Choice and Priority Using Planguage:
A wide variety of specification devices and analytical tools.

Iver Holtersvei 2, NO-1410 Kolbotn, Norway, Tom@Gilb.com, www.Gilb.com, +47 66801697

Abstract:

- Planguage (the Planning language defined in Competitive Engineering [CE]) has a variety of methods and tools to help us identify and choose candidates for solving a defined problem.
- There is no single method or tactic for making a 'best' choice.
- There is no concept of finding a 'perfect choice' either.
- By rational application of a suitable set of methods, within the time and resources available, a candidate solution can be found, which is probably one of the most satisfactory available. It is probably satisfactory enough. And, we will be able to say something about the solution's risks, uncertainties, issues, dependencies, and side effects.
- We can substantially improve the probability of successful choices. Only in some unreal world, if we had infinite time, infinite knowledge, and static circumstances, could we hope to make a 'perfect' choice.
- In the competitive world, the necessity is making very good choices rapidly, under conditions of uncertainty, and risk of change.

Available Tools

- The available tools in Planguage are:
  - Requirement Specification RS (the problem formulation)
    - A disciplined, thorough, and *quantified* way to specify the problem to be solved. Planguage helps give an accurate description of the nature of the best

choices. Many so-called requirements languages have little or no facility to specify quantified qualities and costs aspects of a problem.

```
Adaptability:
Type: Quality Requirement.
Scale: Time in hours needed to re-configure the defined [Base Configuration] to any
other defined [Target Configuration] using defined [Methods] and defined
[Reconfiguration
Staff ].
Expert Reconfiguration: Defined As:
{Base Configuration1/4Novice Setup,
Target Configuration1/4Expert Setup,
Methods1/4Selection of Library Reconfiguration Process,
Reconfiguration Staff1/4Qualified Expert}.
========================== Benchmarks ==========================
Past [Expert Reconfiguration, Version 0.3, Asian Market]: < 1 hour.
======================= Performance Targets =======================
Authority [Goals]: Federal Drug Administration.
Goal [Expert Reconfiguration, Deadline1/4Version 1.0]: < 0.5 hours.
Goal [Expert Reconfiguration, Deadline1/4Version 2.0]: < 0.1 hours.
========================== Constraints ==========================
Fail [All USA Products]: < 0.7 hours.
Fail [Expert Reconfiguration, Deadline1/4Version 2.0]: < 0.5 hours.
Survival [Expert Reconfiguration, European Market]: < 1 Working Day.
```

- *Example 1: a typical Planguage requirement specification with multiple quantifications of benchmarks, targets and constraints in a single requirement specification. Source CE page 55. The numeric clarity given by this type of specification is a critical cornerstone of making rational choices and prioritizations.*
- These requirements tools include multiple *simultaneously required* performance (including qualities) *requirements* (how good the system is), function *requirements* (what the system does), resource *constraints* (budgets for time, people, money, space, and other limited developments, and operational resources), and all other *constraints* (like legality, conformance to culture or standards), on the solution of the problem.

- At a management problem level (organization, process, marketing), somewhat differently from the technical level, these problem formulations might be called 'objectives' and 'constraints'. They would amount to the same basic concept: a definition of a *future desired state* of a system. The problem is, how to do reach those states?

