Chapter
4

PERFORMANCE

How Good?

GLOSSARY CONCEPTS

- Performance
- Quality
- Resource Saving
- Workload Capacity
- Scale
- Meter
- Benchmark
- Past
- Record
- Trend
- Target
- Goal/Budget
- Stretch
- Wish
- Constraint
- Fail
- Survival

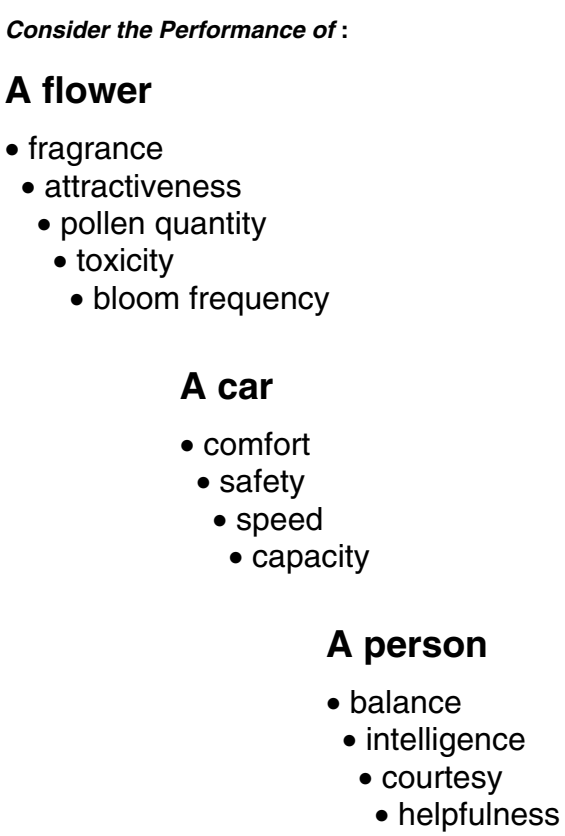


Figure 4.1
Some examples of performance concepts.

4.1 Introduction

Performance: Quality, Resource Savings and Workload Capacity

Performance describes the system benefits: how good the system is and how it affects the external world. Performance attributes state the actual and/or potential benefits and effects experienced by stakeholders in their environments.

Performance attributes are the output attributes; they state the effectiveness of a system. By contrast, the input (or 'fuel') attributes are the resources/costs of developing and/or maintaining a system that exhibits

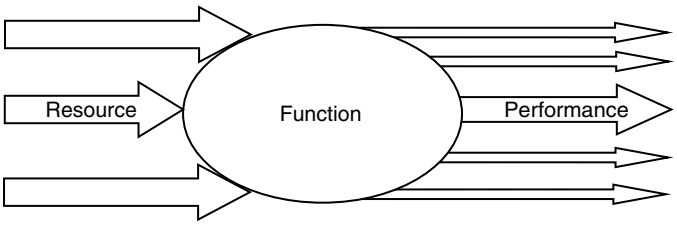


Figure 4.2
A simple system representation. It consists of a function, and its performance and resource (cost) attributes.

those performance attributes. The performance to cost ratio for a system is a measure of its efficiency.

Performance attributes are scalar. As discussed in Chapter 2, there are three basic types:

- **Quality**: The quality attributes specify *how well* the system performs. The term 'quality' is used here in the ordinary widest sense of the word. It is by no means limited to the narrow 'defect free' notion that some people mean when using it. How many qualities can you list of a great car, a dream house, an excellent employee, a great personal computer or a good wine? We include all such ideas in our broad concept of quality.
- **Resource Savings**: These specify *how much* resource is required to be saved compared to the resource usage/consumption by some reference or benchmark system. Achieving specific savings is frequently a driver for system development; for example, cutting the financial cost of carrying out transactions or reducing the time taken to carry out a task. Note, for this performance type, the *saving* of resource is the prime requirement, it is not simply a fortuitous by-product of the system improvements; it is what a stakeholder has specifically demanded.
- **Workload Capacity**: These specify *how much* work the system can perform: the capacity of the system. For example, the average speed for completing certain tasks, the capacity to store information and the maximum number of users supported.

Performance requirements must express *quantitatively* the stakeholders' requirements. I have come to believe, through experience, that all the performance attributes we want to control in real systems are capable of being expressed measurably. I find it intolerable that critical performance ideas are expressed in mere non-quantified words. Expressions like "vastly increased productivity" annoy me! Not one of those three words has a precise and agreed unambiguous interpretation. Yet, I have consistently encountered a world in multinational high-tech companies, amongst educated, intelligent and experienced people, where such vague expressions of performance, especially of

quality, are tolerated; such expressions seem not even recognized as being dangerous and capable of improvement.

Performance attributes are more than a collection of names like 'reliability,' 'user friendliness,' 'innovation,' 'transaction time' and 'cost saving.' Each performance attribute needs to be precisely defined by a set of numeric, measurable, testable specifications. Each performance attribute specification will include different specified levels for different conditions [time, place and event]. Unless there is clear communication in terms of numeric requirements, there is every chance of the real requirements not being met; and we have no clear indication of the criteria for success and failure.

