Chapter
7

Design Ideas and Design Engineering

How to Solve the 'Requirements Problem'

Glossary Concepts
- Design Idea
- Design Specification
- Design Engineering

7.1 Introduction

To Design and to Engineer

The basic design process is finding 'means' for 'ends': it is finding designs that match the requirements.

What is the difference between design and design engineering? They are both essentially the same generic, and basic, process of 'finding satisfactory designs.' However, engineering disciplines are characterized, in my opinion, by the following distinctive traits:

- *quantification* of variable ideas (not 'high', but '42')
- concern for *all* necessary factors (all stakeholders, all requirements and all known design options)
- concern for more than mere 'satisfaction'; concern for *competitive* optimization – 'being the best', rather than just 'getting along'
- rational and systematic argument (for example, the use of Impact Estimation tables to discuss or present design quantitatively with respect to facts, not 'less formal' design or 'emotional' design.

> Design asks, ''Is this a good design?''
> Design Engineering asks, ''What are the totality of performance and cost attributes expected from *this design* in relation to the multiple, quantified, performance and cost *requirements*? What are the risks, priorities, uncertainties, issues, relationships, dependencies and long-term lifecycle considerations, that we should responsibly consider about this design?''

Requirement Specification, Design Engineering and Evo are all Iterative Processes

Design ideas emerge, and are refined, throughout the lifetime of a system. Iteration is necessary in order to improve both the design ideas and the related requirements. Requirements and design ideas *cannot* be determined well in one single pass. Feedback from initial *design engineering* processes is necessary to get a *realistic* idea of which *design ideas are possible*, to determine how much design ideas *might cost* and to identify which tradeoffs amongst performance levels *might have to* be made. Until *the design and the requirements* are adjusted to this 'balanced level' with regard to reality, it is not possible to 'finalize' a competitive *design* for implementation.

In addition, after implementation starts, as a result of the measurable *feedback* obtained from the delivery of each of the Evo steps, even

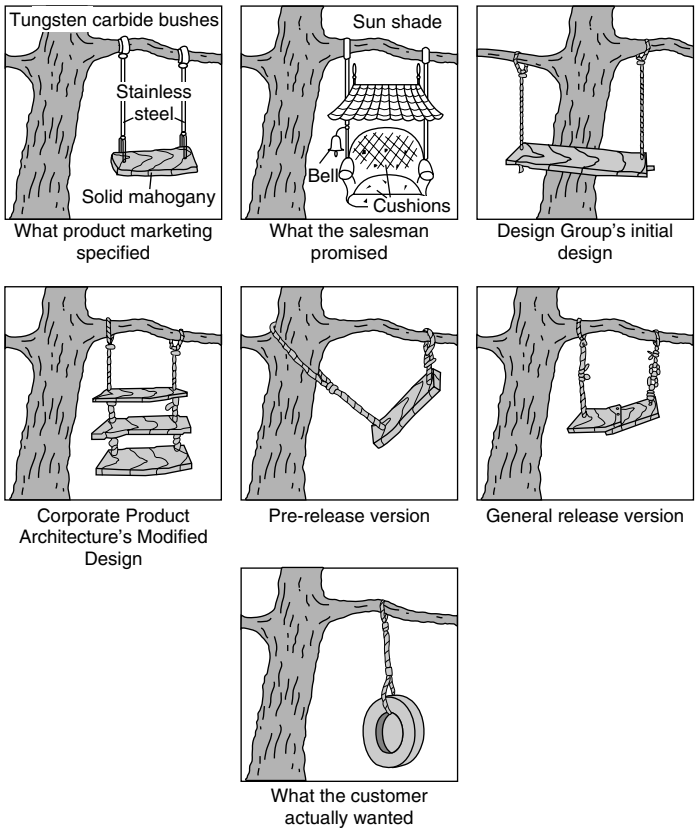

Figure 7.1
The swing solutions. Source: Anon.

further refinement has to be considered for the design ideas, the requirements and the *implementation plans*.

Requirement Specification, Design Engineering and Evo are intimately linked, and some iteration linking them is necessary to get the best competitive results.

Requirements Dictate and Constrain the Design, but *Detailed* Requirement Specification can Wait

What stakeholders perceive as 'value' drives us to state 'what stakeholders want' and 'how much stakeholders might be willing to pay for such change': in other words, to state the *requirements*. Requirements, which reflect values, give us a sound basis for evaluating a design idea: a basis for deciding if we might get what we will find of 'value' from a potential design idea.

> 'Interesting' results are our 'values.'
>
> *Keeney (1992)*

We have to have at least a preliminary set of requirements, before we are ready to '*design*.'[1] These requirements could, even for a large project, be as simple as a statement of the handful of most critical requirements. (After all, these critical requirements are in fact usually driving the investment and the project!) The more detailed requirements can be derived gradually, as needed, during the Evolutionary Project Management (Evo) process. There is no need to try to get all the detail immediately. In fact, there is some virtue in letting the detail emerge as a function of experience and of interaction with key stakeholders.

Note: The detail is, however, ultimately important, and must be eventually specified, so we can fully understand the meaning, intent, risks, assumptions and dependencies of all the requirements. For example, we need to understand which requirements are targeted only at specific system components.

Any Design Idea[2] can be Considered

Any design idea that potentially contributes to the solution of the requirements, can be *suggested*. It is a question of how much a design idea contributes towards meeting the requirements, and at what costs, which determines whether a design is finally selected and implemented. It is then the design idea's *real* performance, on delivery, that will determine whether it survives, or must be replaced by another design idea.

EXAMPLE Some Design Ideas:

- using process improvement teams
- allowing the project team an extra day's time off if a deadline is successfully met (motivation)
- buying an extra server (buy hardware)
- giving discounts to customers who field trial new products (monetary motivation)
- buying a standard component (buy hardware with known characteristics)
- contracting for a special tailored component (subcontracting and tailoring)
- building our own software component (development in-house)
- improving testing process (improving a specific development process).

This example shows a wide variation of types of design.

---

[1] In this book generally, I use the term 'to design,' but with regard Planguage processes, I actually mean 'to design *engineer*,' that is, to use rational and quantitative approaches.
[2] The term 'design idea' is used in this chapter. Solution, idea, strategy, design, means, idea and design solution are all synonyms amongst many other synonyms for 'design idea.'

Design Ideas can be Identified during Requirement Specification

Even while you are initially specifying *requirements*, you should, if you feel that design ideas are flowing into your mind or the minds of colleagues, develop two separate lists of design ideas: *potential design ideas* and *design constraints*. The potential design ideas can then be kept aside for serious consideration in the *design* phase.

Potential Design Ideas

These are either design ideas that were first assumed to be *requirements*, but then were recognized as *really* being *optional* designs, or they are simply design ideas that surfaced during requirement specification. Sometimes such design ideas are deliberately 'brainstormed' (for example, if experts in a specific area are available only during the initial requirements' gathering, then capturing their design ideas might be opportune). Here is an example of a way of keeping track of any potential design ideas; there is *no commitment* to implementing them at *this* stage.

EXAMPLE

Availability:
Type: Quality Requirement.
Scale: % Uptime.
Goal [USA, Version 1.0]: 99.90% <- Marketing Plan [April 20, This Year].
Design A [Availability = 99.90%]: Design Idea: Reuse of <high MTBF> Components <- Ed's suggestion.
Stretch [Worldwide, Version 3.0 and on]: 99.998% <- CEO Vision, "World Class."
Design B [Availability = 99.998%]: Design Idea: Triple Redundant Distinct Software <- Mike.
The two design idea specifications are local to the two different target specifications. They are not design constraints. They are clearly suggestions that need to be evaluated like any other suggestion.

Allowing systems engineers to note design ideas at an early stage is useful in several ways:

- it keeps track of potentially valuable design ideas which otherwise might get forgotten
- it helps make the distinction between the requirements and the design technology clearer ('clear ends–means separation')
- it lends credibility to the proposed goal levels (there exists some credible technology for the goal level suggested)

- it avoids the 'frustration' that some systems engineers feel when they are not allowed to be specific about the technology they have in mind
- it allows us to send a message that we have noted a systems engineer's suggestion or 'pet idea' and credited them with it – without yet officially approving it.