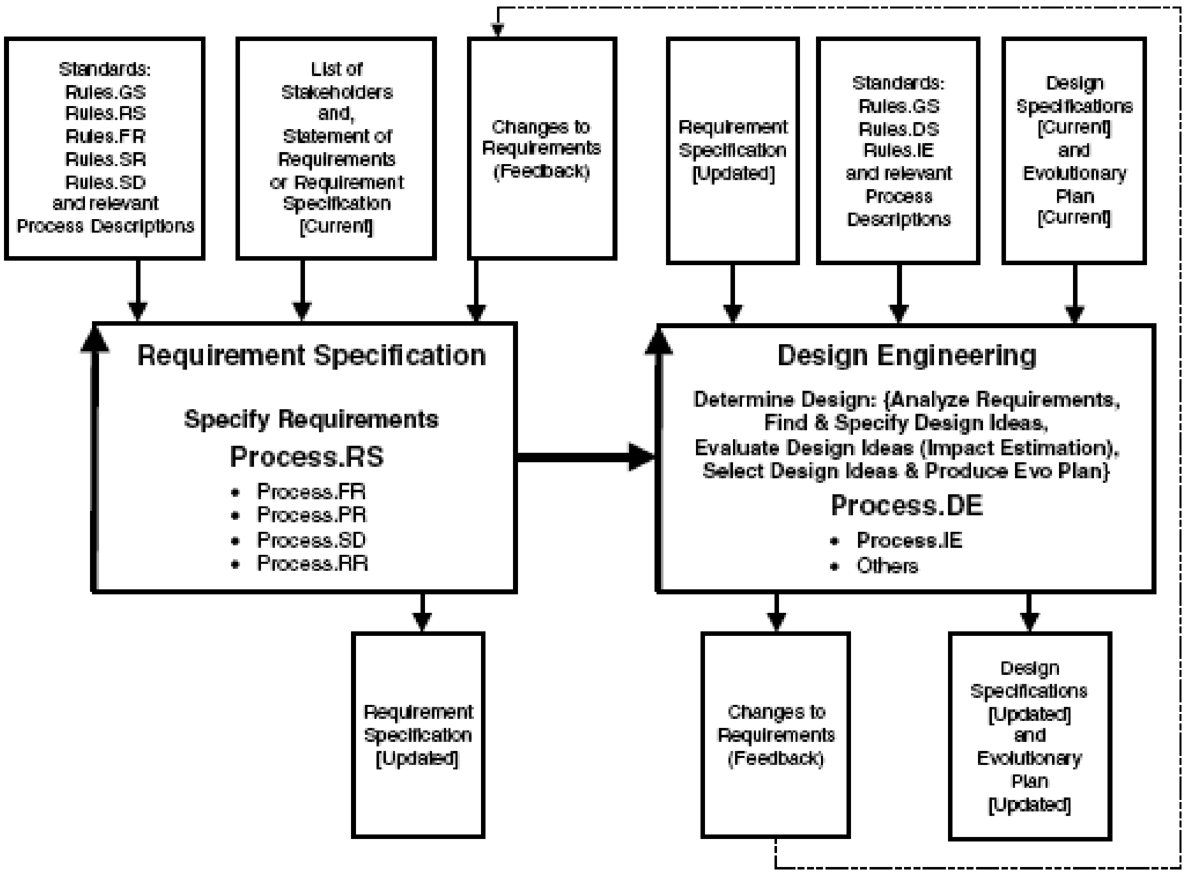
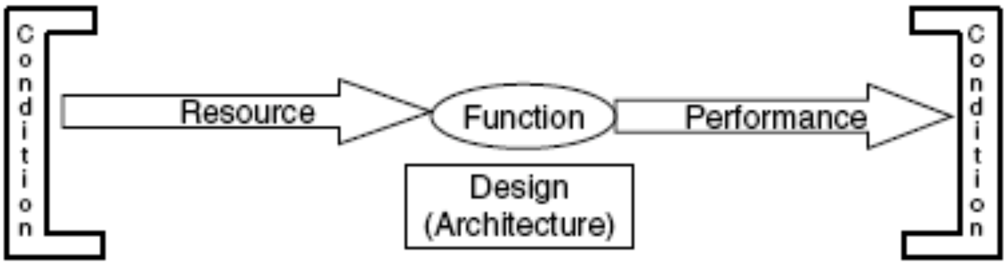

Standards: Rules.GS Rules.RS Rules.FR Rules.SR Rules.SD and relevant Process Descriptions
List of Stakeholders and, Statement of Requirements or Requirement Specification [Current]
Changes to Requirements (Feedback)
Requirement Specification [Updated]
Standards: Rules.GS Rules.DS Rules.IE and relevant Process Descriptions
Design Specifications [Current] and Evolutionary Plan [Current]
Requirement Specification
Specify Requirements
Process.RS
Process.FR
Process.PR
Process.SD
Process.RR
Design Engineering
Determine Design: {Analyze Requirements, Find & Specify Design Ideas, Evaluate Design Ideas (Impact Estimation), Select Design Ideas & Produce Evo Plan}
Process.DE
Process.IE
Others
Requirement Specification [Updated]
Changes to Requirements (Feedback)
Design Specifications [Updated] and Evolutionary Plan [Updated]

