Chapter

6

RESOURCES, BUDGETS AND COSTS

Costs of Solutions

GLOSSARY CONCEPTS

Resource

Budget

Cost

6.1 Introduction

You only get what you pay for.

Folk wisdom

A system designer tries to meet the specified system requirements by identifying value-producing design ideas (solutions). At the same time as looking for function and performance, the designer must also consider the *resources needed*, specifically respecting any *constraints placed on resource usage*.

Relationships amongst Resources, Budgets and Costs

Resources are the inputs, or the 'fuel,' for a system. They are needed to produce the system's performance attributes. They are analogous to the capital expense, air and fuel needed for a car engine (function attribute is to provide power) to deliver the engine's performance attributes.

Resource requirements specify *how much* we plan to use of a limited resource to bring about change (new systems, improved systems) and/or to operate a system. Resource requirements are also known as *budgets*.

Stakeholders' *resources* pay all the real project and system *costs*. In other words, *costs* are the actual consumption of resources. Resource requirements are therefore sometimes termed *cost requirements* (*or cost budgets*).

The term 'resources' is used here in the broadest sense of that word. It covers money, time, people, space and any other 'currency' with which we pay for system changes and the operational system.

Stakeholder Requirements and Resources

Projects exist *primarily* to deliver stakeholder performance requirements. A system's *functions* are probably already in place, and may well have been for ages in earlier generations of the system, but the projected performance outputs (qualities, workload capacities and/or resource savings) of the system are probably not satisfactory – or they will not be in the future. That is what puts you in the 'business of change,' in other words 'creates your project.'

Any project sponsor has limited resources, and is faced with alternative ways to use them. Projects must control costs, or they will either exceed their project sponsors' capability for providing resources or be seen as a less attractive (read 'less profitable') investment for those resources.

For most of today's projects, controlling cost (resource expenditure) is quite a juggling act: you have to balance and trade off performance against budgets. Your stakeholders want a better system, but not at too high a cost. They can usually specify a 'budget' for what they are willing to pay for each system improvement, based on their knowledge of their current system and their competitors' systems. Of course, their budgets may or may not be realistic!

There may also, in practice, actually be *real and absolute limits* on their budgets, which are in no way just 'hopeful plans'. These limits will more severely restrict the amount of resource that can be made available.

'Limited resources' means that either there are necessary economic *limits (it would not be profitable to spend more, or other projects need these resources more) or* finite *availability (there is really no more resource available at all).*

To complicate matters further: it could also be the case that some of your competitors are *also* willing to provide your stakeholders with improvements. Maybe, only if you bid the *lowest-cost* solution for the defined system performance levels, will you get any development business whatsoever.

The Relationship between Costs and Performance Delivery

Many, but not all, system performance attributes are directly related to the operational costs of using the system or to the costs of changing the operational system. As examples, think of qualities such as 'Maintainability' and 'Reliability' and workload capacities such as 'Response Time.' To give a specific example, a project might invest some resources to produce a system with a 'higher ease of maintenance.' The resulting system, in *operation*, will have long-term *lower Maintenance Costs* – due to the improved Maintainability attribute level (Scale: <mean time to repair>.) which was the result of a one-off *investment (an implementation cost)*.

Ultimately, every system requirement can be viewed in terms of resources. When making decisions about system changes, a stakeholder is merely exercising choice over where resources are to be expended – by choice (now) or by default (later)!

The key point is that there usually is *choice* about where and when resources are expended. To be competitive, not only must a stakeholder consider if an investment bears a clear relationship to producing the required benefits, they must also be sure that the specified 'required benefits' are the 'correct' objectives and that the selected investment is going to give the best available payoff.

Look at the Use of Resources across the Entire System Lifetime

There is no point in narrow cost control. We need 'value for money' control instead. We must learn to balance the use of resources across the entire system lifetime. To give some examples:

- there is little use in simply controlling an implementation project's financial investment, if the result is excessive operational costs for the resulting system, or excessive system retirement costs
- it is no good constraining the time to market, if the consequence is that the product cannot achieve the necessary performance levels for sales on that market
- there is no point in constraining head count on a project only to experience that the consequence is project delays to market, which threaten profitability.