Some of the early design ideas may be *politically* wise to consider, due to the fact that influential stakeholders have suggested them. There is no risk of any unfairness in considering these design ideas, because they will have to compete with the later design ideas. All design ideas must win their place for implementation by being the best, in terms of numeric satisfaction of the requirements.

Design Constraints

These are design ideas within the requirement specification, which have to be implemented at some stage. They can either specify or veto the use of specific designs. Usually, specific qualifying conditions apply.

EXAMPLE

Project Interface [Product Line = New Generation, European Market]:
Type: Design Constraint.
Description: The full Project Interface shall be implemented using the most <current version> available. It shall be updated whenever <newer versions> are available.
Rationale: Project Consortium Agreements.

This design constraint (a requirement) applies only to the Product Line of New Generation within the European Market.

The Need for Alternative Design Ideas

Choosing the Best from the Alternatives

When searching to find design ideas, it is important to look for alternative design ideas. Each individual design idea will produce different effects on a system's scalar attributes: the resource usage and performance levels. It is a question of selecting the design idea which has the best performance to cost ratio or the 'best fit to the requirements' with regard to 'delivering stakeholder value' compared to 'resources used' (value to cost ratio).

Choosing the Best Combined Set from all the Alternatives

Design ideas put together in different combinations will interact with each other in different ways: there could be negative side effects and/or positive 'combining' effects (synergy). By having several alternatives, it is possible to select the combination of design ideas, which has the

best, estimated impact on the requirements. (Of course, the chosen combination can always be altered over time, in the light of feedback from evolutionary delivery.)

Reducing Risk by Use of Alternative Design

Another main reason for having alternatives is to reduce risk. If there are several candidate design ideas, then if the first choice fails there is always a backup. At an extreme, alternative design ideas may be implemented in parallel to ensure that specific requirements are fully met.

Design Optimization

When you are designing, you need to decide what type of optimization strategy you intend to use. The strategy options for Design Optimization Tradeoff include:

- **Cost Minimization**: When performance targets are met by specified designs, we can choose to continue to find alternative designs, that are at least equally well performing, with a view to reducing costs to the cost targets (if not below them!).
- **Design to Cost**: Another approach would be to design *to fully use* the all budgeted resources and to look for the designs that give maximum impact on the performance targets. In other words, the *most value* for a specific amount of limited resources. This is called 'Design to Cost.' '*Cutting your coat to suit your cloth.*'

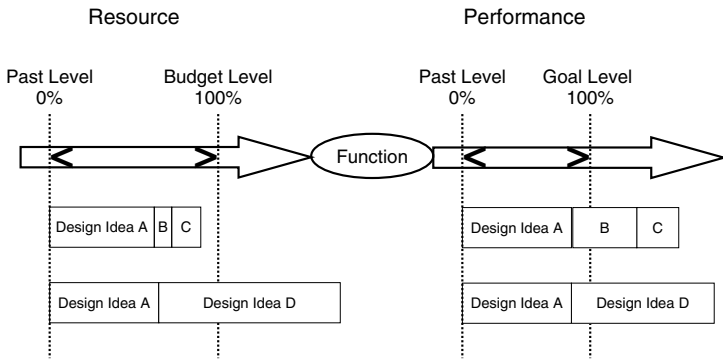

Figure 7.2
To 'design' is to find design ideas, like A, B, and C, which will contribute towards planned performance and resource levels, while simultaneously respecting all constraints. Design Idea D is 'good,' but costs too much.

- **Design to Performance Targets within Cost**: Another option would be to design to meet all the performance targets within cost, but to stop the process once all the planned performance levels were met. In other words, do not use additional time to reduce resource utilization further. This could be a possible approach when Time to Market is the most critical resource.
- **Design for Risk**: Another optimization concept would be to design with regard to risk. The *most pessimistic* estimates of performance impact, and of costs, would be used to determine the 'best design'. There are many other devices in Planguage that help us consider risk when designing *(see specifically, Impact Estimation).*

There are more combinations than those mentioned above. But you can see some basic choices. It is important in any project that you recognize how you are approaching the design optimization process, and that you communicate with your management about it. It could be there is some misunderstanding – maybe there *is* more financial budget available from them, as long as you show a track record of successful delivery.

If there are specific resource budgets that are critical to you, such as 'Time to Market', we recommend that you *initially* 'Design to Cost' with respect to 'calendar time' for delivery to market. Generally, you will want to design to meet the most critical constraints first, then see if you can maximize delivery of performance attributes and minimize other cost aspects in a second round of design effort.

Brief Recap of Planguage Methods and the Design Engineering Process

See Figures 1.3 and 1.6 and Table 1.1 in Chapter 1, 'Planguage Basics and Process Control,' and also the generic project process in Section 1.5. These show how the Design Engineering Process fits into the overall Planguage process model.

Specifically, with regards the Design Engineering Process:

- **Requirement Specification** supports the design engineering process by capturing the requirements. The requirements include specific information required for design decision-making. (For example, see further discussion on 'Priority Determination' in Section 7.7.)
- **Impact Estimation (IE)** is part of the Design Engineering Process. It is the Planguage method used to *evaluate and choose* design ideas. It also incorporates risk evaluation. In addition, IE can also be used to monitor the actual progress towards meeting the requirements. The actual step measurements, obtained after each Evo step has been completed (with delivery of one or more design ideas), can be

input into an IE table and compared against the original estimates. This feedback is used by the design engineering process, to understand where the gaps in design actually exist (that is, the gaps, which require additional design). (*See Chapter 9 for further details on IE.*)

- **Evolutionary Project Management (Evo)** is used to *actually deliver* the design ideas. Evo handles risk by several means:
  - implementing design, step by step
  - demanding that we choose the design ideas *most likely* to provide *high benefit* (highest value to cost ratios, highest performance to cost ratios) for *early* delivery (design ideas are 'sequenced' by some chosen evaluation of their potential benefits and costs into an Evo step plan)
  - testing the reality of the design ideas 'in the field'
  - providing and using feedback data after each step. We can then realistically understand the accuracy of our estimates, concerning design ideas, and can take appropriate measures, depending on the level of risk we perceive
  - we have incrementally 'banked' *some* results and eliminated *some* risk, which maybe means we can afford to discuss taking some higher risk steps.

Note: Both the requirement specification process and the design engineering process are incorporated into Evo; each result cycle demands re-evaluation of the design and brief re-evaluation of the requirements (possible adjustments and tradeoffs) (see Chapter 10). As stated earlier in this section, there is continuous iteration amongst these processes.

7.2 Practical Example: Beginning the Design Engineering Process

Let us say we have specified the following requirements for a project 'Staging a Conference':

Staging a Conference: Type = Function.
========== Conference Performance Requirements ==========
Participation: Quality Requirement:
Scale: Percentage of Worldwide Membership participating.
Goal: 10%.
Representation:
Scale: Percentage of Worldwide Membership represented within defined <groups>.
Goal [Age under 25 or equating to <Student Status>]: 10%.
Information:
Scale: Percentage of Talks rated as 'good' or better (5+ on feedback sheet scale).
Goal: 50%.

Conviction:
Scale: Percentage of Participants wanting to return Next Conference.
Goal: 80%.
Influence:
Scale: Percentage of Participants who <improve as result of the Conference>.
Past: 90%.
Goal: 95%.
Fun:
Scale: Percentage of Participants rating the Conference City quality as 'good' or better (5+ on Feedback Sheet scale).
Past: 45%.
Goal: 60%.
========== Conference Budget Requirements ==========
Financial Cost: Resource Requirement [Financial]:
Scale: Average Participant Conference Cost for an individual Participant including Travel Costs.
Fail: Less than $2,000.
Budget: Less than $1,200.
A set of requirements for a conference, mostly performance requirements and one budget.

Now we can, driven by these relatively clear requirements, start designing.

We begin by listing any design constraints – the 'given' *design* ideas. Here there are none, the only 'given' is the main function that we are to stage a conference.

