Managing Priorities:

A Key to Systematic Decision-Making

Tom Gilb
Tom@Gilb.com
Mark W. Maier
Mark.w.maier@aero.org

Abstract. A central concern of systems engineering is selecting the most preferred alternatives for implementation from among competing options. The selection process is sometimes called tradeoff analysis, and is often built on the methods of decision analysis and utility theory. The process can be loosely divided into two parts, a first part in which one determines the relative priority of various requirements, and a second part, a design selection phase, in which alternatives are compared, and the preferred alternatives chosen.

This paper discusses the means of determining the priority order for implementing system changes. It also outlines the implications on the selection process of evolutionary systems development.

BACKGROUND

There is an enormous literature on tradeoff analysis and decision theory. A typical generic process for making tradeoffs as currently found in the literature consists of the following steps:

#1. Decompose one's understanding of the overall problem to be solved into discrete requirements;
#2. Determine the relative preference for the different requirements. Mathematical functions are commonly used to express stakeholders' preferences for different *functional requirements*. They go by a variety of terms: 'figure of merit', 'value function' and 'utility function' are all used, though with somewhat different meanings (For example, in the formal literature, a utility function possesses formal properties beyond simply capturing a given stakeholder's preferences between two deterministic alternatives) (Kreps, 1988, Daniels et al. 2001);
#3. Evaluate the utility (value) for a set of alternative choices of system configuration;
#4. Make a decision based on the evaluated utility of the alternative choices.

In this traditional process, we make an exclusive decision in step #4, we select the most preferred alternative. In many realistic cases we might choose other courses of action, such as expanding our search, or rethinking the original problem because of poor performance of all known alternatives, returning to the sponsors to recommend constraints be relaxed, or make an incremental implementation decision on the elements that seem to be most certain, and restart the selection process in a new increment.

All of these steps have been addressed in the literature, at varying levels of formalism from entirely heuristic to rigorously mathematical. Two basic references are (Keeney and Raiffa 1976) and (Keeney 1992). The first of these is a canonical textbook on the construction of utility

functions and their use in decision analysis. It emphasizes the use of additive utility functions, but does not advocate the use of the common (and incorrect) subjectively weighted additive utility functions. The second of these books (Keeney 1992) is closest to the subject of this paper. In it, Keeney describes a wide variety of heuristics for determining the objectives for a decision, and discusses creating scales of measure, and the suitability of sets of objectives for making decisions. Keeney also discusses alternatives to hard decisions in step #4. In a recent paper, Daniels et al. (2001) discusses detailed issues pertinent to step #2 if one is using additive utility functions, although the terminology, "figures-of-merit" is used there. The Analytical Hierarchy Process (AHP) (Saaty 1980, Saaty 1988) takes its own perspective on steps #1 through #4. In AHP objectives are decomposed hierarchically downward with priority determined by a specific weight elicitation process. Alternatives are compared on the objectives without detailed scales, but using a specific pair-wise scale unique to AHP. In place of a value or utility function there is an additive roll-up based on the pair-wise comparisons. The AHP method is not formally equivalent to utility theory - a subject that has generated many papers over the years.

While the decision theory literature defines the generic steps with great rigor, real-world applications virtually never reach the idea of rationality captured in decision theory. Some of the reasons for imperfect practice are themselves theoretically sound. For example, we know from a variety of psychological experiments that people's actual behavior and preferences do not correspond precisely to the assumptions of utility theory. In addition, real decisions have to be made on deadlines, and information can never be perfect. Real decision makers are forced into the use of heuristics for such matters as choosing when to stop gathering information, how much complexity to deal with, and how to trade off risks. Moreover, it is also observed that actual practice falls short in adopting overly simplified models and ad-hoc methods simply because well-grounded, but pragmatic tools are not widely available (Gilb 1976).

This paper proposes that Planguage, a specification language and set of methods, which one of the authors (Gilb) has developed over many years, has the capability to address many of the issues discussed above regarding the selection process. Specifically, the paper discusses specifying priority information within requirements specifications (Step #2) (Gilb 2005a: Chapters 2, 4 and 5), and the implications of an evolutionary systems development context (Gilb 2005a: Chapter 10, Larman and Basili 2003, MacCormack et al. 2003).

CONTEXT FOR PRIORITY DETERMINATION

Priority determination should be:

- An *information-based process*, which makes full use of the current and continuously available factual information and, is able to reuse information;
- A *dynamic process*, which uses feedback from the on-going implementation, and is open to instigating and catering for changes in both requirements and design ideas;
- A *resource-focused process*, which considers Return on Investment (ROI) and takes into account resource availability.

For the purposes of this paper we assume a fairly large-scale systems engineering environment, which includes:

- Multiple stakeholders, each probably ignorant of the environments of some of the other stakeholders;
- Multiple competing (for common scarce resources) requirements from the stakeholders;

- (Understandable) ignorance by stakeholders of the real resource costs of delivering their requirements;
- Lack of an agreed understanding of the overall system-wide picture: both the system-wide requirements and the actual, available system-wide resources;
- Complex decision-making processes (including meetings and reviews), which are affected by personal management agendas;
- Uncertainty about the future, in which resources, requirements, authority, power and technology will change;
- An evolutionary delivery environment where we can deliver systems in increments, learn about objectives and priorities at each increment, and can (and must) feed that learning back into subsequent development.

WHAT IS PRIORITY?

This paper defines priority as:

> **"Priority is relative right of a requirement to the utilization of limited (or scarce) resources ".**

In other worlds, if resources were unlimited, there would be no need to prioritize things. You could have it all. In this definition, 'resources' means all types of resources including time to deadline, human effort, money, space, and any other resource notion imaginable.

Systems engineering is a constant stream of priority evaluations. Therefore the concept of priority determination (deciding the priorities for system change) is a primary concept, in both systems engineering, and the management of systems engineering projects. However, it seems to us that we do not even start our discussions about the 'priority' discipline with well-defined 'priority' concepts. In particular, we mix up the specification of stakeholder requirements with the evaluation or determination of priority – even though it is an essential tenet of the literature on decision theory to avoid confusing these two separate elicitations! We must distinguish the specification of the requirements (the attributes) clearly from the determination of priority (which includes the tradeoff process to negotiate the preferred set of attribute levels). Stakeholder requirements are what they want (because they value it), priorities are how they are willing to trade among varying levels of requirement satisfaction – given limited resources. This paper therefore discusses these two, albeit related, topics separately as follows:

- Specification of priority – what are the specific priority signals, which a systems engineering specification can give us to evaluate?
- The priority evaluation process - how do we determine what to give priority to – what to do now, what to do first; given that interesting systems engineering projects don't make decisions for all time? Projects will inevitably develop incrementally, and the incremental development will affect later priority determinations.