Sometimes, it seems difficult to identify satisfactory scales of measure. Often, the only ones that can be found are indirect, imprecise and have other problems associated with them. From my point of view, these problems can be tolerated. The specific scales of measure, and meters for measuring, can always be worked on and improved over time. In all cases, an attempt at quantified specification is better than vague words.

Over the years, I have found people immediately receptive to the idea that they should quantify all their performance ideas. The only question has been "How?" This chapter begins to answer this question. It describes the main Planguage parameters you can use to specify quantified performance attributes.

4.2 Practical Example: Performance Requirements

Let us start with an example of how to quantify a typical performance requirement.

"Increased ease of service" is a term I found in a set of design specifications for a mobile phone handset. It was not defined further. It sounded like a nice thing to have. The telecommunications supplier's culture allowed it to go unchallenged. In fact, in the few dozen pages of specification, there were actually 10 distinct design ideas, each one with a bullet-point claiming it would contribute to "better serviceability" for the new phone.

I asked my client if they had any quantified performance requirement specification for the "increased ease of service" for the phone. They had not, so we agreed to explore some possible definitions. We soon discovered that 'Serviceability' for the mobile phone had numerous elementary components; it was a *complex* objective.

Here is an extract of what we did. It was a three-step process.

Step 1

We identified the different components of Serviceability and gave each of them a name:

Serviceability: ''is comprised of the following elementary and complex objectives.''
Type: Quality Requirement:
{Repair,
Enhancement,
Fashion Changes,
Installation,
Reconfiguration}.

Step 2

We described each of these objectives by defining and agreeing a Gist ('Gist' meaning the essence or main point):

Repair: Gist: Speed of repair of faults under given conditions.
Enhancement: Gist: Speed of introducing hardware or software enhancements.
Fashion Changes: Gist: Ease of customer varying fashion attachments.
Installation: Gist: Speed of installing telephone in defined settings (for example, in an automobile).
Reconfiguration: Gist: Work-hours to modify for varied uses (for example, coupling to personal computer or power supplies).

In fact, we then further decomposed these into a total of 18 elementary objectives. However, such detail is not required for this example!

Step 3

Once we were satisfied that we had captured the scope of Serviceability, we added further definition to specify the requirement in measurable and testable terms: we identified a Scale and a practical way of measuring on that Scale (a Meter). For example:

Repair:
Ambition: Improve the speed of repair of faults substantially, under given conditions.
Scale: Hours to repair or replace, from fault occurrence to when customer can use faultlessly, where they intended.
Meter [Product Acceptance]: A formal Field Test with at least 20 representative cases, [Field Audit]: Unannounced Field Test at random.
========== Benchmarks ==========
Past [Product = Phone XYZ, Home Market, Qualified Dealer Shop]: {0.1 hours at Qualified Dealer Shop + 0.9 hours for the Customer to transit to/from Qualified Dealer Shop}.

Record [Competitor Product XX]: 0.5 hours average. "Because they drive a spare to the customer office."
Trend [USA Market, Large Corporate Users]: 0.3 hours. "As on-site spares for large customers."
========== Targets ==========
Goal [Next New Product Release, Urban Areas, Personal Users]: 0.8 hours in total, [Next New Product Release, USA Market, Large Corporate Users]: 0.2 hours <- Marketing Requirement, February 3 This Year.
========== Constraints ==========
Fail [Next New Product Release, Large Corporate Users]: 0.5 hours or worse on average <- Marketing Requirement, February 3 This Year.
Survival [Next New Product Release, All Markets]: 1.0 hours <- TG.

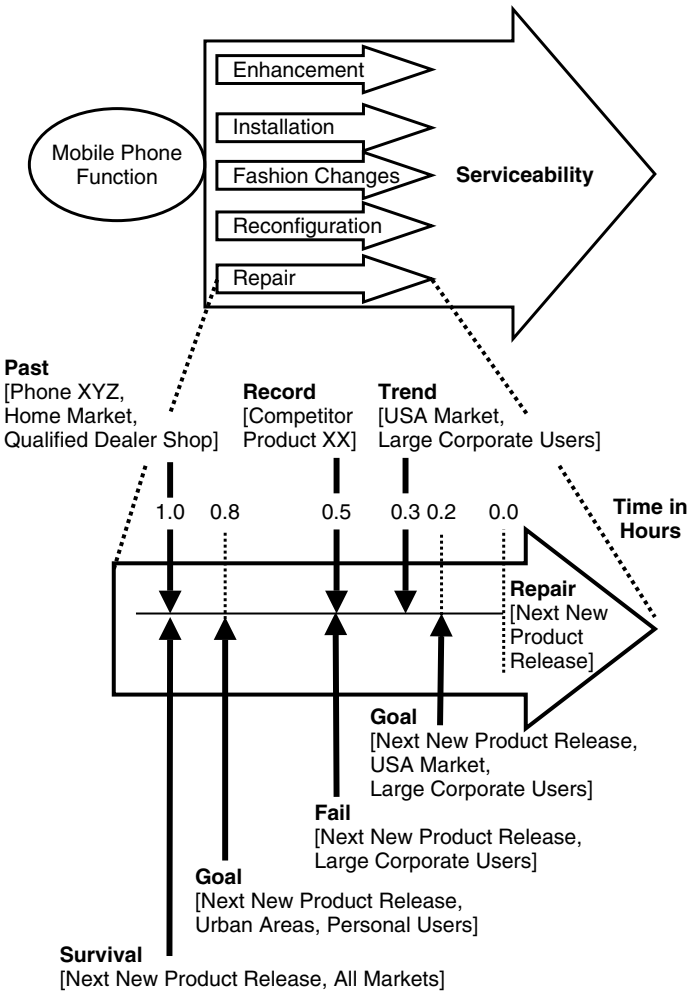


Figure 4.3
Serviceability, a complex performance requirement, decomposes into numerous performance attributes. One of these, the quality, Repair is expanded to show its targets and constraints and, the supporting benchmark information.