The design engineer must be able to intelligently trade off and balance, to some reasonable degree, all the many performance and cost requirements. To do this, a full set of requirement specifications is required across the entire system lifetime. Otherwise, any tradeoffs will be carried out without knowledge of the 'full picture,' and short-term priorities will tend to dominate.

Numeric Performance Levels Enable Us to Understand the Associated Costs Better

Numeric performance requirement specification, with sufficient precision for purpose, is necessary in order to be able to calculate the costs of achieving the performance levels with any precision. Conventions such as specifying performance levels as 'low,' 'medium,' 'high' and 'extremely high,' will not allow us to exercise reasonable control over costs. For example, availability levels like 99.90%, 99.98% and 99.998%, which can easily have order-of-magnitude cost differentials to achieve, would be impractical to distinguish amongst by merely using such non-numeric terms.

The Cost of Perfection – Beware Infinite Cost Increases

'Perfect quality' does not seem possible in our world and lifetime. The stakeholder would always *like* to have it (the 'Ideal' level), but cannot *ever* afford it in practice. It seems that the 'cost of perfection' is *infinite*

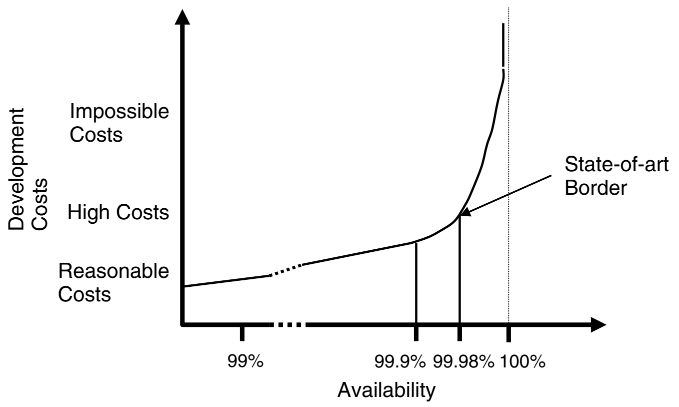


Figure 6.1
As we move any performance level towards nearing perfection, we increase costs dramatically in the direction of infinite costs.

resources. More practically, the costs of performance levels 'nearing perfection' have a nasty tendency to accelerate *towards* infinity. So, as we become more *ambitious* regarding performance, we must become much more *exact* at specifying the performance levels, if we are to hope to understand and control the cost implications.

Further, we must also understand that our systems can be *sensitive* to very small changes in *any* attribute or design specification. These seemingly small changes can give unexpectedly large cost increases, incalculable in advance.

Specify Costs Down to a More Detailed Level – Not Just Total Costs!

We also need to specify the cost requirements in far more detail than people usually do. Not a simplistic 'bottom-line-for-everything' cost budget, but in *detail*. What costs are associated with *every increment* of performance? What costs accompany each increment of function? Estimating and tracking detailed costs will improve our capability of getting feedback early and correcting any situation where the costs are getting 'out of line'. One practical way to view such cost information is by using an Impact Estimation table (IE table) (*see Section 6.8 and Chapter 9, 'Impact Estimation'*).

Accurate Estimation of Costs in Advance is Unlikely for Complex Systems

In advance of building and delivering complex systems (or parts of them), there is no *reliable* way with reasonable *accuracy*, to compute the real, final cost or, to compute the consequential, longer-term cost (Morris 1994). History shows that it is more successful to *stipulate* a reasonable budget amount, and then 'see how much you can get out of it' (MacCormack 2001; Mills 1980). *This means that cost budgets cannot really (and should not) be unilaterally fixed in advance for defined performance requirements.*

Multiple cost budgets and multiple performance goal levels must somehow be set together in some reasonably 'balanced' way. The exact balances amongst them may well be difficult to estimate or know in advance: only the inexperienced believe they can accurately calculate such effects. But we can 'learn as we go' about expected costs in small increments of experience.

Use Design to Cost and Evolutionary Project Management (Evo)

In practice, the best approach to controlling costs for complex systems must be to '*Design to Cost*,' and then to use the Evolutionary Project Management (Evo) method (*see Chapter 10*) and track actual costs.