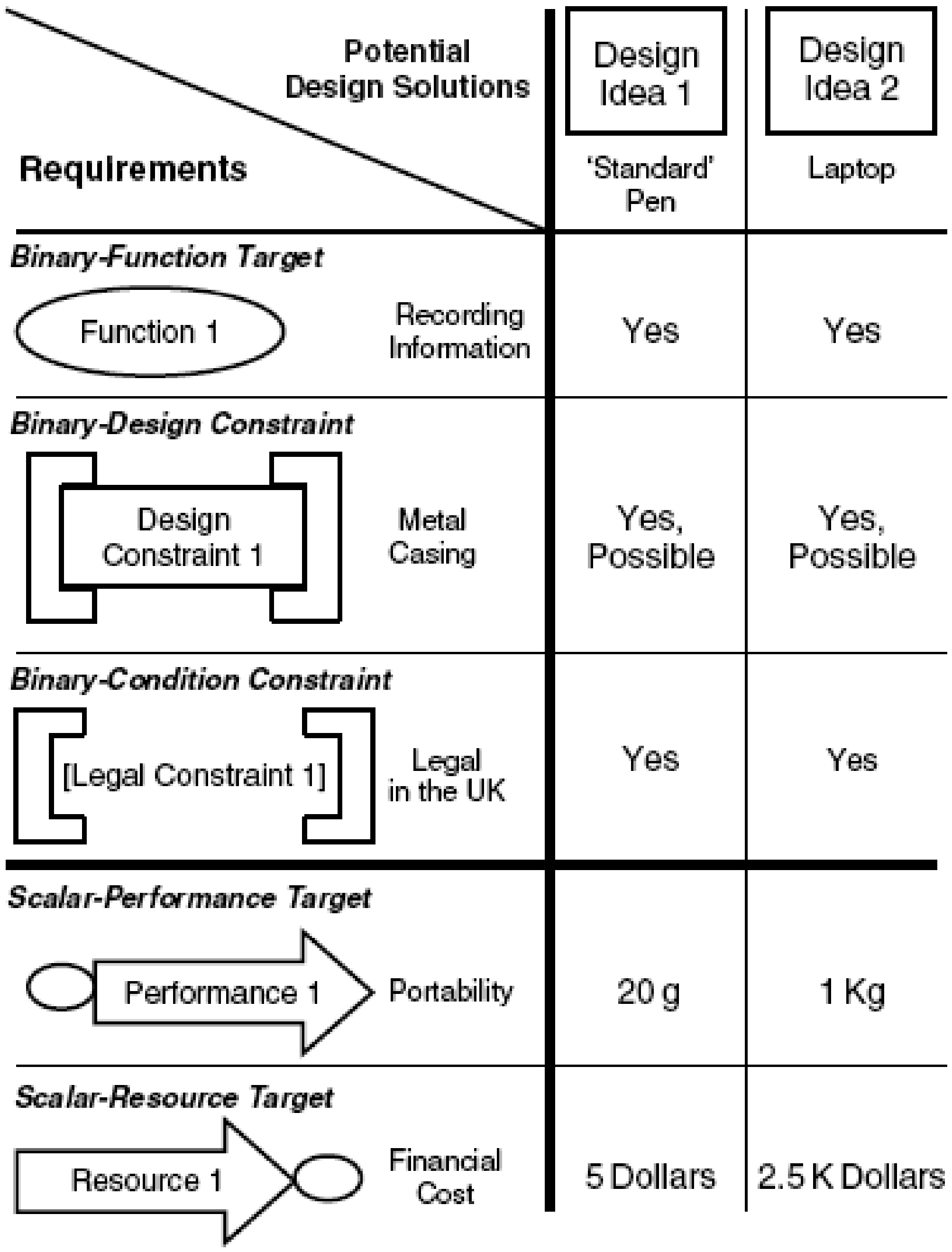
Figure 1: the Planguage relation between requirements specification process and the design engineering process. Source: CE, Fig. 1.5, pg.18, see note 1 for decoding of acronyms. The defined processes and defined Rules noted above are some of the specific tools in Planguage, defined in detail in [CE].