We can then list 'at least one potential design idea, for *each* of the requirements.' This is an arbitrary way of covering the requirements with 'some' design.

Design Ideas:
Central: Choose a location in the membership center of gravity (New York?).
Youth: Suggest and support local campaigns to finance 'sending' a young representative to conference.
Facts: Review all submitted papers on <content>.
London: Announce that the conference is to be in London the next year.
Diploma: Give diplomas for attendance, and additional diplomas for individual tutorial courses.
Events: Have entertainment activities organized every evening, such as river tours.
Discounts: Get discounts on airfare and hotels.

Now, are these design ideas going to make the conference what we want it to be, as defined by the target levels? Nobody really knows and nobody can say. Why not? Well, it depends on the *interpretation* of the design ideas and

Table 7.1 Impact Estimation table

Design Ideas Requirements	*Central*	*Youth*	*Facts*	*London*	*Diploma*	*Events*	*Discounts*	*Sum for Requirement*
Performance Requirements								
Participation	80% ±50%	60% ±70%	0% ±50%	0% ±50%	30% ±50%	20% ±50%	30% ±50%	220% ±370%
Representation	80% ±50%	80% ± 50%	10% ±50%	0% ±50%	10% ±50%	20% ±50%	50% ±40%	250% ±340%
Information	0% ±50%	20% ±40%	80% ±50%	0% ±20%	20% ±50%	0% ±50%	0% ±30%	120% ±290%
Conviction	0% ±10%	20% ±50%	60% ±30%	80% ±50%	10% ±50%	80% ±50%	0% ±50%	250% ±290%
Influence	0% ±50%	40% ±40%	60% ±50%	0% ±50%	80% ±50%	80% ±5%	0% ±50%	260% ±340%
Fun	50% ±50%	40% ±50%	10% ±50%	0% ±0%	0% ±0%	80% ±50%	0% ±0%	180% ±200%
Sum of Performance	210% ±260%	260% ±300%	220% ±280%	80% ±220%	150% ±250%	280% ±300%	80% ±220%	
Resource Requirements								
Financial Cost	20% ±30%	1% ±1%	1% ±1%	1% ±1%	1% ±5%	30% ±50%	30% ±50%	111% ±135%
Performance to Cost Ratio	210/20	260/1	220/1	80/1	150/1	280/30	80/30	

their *execution* in practice. Can we influence that? Yes. By specifying a more detailed design specification with precise details of what we are going to do, and exactly how it is to be done in practice (the implementation and operational design detail). In other words, we now have that first 'sketch of the building,' but we need to get down to the detailed 'blueprints' (engineering) needed by the 'bricklayers and carpenters.'

The first step is to assess what we can evaluate about the impact of our proposed design ideas on the requirements (perhaps a little exaggerated to make our point).

The Impact Estimation table is a way to 'see' what we are doing. A 100% estimate on this table is a belief (right or wrong, well founded or not) that we *will* reach the planned level on time. The plus/minus estimate is a *rough* notion of the uncertainty. Until we get better definition and justification, these numbers are of only slightly better value than words, such as good, bad, excellent. But they do give us *a systematic basis for improvement* in our planning.

(I ask the reader to be patient; a proper version of Impact Estimation is presented in Chapter 9. All I am doing here is illustrating how one might define some requirements and design ideas, and then evaluate the impacts of each of the design ideas on all the requirements.)

The first observation I would make *here* is that we need to *redefine* the design ideas, *with more detail*. This is because of the high plus/minus uncertainties specified.

EXAMPLE Central: Town must be cheaply accessible by most Participants. Location itself must offer reasonable priced Accommodation (like university dorms) within walking distance of the Conference Facilities. Easy access to shops, restaurants, entertainment. <Add even more, and give concrete suggestions>

The ideal would be to create a hierarchy of the components of the design idea, Central and evaluate each separately, to home in on exactly what aspects of the design idea gave most stakeholder value.

7.3 Language Core: Design Idea Specification

Design specification is not just writing down the bare outline of the 'design idea' itself. You have the option of including a large number of additional parameters to describe the design idea. Why bother? Well, it is a matter of how much you want to force yourself to think about your idea, how much you want to share in writing with others, and how much you want to control any risks involved with the design. You must have reasonable confidence that the design idea really will deliver the results you have estimated.

In the design specification, you should ensure that you:

- Supply more detail in the Definition parameter, as this will lead to better understanding of its specific performance and cost impacts. This can be done using structural breakdown (see the Definition parameter in 'Transport by Buses' example below). Each of the sub-design ideas can be refined, until you feel that you have enough detail in the design ideas to guarantee the results levels and result timings, which are planned across all the requirements (or you identify that you need additional design ideas).
- Clarify and limit the design ideas to the specific ones that you want. Avoid ambiguity so that other people can't misinterpret your design intent.
- Identify and specify designs that clearly deliver at least partly one required performance attribute. Any 'side-effect' impacts of each design, on the other requirements, must also be analyzed and estimated. Use the '->' Impacts parameter to explicitly declare which attributes you hope, or expect, will be impacted by specific sub-design ideas. (''Design Idea A -> Safety.'')

Of course, you need to tailor your design specifications to suit the circumstances. A simple rule to guide you is 'to try the design specification parameters out at least once.' Too much description of a design idea will not hurt you, and can easily be deleted if it does not serve a useful purpose. Observe what the engineering team feels is worthwhile, and use that level of specification.

Here is an example of specifying a design idea. It was actually used in charity relief-organization work. (It shows how the main idea can be supported by sub-designs. The aim being to get better control over the results.)

EXAMPLE

Transport by Buses: Design Idea.
Description: Drive Refugees back across the border by bus.
Definition [Sub-designs]:
Village: Refugees should be selected from the same, or nearby, village -> Financial Cost.
White Paint: Buses should be painted UN white, and UN marked -> Safety <- Geneva Convention, Article 6.3.
Agreement: <Agreement with government> to allow transport and resettlement, without harassment, shall be made *before* crossing the border. Agreement papers will be onboard the bus -> Safety.
Radio: Buses shall have radio or mobile telephone contact with our headquarters *during* the transport -> Safety. ''Maybe also video and tape recorder?''
Witness: UN employees, or relief agency employees, perhaps UN soldiers will accompany the buses -> Safety.

Example of a Design Specification

Tag: OPP Integration.
Type: Design Idea [Architectural].
========== Basic Information ==========
Version:
Status:
Quality Level:
Owner:
Expert:
Authority:
Source: System Specification Volume 1 Version 1.1, SIG, February 4 – Precise reference <to be supplied by Andy>.

Gist: The X-999 would integrate both 'Push Server' and 'Push Client' roles of the Object Push Profile (OPP).
Description: Defined X-999 software acts in accordance with the <specification> defined for both the Push Server and Push Client roles of the Object Push Profile (OPP).
Only when official certification is actually and correctly granted; has the {developer or supplier or any real integrator, whoever it really is doing the integration} completed their task correctly.
This includes correct proven interface to any other related modules specified in the specification.

Stakeholders: Phonebook, Scheduler, Testers, <Product Architect>, Product Planner, Software Engineers, User Interface Designer, Project Team Leader, Company engineers, Developers from other Company product departments which we interface with, the supplier of the TTT, CC. "Other than Owner and Expert. The people we are writing this particular requirement for."

========== Design Relationships ==========
Reuse of Other Design:
Reuse of This Design:
Design Constraints:
Sub-Designs:

========== Impacts Relationships ==========
Impacts [Functions]:
Impacts [Intended]: Interoperability.
Impacts [Side Effects]:
Impacts [Costs]:
Impacts [Other Designs]:
Interoperability: Defined As: Certified that this device can exchange information with any other device produced by this project.

========== Impact Estimation/Feedback ==========
Tag: Interoperability.
Scale:
Percentage Impact [Interoperability, Estimate]: <100% of Interoperability objective with other devices that support OPP on time is estimated to be the result>.
========== Priority and Risk Management ==========
Rationale:
Value:
Assumptions: There are some performance requirements within our certification process regarding probability of connection and transmission etc. that we do not remember <-TG.
Dependencies:
Risks:
We do not 'understand' fully (because we don't have information to hand here) our certification requirements, so we risk that our design will fail certification <-TG.
Priority:
Issues:
========== Implementation Control ==========
Not yet filled in.
========== Location of Specification ==========
Location of Master Specification: <Give the intranet web location of this master specification>.