First however, it is important to consider some of the problems with the determination, and use of, simple weighting schemes in prioritization.

WHAT IS WRONG WITH THE *WEIGHTING PROCESS* FOR DETERMINING PRIORITY?

In many priority processes, each element of a set of requirements in a decision-making model is subjectively[1] assigned a numeric value, a 'weighting', indicating its priority (for example, a value on a scale of 1 to 10 or, a percentage weight). The degree of subjectivity involved in these weightings is determined by such factors as the actual people asked (the number of people, their roles and, their expertise) and how they arrive at their decisions (their decision processes, including such things as their influences). In many cases, people are asked on a one-off basis during a group meeting to assign numerous comparative weightings 'off-the-top-of-their-heads'. Inadequate documentation of who, when and, why (experience and/or fact) is widespread.

The reasons why a weighting process is weak can be summarized as follows:

1. **Information 'Overload':** There is a great deal of information about a requirement that any reasonable person would want to know and use in order to determine the priority. For example, the delivery timing, the geographical positioning, the assumptions, the conditions, the level of risk, the level of authority of the people proposing it, the known set of stakeholders it effects, and much more. However, for the purposes of subjective weighting, the problem is that there is too much information in existence for any given requirement: there is typically more real information available[2] than can be utilized in a quick subjective evaluation to determine a weight. Subjective assessment all at once, of all the numerous factors, is impractical;
2. **Lack of Complete Information**: Conversely, another common problem is that requirement specifications often fail to capture all the information necessary for determining priority. The full range of contributory information is typically lacking (for example, there is usually no defined scale of measure for the requirements that vary in level (like Usability, Portability) – also known as performance requirements). Lack of certain information will seriously hinder making any correct priority decision;
3. **'One-off' Weighting:** The weightings tend to be 'frozen', that is they are not reassessed frequently throughout a project. 'Real' priority is not static (Consider, for example, your priority for sleep or food). Priorities are 'dynamic' and 'computable', depending on satisfaction of needs, and on residual scarce resources. Even more

---

[1] A 'subjective' decision is one that is made based on points of view in the absence of, and without specification of, a determination process and the evidence used to arrive at the decision.

(Subjective: "depending on personal idiosyncrasy or individual point of view, not producing the effect of literal and impartial transcription of external realities."

Objective: "dealing with outward things, exhibiting actual facts uncoloured by exhibitor's feelings or opinions." The Concise Oxford Dictionary)

Note: In this paper, we are objecting to subjective *weightings*. We are not objecting to stakeholders proposing their own subjective *requirements*.

[2] Unfortunately, although the information is somehow available, it is typically not accurate, written, easily accessible, and verified. Consequently it will not be used in practice, unless it is in the head of the individual suggesting weights. The purpose and opportunity of Planguage specification is to allow relevant priority-determining information to be collected, written, verified and available for all decision-makers, at all times thereafter.

importantly, for an incrementally-developed system, is that priorities are likely to change, as stakeholders gain experience with early versions, as the system environment changes, and as the process of development reveals new facts about both values and costs.[3]

4. **Lack of Consideration of Resources:** Resources are usually not considered when allocating weightings. It should be clear that ultimate priority (not just need or value), by its nature, totally depends on the type and quantity of available resources. We cannot give implementation priority to something if the necessary resources for implementing it are, in practice, not available. Therefore allocation of weights, without knowing or considering resources, is an illogical exercise, or at least, an incomplete assessment of the information needed to determine the real priority. In our view, methods without resource consideration that rely just on stakeholder need and value, are condemned to the risk of bad decisions in the real world. (See also discussion in a later section.)

   Of course, there is a caveat here: it is important that 'great requirements' are not discarded 'upfront' because of their resource requirements. It may be that the correct solution is that the budgets are increased. So what is being proposed here is that there is are at least, two iterations: the first iteration specifies the required levels, and then a second iteration considers the associated resource requirements, and asks stakeholders if they want to modify their requirements.

5. **An Individual Stakeholder's Viewpoint is Limited:** A person's subjective judgment depends on experience, access to information, and evaluation of that information. Typically, individual stakeholders do not have access to sufficient system-wide information. Even if they did have information they might not be able or willing to apply it. They can participate in supplying their requirements (including their stakeholder values, without knowledge or respect for the other stakeholder values) and committing their stakeholder resources, but they are certainly unable to make a globally optimal priority decision on behalf of the entire stakeholder community (They would at best probably sub-optimize the decision based on their information).

   So stakeholders will produce weightings skewed to their individual viewpoints. Further, if a group of diverse stakeholders meets to determine weightings, then authority, organizational politics and personality are also likely to affect the outcome. Fundamentally, in multi-stakeholder situations there may be no rational, stable set of priorities determinable by considering only the set of all the individual stakeholder's preferences. To determine priority order, solely by means of a meeting, is foolhardy.

   Of course, you should always ask the stakeholders for their opinions on their requirements and their values, so that you have some pointers that you might have missed otherwise. However, individual stakeholder needs must be determined as part of the process – it must not be the end of the process. It is essential that priority decisions are made in an evolutionary manner, and are based on explicitly stated priority policy, the specified set of requirements, and the specified set of design ideas, evidence and feedback. You can of course choose to review any priority decisions

[3] This should be obvious to the reader, and it is well supported by all the referenced literature about Evolutionary Project Management methods such as (Larman 2003), (MacCormack 2003) and others.

with any stakeholders, to get agreement, consensus, and to identify any residual or consequential implementation or operational problems.

WHAT IS WRONG WITH USING *WEIGHTINGS*?