- Design Specification - DS (solutions)
  - 'Design Specifications', in Planguage, are a disciplined, thorough, and *quantified* (impact of solution on requirement) way, to specify potential and final solutions, for evaluation (choice and priority) against the problem statements in the requirements.

- 

- *Figure 2: Basic elements of choice and priority. The requirements elements and the design solutions. Source CE Fig 2.1, pg.47.*
- Ultimately the design solutions must include the set of all solutions that will be applied to solve a defined set of requirements. One solution alone from that set of many solutions is almost impossible to pass final judgement one. We can only pass judgement on a complete set, a total solution.

o Impact Estimation (IE)
- Impact Estimation is a systematic discipline for evaluating the numeric effects ('impacts') of solutions on the requirements. IE looks at both impacts on performance requirements (how good the solution must be) and limited resources (what we can spend to pay for the solution).



Requirements / Potential Design Solutions		Design Idea 1 'Standard' Pen	Design Idea 2 Laptop
Binary-Function Target Function 1	Recording Information	Yes	Yes
Binary-Design Constraint Design Constraint 1	Metal Casing	Yes, Possible	Yes, Possible
Binary-Condition Constraint [Legal Constraint 1]	Legal in the UK	Yes	Yes
Scalar-Performance Target Performance 1	Portability	20 g	1 Kg
Scalar-Resource Target Resource 1	Financial Cost	5 Dollars	2.5 K Dollars

