applied, e.g., review, inspection, analysis, formal verification, simulation) the degree to which a design fulfils these "-ilities".

2.2. Design by shopping

"Design by shopping" has its emphasis on revealing the space of options available from which to choose, without presuming that all selection criteria have previously been elicited. Its origins as an approach that hinges on "a-posteriori articulation of preference" [1] are discussed in [2], where the following statement is made on its viability as a method:

"For the design by shopping paradigm to take hold, research is needed in two areas. First, efficient methods for obtaining rich Pareto sets are needed. Second, interactive graphical computer tools are needed to assist decision makers in the shopping process."

Progress in both of these areas has since occurred For example, [3] employs a variety of rich visualization capabilities to present the option space of designs, and couples these with the design models from which the options are calculated

2.3. Model-based engineering

"Model-based engineering" emphasizes a formal representation of all aspects of system design, from development through operations. In the *software* engineering milieu, UML represents a consensus on the means to represent many aspects of the system to be developed. In *systems* engineering, a similar movement is underway (e.g., the SysML Partners "Systems modeling Language (SysML)"), with the aspiration of encompassing not just aspects of the system to be developed, but also of the development process itself (e.g., taking into account cost, schedule and work allocations). Vendors offer powerful tool suites that support the practical application of these principles, for example Vitech Corporation's CORE® product family of engineering development tools, or 3SL's Cradle® model based systems engineering environment.

2.4. Our vision – a fusion of all three

Our vision it to fuse key elements from all three of the above. We employ model based engineering as the linchpin of our approach because it spans both *development* aspects (cost, schedule, work breakdown) and *design* aspects (system decomposition, functional behavior, measures of performance). This wide scope is key to supporting reasoning about the "-ilities". Furthermore, model based engineering is rapidly gaining acceptance, thus we will not have to persuade engineers to adopt an unfamiliar toolset.

We augment model-based engineering in two key ways: we *reduce the effort* it takes to construct models and variations on them, and we *quantitatively couple the models' development and design aspects* to reflect the ways that development choices affect the operational qualities of the resulting designs. We achieve the effort-reduction by encoding "reference" design practices and knowledge in our domain as templates, and use these to expand problem-specific design descriptions. We achieve the quantitative coupling by interposing a risk-centric model that links development steps to their effects on risks, and risks to their effects on the expected operational behaviors of the system being designed. For example, choosing to construct the system out of higher quality components reduces the risks of certain kinds of system failures.

Both augmentations are essential to our vision. Motivated by the "design by shopping" paradigm, we wish to reveal a tradespace of alternatives. In our vision, however, generation of that tradespace is to encompass design *and* development choices. Work in the "design by shopping" field tends to have a design-centric focus (e.g., in the spacecraft design study reported in [3] the design variables were things like vehicle mass, and thickness of material). How a design is to be developed is rarely considered, so relatively few development concerns (usually only cost) can be taken into consideration. In order for our vision to function, we need the aforementioned coupling of development steps to their operational (run-time) implications. Furthermore, for our vision to be viable, generation of the tradespace should not require a level of effort disproportionately greater than it would take to construct a single model. Some form of automated generation of model variations is essential.

Requirements engineering aspects are brought to the fore in this vision: functional and non-functional requirements and preferences of all kinds can be understood in the context of the aforementioned tradespaces. For example, the consequences on each of the "-ilities" of tightening (or loosening) a scheduling requirement can be revealed. There is no longer the present-day gulf between reasoning about the design, and reasoning about the development by which the design is realized.

3. An approach to realization of our vision

In this section we outline the prototype we constructed to conduct an initial foray towards our vision and demonstrate its soundness. Our prototype takes the form of an assemblage of several capabilities, applied in the following order:

3.1. "Quantum" model creation

We begin with a standard systems engineering tool; in our experiments to date we used Vitech Corporation's CORE® for this purpose. Using this we represent two kinds of systems engineering information:

- *Specific* information on the design problem at hand. The operational scenarios it is to exhibit, the functions to be exercised in those scenarios, and

the decomposition of the system into subsystems, and subsystems into components.

- *Reference* information on the capabilities of standard subsystems and components, and on standard processes to be followed. Example capabilities are the functionality and failure modes of standard subsystems. Example processes are the sequence of design, development or acquisition, integration, and testing steps applied to development of any major subsystem, interspersed by various reviews, inspections and analyses.

A key aspect of our approach is that both kinds of information can include *choices – design* choices (e.g., choice of parts and materials) and *development* choices (e.g., optional inclusion of additional review and testing steps).

We developed code to automatically *combine* the information from these two sources. Roughly speaking, the problem-specific information is used as "seed" data, and the reference information is interpreted as templates, used to expand the seed information.