'Design to Cost' means that you intentionally select designs which *fit within* your committed cost budgets. You may even trade off some marginal performance levels in order to stay within your resource constraints and meet resource targets (budgets). It depends on your priorities. (The alternative is to design for performance alone, and be surprised at the budget overruns!)

Using the Evo method for your project means delivering to your customer or market a succession of improvements in the system's functionality and performance levels. The highest priority improvements must be delivered 'first' (at the earliest opportunities). You must be prepared to learn from the *frequent feedback* from the partial deliveries and to make any *necessary adjustments* in cost budgets. In practice, this is in your interest because, with early warning, you can 'change course' early and so avoid many cost problems.

When, eventually, the budgeted resources do run out – even if you have not delivered all the requirements yet – you can ask, like Oliver Twist, for 'another cup of broth.' If you have been good at delivering value in relation to the resources you have used, then one would expect that your stakeholders would want to keep you in business (the next 'round', at least).

The reader may well find that the ideas of 'Design to Cost,' and of taking an evolutionary approach to costs are strange. But we argue that they are both necessary and possible. Planguage has these approaches in-built in the form of the Impact Estimation and Evolutionary Project Management methods. Even when performance requirements are set at the highest levels, Evolutionary Project Management has a successful past history of being in control of costs and deadlines (for example, the space and military projects in the late 1970s). *(More on this method can be found in Chapter 10, 'Evolutionary Project Management.' More on 'Design to Cost' can be found in Chapter 7, 'The Design Process' and in Chapter 9, 'Impact Estimation.')*

We can control costs if we get early warnings of unexpected costs and we are able to react to these warnings. We must have early, frequent, feedback mechanisms in our planning, our systems engineering and our project management. We can get this degree of control:

- by budgeting resources in *small* (say, 2%) increments
- by *designing* to stay within the budget
- by *reacting to experience* with cost expenditure (changing designs or requirements as far as it is realistic to do so)
- by monitoring a *multiplicity* of resource budgets and a *multiplicity* of performance goals
- by specifying all the *constraints* that apply to the problem, in advance of solving it.

6.2 Practical Example: Resources, Budgets and Costs

Resource Requirement Specifications: Allocation of Resources

We are all familiar with the simplest types of 'resource limitation' specifications: 'the total budget is a million' and 'the deadline is January next year.' There is a real human need for these simple ideas.

However, in order to control and deliver 'within budget', we must take a more sophisticated approach to budget specification. We must, for example, relate resources more carefully to *exactly* what is to be achieved or delivered to stakeholders (the required function and performance attributes), and we must consider the resource constraints. If we fail to do so, then both time and money will run out but we will not have achieved our 'real aims,' which are the function and performance improvements. For example, if only 2% or 20% of the work is accomplished by using 80% of our budget, then we are usually in deep trouble.

Here is an example of a generic financial budget specification, which helps ensure more specific detail:

EXAMPLE

Financial Budget:
Scale: Percentage (%) of total initial Project Money Allocation.
Type: Resource Requirement.
Meter: Project Accounting.
========================= Constraints =========================
Survival [Final Deadline]: 100% "Must not use more than this by final deadline".
Rationale [Survival]: If >100% we have a loss on this project, and it can be deemed a failure.
Fail [Final Deadline]: 90%. Rationale: This gives us 10% profit.
=========================== Targets ===========================
Budget [For each 2% of Total Project Calendar Time, If 2% Benefit]: 2% "of total budget. See Scale above."
2% Benefit: Defined As: At least an incremental 2% of the total of all planned performance improvement ('benefit') shall be delivered.
Rationale [2% Benefit]: This Evo approach will give us consistent control and feedback throughout the project, so we can take action early if necessary, to avoid disaster.
Stretch [Final Deadline]: 80%.
Rationale [Stretch]: This gives us 20% profit. "Double the normal."
Notice the subtle distinction between a Survival level (a hard budget constraint level to avoid unacceptable losses), and a Fail level (a softer budget constraint level to avoid some sort of failure or pain). The Budget is the actual required target budget for some degree of success. The final target set, 'Stretch,' is intended as a motivating cost target. Consider how the resulting 'differentiated' project budget plan will differ from the simplest budget maximum specification.