We acknowledge the need to give information, from stakeholders to decision-makers, about stakeholder values. One widely used direct way of doing so is using specified weights. What do these weights tell us? They convey the information that the 'accomplishment' of a given objective is viewed by a specific set of stakeholders as being 'this important' compared to other objectives. Weightings are typically expressed by using numeric rankings. Usually the notion of the accomplishment for the objective is not clearly specified. It is typically not specified numerically, and typically only refers to a 'name' (like 'better productivity', 'increased availability').

Gilb (one of the authors of this paper) used this form of weighting method for about 27 years up to 1987. But around that time he had developed the art of quantifying any performance dimension, including any quality dimension, of a decision-making problem. He consequently realized the following:

- The 'priority' is for reaching a specific level of performance under stated conditions (that is, at a given time and place, or on a given event occurring);
- When that level is reached on time, the performance dimension clearly has no more 'claim on scarce resources'. The job is done, and the priority changes (Just like when you have enough to eat, your priority for food changes).

Consequently, Gilb realized that fixed weights are a bad way to represent priorities. They do not relate clearly to the target levels, the degree of fulfillment of the objective, the timing, or the conditions. The better way to represent priorities had to be in terms of the numeric target levels for the critical dimensions of a project or system; and in terms of the degree these target levels were reached. This method of determining priorities is exactly the same one used by nature, when your body decides on its priorities.

So, why do people feel they need weighting schemes? We believe it is because they have not mastered the necessary art of quantifying all their related dimensions[4]. They therefore do not have the logical basis for an alternative method, and so fall back on weighting.

Now clearly all planning, and consequent decision making, is better off if all the critical variable dimensions are clearly specified, and are quantified. We need to mandate this practice of quantification of critical variables (as many of our clients have in fact done): define a scale of measure; specify the goal level, the conditions and the deadline. We need to do this anyway - irrespective of our need to use models to make comparative decisions.

Once we have accepted the necessity of quantification, we can throw away the unnecessary crutches of weights to determine what has currently got the biggest 'claim on scarce resources', and instead make use of:

- The natural weighting information given by the quantified specification;
- The natural evolutionary movement towards target satisfaction (eating reduces priority for food);
- Other information (merely outlined in this paper), such as risks and dependencies.

[4] The art of quantification is described in the Competitive Engineering chapter on Scales of Measure (Gilb 2005a).

SPECIFICATION OF PRIORITY USING PLANGUAGE

Planguage[5] is a specification language and set of methods, which gives a wide variety of ways of identifying, structuring, and evaluating information about requirements and design ideas (solutions) (Gilb 2005a, Gilb 1988). One of its aims is to enable systems engineers to make better decisions about priority.

Some aspects of Planguage, relevant to priority, are discussed in the following sections: the Planguage specification language, and then two of the Planguage methods, Impact Estimation (IE) and Evolutionary Project Management (Evo).

PLANGUAGE SPECIFICATION LANGUAGE

The Planguage methods for priority are a pragmatic response to the central problems discussed above. The basic idea of the Planguage specification language is to allow any systems engineer to collect all the information that might be useful for understanding a requirement specification, and its priority, in one specification place, and to capture it in a fashion easy to communicate and analyze. The specification becomes a detailed, reusable 'object'. Planguage is built on a specific glossary of terms (Gilb 2005b). The most important terms are discussed below.

Within Planguage requirement specification, five main types of requirement attribute are identified: performance, resource, function, design, and condition. All the scalar requirement types can be specified simultaneously as target levels (levels we aim to get to) and constraint levels (levels we are warned to avoid). As a simple analogy, consider room temperature: we aim for just right, and avoid too hot or too cold. The reader might like to consider our 'priority', or values, in this case.

Each requirement attribute is can be partially described using a set of parameters. Some key Planguage specification parameters[6], which assist priority determination, include:

- Source;
- Authority;
- Stakeholder;
- Qualifiers {time, place, event};
- Constraint {Fail, Survival};
- Target {Goal (for performance attributes) or Budget (for resource attributes), Stretch, Wish}.

Source. Information about the Source of a specification tells you where it originated and helps you to track it back to its roots. This is especially important if you need to question or check something. A Source can be a person or a document. When source is known, a lot of priority information is available as a consequence.

Authority. Information about the Authority, which stated a requirement, enables you to assess the importance to attach to it. An Authority is usually a role (like CEO), or corporate standards or plans, but it can additionally state the name of the person responsible for it.

Stakeholder. In our experience our clients easily identify 20 to 30 interesting stakeholders for a

[5] Pronounced plan-guage, rhymes with language.

[6] If the reader would like to go far deeper into the grammar of Planguage they should refer to the Competitive Engineering book (Gilb 2005a), or www.gilb.com. In this paper, we will normally stick to the priority concepts and not worry too much about other aspects of Planguage.