Figure 7.3
Here is a real (doctored!) example of a design specification using a version of the Design Specification Template given later in Section 7.9. Not all parameters are filled out yet. Notice that even the parameters which are not filled out (like Impacts [Side effects] and Issues) are asking important questions about the design – and hinting that responsible designers should answer such questions!

7.4 Rules: Design Specification

Tag: Rules.DS.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Note: Design specifications are either for optional design ideas (possible solutions) or required design constraints (that is, actual requirements AND consequently, pre-selected solutions).

Base: The rules for generic specification, Rules.GS apply. If the design idea is a design constraint (a requirement), the rules for requirement specification, Rules.RS also apply.

R1: **Design Separation**: Only design ideas that are intentionally 'constraints' (*Type: Design Constraint*) are specified in the *requirements*. Any other design ideas are specified separately (*Type: Design Idea*). *Note all the design ideas specified as requirements should be explicitly identified as 'Design Constraints.' (Repeat of Rules.RS.R9: Design Separation.)*

R2: **Detail**: A design specification should be specified in *enough detail* so that we know precisely what is expected, and do not, and cannot, inadvertently assume or include design elements, which are not actually intended. It should be 'foolproof.' *For complex designs, the detailed definition of its sub-designs can satisfy this need for clarity, the highest level design description does not need to hold all the detail.*

R3: **Explode**: Any design idea (Type: Complex Design Idea), whose impact on attributes can be better controlled by detailing it, should be broken down into a list of the tag names of its elementary and/or complex sub-design ideas. *Use the parameter 'Definition' for Sub-Designs.*

If you know it can be decomposed; but don't want to decompose it just now, at least explicitly indicate the potential of such a breakdown. *Use a Comment or Note parameter.*

R4: **Dependencies**: Any known dependencies for successful implementation of a design idea need to be specified explicitly. Nothing should be assumed to be 'obvious.' *Use the parameter, Dependency (or Depends On), or other suitable notation such as [qualifiers].*

(For design constraints (requirements), this is a repeat of the rule, Rules.RS.R5: Dependencies.)

R5: **Impacts**: For each design idea, specify *at least one* main performance attribute impacted by it. *Use an impact arrow '->' or the Impacts parameter.*

Comment: At early stages of design specification, you are just establishing that the design idea has some relevance to meeting your requirements. Later, an IE table can be used to establish the performance to cost ratio and/or the value to cost ratio of each design idea.

EXAMPLE Design Idea 1 -> Availability.
Design Tag 2: Design Idea.
Impacts: Performance X.

R6: **Side Effects**: Document in the design specification any side effects of the design idea (on defined requirements or other specified potential design ideas) that you expect or fear. *Do this using explicit parameters, such as Risks, Impacts [Side Effect] and Assumptions.*

Do not assume others will know, suspect or bother to deal with risks, side effects and assumptions. Do it yourself. Understanding potential side effects is a sign of your system engineering competence and maturity. Don't be shy!

EXAMPLE Design Idea 5: Have a <circus> -> Cost A.
Risk [Design Idea 5]: This might cost us more than justified.
Design Idea 6: Hold the conference in Acapulco.
Risk: Students might not be able to afford attendance at such a place?
Design Idea 7: Use Widget Model 2.3.
Assumption: Cost of purchasing quantities of 100 or more is 40% less due to discount.
Impacts [Side Effects]: {Reliability, Usability}.

R7: **Background Information**: Capture the background information for any estimated or actual *impact* of a design idea on a performance/cost attribute. The evidence supporting the impact, the level of uncertainty (the error margins), the level of credibility of any information and the source(s) for all this information should be given as far as possible. For example, state a previous project's experience of using the design idea. *Use Evidence, Uncertainty, Credibility, and Source parameters.*

Comment: This helps 'ground' opinions on how the design ideas contribute to meeting the requirements. It is also preparation for filling out an IE table.

EXAMPLE Design Tag 2 -> Performance X <- Source Y.

R8: **IE table**: The set of design ideas specified to meet a set of requirements should be validated at an early stage by using an Impact Estimation (IE) table.

Does the selected set of design ideas produce a good enough set of expected attributes, with respect to all requirements and any other proposed design

ideas? Use an IE table as a working tool when specifying design ideas and also, when performing quality control or design reviews on design idea specifications.

See Chapter 9, 'Impact Estimation.' Failing that, at least ask the 'Twelve Tough Questions' about the design ideas! (Can you quantify the impacts?) See Section 1.2 for details of the 'Twelve Tough Questions.'

R9: **Constraints**: No single design specification, or set of design specifications cumulatively, can violate any specified constraint. If there is *any* risk that this might occur, the system engineer will give a suitable warning signal. *Use the Risk or Issues parameters, for example.*

R10: **Rejected Designs**: A design idea may be declared 'rejected' for any number of reasons. It should be retained in the design documentation or database, with information showing that it was rejected, and also, why it was rejected and by whom.

EXAMPLE
Design Idea D: Design Idea.
Status: Rejected.
Rationale [Status]: Exceeds Operational Costs.
Authority: Mary Fine. Date [Status]: April 20, This Year.

7.5 Process Description: The Design Engineering Process

Process: Design Engineering Process

Tag: Process.DE.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Assumption: We have clearly-stated and reasonably complete requirements.

Notes:

1. Design is an iterative process. The process given in this section should be viewed with this in mind; the procedure is written as if it were carried out in a single pass, but in practice, a much more complex pattern of cross-checking, backtracking and tradeoffs would actually be carried out.
2. This procedure is much longer than it needs to be, due to the nature of this book. You should probably use a more concise version (say, one statement for each procedure step).

Overview of a Design Process

Design is an *intellectual* process, which is supported by problem definition, requirement specification, best-practice design process standards and analysis tools. The following fundamental questions arise in designing:

1. **Analyze the Requirements:** Which requirements are of high stakeholder value? What constraints apply? What is the priority time sequencing for delivery of requirements?
2. **Find and Specify Design Ideas:** How do we *find* and *specify* potential design solutions for our requirements?
3. **Evaluate Design Ideas:** How do we *evaluate* potential design solutions?
4. **Select Design Ideas and Produce Evo Plan:** How do we *choose* from several 'good' design alternatives? What do we do about *uncertainties*, and about the *risk* that the selected designs are not as good as we thought?

Notes:

Ansoff points out that "Simon has shown that solution of any decision problem in business, science, or art can be viewed in four steps:

1. *PERCEPTION of decision need or opportunity. Simon calls this the INTELLIGENCE phase.*
2. *FORMULATION of alternative courses of action.*
3. *EVALUATION of the alternatives for their respective contributions.*
4. *CHOICE of one or more alternatives for implementation." (Ansoff 1965)*

This supports the choice of the four main sub-processes of the Design Process!

Figure 7.4
Overview of a Design Process. This applies with or without a quantified engineering approach to design.

Entry Conditions

E1: The Generic Entry Conditions apply. The requirement specification should ideally have exited from Specification Quality Control (SQC).

E2: Any existing feedback, from Impact Estimation (design idea analysis), or practical trials, is made available to the design engineer.

Procedure

P1: **Analyze the Requirements**: You may well identify stakeholder conflicts and overlaps[3] amongst the requirements while analyzing them. These need conflict resolution: consider if the 'tougher' requirement level can be used, identify the 'owning' stakeholders for all values and negotiate with the stakeholders. It may well be worth waiting until you have some alternative design ideas before you negotiate with the stakeholders, as by then you will have a better understanding of

[3] Overlaps in requirements represent either an opportunity for additional value to be delivered or, the possibility for over-estimation of value *('double accounting').*

what solutions can be delivered. See also discussion in Section 7.7 on Priority Management.