In the example, we are specifying in the Budget level that for every 2% of our budget we had better not plan to use more than 2% of the calendar time budgeted, and we had better plan to deliver corresponding planned performance improvements in measurable increments. I remind the reader that the previous two chapters tried to introduce the notion that performance measures (such as 2% of any planned performance improvement) *can* be specified and measured.

If you want project control, you will insist on doing things on such a 'pay as you go' basis (or even, 'no cure, no pay'). If you let projects spend money, without demanding clearly measurable results, I promise you 'they' will spend your money, take *your* time and be unable to give you *anything* worthwhile in return. On several occasions, I have investigated very large projects, in the UK, Sweden and Germany, which have managed to consume hundreds of millions of dollars without delivering a single solution of any value to any stakeholder. Take steps to ensure this doesn't happen on your project!

EXAMPLE

Engineering Hours:
Gist: To help ensure resource usage is balanced with the business progress and value by controlling the allocation of engineering hours to different stages and types of work.
Ambition: Low resource-use in beginning, more as value increases.
Type: Resource Requirement [Engineering Work-Hours].
Scale: % of total Engineering Work-Hours allocated.
Budget [Early Pilot Trials]: 10%,
  [Domestic Deliveries to Contracts]: 10% <- Marketing Plan 6.8,
  [From Next Year, Domestic Deliveries, Wholesalers]: 20%,
  [European Deliveries, Contracts [At least 10 signed], If Authority Given]: 30%,
  [European Deliveries, Wholesalers [1 in each country]]: 30% <- The Board.
Authority Given: Authority: Board Approval granted for this budget fraction <- The Board.
An example of allocating a budget. Notice the conditions "[From Next Year], [At least 10 signed], [1 in each country], [If Authority Given]." We could call this a 'conditional budget.'

6.3 Language Core: Resources, Budgets and Costs

Resource Requirement Specification

Resource requirements (Budgets) are specified in a similar manner to performance requirements, because they are *also* scalar requirements (that is, they are variable along a defined scale of measure). See Section 4.3, 'Language Core: Scalar Attributes.'

EXAMPLE

Logic Space:
Type: Resource Requirement.
Scale: Maximum Storage Space in megabytes.
Owner: System Architecture.
Stakeholders: {Architect, Hardware Storage Designer, Handset User}.
Fail [Any One Function]: 100 Mb. "A resource constraint".
Budget [Any One <Frequent> Function]: 50 Mb. "A resource target".

6.4 Rules: Resource Requirement Specification

The rules for scalar requirement specification (Rules.SR) apply (*see Section 4.4*).

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH006.3D – 165 – [165–184/20] 29.6.2005 12:40PM

6.5 Process Description: Resource Requirement Specification

Process: Resource Requirement Specification

Tag: Process.RR.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: A process for specifying resource requirements and for cost estimating, resource budgeting, and project adjustment to stay within budgets.

Note: This process is highly iterative, and needs to be done early and often. It should actually be embedded in the Evolutionary result cycles. It is described here so that the reader sees the multiple elements of determining budgets. This is certainly not a simple procedure within a real project. Budgets will probably need to be estimated and adjusted several times, in the course of attempting to achieve a balance of the performance and function requirements with the resources.

Entry Conditions

E1: The Generic Entry Conditions apply. The specific source documents that should have already exited successfully from Specification Quality Control (SQC) include:

- the current requirements
- the design specifications
- any Impact Estimation tables (giving cost estimates for designs, or for Evo steps).

Note: If any of the source documents has failed to successfully exit SQC, then you can 'stipulate' desired costs, but you do not have a reasonable basis to confidently 'estimate' the costs of the designs/plans, which are needed to deliver the desired functionality and performance levels, on time.

Procedure

P1: **Identify Resource List:** Get existing lists of 'critical resources to be controlled' for this sort of project. These lists should include such things as project elapse times (long and short term), people, people work hours, project space, investments, development costs, production costs and operational costs (including maintenance costs).

P2: **Analyze Benchmark Costs:** Examine earlier similar projects for cost levels, and cost deviations from plans.

P3: **Determine Project Costs:** Determine acceptable and unacceptable cost levels for this project. *Consult any contracts, marketing plans and product plans.*

P4: **Produce Initial Project Budgets:** Specify an initial draft of project resource budgets.