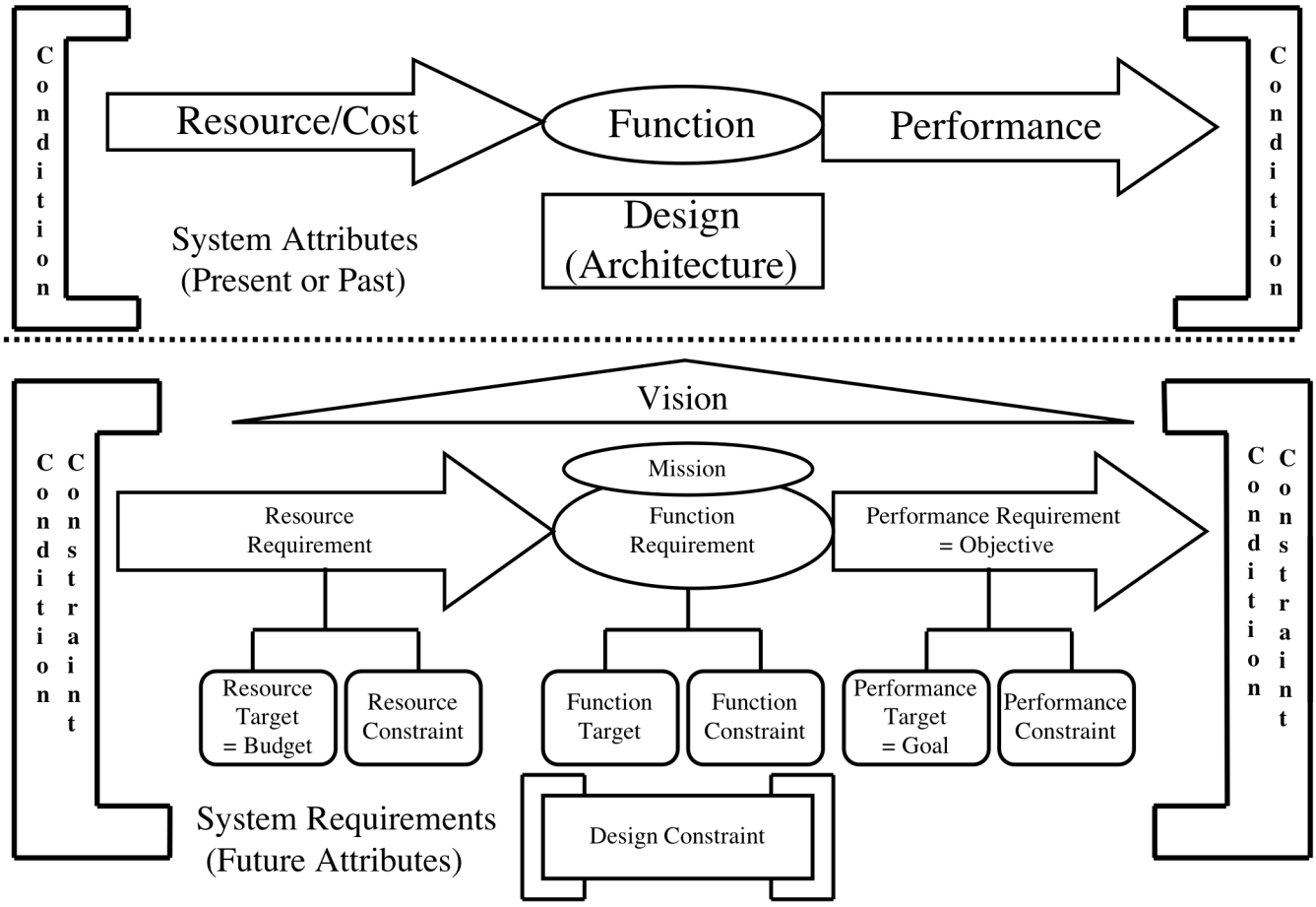

Figure 1. Various classes of requirements (Gilb 2005a: Figure 2.2). Notice the existence of both 'target' levels and 'constraint' levels in both performance and resource requirements. This is more complicated than simplistic models – but it is arguably more realistic and useful

project. An important element of priority thought is to recognize that any *single* requirement may have *more than one* stakeholder. By analyzing requirements against stakeholders, it is possible to understand who will be affected by delivery of specific system changes. Ideally a matrix of requirements versus stakeholders would be constructed to show potential priority conflicts. In practice, Planguage specification collects the basic information for such a matrix at the level of the single requirement.

'Stakeholders' is a defined parameter in Planguage. You should list the different stakeholders as follows: (*Note the '{…}' indicates a set or list.*)

Stakeholders: {Installers, Testers, Users, European Union}.

We need also to take into account when considering priorities, that different stakeholders have different levels of authority, and that specifications have different economic consequences for different stakeholders.

Qualifiers. A specification will normally have a set of 'qualifiers' to specify the **time, place and event conditions** for which it is valid. This information is key for priority determination.

See Figure 2 for an example.

Availability:

Scale: % Uptime for defined [Environment] during a defined [Time Period].

Meter: Automated service logs and uptime reports from them.

Survival [USA, Environment = Air Traffic Control, Time Period = End 2005, If FAA Reg. 23.39 is in force]: 99.90%.

Fail [Environment = Los Angeles Airport And Single Runway, Time Period = Initial Delivery]: 99.98% <- LAX Contract.

Figure 2. Note Qualifiers are specified by using square brackets

Constraint. A constraint is a specification that should be met, otherwise certain definable negative consequences will follow. Constraints cannot be absolute, because the cost of respecting the constraint can exceed the negative consequence in the overall value schemes of the authoritative stakeholders. Sometimes constraints will conflict with other requirements, including limited resources, and then conflict resolution is required. Constraints are either binary or scalar. Types of binary constraints include function constraints, design constraints, and condition constraints. Scalar constraints are either performance constraints or resource constraints. There are two basic kinds of scalar constraint: a Fail level which is a point where defined negative consequences appear (such as loss of market share, or safety risks) and, a Survival level, which indicates the level which is required for system survival. The Fail level is a weak priority signal. It says – something is wrong and you should not be at this level – so get out of the fire. The Survival level says, 'if you get any worse' you risk death of the system. For example, a Fail level is like 'I am having difficult breathing', and a Survival level is like 'I am just about to suffocate and die'. It is obvious that Survival levels send a stronger message about priority than Fail. These levels are part of a continuum along the scale of measure. We assume that there are varied degrees of failure, and varied degrees of threat to survival as we move about the scale from an edge of the Fail range or the Survival range. Survival and failure may be limited to the viewpoint of a single stakeholder. See Figure 3.

Personal Survival:

Scale: Average amount of air, in liters, needed per defined [Time Period: Default: 5 minutes] in defined [Situation] to avoid death by suffocation.

Survival [Situation = Escape Burning Building]: 50 liters.

Survival [Book Reading]: 5 liters.

Figure 3. Specification terms in [qualifier brackets] are conditions that must be true for the requirement to be valid. Obviously priority changes, depending on the truth of the specified conditions

Target. A target is a specification that is aimed at to achieve success of some kind. It can be a binary function target, a scalar performance target (a goal level), or a scalar resource target (a budget). There are three kinds of *scalar* target: Goal/Budget, Stretch and Wish. A Goal/Budget level is the required level. The Goal/Budget levels are the most important targets. A Stretch level is an ambitious level, which is set as a challenge: meeting a Stretch level would be great, but it is not essential. A Wish level is stated to capture an 'ultimate dream'. A 'Wish' level is not intended to be reached, if there is no budget allocation to achieve it, or if the technology is

outside the state of the art. It does however express a stakeholder value – independent of cost or feasibility.