P1.1: **Establish the Stakeholder Value on Delivery of each Requirement**: The value on delivery of each requirement to the system/organization should be assessed. You are looking for the requirement areas where there are major benefits. (What is of value depends on the stakeholders: it might not be just financial resource.) Identify the volume of use associated with each requirement.

Ideally, the stakeholders will have already selected the highest value requirements as the *critical requirements*.

For example: resource savings for performance requirements will be relatively simple to determine. Say you wanted to bring the time of carrying out a transaction down from two minutes to one minute. The benefit to the business, assuming 200 such transactions were carried out a day, would be 200 minutes per day. If operators had to wait while each transaction went through – this could amount to freeing up over three work-hours each day to carry out additional activities. In other words, assuming 250 work-days each year, 250 multiplied by 200 minutes each year. To the business, the ability to free up staff or the cost of employing the staff in this area, is the 'value' gained on delivering the requirement.

P1.2: **Sequence the Delivery Order of the Requirements**: Sequence the requirements for attention in the order of maximum stakeholder value first. Adjust to cope with any dependencies amongst the implementation of requirements (These dependencies either prevent implementation or prevent some level of benefit being achieved).

Note also the possibility for delivering each requirement in stages either by gradually improving a performance level or cost level or by delivering into different areas at different times (that is, to divide according to qualifier conditions; for example, by geographic area, by role and/or by timescale).

P1.3: **Establish Scope of Design Interest**: Establish and specify the scope of interest for the system design. This will be a set of qualifying conditions covering a specific timeframe and specific system space (locations/components/functionality). For larger systems, you might want to divide the system into major subsets – maybe, say, by functionality and/or by timescale, so that you can work on a series of smaller design areas.

P1.4: **Make a List of Requirements within the Scope defined**: Identify any function, performance, resource or condition *constraints* specified in the requirements. Then identify the *target* requirements. Note the qualifying conditions, which apply to each one of them.

P2: **Find and Specify Design Ideas**:

P2.1: **Exploit any Earlier Notes of Design Ideas**: Check to see if there is already a list of potential design ideas developed at the same time as the requirement specification. If there is, include those design ideas for consideration.

P2.2: **Establish the Design Constraints**: Read the requirements to see if there are any design constraints specified (Type: Design Constraint). If there are, then note them and any specific conditions qualifying them [time, place, event].

P2.3: **Brainstorm Design Ideas**: Search all available sources of design information for good matches to our stated requirements. (Specifically with regard to meeting the function requirements, achieving the performance levels and, delivering within the budgets. Any constraints and conditions must also be considered.)

Identify any dependencies amongst design ideas.

Also identify any design ideas that are alternatives.

Attention needs to be focused especially on the areas of greatest benefit to the system/organization. It is *the gaps* between our current updated system design process benchmarks (how well we have satisfied requirements until now) and our specified targets, which are of interest.

We are totally dependent in our search on the following:

- Knowledge of the *existence* of good design ideas (Where are they? Do we have the best ones?).
- Having complete and reliable *information about the likely impacts* of the design ideas on system attributes, so we can match the best design ideas to our residual requirement gaps. Most design ideas have too little specified, or available, data about their performance and cost characteristics.
- Understanding how design ideas *mix and interact* with each other. (Maybe the mixture will conflict? A design idea, in itself, might seem satisfactory, but the effect of combination with other design ideas, already in place, or under consideration, could be a counter-productive. Alcohol and driving don't mix well; though each in the right time and place might be acceptable.)

Hint: Select design ideas from available knowledge: books, periodicals, conference proceedings, past products, memory, colleagues, web searches, company experience, competitive analysis, benchmarking and others.

Stop the search when a set of satisfactory design ideas has been found, or when you run out of time to search for more.

P2.4: **Draft Design Ideas**: Draft a set of the design ideas, which might satisfy the requirements. See the design specification template outlined in Section 7.9.

P2.5: **Collect Specification Detail (to support later Impact Estimation)** : Add detail to the design idea specifications in order that there is sufficient information to enable estimates of the impact that each design idea will have on each performance and cost attribute. Refine the design idea specifications to the levels of detail, which reflect the 'level of uncertainty' and 'risk of deviation' from planned levels, which *you* are prepared to accept. Ensure all suspected risks, assumptions, and uncertainties are documented.

Document clearly, where any design idea has weaknesses with regard to the requirements. (For example, "Risk: Too long an implementation time." and "Risk: High risk of user dissatisfaction over usability.")

P2.6: **Consider Design Implementation**: Once you have defined the design ideas themselves, then turn *your attention to their implementation processes*. What qualifications are required for the implementers or subcontractors? What process should they follow? How will they be required to prove or measure their results? Leave nothing essential to the whims of others! Get control over your design ideas. *See 'Implementation Control' section in the design specification template in Section 7.9*.

P2.7: **Consider System Capacity and Growth**: Even when things work well in practice initially, there is no guarantee they will *continue* to do so. Success breeds volume. Volume breeds *capacity* problems. The 'good' system is no longer good enough. So we must be prepared to undertake a *continuous* responsibility for modifying the system design to meet *changed* circumstances. Sometimes 'gradual adjustment' is all that is necessary. Sometimes major new architecture is necessary. You need to be explicit about system capacity and give, if relevant, an outline of your plans of how to cater for system growth.

P3: **Evaluate Design Ideas**: We must evaluate the effects of a design on the system to which it will be added, and with regard to how it will mix with 'design changes yet to be implemented,' or even 'yet to be imagined,' by considering our long-range requirements, and architecture, for adaptability. We must ensure that we evaluate design ideas for their incremental effects on *all* of our required attributes – not just the requirements we initially designed them to primarily impact. Consider side effects – good and bad, intended and unintended.

P3.1: **Filter for Violation of any Constraint**: A design idea must not violate any applicable constraint(s). Check each design idea against

each constraint. Mark the status of any design idea that violates, or potentially might violate, any constraint as 'Rejected' giving the reason in a 'Rationale' parameter. *(This is a more systematic check than might have been carried out when brainstorming during P2.3.)*

P3.2: **Estimate all Impacts**: Using an Impact Estimation (IE) table, estimate the impact of each design idea (or set of design ideas), on each performance target and each cost budget.

Cite evidence, plus/minus uncertainty and source(s) for each impact estimate. Determine the credibility of each estimate (using the credibility ratings scale from 0.0 to 1.0).

See Chapter 9, 'Impact Estimation' for further detail.

P3.3: **Consider Side Effects**: It is not just a case of checking that a design idea delivers the *required* benefits, we must also consider whether a design idea has any *unintended* negative side effects, which are *unacceptable (some negative effects may be tolerable overall).*

P3.4: **Consider Safety Margins**: You also need to assess whether the safety factors are met. Maybe a factor of two times 'over-design' is required?

If needed, return to P2 to look for further design ideas or, consider if the requirements need modifying.

P4: **Select Design Ideas and Produce Evo Plan**:

P4.1: **Initial Sequencing of Design Ideas**: We are then faced with decisions about which design ideas to select *for implementation* and which to reject. We will often be faced with several 'sufficient' alternatives; any one of which would be adequate. So how do we choose? Usually, no one dimension (for example, 'cost') is decisive.

In general, the selection decision must be made based on the *many* dimensions of measurable performance and cost. Any conditions (such as those specified in qualifiers) also impact selection.

Selection means *prioritization*. We need to determine which requirements we intend to satisfy first. *(We have an initial selection of critical requirements from P1.)* Evolutionary project management (Evo) will have the final say in determining the actual implementation sequence, but during the design engineering process we must attempt an initial sequencing of the design ideas to meet the requirements' priorities.

Establish which design ideas impact each of the constraints.