At this point, everyone realized that we needed to do some 'serious' work defining our Serviceability objective. Only with an improved requirement specification, could we begin to evaluate our ten specified design ideas that claimed ''increased ease of service''!

Whoever had originally written the phrase ''increased ease of service'' had failed to communicate a precise, unambiguous requirement. Of considerable concern, there was clearly no means of agreeing the specific requirement level with other stakeholders. Nor was there any means of verifying we had met the requirement on delivery. In practice, the engineers were designing the phone without any significant input from Marketing.

These same problems, of 'lack of clarity' and 'lack of necessary detail', also occur elsewhere. In *your* business too!

4.3 Language Core: Scalar Attributes

All performance and resource/cost attributes are scalar. The Planguage parameters used for specifying scalar attributes[1] include:

- Ambition
- Scale
- Meter
- Past
- Record
- Trend
- Goal (abbreviation for 'Planned Goal' for performance attributes)
- Budget (abbreviation for 'Planned Budget' for resource attributes)
- Stretch (abbreviation for 'Stretch Goal' or 'Stretch Budget')
- Wish (abbreviation for 'Wish Goal' or 'Wish Budget')
- Fail
- Survival (abbreviation for 'Survival Limit').

Each scalar attribute must be specified using a tag, and an appropriate set of these parameters. Past, Record and Trend are used to specify *benchmarks*, Goal or Budget,[2] Stretch and Wish are used to specify *targets*, and Fail and Survival are used to specify *constraints*.

---

[1] For the sake of simplicity, only the abbreviations for these parameters tend to be used in the main text of this book. For example, where 'Stretch' is used, distinction is not made between 'Stretch Goal' and 'Stretch Budget' as it is evident from the context whether you are specifying a goal or a budget.

[2] In the past, 'Plan' was used instead of 'Goal' and 'Budget.' Use of 'Plan' is still perfectly acceptable if it better suits your organizational culture. 'Plan' does have the advantage of being simpler and of better conveying to people that it is a level that is subject to stakeholder acceptance and negotiation.

The numeric values of the target, constraint and benchmark parameters define the attribute levels. Note that benchmarks refer to *existing* or *past* values, or future estimates extrapolated from *past* values, whereas targets and constraints are *future requirement* values.

Here is some further detail about the main specification concepts:

Ambition

This is a descriptive parameter used to express the level of expected performance in words.

Scale

The heart of a scalar attribute specification is the 'scale of measure' parameter, Scale (sometimes also known as the 'units of measure' parameter). The Scale states the specific definition of a scalar attribute that we intend to use. It defines the units for measurement (for example, 'bits per second,' 'miles per hour,' 'mean time between failure' and 'number of new customer contracts per year').

While suitable scales of measure for resources/costs are relatively obvious to most people, identifying suitable Scales for performance attributes, especially for qualities, is more challenging. There are, as yet, few widely recognized standardized Scale definitions available. (*See further discussion in the next chapter, 'Scales of Measure'.*)

Scale: A scale of measure, stating the units to be used for numerically expressing a scalar attribute level.

Meter

The Scale parameter is supported by the Meter parameter, which defines, references or outlines a practical and economic method for actually carrying out the measurement of the numeric level of the attribute in a real system. More than one Meter may be required if the means of measuring is going to alter over time or vary according to the place and conditions.

Meter: A practical method for measuring and testing a scalar attribute level, on a defined Scale.

Benchmarks

The benchmark parameters, Past, Record and Trend are used for specifying historical, current or extrapolated performance levels. It is

important that we understand what our own existing systems, our competitors' systems and, more generally, any other relevant systems, are achieving.

Past: A relevant benchmark level worth consideration that has already been achieved in some defined [time, place, event] conditions. We are interested in the levels achieved by any relevant existing system (our own, competitive, or any other system).

Record: A Past, which is the best-known result; a state-of-the-art numeric value.

Trend: An extrapolation of past data, trends and emerging technology to a defined [time, place, event] conditions. Aside from our own project's plans to improve this level, what future levels are likely to be achieved by others? What will we be competing with?

Targets

The target parameters, Goal or Budget, Stretch and Wish, define the attribute levels for success, motivation and dreams respectively.

Goal: A future required level under defined [time, place, event] conditions, which has to be achieved to claim success in meeting a performance attribute requirement. Goal levels are also a signal to stop investing in levels better than this one, as the value gained is most likely insufficient to justify additional costs.

Budget: A future required level under defined [time, place, event] conditions, which has to be achieved to claim success in staying within a resource attribute requirement. A signal to worry about resource usage when more resources are estimated to be needed, or are really used, than this 'acceptable' level of cost.