The net result is a systems engineering model that, as is common for such models, spans development through operation. Unusual, however, is the inclusion in the model of *choices* (among development alternatives and options). These will have ramifications on the later steps in the development and on the likelihoods of the various operational scenarios. At this point therefore our systems engineering model is in an indeterminate "quantum" state embodying *all possible consistent selections among those choices*. Additionally, some of the ramifications of the model (e.g., how long it would take and how much it would cost to develop a design characterized by a set of selections) have yet to be calculated.

To evaluate and explore those ramifications, and to help select from among them, we automatically transfer the information from the systems engineering tool to our risk-based decision support tool, described next.

3.2. Risk-based decision support tool

We use our home-grown risk-based decision support tool DDP [4] to perform the evaluations of various development and design alternatives and options, and to help guide the users in selecting among those alternatives.

For a specific selection among the development and design alternatives and options, we use the risk-based model encoded in DDP to calculate the measures of interest (the "evaluate" part of a typical "generate-evaluate-decide" cycle). Our approach encompasses both measures of the *operation* of the (candidate) system, and of the *development* of that system.

The measures of operation are usually system dependent, based on what the system is intended to achieve; classical requirements engineering methods have a role here in ascertaining what these are. For safety-critical systems, there is often a clear distinction between acceptable and unacceptable behaviors (those that lead to injury or death). In such situations, the focus is often on estimation of the likelihoods of these unacceptable outcomes. For other systems, or within the envelope of safe behaviors, there are often graduated measures of success, e.g., "how much science data will this spacecraft yield?" These are typically problem-specific.

The measures of development include ones of almost universal concern ("how much will it cost to develop the system?" "how long will it take?"), refinements of these (e.g., "how *evenly* is the workload spread over time?"), and perhaps some problem-specific aspects (e.g., "how much of a lead system architect's time will be needed?").

The systems engineering tool is used to calculate for each scenario its measures of success, and passes this information over to DDP. Combining these with its calculation of scenario likelihoods, DDP can assess the *expected* amounts of measures of success (in principle we could calculate probability distributions). The fine details of how we arrange the information transfer from the systems engineering tool to DDP so as to make this possible are expounded in [5]. Underpinning it is a probabilistic treatment of uncertainty somewhat akin to that in [6] and [7].

It is important to note that in our integrated approach we calculate implications of development-time decisions on both the subsequent steps of the development itself, and on the reliability of the design that results. For example, a development-time decision to use high-quality (rather than low-quality) parts decreases the likelihood that subsequent testing will reveal the need to replace faulty parts, and will improve the reliability of the design. Likewise, adoption of an early-phase review will, by catching defects early, save the (typically far greater) downstream cost of repairing them. These calculations take into account:

- The "unit" cost of reworking (repairing) a defect,
- A "multiplication" factor capturing the escalation of expense of rework/repair of defects the later they are left (e.g., the oft-reported phenomenon that the cost of correcting a software requirements bug escalates through the development lifecycle).
- The "amount" of defects to be reworked. This is proportional to the prevalence of defects, and the

fraction of them that the measure detects. Prevalence depends on how likely those defects were in the first place (e.g., less so if high-quality parts were utilized), and the net effect of other detection measures that have already taken place.

3.3 Generation, visualization and selection

Our highly integrated model now contains a large quantity of design and development information. Furthermore, this information includes *choices* (our so-called "quantum model"). These choices give rise to vastly many permutations of developments, and designs they yield. It would be infeasible to generate them all. Instead, we use heuristic search to explore favored regions of the design space.

This exploratory approach is widely used for design optimization of many forms. The community represented by this conference may be familiar with Sutcliffe's use of evolutionary computing techniques applied to selecting components in socio-technical system designs [8]. At the heart of these approaches is a "generate-evaluate-decide" cycle. Somehow, a set of choices are made that yield a candidate design. This is then evaluated to yield the measures of interest (e.g., cost, performance). That evaluation, together with the history of previous such design evaluations, is then used to determine the next set of decisions, and so on.

We use "simulated annealing" to guide the exploration towards the more optimal regions of the tradespace. We also provide a convenient interface through which the users can manually make their own selection decisions.

Once exploration is underway, cogent visualization is highly effective as a means for allowing skilled designers to comprehend the rich tradespace. We have experimented with a number of visualization capabilities. One is the ATSV tool [3], which includes the capability to utilize 3 spatial dimensions, plus other visual cues such as color, size, orientation and "glyphs", and dynamic mechanisms of filtering and highlighting. Taken together, these give us the means to scrutinize multi-dimensional portrayals of designs' characteristics. Our DDP tool has several (comparatively modest) visualization capabilities for study of the model of development and design. Finally, CORE can export schedule information to tools such as Microsoft Project®, whose forms of presentation are familiar to a wide audience.