Notice the priority information in each of these three target concepts. The specification of each target level is itself an elicitation from stakeholders of their preference for specified levels of that attribute. The structure of the elicitation is important, and forms a bridge between the heuristic and pragmatic and the theoretical. The levels (Survival through Wish) can correspond to points on a single attribute utility function. Single attribute utility functions are often (though not always) "S" shaped. They rise slowly for low values of the attribute (too low to care about), rise quickly when the performance is valuable, and rise slowly again as we are nearly fully satisfied. It is much more likely that a weight can be found that can represent the relative priority of such a function of the attribute, than such a weight can found for the original performance measure. A paper by Daniels et al. (2001) references a variety of shapes for such functions. References on multi-attribute theory (Keeney and Raiffa 1976) discuss the formal conditions under which a utility function can be broken down into single attribute functions with simple combination. See Figure 4, which shows a simple example of a scalar performance requirement containing five specific requirements: two constraints and three targets.

Availability:

Scale: % Uptime for defined [Environment] during a defined [Time Period].

Meter: Automated service logs and uptime reports from them.

-------- Scalar Constraints ----------==========----------

Survival [USA And Air Traffic Control, End 2005, If FAA Reg. 23.39 is in force]: 99.90[7]%.

Fail [Los Angeles Airport And Single Runway, Initial Delivery]: 99.98% <- LAX Contract.

-------- Scalar Targets (Performance Goals) -==========--

Goal [Worldwide And Major International Airports, If Required by Law]: 99.95%.

Authority: {Applicable Laws, Regulations And Contracts}.

Stretch [For Selected Airports, If Contracted]: 99.998%.

Stakeholders: Selected Airports [Worldwide].

Wish [Any Airport, If No Additional Cost]: GE 99.999%.

Authority: Market Director.

GE: "Greater than or equal to".

Figure 4. Note 'Availability' is a name tag, that gives us a permanent cross-reference name to access all information about that requirement. Everything that follows the tag either defines the requirement, or enriches our (priority) understanding of it (like Authority or Stakeholder). A 'Scale' defines the basic 'Availability' concept as a scalar variable. We can now put numbers on the scale. A 'Meter' defines a way to measure where a system is operating on the scale.

[7] Within Planguage specifications; there will be situations where subjective numeric levels have to be stated. Clear indication should be given if there is no, or insufficient, evidence for a numeric level (For example, indicate clearly that the value is a "Guess" in the Source parameter information). There will also be situations where there is predicted or known variation in a required numeric value. In such cases, the specific range of the numeric value should be specified by using a ± symbol.

PLANGUAGE METHODS SUPPORTING PRIORITY DETERMINATION

Impact Estimation (IE). IE tables are usually based on the use and definitions of the scalar requirements described above. They are in principle of the same outline structure as Quality Function Deployment. However, with Planguage's use of numeric scalar requirements they are more specific in the analysis of the impact of design ideas, and more specific about the likelihood of achieving requirements. In addition, the resource utilization (cost) of design ideas is captured, which permits benefit-to-cost ratio comparisons amongst design ideas. For priority determination purposes, an IE table is useful in identifying which design ideas are needed to achieve a specific prioritized requirement (Gilb 2005a, Gilb 1998, Gilb 1988).

Evolutionary Project Management (Evo)[8]. Evo sub-divides a project into many steps (say 50 steps, or 2% of cost or time budgets steps), each of which deliver some stakeholder benefit (as defined by formal requirements). Most of the decision theory based work on priority is built on the assumption (perhaps implicit) that we are deciding once. The classic case assumes we will select a configuration and then have to live with it. This is rarely the case in reality. Almost any type of system can be brought to ultimate fruition through a series of value delivery steps to various stakeholders. This is as true of aircraft, electronics systems as it is software and organizations. Rolls Royce advertised with the "20,000 evolutionary steps that makes a Rolls Royce". Since value is delivered to some stakeholder at each delivery step, subsequent value delivery steps can be based on feedback, and enhanced understanding from the earlier steps. This has as a consequence that priority can be re-evaluated at each step! The stakeholder values themselves might also change as steps progress, but even if values are held constant, plenty of other things change (remaining resource, external competition, laws, social change, management personalities); things that can necessarily be inputs to determining our current priorities for the 'next step'. For priority determination and implementation this puts a premium on evolution, feedback, and reuse in the priority and decision models.

At each step, two factors will immediately be measured for feedback purposes:

- the degrees to which the requirements (function and performance/quality) are incremented over the previous step;
- the degrees to which resources have been expended by the step.

The results will be compared to step plans or expectations, both on an individual step basis and a cumulative basis.

Final commitment to delivering a specific Evo Step is only made for the next step, after delivery and feedback from the previous Evo step.

For priority determination purposes, Evo means that we can track step-by-step the delivery of requirements and the expenditure of resources. And, after each Evo step, we can reassess our current priorities and, if necessary, modify our Evo plan. This is a much more dynamic approach to priority handling than permitted by Waterfall (also known as 'Traditional' or 'Grand Design') project management methods. It also challenges the conventional 'weighting' process, which as explained earlier, tends to be a 'one-off' process.

---

[8] Note Evolutionary Acquisition is now a mandatory guideline for the US Department of Defense (DoD 1998).

Step-> Target Require-ment	**STEP1** Plan % (of Target)	actual %	deviation %	**STEP2** to **STEP20** Plan %	plan cumulated to here %	**STEP21** **[CA,NV,WA]** Plan %	plan cumulated to here %	**STEP22** [all others] Plan %	plan cumulated to here %
PERF-1	*5*	*3*	*-2*	*40*	*43*	*40*	*83*	*-20*	*63*
PERF-2	*10*	*12*	*+2*	*50*	*62*	*30*	*92*	*60*	*152*
PERF-3	*20*	*13*	*-7*	*20*	*33*	*20*	*53*	*30*	*83*
COST-A	*1*	*3*	*+2*	*25*	*28*	*10*	*38*	*20*	*58*
COST-B	*4*	*6*	*+2*	*38*	*44*	*0*	*44*	*5*	*49*

Table 1: This is a conceptual example, but is very close to actual practice by our clients (Johansen 2005). Three goals (performance targets) and two resource targets are having the *real* impacts on them tracked, as steps are delivered. The same Impact Estimation table is also being used to specify the impact estimates for the *future* planned steps. So at each step, the project can learn from the reality of the step's deviation from its estimates. Plans and estimates can then be adjusted and improved from an early stage of the project. Priorities can be changed as new data emerges from previous steps, or as requirements are changed as a result of experience, or for other logical reasons (Gilb 2005a: Figure 10.5)