Stretch: A future desired level, under defined [time, place, event] conditions, which is designed to challenge people to exceed Goal levels or use less than Budget levels.

Wish: A future desired level, currently a 'dream' target level, under defined [time, place, event] conditions, which has some value to a stakeholder. The requirement is not planned or promised, due to technical or cost reasons, but it is recorded and kept in the requirement database (even if not acceptable now) so that it can be borne in mind.

Constraints

The constraint parameters, Fail and Survival, state the levels for some kind of system failure and system survival respectively.

Fail: A future required level under defined [time, place, event] conditions, that is necessary to avoid some kind and degree of system failure, while still allowing the system to operate.

Survival: A future required level under defined [time, place, event] conditions, which is necessary to avoid total system demise. In other words, it is a level necessary for the survival or viable operation of the system.

Conditions

It is also important to distinguish amongst the different numeric levels required for different conditions. Different times, different places and different events can all give rise to requirements for different attribute levels. Qualifiers should be used to specify the qualifying conditions for the different specified levels. Each of the above Planguage parameters will normally contain a qualifier and/or be within the scope of a qualifier defined elsewhere that applies to this specification. A qualifier can be used after almost any tag or parameter to be specific about dates, markets, products, components, stakeholders and other conditions. (*See also further discussion of qualifiers in Section 2.7.*)

[Time, Place, Event] or [When, Where, If]: Qualifiers specify the conditions for a specification being valid, in force or effective. All conditions must be 'true' for the associated specification to be valid.

Here is an example to illustrate the parameters just defined:

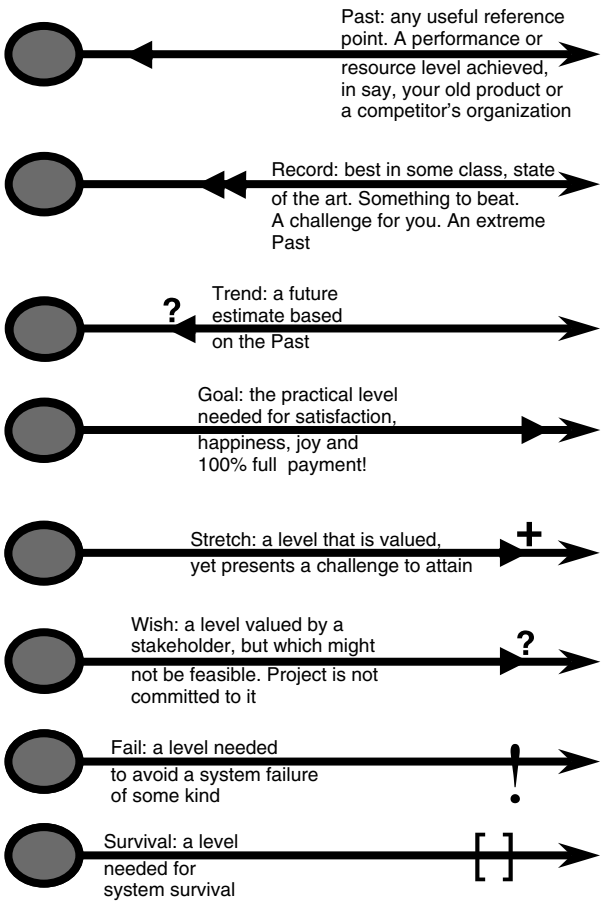
EXAMPLE

Usability [New Product Line, Major Markets]:
Ambition: To achieve a low average consumer time to learn to use our telephone answerer under stated conditions.
Scale: Average number of minutes for defined [Representative Consumers and all their household family members over 5 years old] to learn to use defined [Basic Daily Use Functions] correctly.
Meter [Product Acceptance]: A formal Field Test with at least 20 representative cases, [Field Audit]: Unannounced Field Test at random.
Past [Product XYZ, Home Market, People between 30 and 40 years old, in homes in Urban Areas, <for one explanation & demonstration>]: 10 minutes.
Record [Competitor Product XX, Field Trials]: <5 minutes?> <- one single case reported,
[USA Market, S Corporation]: 10 seconds <- Public Market Intelligence Report.
Fail [Next New Product Release, Children over 10]: 5 minutes <- Marketing Requirements February 3 Last Year.
Goal [Next New Product Release, Urban Areas, Personal Buyers]: 5 minutes, [Next New Product Release, USA Market, Large Corporate Users]: 5 minutes <- Marketing Requirements February 3 Last Year.
Stretch [Next Year]: (Record [USA Market] – 10%).

Table 4.1 Basic Planguage parameters for scalar attribute specification. See also Table 1.1 for the additional basic generic Planguage parameters.