P5: **Perform SQC:** Perform Specification Quality Control (SQC) using Rules.GS, Rules.RS, Rules.SD and Rules.SR. The source documents (process inputs) are listed in the entry condition E1 above. If the specification is not 'clean enough' (the SQC process calculates that there are one or more remaining major defects/page), then return to P1 and cycle through the procedure again as required.

P6: **Carry Out Evo:** Perform an Evo step. (Deliver some results!) Measure *real* costs, for the delivery, versus the budgeted step costs. Replan either costs or other things (such as designs, performance levels, and timing) in order to keep within the project resource requirements. Continue 'cycling' with this step until all the planned Evo steps for the project are completed.

Exit Conditions

X1: The Generic Exit Conditions apply.

The entire process of cost adjustment, and learning, goes on as long as money is still being spent on the project, or spent on the operational system. *'Project end' allows exit from the project. 'System/Product end' (that is, the system or product is no longer sold or distributed) allows exit from the system/product support process. This is a formal way of saying 'this is a continuing process, as long as resource is being consumed.'*

6.6 Principles: Resource Requirements

1. **The Principle of 'Many Critical Risks'**
   There are *many* resource, performance and condition dimensions critical to any system, not just one or a few.
2. **The Principle of 'You Can't Have It All, Trade-offs are a Necessity'**
   Fixing the required level of *one* resource dimension arbitrarily can only be done at the probable expense of *other* attributes.
3. **The Principle of 'You Get What You Pay For'**
   It is really the availability of resources, which limits the levels of performance that can be delivered in practice.

4. **The Principle of 'Attribute Balance'**
   Once you have found a balance between performance and costs, management cannot cut the financial budget, people or time without negative consequences.
5. **The Principle of 'The Cost of Perfection'**
   Perfect quality costs *infinity.*
6. **The Principle of 'The Rolls Royce'**
   Near-perfect performance levels cost more than most people would pay.
7. **The Principle of 'Natural Ambition'**
   The pressure on resources will *always* be at a 'level of discomfort', not to say downright intolerable – this is a natural management strategy to find out how far they can push!
8. **The Principle of 'The Traffic Bottleneck Illusion'**
   Increasing your allocated resources will *not* relieve the pressure on you, but only raise that sponsor's expectations.
   Removing one bottleneck serves mainly to discover others.
9. **The Principle of 'Really Useful Resource Management'**
   The only *practical* way to control costs and performance in large complex dynamic systems is by *early*, *frequent* realistic evolutionary *feedback* on costs, and consequent *adaptation* to realities.
10. **The Principle of 'Shifting Conflicts'**
    Conflicts amongst budget targets, performance targets and design ideas are natural; there's no blame. You just keep resolving them: it's the name of the game.
    Budget constraints will always exist and, will always be subject to change.

6.7 Additional Ideas

Using Impact Estimation and Evolutionary Processes to Balance Requirements

There are two Planguage methods that are worth outlining[1] at this point, because they are fundamental to the control of costs (and also performance).

One is Impact Estimation (IE), which enables *design evaluation* against *multiple* resource budgets and *multiple* performance targets. It produces

[1] See Chapter 9, 'Impact Estimation' and Chapter 10, 'Evolutionary Project Management' for more detail.

Table 6.1 A simple IE table.

Designs-> *Requirements*	*Contract*	*Supplier*	*Motive*	*Architect*	*Parts Used*	*Sum* *% Impacts*
Quality 1	0%	100%	50%	30%	−20%	160%
Quality 2	100%	50%	0%	20%	50%	220%
$Investment Cost	5%	10%	1%	10%	110%	136%
$Operational Cost	5%	50%	20%	1%	10%	86%
Staff Resource	10%	20%	10%	5%	0%	45%
Performance to Cost ratio	*100*/20	*150*/80	*50*/31	*50*/16	*30*/120	

an IE table that provides, amongst other information, performance to cost ratios, which allow relative assessment of the proposed designs.

The other method is Evolutionary Project Management (Evo), which plans and implements design delivery in a sequence of Evo steps. The choice of design for the next Evo step is re-evaluated once the feedback from the implementation of the latest Evo step is received. Evo can use the IE table information to select the design(s) for the next Evo step and to capture the feedback from past Evo steps.