LACK OF CONSIDERATION OF RESOURCES IN PRIORITY DETERMINATION

Conventional priority determination methods always oversimplify consideration of resources. Sometimes, they even manage to totally eliminate all discussion of resources. Specific concerns with these priority determination methods are as follows:

- they do not look at *multiple* critical resources (for example, project time, effort, project expense, capital expense, maintenance expense, decommissioning expense, space and real time to solve problems) where cost benefit ratios are needed to capture the full nature of the trade off decisions;
- they do not look at those resources on *real-world* scales of measure (for example, they simply assign subjective cost weightings (that is numeric rankings of between 1 and 10), or they reduce all resources to simply being considered in terms of money);
- they often look at costs at too high a level (that is, at project-, not component-level);
- they do not cater for Evo projects. They never look dynamically at 'remaining resources' (the resources left after all the actual expenditure up to this point), nor do they consider any *additional* resources, which might be obtained by infusion of new resources, as a project evolves;
- they frequently deal with costs as a complete afterthought.

To give some examples that illustrate how consideration of resources is sidelined in current thinking on priority determination (We are sure the authors cited below understand about resources, the point is that consideration of resources together with priority is not seen as essential to their methods):

- Thomas Saaty (Saaty 1988, pp.113-120) is at least quite clear about the logical necessity of evaluating benefit to cost ratios. Saaty states: "We must prioritize

Design Idea -> Requirement	**On-line Support**	**On-line Help**	**Picture Handbook**	**On-line Help + Access Index**
Learning Past: 60min. <-> Goal: 10min.				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3min.	±5 min.	±10min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%	±10%
Evidence	Project Ajax, 1996, 7 min.	Other Systems	Guess	Other Systems + Guess
Source	Ajax report, p.6	World Report p.17	John	World Report p.17 + John
Credibility	0.7	0.8	0.2	0.6
Development Cost	120K	25K	10K	26K
Benefit-To-Cost Ratio	110/120 = 0.92	100/25 = 4.0	60/10 = 6.0	104/26 = 4.0
Credibility-adjusted B/C Ratio (to 1 decimal place)	0.92*0.7 = 0.6	4.0*0.8 = 3.2	6.0*0.2 = 1.2	4.0*0.6 = 2.4
Notes: Time Period is two years.	Longer timescale to develop			

Table 2: An Impact Estimation (IE) table showing the impacts of the design ideas on one requirement, 'Learning'. This IE table compares several design ideas, which are not cumulative. It is not appropriate here to discuss all the detail. However, some notes are provided here to give a feel for IE evaluation. The design idea of Picture Handbook is seen as very cost-effective (60/10), but it doesn't, on its own, meet the goals (60%). Maybe there is a complementary design idea that could be found? On-line Support is seen as achieving the goals (110%, though the safety margin (10%) is not extremely comfortable) but, it is not very cost-effective (110/120) compared to On-line Help (110/25) and the development timescales need considering. Overall, there is a need to review the long term strategy. Short term, On-line Help (3.2 credibility adjusted benefit/cost ratio) seems an ideal design idea to start considering further

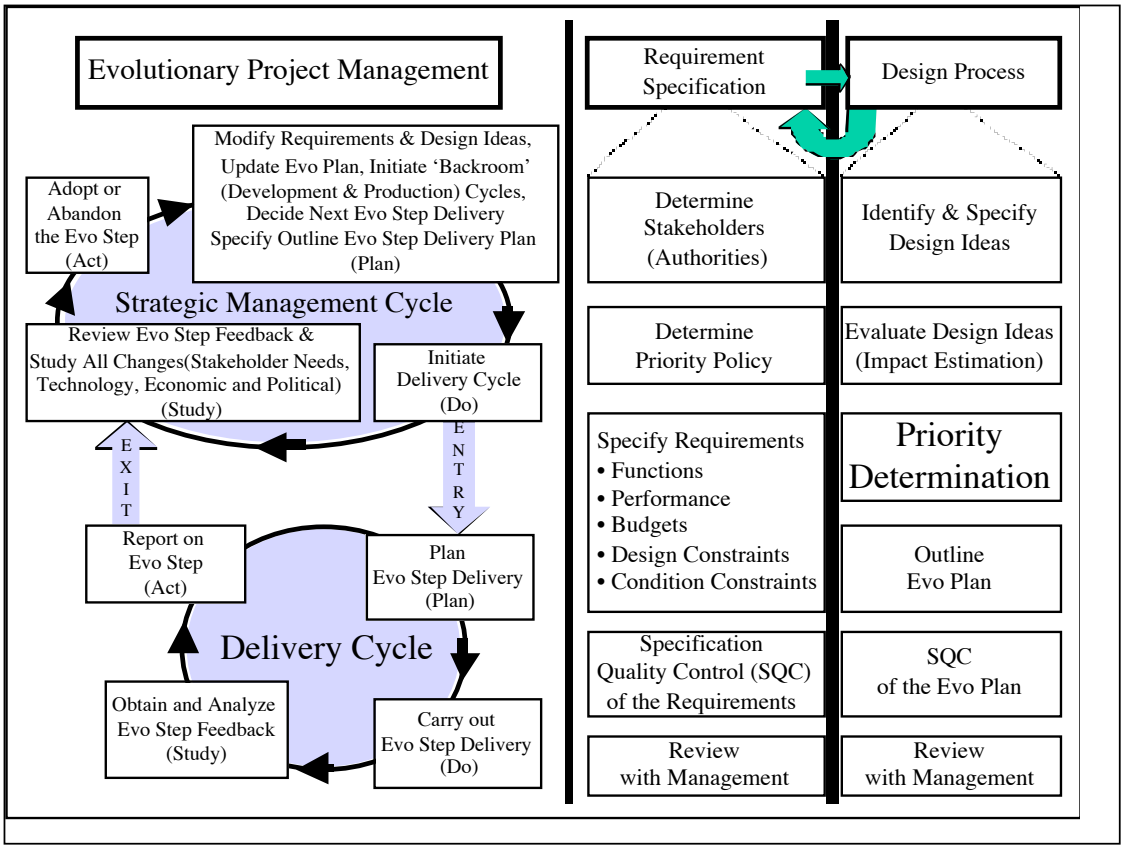

Figure 5. Priority management as an iterative process: The left-hand of the diagram shows the Strategic Management Cycle and the Delivery Cycle within Evolutionary Project Management (The Development Cycle and Production Cycle are not shown). Within the Strategic Management Cycle, the system requirements and design ideas can be updated to reflect feedback, and any changes that have occurred. Within the Delivery Cycle, actual delivery of the Evo step occurs. The right-hand of the diagram shows the main sub-processes of Requirement Specification and the Design Process. Note Impact Estimation and Priority Determination are within the Design Process