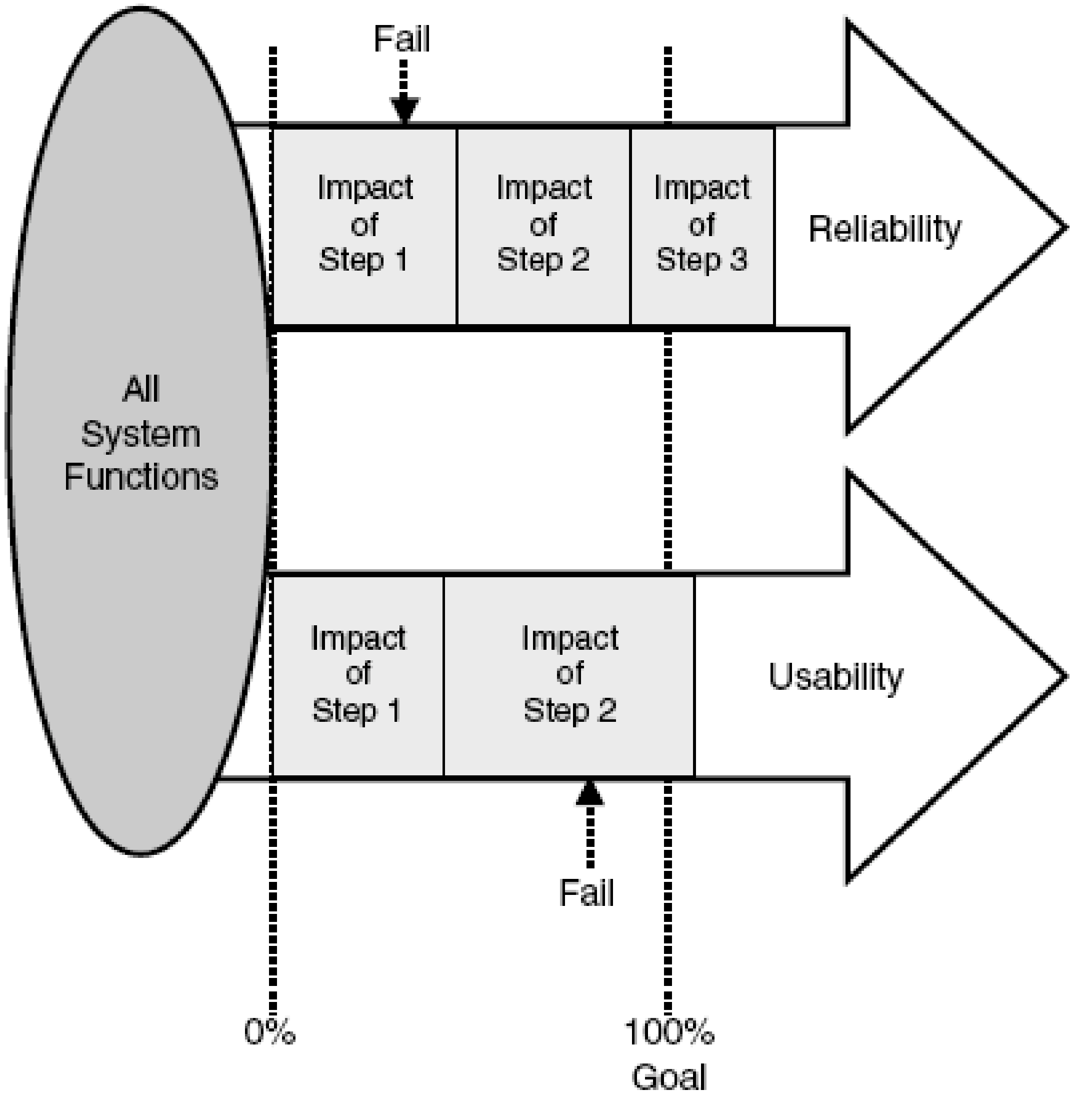
▪

▪ Figure 3: A conceptual Impact Estimations table showing how design alternatives rate in relation to various types of requirements. Source CE, page 58. The large Planguage graphic symbols are purely for teaching purposes here.

▪

▪ It is a common misunderstanding that IE is the major or only selection mechanism. This is not true. It is the interplay of earlier (choice and elimination mechanisms) and later (evolutionary feedback in practice) mechanisms with IE that help us to finalize our decisions about solutions.

- Evolutionary Project Management. (Evo).
    - Evolutionary project management allows us to systematically test our preliminary choices in a real environment. We can test them very early in the project life, so that disappointing choices can be dropped at little penalty, and replaced with better choices in fact.
    - Evo gives us an environment, not available in earlier models and methods, to see the effect of combining a particular solution with both a real systems environment, and other recent partial solutions to the overall problem.



▪

        - *Figure 4: Incremental deliveries of quality requirements have the effect of changing the next step priority, depending on the actual cumulative impact in relation to constraint requirements (Fail level) and targeted requirements (Goal). When a Goal level is reached then the priority for that attribute is gone.*
    - **Principles**
    - There are at least 100 principles formally stated in Competitive Engineering (and about 124 in Gilb88, and many more in other Gilb papers and slide presentations, for example on Risk, many in papers at INCOSE 2003-5, and earlier) they are the heuristics that can be used to guide us in our decision-making process. A key subset of these principles for making choices will be explored here.

5. The Principle of 'Deadline or die'

- *There is no point in demanding a performance requirement, if you would always give priority to something else, for example, a deadline*.

Source: CE page 126 performance Principles.

    - The principles are intended used in connection with the other tools mentioned above.
    - The principles stated *in this paper* are original to this paper, and new formulations, but they have their roots in Planguage, and previously stated principles. They are somewhat more-focussed on our choice-making problem than the more general principles in CE.
- Of course anyone is able and welcome to use any alternative, supplementary, or additional tools as they please. However this paper is specifically written to clarify the toolset within Planguage itself, and will limit itself to the Planguage set of methods as described in Competitive Engineering.