Planguage Parameters for Scalar Attributes			
Scalar Attribute Parameter	*Meaning*	*Used for*	*Note also*
Scale:	Definition of the scale or units of measure	Defining the variable varying performance or cost concept with precision and clarity	Contractual use Icon: -|-|-
Meter:	Definition of how we are going to measure or test the level of this attribute	Determining the real world numeric levels in practice	Contractual use Icon: -|?| -
Past:	A known measured benchmark of an interesting past or current level	Providing a baseline attribute level	A useful reference point A benchmark Icon: <
Record:	A 'state of the art' level	If a Goal or Budget is near to or better than the Record, then a warning of extra risk and cost is implied	A useful reference point A benchmark Icon: <<
Trend:	From extrapolation of existing data, an estimated future level	A cross-check that the Goal or Budget level is ambitious/ competitive enough	A useful reference point A benchmark Icon: ?<
For performance attributes, Goal: *For resource attributes*, Budget:	A future, target requirement level, to be met or bettered for *success* and stakeholder *satisfaction*	Understanding the future requirement level. *Knowing when to stop* designing, investing and developing	A contractual *full* payment level A target Icon: >
Stretch:	An interesting, but difficult to attain, target level	To motivate teams to do better than currently considered practical or economic	No contractual commitment A target to strive towards as a challenge Icon: >+
Wish:	A desired level, dreamed of by some stakeholder, even if unrealistic	Better understanding of the stakeholder needs. Potential requirement direction when feasible later	No contractual commitment whatsoever – completely unbudgeted A stakeholder 'dream' to bear in mind Icon: >?
Fail:	A future requirement level, which is necessary to avoid *some sort* of system failure. A constraint	Stating a minimum requirement for delivery levels Understanding the minimum levels for any tradeoffs	Contractual use as a *minimum* payment level Icon: >>
Survival:*	A future requirement level, which is necessary to avoid *total* system failure. A constraint	Stating an absolute minimum requirement for any valid delivery or operation Failure to meet a Survival level means total system failure	Contractual use as a 'non-payment' level Icon: [ ] *Note this icon is deliberately similar to that used for qualifiers*

Note: * *The Catastrophe parameter is an alternative to the Survival parameter. See the Glossary. 'Survival' is used in the text of this book.*



Note: This diagram applies to performance attributes and shows performance scale arrows. With a change to show scalar resource arrows, all the parameters also apply to resource attributes, apart from the Goal parameter, which is replaced by Budget for resource attributes.

Figure 4.4
Performance benchmarks, targets and constraints.

Note, terms defined by the project such as 'Major Markets' are capitalized to indicate that they are already, or will be shortly, more formally defined elsewhere. They will not necessarily be defined in these textbook examples. For example,

Major Markets: Defined As: {USA, Japan, Europe, India}.

There are other additional parameters that can be used to describe a scalar requirement. Some of these are shown in Figure 4.12, Scalar Requirement Template.

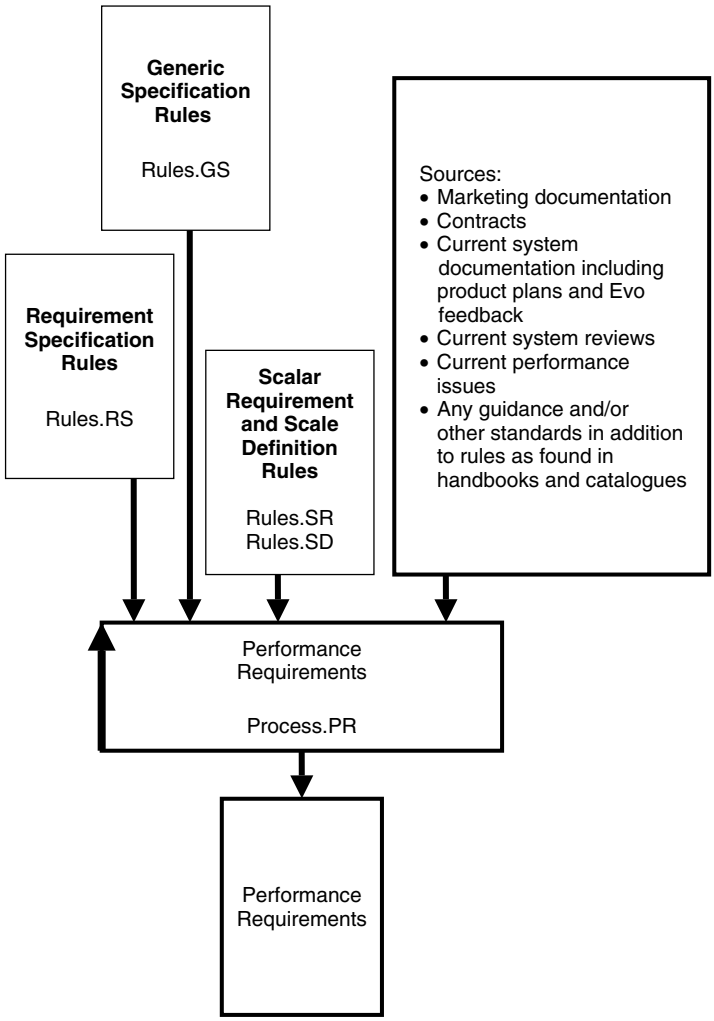


Figure 4.5
Performance requirements are subject to at least three types of rules for specification. These rules should be used in the SQC of performance requirements.

4.4 Rules: Scalar Requirements

Tag: Rules.SR.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: Rules for Scalar Requirement Specification.

Note: These rules apply to both performance requirement specification and to resource requirement specification.

Base: The generic rules for specification (Rules.GS), the rules for requirement specification (Rules.RS) and the specific rules for scale definition (Rules.SD) apply.

R1: **Completeness**: All scalar attributes, that are arguably critical to success or failure, shall be identified, specified and thoroughly defined.