The key point is that these two methods can evaluate *realistic feedback* from *partial implementation* of our designs. We get a *more reliable* picture of the *real* costs of what we are doing, and can then *make adjustments* to *anything* necessary (design, resources, performance levels and/or timing) to achieve the performance-to-cost ratio we are satisfied with.

Table 6.1 shows an example of a simple IE table.

This IE table has three *resource requirements*: $Investment Cost, $Operational Cost and Staff Resource. These are *defined* somewhere else, with a Past (Benchmark) level, which is represented by the 0% level on this table, and a Budget (or other Target) level, which is expressed by the 100% level on this table.

The *referenced* designs (Contract, Supplier, Motive, Architect, Parts Used) are also defined somewhere else with enough detail to permit us to estimate their impact, sufficiently well for our current purposes, on the performance goals (Quality 1 and Quality 2). Interpretations of impact are as follows:

- 0% is *no* change from the benchmark
- 100% reaches the target level on time
- 50% is *halfway* to the target level
- −20% is a '*negative*' impact compared to the benchmark.[2]

[2] For costs this would imply that a design earned resource rather than consumed it. This is not unthinkable.

We should arrive at the estimates of impact based on evidence (such as experiences with the defined design ideas).

Once each design idea has a numeric impact estimate for each performance cell and each cost cell, we can use these cell estimates to calculate a 'performance to cost' ratio. This is the overall ratio of performance delivered with respect to our objectives by the design idea (the sum of Quality 1 and Quality 2), over the sum of the estimated use of resources in relation to the plan by the design idea (the sum of the costs: {$Investment Cost, $Operational Cost, Staff Resource}).

Using a basic IE table, the impact of any design idea on performance with respect to its estimated costs can be evaluated. Design decisions, such as "what happens if we *drop* the design idea, Parts Used?" can also be assessed. Of course, the IE table simplifies, as all models do, but it still gives useful insights.

When there is a sufficient set of design ideas, that is likely to meet the planned levels, on time, with reasonable 'safety factors' (for example, all the 'Sum for Requirement' values are in excess of, say, 200%), then Evo can start to use the IE information in a slightly modified IE Table format to plan the implementation steps of the project.

In simple terms, an Evo plan would sequence the implementation of the design ideas to get the best results (the highest performance-to-cost ratios) delivered to stakeholders early. An IE table can be used after each evolutionary step delivery to capture the numeric feedback from the implementation of any set, or sub-set, of the designs, for any target market of interest *(see Chapter 10, 'Evolutionary Project Management,' for more detail).*

Instead of relying solely on estimates, *real* performance and cost experience is captured step by step and, of course, it can then be compared against the estimates step by step. This feedback on *real* cost and *real* performance levels allows better understanding of the true *future* cost levels, at an early stage of the project. This leads to better control over costs, system performance, design and projects. The heart of good project management is such multidimensional, numeric feedback and consequent improvement in plans.

6.8 Real Example: Resource Target and Resource Constraint Specification

Here is an example of a resource requirement specification, which includes some resource constraints. It also includes price specification. It is based on a real case study, but edited for confidentiality and to reflect the latest Planguage terminology.

EXAMPLE

Installation Time:
Ambition: Installation time must not be more than that of an unlicensed system <- RSW 3.
Type: Resource Requirement.
Installation Effort: Scale: Work Hours.
Budget [USA]: 15 <- Requirement Specification, Feb 5.
Installation Duration: Scale: Calendar Days.
Budget [USA]: 2.5 <- Requirement Specification, Feb 5.
Installation Costs:
Scale: Total Installation Cost of all Involved Parties.
Type: Resource Requirement.
Total Installation Cost: Defined As: Financial Cost of {Education of Customer People, Involvement during Installation of Customer People, Involvement during Planning of Customer People, Loss of Service in a PBX, Special Tools for Strange Cabling, any other thing even if not on this list!}
Past [DECT, USA, Last Year]: <not known exactly>.
Fail [Per Installation, USA, Release 1]: Maximum of twice DECT Installation Costs. "A constraint."
Budget [Per Installation, USA, Release 1]: Within ±20% of DECT Installation Costs. "A Target."
Per User Price:
Note: The actual price targets may vary from time to time and market to market.
Type: Performance Constraint.
Note: this is NOT a budget for the project or the Base Station system. This is a result of the design of the new system.
Scale: $ Per User Price for defined [Number of System Users] to use at a defined [Location] for defined [Release] of total Base Station {CE and RH}.
Past [Last Model]: $1,000.
Fail [30 to 250 System Users, USA, Release 1]: $700 or more <- RSW 2.
Survival [More than 250 System Users Or Larger Building Or tougher than Normal Radio, USA, Release 1]: $700 or more. <- RSW 2.
Subscriber Cost:
Type: Performance Constraint.
Note: The actual customer cost targets may vary from time to time and market to market.
Scale: $ Cost for a defined [Number of Users] of System per Subscriber, including TK and SW licenses cost to TeleCo.
Past: $600.
Fail [100 Users, USA, Release 1]: $400 or more <- RSW 2, Cost Assumptions (Page 2).
This is a real example, but not in its final form: it is only the first draft translation of a customer's older, non-Planguage specification. It is also upgraded with recent Planguage changes.