alternatives based on their benefits and costs" (op cit. p. 113). But he suffers in our view by discussing costs mainly at the project selection level (not at the systems engineering design level), and by focusing on *monetary* resources (and not on other types of resource, such as time to market, and human effort). Preference is determined without any mention of looking at the costs at all, let alone costs in relation to specific limited resources;

- Akao (1990) states regarding Quality Function Deployment (QFD) methods, that "there is a strong need to ensure at the product planning stage our ability to achieve expected costs or target costs while keeping a balance between quality and cost." (op cit. p. 214). However, there is little further mention of cost. To be fair, he does note that the cost aspect of QFD is "still in the trial stage" (op cit p. 219).

RESOURCE FACTORS FOR PRIORITY

As stated earlier, priority depends on resources. It is only some real or projected limitation on resources, which forces us to choose to satisfy one requirement, or to select one particular design idea (to satisfy prioritized requirements) instead of another.

Benefit to cost ratios as priority decision factors. One generally agreed method for deciding on the priority of a requirement is to look at its 'stakeholder benefit' in relation to the 'resources needed' (the cost[9]) to implement the requirement. In other words, the method is to look at the 'Return On Investment' (ROI) or 'profitability'.

There are also other ways in which consideration of resources can effect the selection of the requirement to be implemented next. They are concerned with practical optimization of the available resources, and are discussed immediately below:

Substituting one resource for another. Take the 'resource' factor one step further. In general any combination of resources might be sufficient to deliver any system requirement. For example, if you run out of time, then you can perhaps use another type of resource, say money or human effort, to recover the situation. The consequence of this is that you cannot make the optimum priority design selection decisions without knowledge of the current degree of availability of all the potential resources.

Selection choice might be determined by lack of resources. In spite of the general observation above, in some cases a specific requirement might be totally dependent for implementation on some resources that are simply not available; and not capable of being substituted by other resources at the moment. In such cases, the next highest priority requirement, that can be implemented using currently available resources, should become the selected choice for implementation. This is a 'next best option' choice. (Of course, there is also the option of waiting, hoping resources become available!)

SUMMARY

There are many possible factors that could be used for determining priority. We firmly believe that each project and organization should use whatever priority-determination rules are right for their culture and their needs.

Here is a summary list of some of the factors to use to determine priority:

- **Constraints:** Constraints are the *first* consideration of decision-making about priorities. There are a variety of constraint types, and they each will have different authority and economic consequences. They will, as noted above, not be absolute. There will be situations where they are uneconomic. They can also be illegal, impractical – and just plain ill considered. A common case of this is when a technical design is required (a design constraint) in the belief that it has some necessary properties (such as reliability or compatibility). This assumption, may turn out to be untrue, or better designs might be found – in which case the 'design constraint' loses its priority;

---

[9] Many people use the terms cost and resource interchangeably. We will use them in a more defined way here. Resources are the various assets or 'fuels' needed by systems (For example, time, effort, money and space). In real systems resources are finite or limited. Cost is the term for the expenditure (past, present or future) of a resource needed by a process or component of the system.

- **Dependencies:** An item that something else depends on for implementation will have the same priority as that 'something else' has, even though it might have a lower priority in isolation;
- **Stakeholder Benefit:** Specifications (requirements and design ideas) can be prioritized depending on the benefit they produce. This should be determined, less by subjective '1 to 10' weighting scales, and more by the *real* performance levels delivered, and the consequential real world benefit (real stakeholder value) of that product performance level;
- **Stakeholder Benefit in relation to Cost:** In other words, the return on investment (ROI), profit and net value. Again, the cost should be determined realistically (for example, calculate total life cycle cost including maintenance, adaptability and decommissioning costs);
- **Benefit to Cost in relation to Risk:** The most pessimistic (worst imaginable case) values for stakeholder benefit and cost can be used to evaluate alternative choices.

Notice that even with a constant set of information about the priority specification options, it is our choice of priority *determination* factors that determine the option that will be chosen.

Priority determination is a richer area for systems engineering than this paper indicates. This paper does not cover all the mechanisms available in Planguage to deal with priority information. Instead, it focuses on showing that there are more-realistic and capable alternatives to traditional weighted priority model methods. A study of Planguage (Gilb 2005a) would show a large number of other devices for understanding priorities. For example, the devices for dealing with risk and uncertainty are one major angle of analysis that we have barely mentioned here. Should the degree of risk determine priority? Of course!

This paper can be summarized in terms of ten principles of priority determination:

1. Priority is what has first claim on limited resources.
2. Stakeholder requirements are the major basis for determining priorities.
3. Benefit to Cost Ratios for design impacts help to realistically determine the current priority design (or 'strategy').
4. Priority decisions should be based on a detailed, rich, realistic set of information about the options
5. Constraints are your first priority. Stay within constraints before optimizing towards targets.
6. Targets are your second priority. But when all targets are reached – stop using resources.
7. Priority needs to be determined periodically, not simply at the beginning of a project.
8. The 'most threatening' gap to reaching a requirement level has highest priority, other things being equal. By 'most threatening,' is meant the one threatening the biggest risk in terms of consequences to the organization
9. More-objective requirement statements (that is, fact based, citing the supporting evidence, measurement based from past relevant experience) are better than more subjective statements (such as opinions without facts or measures).
10. Priority should be determined based on risks, benefit and cost.

To work well, priority determination is a far more complex process, than applying mere subjective weightings. Only if we take a more rigorous approach and document our priority processes and priority decisions are we going to gather experience data, and to identify the best practices.