The purpose of all of these is to inform designers of the tradespace of available options, and its makeup. They can use this information to help select their preferred designs, to explore "what-if" scenarios to (perhaps) motivate revising requirements (e.g., "what if we had 10% more budget?), and to explore sensitivities of designs to the information on which they are based.

4. Demonstration

In this section we present fragments from an example we ran through our prototype. Since we work to help NASA, our example models a space mission design. Nevertheless, its salient aspects as regards requirements engineering and design are common to many terrestrial systems: cost, schedule and other resource limitations constrain the development options; the system's intended operation takes the form of a sequence of steps, with possible failures along the way; multiple measures characterize the performance of the system.

We encoded within the systems engineering tool some reference information, of institutional processes (on the makeup and sequencing of development steps for systems and subsystems) and the relevant capabilities of a few standard subsystems. We also encoded within the systems engineering tool the mission-specific information – a space mission that would go from the earth to the moon, gather some moon rocks and perform some science experiments there, and return to earth. We embedded choices within both forms of information – e.g., in the reference information, choices of whether to perform certain optional reviews, tests, etc.; in the mission-specific information, choices of make vs. buy decisions for subsystems, and choices of component parts of different quality levels. We used our code to generate the "quantum schedule" that represents these choices.

This representation is transferred to our DDP tool, which is used to compute the following measures of a given selection of choices:

- Total development cost
- Elapsed time from start to finish of development (determined by the development's critical path)
- Likelihoods of classes of scenarios (e.g., aborted missions)
- Expected values for:
  - science data transmitted back to earth, and
  - lunar rock mass gathered and returned to earth.

DDP has several built-in visualization capabilities that help reveal the detailed makeup of a design's contribution to these measures.

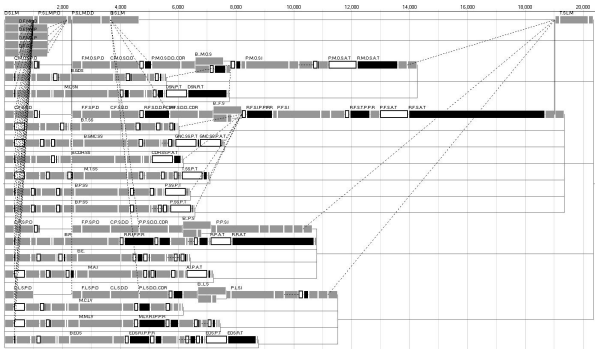

Figure 1 – DDP-generated Gantt chart

An example of a Gantt chart generated by DDP for one of the possible development plans is shown in Figure 1. The rectangles represent tasks, with each rectangle's length proportional to the task duration. Rectangle fillings distinguish three kinds of tasks:

- grey = standard development activities
- hollow = activities that potentially detect defects
- black = rework/repair of detected defects

Note the strikingly long black rectangle extending towards the right of the figure, indicating a long-duration activity to repair defects. It is on the schedule's critical path, and so might motivate the designer to seek earlier-lifecycle ways to reduce the likelihood of those defects, to decrease the duration of that repair, and therefore decrease the overall development duration.

Even in our simple demonstration example there are almost 200 distinct choices. The size of the search space this represents is approaching 2^{200}, or roughly 10^{60}. We use a "generate-evaluate-decide" cycle to yield interesting points in this space.

To help designers scrutinize this space, we use the visualization capabilities of the ATSV tool [3]. Figure 2 shows a relatively simple example of this. The three axes have been set to portray:

- Cost (axis towards the lower left)
- Duration (axis towards the right)
- Total benefit (axis towards the top)

Total benefit is an aggregate of our model's two measures of value, the proportion of mass of rocks returned to Earth out of the maximum possible such, combined with the proportion of science instrument data transmitted back to Earth out of the maximum possible such. Should we wish to do so, we could display these in separate axes.

Each tiny cube in the display represents a different development plan and resulting design. The two sizes of icons distinguish whether or not each plan includes a particular optional activity:

- Small = design *omits* that activity
- Large = design *includes* that activity

The particular activity is an optional design peer review for one of the subsystems. From the figure it is discernable that the large points (designs that include the activity) tend to lead to lower overall durations. This indicates that the activity catches problems early, and so more than pays for itself later.

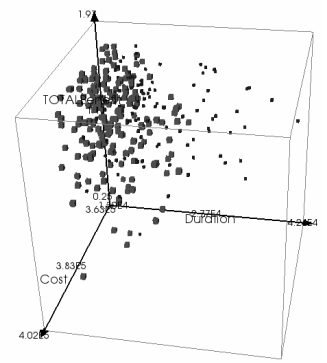

Figure 2 - Using ATSV to reveal an option with a dramatic effect on duration

The point is that ATSV makes it easy to quickly step through the options, and see patterns such as this. More generally, ATSV is adept at presenting the overall shape and structure of the search space, a very useful capability for our purposes.