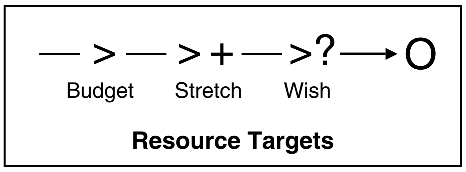
//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH006.3D – 165 – [165–184/20] 29.6.2005
12:40PM

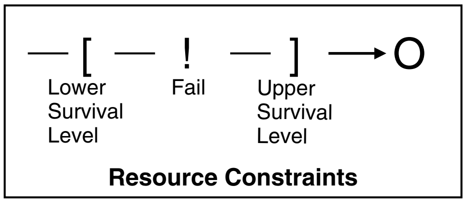
6.9 Diagrams/Icons: Resource Requirement Specification

Resource Requirement Icons

Resource target and constraint icons are scalar icons, identical to those used for performance attributes.

Resource targets specify how much we would 'like' to use of a resource. There are three types of resource target: Budget (>), Stretch (>+) and Wish (>?).





A Resource constraint is defined using a Fail concept (!) or a Survival concept: the '[' is a lower limit and the ']' is an upper limit.

Resource constraints set (relatively) strong framework limits to the use of resources. These strong constraints could be due to legal restrictions, contract limits or other sources, which are relatively inflexible. They are generally outside our control. Constraints are not so easily the subject of tradeoff decisions, as targets might well be.

Resource Requirement Specification Template

The scalar requirement template given in Section 4.9 should be used for resource requirement specification.

6.10 Summary: Resource Requirement Specification

There are many limited resources we must track for building, modifying and operating a system. Budget specifications will include calendar time, people effort, and money to implement, operate and service the system.

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH006.3D – 165 – [165–184/20] 29.6.2005
12:40PM

'Costs' is our term for 'use of resources': resources that are generally in demand for satisfying *other* priorities. Failure to think and document clearly with regard to resources is likely to lead to resource scarcity problems.

We assume that most systems that the reader is likely to use the Planguage methods on are non-trivial and difficult to manage. They are of such a nature that they are very difficult to *predict* costs for, and almost as difficult to *control* the costs of. Planguage addresses these problems in several ways:

- Planguage ensures specification of resource requirements is performed in a disciplined and detailed *numeric* way.
- Through Impact Estimation (IE), Planguage obtains *tightly integrated performance and cost information.* Not just the total final budgets, but *detailed budget allocation* at design idea level and at evolutionary step level, which is linked to the evolution of the stakeholder valued results! Such resource requirement specification information gives a better ability to predict costs in advance. Such resource budgeting is also important to ensure engineers do 'Design to Cost' from the earliest stages. It helps them keep aware that they do have finite limits for resources. It is otherwise too easy for them to focus on performance and technology; leaving serious cost considerations until too late.
- Through Evolutionary Project Management, Planguage provides better cost-expenditure control, because we have a way of *adjusting cost budgets and estimates for resource usage, as we learn, early and frequently, from practical experience.* Alternatively we can get resource control, because we can choose '*tradeoffs*' in order to maintain the budgets we *initially* planned for. 'Tradeoff' means that we can adjust certain performance levels and/or adjust certain design specifications. We can also adjust certain qualifiers [when, where, if]. With Planguage, we can more clearly, and earlier, see the exact options available, and make more intelligent tradeoff decisions.