It is our view that we must inject this more practical and dynamic paradigm, regarding priority into requirements, design and project management. We must recognize that much of the reason for our current unimaginative methods (the subjective weighting) is that we are stuck in the big bang or waterfall method paradigm. Evolutionary methods, which typically allow us to collect far more correct and realistic information about priorities than the waterfall method static models, should be assumed to be the norm for almost all systems engineering projects and practices (MacCormack 2001, MacCormack 2003, Gilb 2005).

There are 'Three Truths' about priority[10]:

1. Priority = Claim on scarce resources;
2. Priorities change continually;
3. Prioritization is aided by rich specification.

REFERENCES

Akao, Yoji (Editor), *Quality Function Deployment: Integrating Customer Requirements into Product Design*, Productivity Press, Cambridge Mass., USA, 1990.

Daniels, J., Werner, P. W., & Bahill, A. T., "Quantitative methods for Tradeoff Analysis." *Systems Engineering*, Vol. 4, Number 3, pp 190-211, 2001.

DoD, *Joint Logistics Commanders Guidance for use of Evolutionary Acquisition Strategy to Acquire Weapon Systems*, The Defense Systems Management College Press, Fort Belvoir, VA 22060-5565, 1998 and later versions. For copies of this guidebook, contact the Defense Systems Management College Press at (703) 805-3364 or visit the DSMC Home Page `http://www.dau.mil/`.

Gilb, Tom, *Competitive Engineering, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, Elsevier Butterworth-Heinemann, 2005. ISBN 0750665076.

Gilb, Tom, `http://www.Gilb.com`, 2005. Various free papers and manuscripts including Priority Management in draft.

Gilb, Tom, "Impact Estimation Tables: Understanding Complex Technology Quantitatively." *Crosstalk*, December 1998. This article can be found in its entirety in PDF format on the Software Technology Support Center Web site at `http://www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/12/gilb.asp/.`

Gilb, Tom, *Principles of Software Engineering Management*. Addison Wesley, 1988.

Gilb, Tom, *Software Metrics*, Studentlitteratur AB Sweden, 1976 and, Winthrop (USA) 1977. Out of Print.

Johansen, Trond and Gilb, Tom, "From Waterfall to Evolutionary Development (Evo) or How We Rapidly Created Faster, More User-Friendly, and More Productive Software Products for a Competitive Multi-national Market." July 2005. *Proceedings of INCOSE Conference*,

[10] As used by Erik Simmons, Intel in teaching our methods.

Rochester NY USA, July 2005.
Keeney, Ralph L., *Value-focused Thinking: A Path to Creative Decisionmaking*. Harvard University Press, Cambridge Mass/London, 1992 ISBN 0674931971. http://www.fuqua.duke.edu/faculty/alpha/keeney.htm/.
Keeney, Ralph L. and Raiffa, Howard, *Decisions with Multiple Objectives : Preferences and Value Tradeoffs*. Wiley, 1976. (Also more recently published by Cambridge University Press, 1993. ISBN: 0521438837. Paperback 589 pages.)
Kreps, David M., *Notes on the Theory of Choice*, Westview Press, 1988.
Larman, Craig, and Basili, Victor. "Iterative and Incremental Development: A Brief History." *IEEE Computer*. June 2003. Pages 2-11. See www.craiglarman.com for a copy of this paper.
MacCormack, "Product-Development Practices That Work: How Internet Companies Build Software", *MIT Sloan Management Review*, pp 75-84, Winter 2001.
MacCormack, Alan (Harvard University), Kemerer, Chris F. (University of Pittsburgh), Cusumano, Michael (MIT), and Crandall, Bill (Hewlett-Packard), "Exploring Trade-offs between Productivity and Quality in the Selection of Software Development Practices." *IEEE Software*, September/October 2003, Volume 20, Number 5, Pages 78-85.
Saaty, Thomas L., "Multicriteria Decision Making: The Analytical Hierarchy Process", *Decision Support Software*, McLean Va, 1988.
Saaty, T. L., *The Analytical Hierarchy Process*, McGraw Hill, New York, 1980.

BIOGRAPHIES

Tom Gilb

Tom Gilb is the author of 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage (Summer 2005), 'Principles of Software Engineering Management' (1988) and 'Software Inspection' (1993). His book "Software Metrics" (1976) coined the term and, was used as the basis for the Software Engineering Institute Capability Maturity Model Level Four (SEI CMM Level 4). His most recent interests are development of true software engineering and systems engineering methods.

Tom Gilb was born in Pasadena CA in 1940. He moved to England in 1956, and then two years later he joined IBM in Norway. Since 1960, he has been an independent consultant and author. He is a member of INCOSE.

Author Contact: Tom@Gilb.com

Mark W. Maier

Dr. Mark W. Maier received the BS and MS degrees from the California Institute of Technology and the Engineer and PhD degrees in Electrical Engineering from the University of Southern California. While at USC, he held a Hughes Aircraft Company Doctoral Fellowship, where he was also employed as a section head. Currently he is a Distinguished Engineer at The Aerospace Corporation where he founded the systems architecting training program and consults on the application of architecting methods to government and commercial clients. Prior to coming to The Aerospace Corporation (Reconnaissance Systems Division, Engineering and Technology Group), he was an Associate Professor of Electrical and Computer Engineering at the University of Alabama at Huntsville. Dr. Maier's research interests are in systems architecting, engineering

computer based systems, space systems, and radar systems. He is co-author, with Dr. Eberhardt Rechtin, of The Art of Systems Architecting, Second Edition, published by CRC Press. Dr. Maier is chair of the INCOSE Systems Architecture Working Group.

Author Contact: Mark.w.maier@aero.org

ACKNOWLEDGEMENTS

To Richard Zultner for challenging me and correcting Tom every single step of the way, in early drafts. We are pretty sure he does not agree with all of this, but he did force Tom to another level of clarity by his dogged criticism and deep knowledge of the Analytical Hierarchy Process (AHP) and Quality Function Deployment (QFD) methods.

To Erik Simmons of Intel, Portland for giving deep, useful feedback on the paper.

This paper was edited by Lindsey Brodie, Middlesex University, who made substantial original contribution in addition to editing. This includes originating the diagrams.
lindseybrodie@btopenworld.com