Principles of Choice: principles for the decision-making framework

- The Top Ten Decision Framework Principles

1. **More is Good**: The more relevant information you specify about the problem, the more likely you are to choose a good solution.
2. **Aspects is Vital**: The more you know about all aspects of a solution, the less likely you are to choose it in error.
3. **Many to Many**: The more attributes of a solution that you match against the problem requirement attributes, the better your decisions will be.
4. **Future is Different**: Previous experiences with a solution are not a certain guide to its attributes in your future.
5. **Priority Decides**: Information about priority of requirements makes it easier to select suitable solutions.
6. **Uncertainty is Certain**: Information about uncertainty in requirements and solution attributes allows you to make better choices with respect to the uncertainty you can tolerate.
7. **Totality Beats Subsets:** It is the total set of solutions that must be chosen to solve the total problem (the total set of requirements). Any small change in a requirement, or a solution, can invalidate the *entire* set of solutions chosen up to a certain point.
8. **Feedback**: The process of choosing solutions is necessarily iterative, and iterative decision-making processes are more likely to provide better solutions in less decision-making time, than straight line processes.
9. **Stakeholders**: Failure to identify all critical and profitable stakeholders for the solution results, can invalidate your *entire* decision-making process.
10. **Trust but Verify**: all specifications need ultimately to be put to tests of credibility. Ask for sources, evidence, uncertainty, and how we are going to verify in time that specifications are true.

Heuristics: heuristics for decision-making, for making specific choices

- The Top Ten Choice-Making Principles.

1. **Best Choice**: In general the best choice will satisfy all the performance Goal levels, at minimum resources, within stated constraints.
2. **Constraint Priority**: Constraints have different priorities, so instead of dropping an otherwise promising solution, because it violates a stated constraint, consider removing a lower-priority constraint itself.

3. **Initial Choice of Candidate Solutions**: scan solution candidates to find the few that give most impact to your highest priority performance requirement.
4. **Tried and True**: when decision-making time is limited we should prefer solutions that are well-understood in terms of performance and costs. This is a conventional engineering paradigm.
5. **Sufficient beats Maximum:** solutions should not be chosen for having a maximum (of other choices) performance impact. They should be chosen for having *sufficient* impact to bring us to the specified goal levels of performance on time, with respect to all other selected solution's effect on those performance requirements. They should fill the remaining gap to the goal. They should not assume they are they only solution. Once we are *at* our goal levels, there is no stakeholder value in being better. So we need to avoid overdoing the solution power – except when that incremental power is free of costs.
6. **Multiple Impacts are Best**: when you have a set of promising candidate solutions picked out by other heuristics, an Impact Estimation Table will give you advice on the generally best solution in terms of impacts on all performance requirements, and on all resource budgets. Only 'acceptable' and 'promising' solutions will make it to the IE table evaluation. This is usually based on one or few dimensions of evaluation. The IE Table allows you to do a broader evaluation in more critical dimensions, including all positive and negative side effects.
7. **Trial and Error**: one or more of the best-looking candidate solutions from an Impact Estimation process evaluation, can be scheduled for early practical trials, by integrating them evolutionarily into a developing, or old, system. The expected performance and cost impacts can be measured, and compared to expected IE Table values. You can thus in practice make a final judgement on the solution, and get *even more* reliable data on the solution, when it is retained, for scaling up.
8. **Simplified Impact Estimation**: there are many possibilities for simplifying the Impact Estimation process, basically using lower credibility impact data (guesses, rough estimates, outside sources, round number estimates, +++ type estimates, 0 to 9 estimates, subjective judgement). This simplified process will reduce the cost of looking at a large number (5-50) alternatives quickly, at some risk of losing some better solutions. But it will allow you to quickly get down

to few (1 to 5) solution alterative where performance and cost data can be studied at higher levels of credibility – using more time per alternative.

9. **Where to Look? Performance!**: look for options using *the highest priority performance goal* as a guide. Look at any source such as expert designers, web info, engineering and management literature, and tried and true solutions in the business.
10. **When to Stop? Satisfaction:** you can stop looking for other solutions only when you have in fact measurably delivered real systems with the goal level of performance attributes reached, within the resource and other constraints. Requirements Satisfied. Of course by then, someone will raise the stakes – so the search for better solutions, in a competitive world, is eternal.