5. Implications, challenges, opportunities

Our vision extends the practicality of model-based engineering to straddle the development / design gulf, and so makes possible the quantitative treatment of requirements over this entire continuum. Its generation of development plans from a combination of reference information and project-specific details helps reduce the modeling effort, and ensures that the plans are correct by construction with respect to that reference information (e.g., institutional practices).

Thus institutions will be more able to capitalize on their design expertise knowledge: individual corporate processes, gate products and various hard-learned sets of rules of thumb, which up until now have been encoded in documentation for human consumption, will become reference knowledge on tap for widespread application and tailoring. Systems engineers will be liberated from low-level consistency checking of plans against standards. Their focus will instead be on getting the coupled design-development right. This will raise the level of design discourse: engineers' time will be utilized to rapidly investigate design *spaces*, not just point designs.

Rather than focusing solely on the accuracy of the produced plans and designs, companies will be focused on the accuracy/validity of the input data from which

they are generated. Reviews will be able to focus on the source of the various inputs and on more subtle, complex interactions that may have been inadequately modeled or represented.

There are, however, some significant challenges that lie ahead if our vision is to be realized. Many of the challenges involve getting users to adopt the approach, as there are some non-recurring expenses which must be 'paid' up front. In particular, the reference information will take time to gather, formalize, and render consistent.

Our prototype is sufficient to illustrate the concept, but will quickly run into problems of scale if used as-is. In the area of risk-based calculation, we must leverage the work that takes place in the reliability and risk assessment community to address large-scale problems. Even in our "simple" example the plethora of choices, their cross-coupling to risks, etc., combine make it difficult to grasp *why* things happen, and *why* the tradespace takes the form it does. This is somewhat reminiscent of program debugging – why does this behavior have these characteristics? We believe there is much to be done with reasoning and visualization to reveal not only the tradespace, but also the factors that shape it.

An area in which our approach is somewhat weak is its treatment of problem decomposition (model refinement). Goal-oriented requirements engineering addresses this head-on. We note the trend towards coupling such work with executable models, e.g., [9], and with quantitative reasoning, e.g., [10]. We seem to be approaching this same problem from the opposite direction. Some blend of these approaches would seem appropriate.

6. 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 and funded through NASA's Exploration Systems Mission Directorate. 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.

We especially thank Matt Brinza for his contribution to an early software prototype of these ideas, Mike Yukish and his group for use of their ATSV tool, and Profs. Tim Menzies and James Kiper for many influential conversations. We are grateful for the ongoing support and guidance of this effort provided by Steve Prusha, Steve Wall and Freddie Douglas.

7. References

[1] C-L. Hwang, and A.S. Masud, 1979, "Multiple Objective Decision Making - Methods and Applications", *Lecture Notes in Economics and Mathematical Systems No. 164*, Springer-Verlag, 1979.

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

[3] G. Stump, M. Yukish, T.W. Simpson, and E.N. Harris, "Design Space Visualization and its Application to a Design by Shopping Paradigm", *Proc. DETC'03 ASME Design Engineering Technical Conferences and Computers and Information In Engineering Conference*, Chicago, Sep 2003.

[4] M.S. Feather, and S.L. Cornford "Quantitative risk-based requirements reasoning", *Requirements Engineering* (Springer), 8(4), 2003, pp. 248-265.

[5] S.L. Cornford, M.S. Feather, and J.S. Jenkins, "Intertwining Risk Insights and Design Decisions", *Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management*, Paper PSAM-0193, New Orleans, May 2006.

[6] V.C. Cortellessa, K. Goseva-Popstojanova, K. Appukkutty, A.R. Guedem, A. Hassan, R. Elnaggar, W. Abdelmoez, and H.H. Ammar, "Model-Based Performance Risk Analysis", *IEEE Transactions on Software Engineering*, 31(1), Jan 2005, pp. 3-20.

[7] R.L. Dillon, M. E. Paté-Cornell, and S.D. Guikema, "Optimal Use of Budget Reserves to Minimize Technical and Management Failure Risks During Complex Project Development", *IEEE Transactions on Engineering Management*, 52(3), Aug 2005, pp. 382-395.

[8] A. Sutcliffe, W-C. Chang, and R. Neville, "Evolutionary Requirements Analysis", *Proc. 11th IEEE International Requirements Engineering Conference*, Monterey Bay, Sep 2003, pp. 264-273.

[9] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso, "Specifying and analyzing early requirements in Tropos", *Requirements Engineering* 9(2), May 2004, pp. 132-150.

[10] E. Letier, and A. van Lamsweerde, "Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering", *Proc., ACM/SIGSOFT 2004/FSE-12*, 2004.