R2: **Explode**: Where appropriate, a complex scalar requirement shall be specified in detail using a set of complex and/or elementary scalar attributes.

Note: In addition to detailing by means of elementary specifications, you can continue decomposing scalar specifications by using sets of [qualifiers].

R3: **Scale**: All elementary scalar attributes must define a single numeric Scale, fully and unambiguously, or reference such a definition.

R4: **Meter**: A *practical and economic* Meter, or set of Meters, shall be specified for tracking levels on each Scale. A reference to a full definition or standard measuring process for all identified Meters must be given. As an initial minimum for a new Meter, an outline of the Meter measuring process is permissible.

R5: **Benchmark**: A reasonable attempt shall be made to specify benchmarks {Past, Record, Trend} for our current system, and for relevant competitive systems. Explicit acknowledgement must be made where there is no known benchmark information.

R6: **Requirement**: At least one target level {Goal or Budget, Stretch, Wish} or Constraint {Fail, Survival} must be stated for a scalar attribute specification to classify as a *requirement* specification. *A specification with only benchmarks is an analytical specification, but not a requirement of any kind.*

R7: **Goal or Budget**: The numeric levels needed to meet requirements fully (and so achieve success) must be specified. In other words, one or more [qualifier defined] Goal or Budget targets must be specified. *The need for target levels to specifically cover all short term, long term and special cases must be considered.*

R8: **Stretch**: When you want to indicate an engineering challenge, in order to motivate design engineers to find designs to achieve better-than-expected levels, specify a 'Stretch' target (*using a Stretch parameter*). You should also include information about the benefits of reaching this target (*using Rationale*).

R9: **Wish**: Any known stakeholder wish level (a level that has some value to a stakeholder, but only a level to be dreamed of, it is an

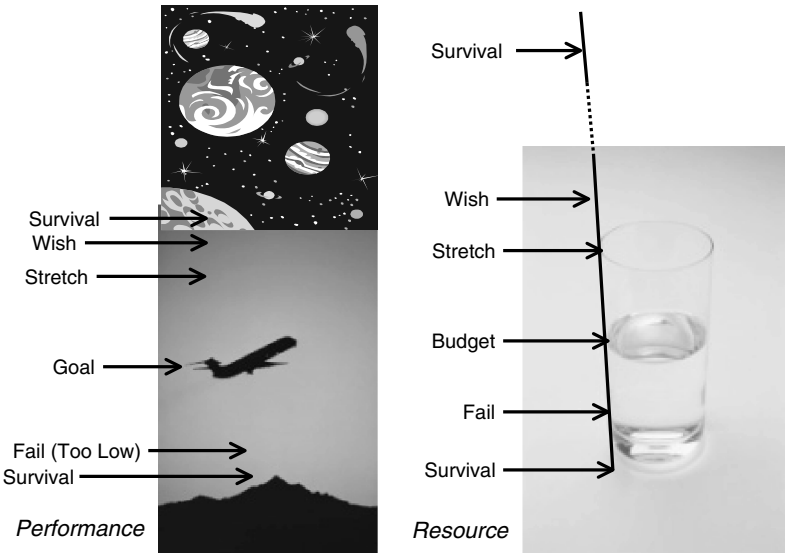


Figure 4.6
How the scalar requirement parameters can be used to describe real-world situations.

uncommitted level) shall be captured in a 'Wish' statement *(with Rationale). Even if the Wish level cannot realistically yet be converted into a practical target level, it is valuable competitive marketing information and may allow us to better satisfy the stakeholder at some future point.*

R10: **Fail**: Any known numeric levels to *avoid system, political, legal, social, or economic loss or pain* must be specified. In other words, one or more [qualifier defined] Fail constraints must be specified. *Several Fail levels may be useful for a variety of short term, long term, and special situations.*

R11: **Survival**: The numeric levels to avoid complete system failure (a totally unusable or unrecoverable system) must be specified. In other words, any [qualifier defined] constraint levels at which system survival is completely at risk must be identified, using Survival parameters.

4.5 Process Description: Performance Requirements

Process: Performance Requirements.

Tag: Process.PR.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Entry Conditions

E1: The Generic Entry Conditions apply. The list of valid source documents could include marketing documentation, contracts, current system documentation, current system reviews, any lists of system performance issues and any initial design specifications. It specifically includes any documentation of standards that applies, such as handbooks and catalogues. The required specification standards include {Rules.GS, Rules.RS, Rules.SR and Rules.SD}.

Procedure

P1: Scan all input (source) documents for implied (for example, via design specifications) or *explicitly expressed* performance requirements. Build a list of performance requirements categorized by stakeholder type.

P2: Next, scan all input (source) documents (including any design documents and strategic plans) for design ideas. Mark the design ideas as requirements, ONLY if they are intentional design constraints (*as they are then true requirements*). Otherwise, if they are not constraints, determine and specify the possible performance requirements that led to these design ideas being specified. Add these 'implied' performance requirements to the overall list of requirements. *You can keep the design ideas, separately, for design phases. But get them out of the real requirements. You might well cross-reference the implied requirements (Impacts: <Requirement X>.) and design suggestions (Is Supported By: <Design Idea Y>.) for future understanding of why they are there.*

P3: Using the P1 lists (*explicit requirements*) and P2 lists (*requirements derived from design ideas*), establish a comprehensive list of candidate performance requirements. Specify at least Tag, and possibly a Gist or Ambition. Include cross-reference to any Sources (<-), Assumptions, Dependencies and Risks you can determine.