Process: a systematic path for decision-making.

- Stakeholder identification
- Stakeholder values analysis
- Constraints analysis
- Draft Requirements Specification: Pretty Good levels
- Requirements Review
- Design Solution Brainstorm
- Weeding Out Solutions based on too costly or impressive performance match
- Impact Estimation simplified (if many solution alternatives)
- Impact Estimation in detail (for a chosen few most-promising solutions)
- Select a solution for early Evoutionary implementation and analysis in practice (old or new system)
- Adjust detailed Impact Estimation values based on experience
- Ask if Goals reached and constraints not violated. No. go up and continue the process, step 6 or below.
- YES. You are done.

The Theory of Priority in short

- Priority: defined as: the thing we choose amongst alternatives.
- Principles of Priority: (from Gilb and Maier)
    1. Priority is what has first claim on limited resources.

2. Priority needs to be determined periodically, not simply at the beginning of a project.
3. Prioritization is aided by rich specification. More-objective requirement statements (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).
4. Stakeholder requirements (or 'objectives') are the major basis for determining priorities.
5. Benefit to *Cost* Ratios for design impacts help to realistically determine the current priority design (or 'strategy').
6. Priority decisions should be based on a detailed, rich, realistic set of information about the options
7. Constraints are your first priority. Stay within constraints before optimizing towards targets.
8. Targets are your second priority. But when all targets are reached – stop using resources.
9. 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
10. Priority should be determined based on risks, benefit and cost.

References

- CE: Gilb, Tom, Competitive Engineering, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, ISBN 0750665076, 2005, Publisher: Elsevier Butterworth-Heinemann.
-
- Gilb and Maier 05:
  - Tom Gilb and Mark W. Maier
  - Managing Priorities: Key to Systematic Decision Making. Presented Orally and in Proceedings (CD) of INCOSE Annual Conference, Rochester, NY, USA 2005
- Gilb88:

- Tom Gilb, Principles of Software Engineering Management, Addison Wesley, UK, 1988. 20th Printing in 2005.

- Notes:
- 1. (reference Figure 1)
- Notes:
- Iteration of the processes has been allowed for by including existing specifications as potential inputs. Qualifying square brackets have been used around descriptive words, which are added to assist understanding. The aim is to show how the rules and process descriptions discussed in this book fit together. This diagram shows procedure steps P1 and P2 of the Generic Project process (Process.GP). These same processes are used during Manage Evolutionary Project (Process.GP.P3) – that is during Evolutionary Project Management – in order to update the requirements, the ideas and the Evo plan (see Figure 1.6 in CE). A standing rectangle is the Planguage graphic symbol for a specification or document, and a rectangle with an arrow up on the left side is the Planguage graphic for a 'process'.
- The abbreviations used in this figure (and in the rest of the CE book) are as follows:
- GP Generic Project RR Resource Requirements
- GS Generic Specification DS Design Specification
- RS Requirement Specification DE Design Engineering
- FR Function Requirements IE Impact Estimation
- SR Scalar Requirements EVO Evolutionary Project Management
- PR Performance Requirements SM Strategic Management
- SD Scale Definition DC Delivery Cycle

Author Bio

Tom has been an independent consultant, teacher and author, since 1960. He mainly works with multinational clients; helping improve their organizations, and their systems engineering methods.
Tom's latest book is 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (Summer 2005).
Other books are 'Software Inspection' (with Dorothy Graham, 1993), and 'Principles of Software Engineering Management' (1988). His 'Software

Metrics' book (1976, OoP) has been cited as the initial foundation of what is now CMMI Level 4.
Tom's key interests include business metrics, evolutionary delivery, and further development of his planning language, 'Planguage'. He is a member of INCOSE and is an active member of the Norwegian chapter NORSEC. He participates in the INCOSE Requirements Working Group, and the Risk Management Group.

Email: Tom@Gilb.com
URL: http://www.Gilb.com

Version Nov 9 2005