Establish which design ideas impact each target requirement (function targets, performance targets and budget targets). The impact estimation table will provide the information for the performance and budget targets. The design specifications will also hold some information depending on how much detail has been captured. *Remember the*

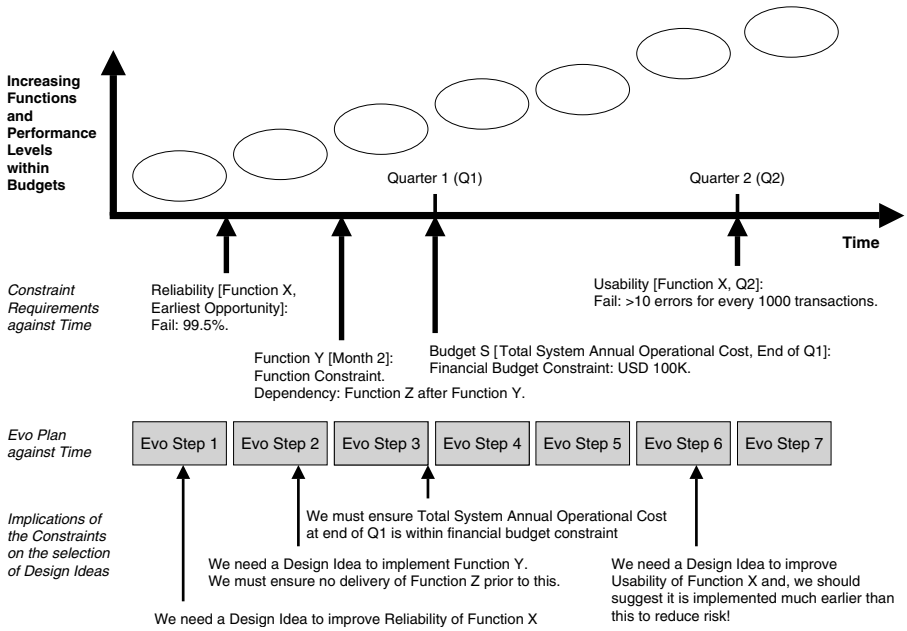

Figure 7.5
Diagram shows the path for a system improving over time as Evo steps are delivered. The points marked on the time axis are the times when specified constraints have to be delivered. The sequenced Evo steps are attempting to deliver the requirements on time. The design ideas making up the content of the Evo steps have, as the first priority, to try to satisfy any constraints. Not shown in this diagram – the second priority is that they deliver the required target functions, and to the target levels for the performance and cost attributes.

performance and budget attributes do not 'hang in mid-air' – they will be attached to some functionality.

Next, identify any design idea dependencies.

At this stage, there should be a sequence of design ideas dictated by the (system scope) conditions – especially by the required timescales.

Where there are alternative design ideas, the performance to cost ratios from an IE table can be used to determine which designs contribute most efficiently towards meeting the requirements.

P4.2: **Ensure adequate Safety Margins to address Risks**: Ensure the sum of the impact estimates for each performance requirement covers the required safety margins for both performance and cost targets.

P4.3: **Consider Design Ideas with regard to the Risks**: We must also consider the uncertainties in our evaluations. The initial, purely intellectual, design process is inherently at risk of giving us false conclusions because *our* system is always somehow different from all *others*. Past design idea experience might not be valid. Our information on design effects could also be too general, or even downright wrong.

We can make allowances to cope with risk by 'over-design.' We may deliberately choose more design ideas than we strictly need in order to have safety margins. For example, we may choose two solutions that back each other up, rather than one.

We must also carefully *validate* our design choices in reality, and be prepared to *re-design* whenever practical experience shows this is necessary.

P4.4: **Consider Optimization**: Once we have an initial set of design ideas, which provide a satisfactory solution, then we can try to *optimize* using a declared optimization strategy (which might be a requirement of an engineering policy statement).

For example:

- Look for the least-cost set of design ideas, which *fully* meets the requirements.
- Select design ideas with the highest performance to cost ratios. This is *generally* good competitive practice.

But there are all kinds of variations on optimization strategies depending on your priorities regarding performance targets, resource budgets and risk (see Design Optimization within Section 7.1).

P4.5: **Re-define Design Definitions**: Re-define design ideas, if you can improve your impact estimates, and get better control over desired results by doing so. Re-define them so that they have substantially different performance and cost impacts, in the direction you need them to be.

P4.6: **Consider Pilot and/or Trial**: Plan to try out design ideas, which *seem* high risk, in pilots and/or trials, or specify them for implementation in early evolutionary result cycles. Feedback any results into *this* process (at P3.2: Estimate all Impacts.). Refine your estimates.

Design knowledge, from past uses of a design, gives us some idea of how we can expect things to work. But, new systems contain many new elements of technology, inputs and people. Thus, only practical use of a new design idea, in the real environment, will assure us that we were correct in our estimates of design effect or will convince us that we were to some degree wrong. So, the design engineering process must somehow be linked with a practical process of trying things out, well before large-scale irreversible commitment is made to design ideas.

Exit Conditions

X1: The Generic Exit Conditions apply. The set of design specifications should have exited SQC with no more than one estimated remaining major defect/page. *(Expectation*: *If you don't demand such*

a low exit level, the specification will have 20 or more major defects/page.)

X2: A safety factor of <*four*> is required for all performance and cost attributes.

Note: (4 times over-design, is NOT 4 times more cost!)

Note: Process 'Exit' means we can *then* use the design specification for planning *trials to get feedback* (that is, using the design in Evo steps, see Chapter 10). It does not mean the design specification is a 'final' specification.

Gap Analysis by Igor H. Ansoff

The procedure within each step of the cascade is similar.

(1) A set of objectives is established.
(2) The difference (the 'gap') between the current position of the firm and the objectives is estimated.
(3) One or more courses of action (strategy) is proposed.
(4) These are tested for their 'gap-reducing properties.'

A course is accepted if it substantially closes the gaps; if it does not, new alternatives are tried.

Igor H. Ansoff, Corporate Strategy *(Ansoff 1965 Pages 25–26). Also quoted in Mintzberg (1994).*

7.6 Principles: The Design Engineering Process

1. **The Principle of 'Design Ideas are only as Good as the Requirements Satisfied'**
   Design ideas cannot be correctly judged or validated *except* with respect to *all* the performance and cost requirements they must satisfy.
2. **The Principle of 'The Best Chess Move'**
   You should try with each increment of design specification or design implementation, to get the best possible satisfaction of your unsatisfied performance requirements, from your unused cost budgets.
3. **The Principle of 'Results Beat Theory'**
   Design ideas are only as good as their *real results*, not their *intent.*
4. **The Principle of 'Early Surprises'**
   You never *know* how it works, until you have actually tried out a design idea in practice. Get surprised as early as possible!

5. **The Principle of 'It's Not Just What You Do, It's How You Do It'**
   Design ideas must try to exercise control over both design *content* and design *implementation*. The devil is in the details!
6. **The Principle of 'Good is Not Always Good Enough'**
   A 'good' design idea might not be good enough to meet all *your* targets on *time*.
7. **The Principle of 'Designs should have Good Return on their Investment'**
   'Good' design ideas might cost *too much*, sooner or later.
8. **The Principle of 'Sneaky Gremlins'**
   Apparently 'good' design ideas might have subtly-hidden nasty side effects. Estimate them, know when you don't know them, measure them, and don't assume they won't hurt you! They will show *you* no sympathy!
9. **The Principle of 'Design Beats Test'**
   Design performance 'in', and design 'to control' costs:
   You cannot *test* quality *into* a badly designed system.
10. **The Principle of 'Eternal Vigilance for the Butterfly Effect'**
    You never *finally* know about a design idea's effects;
    Tomorrow's slightest change might ruin your whole project.
    Even initially successful designs might have to be adjusted for growth and change.

7.7 Additional Ideas

Priority Determination

Systems engineering can be viewed as a constant stream of priority evaluations. So priority determination is a key concern. However, the conventional means of deciding priority are frequently inadequate: a subjective weighting approach is, unfortunately, often adopted. Ideally priority determination for implementing requirements should be:

- a 'performance to costs impact' and 'resource-focused' process. It should consider value to cost ratios, return on investment (ROI) and take into account resource availability.
- an information-based process, which makes full use of the available factual information, and is able to reuse this information. Not a weight-based process.
- a dynamic process, which uses feedback from the ongoing implementation; and is open to instigating, and catering for, change in requirements and in design ideas.