P4: Check handbooks, catalogues and lists of standard performance requirements for ideas of additional performance requirements, which you should consider. *Remember that you need to specify things that are currently taken for granted because they are not problems in any of your current products or systems. We have to keep our system healthy in the future, consciously!*

P5: Consider the total stakeholder environment. This involves not just your one or two users and customers, but more likely, your ten or more internal and external stakeholders, such as help desk, installers, politicians, marketing and customer trainers. Using the input documents, brainstorm to determine each stakeholder's critical qualitative attributes. Ensure that the critical performance attributes are identified on your requirements' list. Interview the stakeholders to get

feedback and confirmation about your specification. Add to the list of requirements or modify them as necessary.

P6: For each identified performance requirement, specify it in detail using the rules that apply (Rules.GS, Rules.RS, Rules.SR and Rules.SD). Ensure each performance attribute is measurable in practice.

P7: Consider which performance requirements are key, and must therefore be controlled. Identify the most important 'top ten' performance requirements. Group the others as 'Diverse' or 'Less Critical' if you like.

P8: Perform Specification Quality Control (SQC) on the performance requirements. Correct any identified defects, and calculate the remaining major defects/page (a page being 300 words of non-commentary text). Check against the rules: {Rules.GS, Rules.RS, Rules.SR and Rules.SD}.

Exit Conditions

X1: The Generic Exit Conditions apply. The maximum possibly remaining major defects/page must be less than one.

X2: The Requirement Specification Owner (usually specified as 'Owner: <name, e-mail address or department>.') agrees to release the performance requirement specification with *their* name on it. They have veto on release.

4.6 Principles: Performance Requirements

1. **The Principle of 'Bad numbers beat good words'**
   Poor quantification is more useful than no quantification; at least it can be improved systematically.
2. **The Principle of 'Performance quantification'**
   All performance attributes can be expressed quantitatively, '*qualitative*' does not mean unquantifiable.
3. **The Principle of 'Threats are measurable'**
   If the lack of a performance attribute can destroy your project, then you can measure it sometime; the only issue will be "how early?"
4. **The Principle of 'Put up or shut up'**
   There is no point in demanding a performance requirement, if you cannot pay or wait for it.
5. **The Principle of 'Deadline or die'**
   There is no point in demanding a performance requirement, if you would always give priority to something else, for example, a deadline.

6. **The Principle of 'Dream, but don't hold your breath'**
   There is no point in demanding a performance requirement, if it is outside the state of your art.

7. **The Principle of 'Benchmarks and targets'**
   Numeric past 'history' levels and numeric future requirement levels *together complete* the performance requirement definition of relative terms like 'improved'.

8. **The Principle of 'Scalar priority'**
   In practice, the first priority will be survival,
   The second priority will be avoiding failure,
   The third priority will be success,
   And the required levels for all of these will be constantly changing.

9. **The Principle of 'Many-splendored things'**
   Most performance ideas are usefully described by *several* measures of goodness.

10. **The Principle of 'Limits to detail'**
    There is a *practical* limit to the number of facets of performance you can define and control.
    It is far less than the number of facets that you can *imagine* might be relevant. (Try a limit of just the Top Ten!)

4.7 Additional Ideas: Performance Requirements

Handling Complex Performance Requirements

Many performance requirements, like the quality requirement, 'Usability', can be expressed in greater detail using sub-requirements (such as Learning Time, Error Rate and Minimum Skills Entry Level). There are many possible interpretations, and they all have some use or validity. We call such decomposable ideas 'complex requirements'. It would be easy to think, "there is no measure to cover such a complex requirement." Our attitude is pragmatic and says, "We *will* define a reasonable number of the sub-requirements quantitatively, and *use* them to define what we mean." We only need to identify sufficient sub-requirements to capture the meaning of the performance attribute in the current system context.

The Planguage structure for a hierarchy is as follows:

Tag:

Sub-Tag 1: Scale: <some scale>. <Other parameters as needed>,

Sub-Tag 2: Scale: <some other scale>. <Other parameters as needed>, . . .

Sub-Tag n: Scale: <yet another scale definition>. <Other parameters as needed>.

For example, the first step of the practical example given in Section 4.2 is primarily discussing this idea of expanding a complex quality requirement, 'Serviceability', into a number of elementary and complex quality requirements ('Repair', 'Enhancement', 'Fashion Changes', 'Installation' and 'Reconfiguration').

Here are some *known* 'classic' decomposition examples in the form of a {descriptive tag set}:

EXAMPLE Performance: {Quality: {Availability {Reliability, Maintainability, Integrity}, Adaptability, Usability}, Resource Saving, Work Capacity: Storage Capacity}.

EXAMPLE Maintainability: {Problem Recognition, Administrative Delay, Tool Collection, Problem Analysis, Change Specification, Quality Control, Modification Implementation, Modification Testing, Recovery}.

EXAMPLE Usability: {Entry Level Experience, Training Requirements, Handling Ability, Likeability, Demonstrability}.