The fundamental assumption of the Planguage method is that we must set things up to learn (this is Shewhart's Plan-Do-Study-Act cycle) as rapidly as possible, before we fail, and before our competitors do things better and 'put us out of business'. The threat of losing your workplace and budget to 'competition' applies even if you are a government agency or a charity!

By use of Planguage practices, the all-too-common project syndromes of 'running out of resource (time or money) without delivering any value' and 'pushing the system out of the door on the deadline; system performance be damned' ought to be eliminated for good! This is more than an optimistic hope. It has been done.

The real price of everything, what everything really costs to the man who wants to acquire it, is the toil and trouble of acquiring it.

Adam Smith (1723–90) Scottish economist,
The Wealth of Nations *(1776)*

Overview of Planguage Methods for Controlling Costs

The prerequisites for effective control over a project are tight integration of cost and performance considerations, 'design to cost' and using feedback on actual costs to modify plans. Planguage methods ensure these prerequisites by demanding:

- detailed, numeric, measurable performance specifications that adequately capture the performance requirements: the qualities (stakeholder-related objectives) as well as the workload capacities and resource savings (the resource-related objectives)
- resource requirement specifications for the resources allocated, and for any known restrictions on resource expenditure
- design specifications with detailed expected cost *and* performance attributes of the design
- impact estimates of the abilities of the various designs to meet both the performance goals and the resource budgets
- selection of evolutionary steps according to their stakeholder value, and their performance to cost ratios
- feedback from live systems of the actual progress towards achieving the performance levels, and the actual resource expenditure after implementing each evolutionary step
- action being taken on the feedback to adjust specifications, or the future evolutionary steps, to ensure realistic plans (revision of budgets or tradeoffs).

A Proposed Resource Requirement Specification Policy

1. **Define Resource Requirements Thoroughly**: In requirement specifications, all potentially critical resources shall be specified as budgets in a *well-defined, thorough* manner.
2. **Specify the Performance and Cost Relationship**: The level of both resource budget and performance goal detail shall be sufficient to enable us to understand the benefit, in relation to resources, of *incremental performance improvements*.
3. **Make All Cost Requirements Visible**: We must be able to 'see' all opportunities to reduce costs by investment in better system design. The budgets must specifically incorporate ongoing *operational costs* requirements (that is, the resources required for such things as installation, adaptation, porting, maintenance, recovery, auditing, servicing and/or customer help lines) so these can *compete for priority* with *short-term investment* costs.

4. **Plan for the Long Term**: All budgets shall consider the *total* lifetime of system perspectives. This specifically includes *long-term* considerations (such as costs of system retirement, pollution and accidents).
5. **Designs Shall Be Cost Estimated for Impact on All Critical Resources**: Costs shall be estimated for all critical and budgeted multiple resource factors for *every discrete design idea* using Impact Estimation Tables.
6. **Let Value Decide the Costs**: Value delivered in relation to costs, not 'resources consumed' alone, should dictate expenditure. If designs provide the opportunity for excellent required 'payback', then we should automatically spend more, and vice versa. *(Given that budgets are formulated in 'performance to cost' terms, and we have Evolutionary feedback, the levels of risk should be under acceptable control.)*
7. **Document Supporting Information**: When defining cost requirements, *full documentation* shall be given about assumptions, benchmarks, risks, uncertainties, ranges, authorities, sources and other related facts so as to give us the best possible background for rapid, confident, independent decision-making by the systems engineers and managers.
8. **Justify Estimates and Perform Specification Quality Control**: When making estimates, the full array of *evidence and sources* of the evidence shall be documented. Worst-case scenarios shall be given explicitly. The estimations shall undergo *Specification Quality Control* (SQC).
9. **Track Costs Early, During Implementation**: Costs shall be tracked and analyzed *at every evolutionary step* of development, so as to learn of problems as early as possible, and take corrective action.

This policy above captures many of the key points discussed in this chapter about Resource Requirements. Note: this policy should also be supported by specification rules to enable the Specification Quality Control (SQC) of Resource Requirements and Cost Estimates.