Ideally the ultimate values to the stakeholders, which are the results of the system performance characteristics, would be evaluated and used to determine priority. In practice it might be difficult for a systems engineer to access the stakeholder domain data needed to calculate the value that the stakeholder would expect to experience. Even the stakeholder might have difficulty estimating the 'value delivered' accurately. So, we might choose to fall back on a more immediate notion of stakeholder value – meeting the required targets.

What is wrong with the Subjective Weighting Approach for Determining Priority?

In the priority weighting (or priority ranking) process, each element of a set of elements in a decision-making model, is subjectively assigned a numeric value indicating its priority (For example, a value on a scale of 1 to 10 or, a percentage weight).

The degree of subjectivity[4] 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 such a process is weak, when determining the priority for implementing requirements, include:

- Information overload: too many things have to be taken into account at once for subjective assessment to work well.
- Lack of specific information: often there are gaps in the information available: evidence and source data are usually missing.
- 'One-off' weightings: weightings tend to be 'frozen', they are not reassessed frequently.
- Lack of consideration of resources: resources are simply not taken into account.
- An individual stakeholder's viewpoint is limited (a person's subjective judgment depends on many things. For example, experience and access to information).
- Typically people can only participate in supplying their requirements and committing their resources. They are unlikely to be able to make a globally optimal priority decision, on behalf of the entire stakeholder community.
- In a group meeting, factors such as authority, office politics and personality interfere with the outcome.

[4] Note, I am objecting to subjective weightings, not to stakeholders proposing their own subjective requirements.

The Role of Resource in Determining Priority

Planguage defines priority as follows:

> A 'priority' is the determination of a relative claim on limited resources. Priority is the relative right of a competing requirement to the budgeted resources.[5]

If resources were unlimited, there would be no need to prioritize things. You could have it all.

Many approaches to priority oversimplify or even eliminate consideration of resources (for example, see Saaty 1988; Akao 1990). Yet return on investment (ROI) is ultimately the key driver when deciding priority. What value shall be obtained in relation to the 'resources needed' (the costs)?

Resource availability can also be a factor in determining implementation priority. Selection of a priority solution might be:

- influenced by a lack of some resources
- affected by the ability to substitute one resource with another.

Planguage Information Supports Priority Determination

Planguage captures a wide range of reusable information that supports priority determination. It quantifies all scalar requirements and caters for individual deadlines at a detailed level, and as a result gives you a greater level of priority control. Some key Planguage specification parameters assisting priority determination are as follows:

- Value
- Stakeholder
- Constraint
- Target
- Dependencies
- Qualifiers [Time, Place, Event]
- Authority
- Source.

The source, authority, and stakeholder information establishes the stakeholders affected by a requirement, and their level of responsibility. When determining priority, meeting any constraints is the first priority. The next priority is to meet the targets. The qualifiers narrow the requirements down to the specific conditions: time qualifiers specify

[5] I mean all types of resource including time to deadline, human effort, money and space.

the timescales, place qualifiers limit the system space and event qualifiers state any specific circumstances, which apply.

Planguage might capture this information, but it requires evaluation to establish the priorities. There may be priority conflicts needing negotiation.

Priority Strategy

One piece of information vital for priority determination is the strategy for priority. There are several different strategies that could be chosen. See discussion on Design Optimization within Section 7.1.

Dynamic Priority Evaluation

Planguage adopts a dynamic, numeric idea of priority. Priority is defined as the claim on resources to develop or operate a system. It is the currently unfulfilled requirements, (the gaps) which have priority. Our highest priorities, at any moment in time, are the unfulfilled requirements that are due next, date-wise.

There are no artificial weighting factors needed in Planguage. We use only direct natural statement of the qualities and costs we want, together with when we want them. We compare 'what we want' with 'what we have' at the moment. The larger the gap between 'wants' and accomplishments, the higher the current priority in that area to do more design work or to do more implementation work.

Using Evo, priority control becomes early, frequent and continuous, throughout the project design and implementation phases. Priorities change as they are satisfied (just as appetite changes as food satisfies it).

Also, the basic requirements can change at any moment of a project. It would be convenient if they didn't, but the real world is not that co-operative! Continuous re-assessment of priority, allows any changes in the requirements to be incorporated into the system design process.

See also Section 9.7 on priority.

7.8 Further Example/Case Study: Design Specifications Masquerading as Requirements

This is a sample of some real design ideas, which were *found in a requirement specification.* (Certain details are changed for confidentiality.)

They are extremely early outline drafts and still need a lot of work! We certainly had not yet enhanced the specifications to

the level required by the Planguage template in this chapter. However, the drafts do give some practical insights into simple Planguage formatting. The most important steps we took were as follows:

- to refuse to treat them as requirements
- to identify the performance attributes they were *intended* to impact (*see 'Impacts' parameters below*) and
- to define the impacted performance attributes properly.

EXAMPLE

Adaptive Channel Allocation: ACA: Design:
Assumption [ACA]: New Product must automatically yield to Macro cellular system, and to re-tuning of the Macro radio network.
Impacts [Co-existence]: Slow or Fast? <- Marketing Specification 3.11.
Note: This is one design idea, not a constraint.
Automatic Roaming Designs: "A rough collection of design ideas."
Impacts: Automatic Roaming.
Note: these may be design constraints! <- New Product Team 4 March.
IS-41 signaling link to the public network <- Marketing Specification 5.2.1.
Signaling, data and messaging interfaces <- Marketing Specification 5.2.
The New Product must support the protocol of Cellular Messaging Teleservice (CMT) over its signaling link over both public and cellular network <- Marketing Specification 5.2.2.
The New Product must support the receipt and acknowledgment protocol for voice-message-waiting indication <- Marketing Specification 5.2.3.
Cell Plan Minimization:
Impacts: {Installability, Maintainability}.
Cooling Fans [**Radio Heads**]: to be avoided to avoid noise, but quiet ones, as defined by Quietness quality requirement, acceptable.
Impacts: Quietness <- Marketing Specification 4.1.5.
Product Evolution: Design Idea. "These design ideas are a rough collection from the Marketing Specification".
Impacts: Evolution.
New Product shall have a modular structure <- Marketing Specification 3.2.
Modular, future proof <- Marketing Specification 4.3.3.
New Product shall be easy to upgrade <- Marketing Specification 3.3.
The switch must support remote SW loading <- Marketing Specification 4.3.4.
'Plug & Play' <- Marketing Specification 4.5.2.
Remote Software Upgrade: for both correction and upgrading proposes <- Marketing Specification 4.5.14.
Software changes shall not require manual physical access to Radio Heads <- Marketing Specification 4.5.15.
New software – upgrades, patches, new releases, etc. should require a minimum of scheduled downtime for New Product <- Marketing Specification 4.5.16.
Home Location Register: HLR:
HLR is part of the Macro cellular system?? <- Marketing Specification 4.3.7.
Impacts: <unspecified>.
Low Power Consumption [**Radio Heads**]:
Low Power consumption will be designed.
Impacts: Quietness "in order to avoid fans and consequent noise" <- Marketing Specification 4.1.5.

EXAMPLE CONTINUED

Low RF Power Output [**Radio Heads**]:
Impacts: {<avoiding interference>, Availability, Co-existing, Per User Cost, Robustness, others} <- Marketing Specification 4.2.1.8.
Remote SW Loading:
The switch must support remote SW loading <- Marketing Specification 4.3.4.
Impacts: Maintenance.
Single Cabinet [**Central Equipment**]:
The Central Equipment must fit into a single cabinet including power, but not batteries <- Marketing Specification 4.5.4.
Comment: This is really a way to achieve Volume of 36 liters as estimated by TW.
RH1: Assumption: RH assumed to be single cabinet.
Impacts: <unspecified>.
Basically, what we did was to identify these design specifications as design ideas, not requirements (design constraints), and to structure them so we could see their Source and their Impact intents.

7.9 Diagrams/Icons: The Design Engineering Process

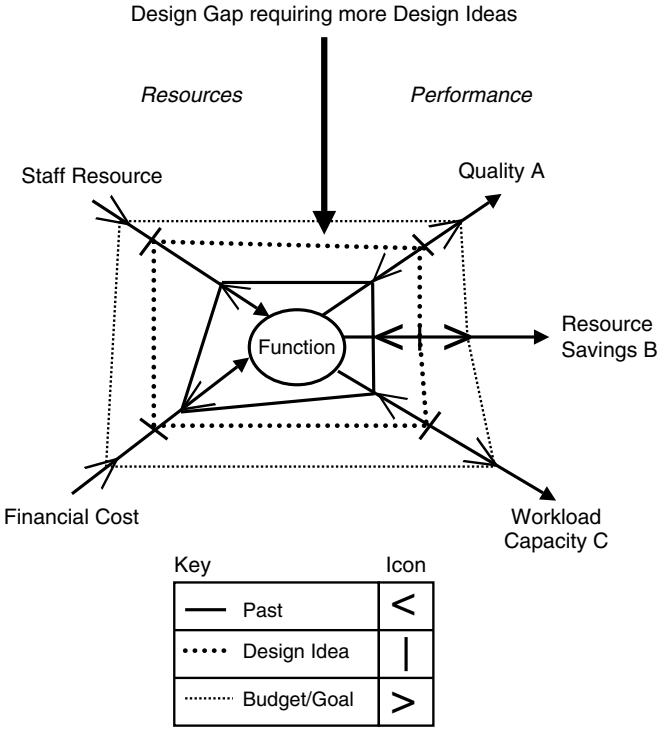

Figure 7.6
Diagram showing the gap between the Past and the Goal/Budget levels and the contribution that a Design Idea makes towards filling the gap.

Design Specification Template <with Hints>

Tag: <Tag name for the design idea>.
Type: {Design Idea, Design Constraint}.

========== Basic Information ==========
Version: <Date or version number>.
Status: <{Draft, SQC Exited, Approved, Rejected}>.
Quality Level: <Maximum remaining major defects/page, sample size, date>.
Owner: < Role/e-mail/name of person responsible for changes and updates>.
Expert: < Name and contact information for a technical expert, in our organization or otherwise available to us, on this design idea>.
Authority: <Name and contact information for the leading authorities, in our organization or elsewhere, on this technology or strategy. This can include references to papers, books and websites>.
Source: <Source references for the information in this specification. Could include people>.
Gist: <Brief description>.
Description: <Describe the design idea in sufficient detail to support the estimated impacts and costs given below>.
<Term Tag here>: Definition: <Use this to define specific terms used anywhere in the specification>. "Repeat this for as many definitions as you need"
Stakeholders: <Prime stakeholders concerned with this design>.

========== Design Relationships ==========
Reuse of Other Design: <If a currently available component or design is specified, then give its tag or reference code here to indicate that a known component is being reused>.
Reuse of This Design: <If this design is used elsewhere in another system or used several times in this system, then capture the information here>.
Design Constraints: <If this design is a reflection of attempting to adhere to any known design constraints, then that should be noted here with reference one or more of the constraint tags or identities>.
Sub-Designs: <Name tags of any designs, which are subsets of this one, if any>.

========== Impacts Relationships ==========
Impacts [Functions]: <List of functions and subsystems which this design impacts attributes of>.
Impacts [Intended]: <Give a list of the performance requirements that this design idea will positively impact in a major way. The positive impacts are the main justification for the existence of the design idea!>.
Impacts [Side Effects]: <Give a list of the performance requirements that this design idea will impact in a more minor way, good or bad>.
Impacts [Costs]: <Give a list of the budgets that this design idea will impact in a major way>.
Impacts [Other Designs]: <Does this design have any consequences with respect to other designs? Name them at least>.

========== Impact Estimation/Feedback ==========
For each Scalar Requirement in Impacts [Intended] (see above):
Tag: <Tag name of a scalar requirement listed in Impacts [Intended]>.
Scale: <Scale of measure for the scalar requirement>.
Scale Impact: <Give estimated or real impact, when implemented, using the defined Scale. That is, given current baseline numeric value, what numeric value will implementing this design idea achieve or what numeric value has been achieved?>.
Scale Uncertainty: <Give estimated optimistic/pessimistic or real ± error margins>.
Percentage Impact: <Convert Scale Impact to Percentage Impact. That is, what percentage of the way to the planned target, relative to the baseline and the planned target will implementing this design idea achieve or, has been achieved? 100% means meeting the defined Goal/Budget level on time>.

Figure 7.7
Continued next page

Percentage Uncertainty: <Convert Scale Uncertainty to Percentage Uncertainty ± deviations>.
Evidence: <Give the observed numeric values, dates, places and other relevant information where you have data about previous experience of using this design idea>.
Source: <Give the person or written source of your evidence>.
Credibility: <Credibility 0.0 low to 1.0 high. Rate the credibility of your estimates, based on the evidence and its source>.
========== Priority and Risk Management ==========
Rationale: <Justify why this design idea exists>.
Value: <Name [stakeholder, scalar impacts and other related conditions]: Describe or quantify the knock-on value for stakeholders of the design impacts>.
Assumptions: <Any assumptions that have been made>.
Dependencies: <State any dependencies for this design idea>.
Risks: <Name or refer to tags of any factors, which could threaten your estimated impacts>.
Priority: <List the tag names of any design ideas that must be implemented before or after this design idea>.
Issues: <Unresolved concerns or problems in the specification or the system>.
========== Implementation Control ==========
Supplier: < Name actual supplier or list supplier requirements>
Responsible: <Who in your organization is responsible for managing the supplier relation?>
Contract: <Refer to the contract if any, or the contract template>
Test Plan: <Refer to specific test plan for this design>
Implementation Process: <Name any special needs during implementation>
========== Location of Specification ==========
Location of Master Specification: <Give the intranet web location of this master specification>.

Figure 7.7
Design Specification Template. This is a form to fill out, with <hints in fuzzy brackets>.

7.10 Summary: The Design Engineering Process

The 'design engineering process' is a systematic, rational process of finding design specifications, which when implemented will satisfy a balanced set of requirements on time.

The term 'design engineering' means a design process based on multi-dimensional quantified requirements and multiple quantified design attributes. It requires concurrent use of an implementation process, like Evo, based on quantified measurement of performance and costs at frequent evolutionary cycles, and of necessary analysis and correction to maintain progress towards (potentially adjusted or traded off) formal and quantified targets.

The selection of design ideas is determined by the need to deliver a set of specified stakeholder target levels within a set of specified constraint levels.

The design engineering process is really concerned with identifying optional design ideas and evaluating the alternative possibilities to find

a satisfactory architecture (that is, the sum of all design ideas), which provides:

- the best fit with the requirements
- early delivery of key results (with high stakeholder values)
- best value to cost ratios and performance to cost ratio, and
- acceptable risks.

The design engineering process may also involve identifying the best reaction (redesign) to any feedback (good or bad feedback from actually implementing design ideas in the real system).

The design engineering process cannot usually be done, competitively, in a single pass. The effects of even a single design idea are too complex to understand without 'experience analysis' from past use of the idea (see 'Impact Estimation', Chapter 9), and especially without actual use on our new system (see Evo, Chapter 10). So it must normally be expected that the 'final' and 'correct' design specification can only be evolved towards (never perfectly or ideally reached) as a result of multiple feedback-and-change cycles.

Refinement of design can be done in parallel with actual use (by at least some early stakeholders) of a version of the product. The practical feedback from this early delivery can be used to improve the design; probably faster and more correctly than by staying in the 'design phase' longer.

A further complication is that as time goes on, both the 'design requirements' and 'potential and selected design technology' will 'expectedly' change, thus requiring yet another set of cycles of learning how to satisfy these new, changed requirements. Never perfect, continuously better, is the watchword.

In terms of 'Competitive Engineering' you can *always* refine the design to be more competitive. However, there is a point where the cost and time of refining the design exceeds any competitive benefit, and it is time to stop designing and to get the product out of the door, this time around.

Design Policy

Design ideas are only really finally validated when they display satisfactory attributes in a real system (that is, after successful delivery in an evolutionary step). Don't kid yourself that they are 'final' before that.

A suggested mental attitude towards design specifications. Don't believe any estimates of performance and cost, only reality as measured!