(See also the next chapter, especially Section 5.7.)

Limit the Amount of Detail

Expanding complex performance requirements into a number of sub-requirements (and the subsequent need to further expand any sub-requirements that are also themselves complex requirements) usually leads to a great deal of detailed information when specification of the parameters is carried out.

Make sure you focus on the critical (key-influence) performance attributes. Tracking the top ten attributes is usually more than sufficient to make a start. Remember, if you are using evolutionary delivery, you can always decide to modify which attributes you are monitoring over time.

Setting Scalar Levels

Implicit Assumptions Supporting a Scalar Parameter Level

When you set a scalar level, there are certain *implicit* supporting assumptions, which apply. For example, when you specify a Goal level, you are very unlikely to mean 'this level and only this level.' You actually are specifying that a stakeholder wants 'this level *or better*.' Of course, all the other simultaneously specified targets and

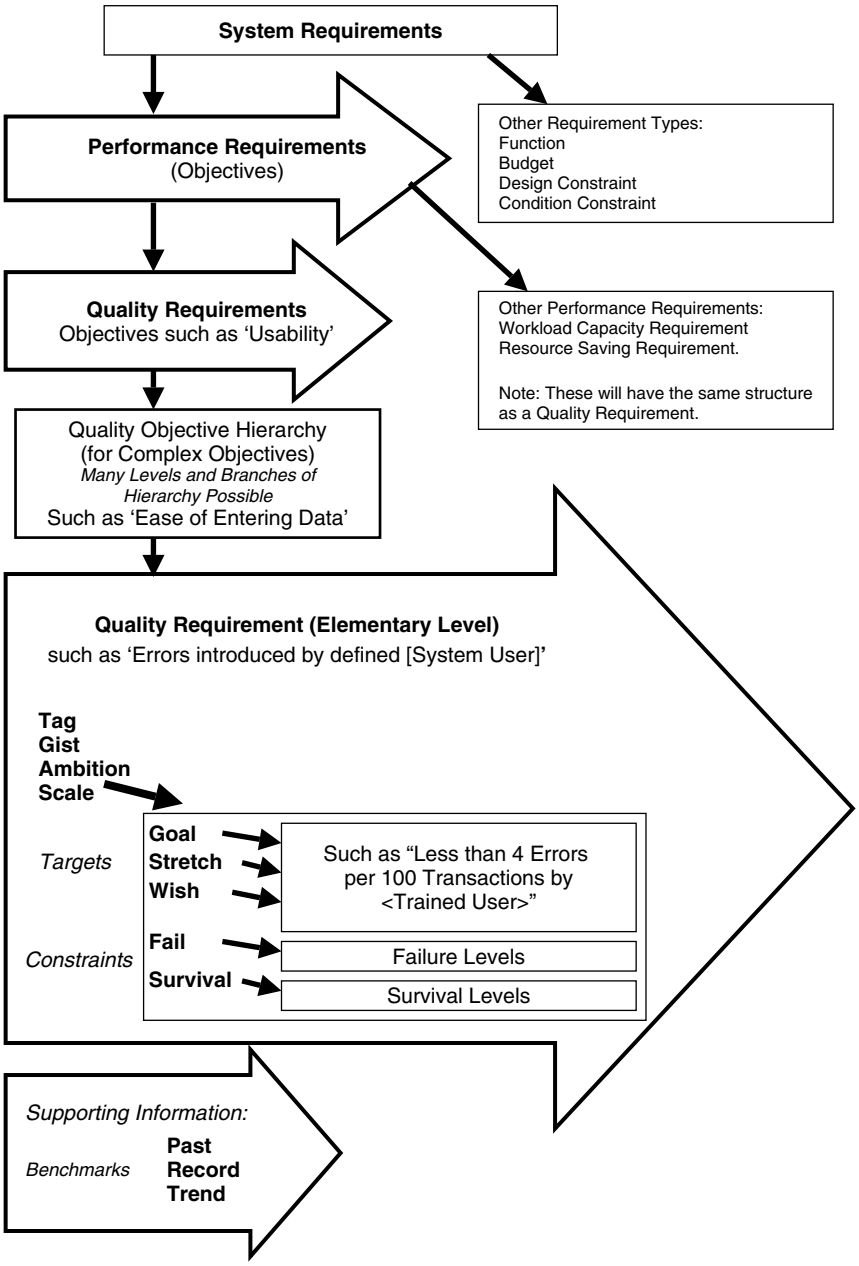


Figure 4.7
Requirement specification hierarchy for a quality requirement.

Table 4.2 A teaching example supplied by Erik Simmons, Intel. The data is not real! Note the explicit direction specified for the Fail levels.

Attribute	*Parameter*		
	Fail	*Goal*	*Stretch*
Performance			
Power (watts)	>10 W	5 W	3–4 W
Product Cost (each unit)	>$21.85	$21.60	$21.50
MTTF (hrs)	<10,000	20,000	25,000
Battery (hrs)	<8	12	16
Weight (lbs)	>5	3	2
Display (diagonal in inches)	<7	8	9
Resource			
Ship Date	>March Next Year	January Next Year	November This Year
Effort (hrs)	>25,000	23,000	22,000
Peak Headcount	>15	12	10

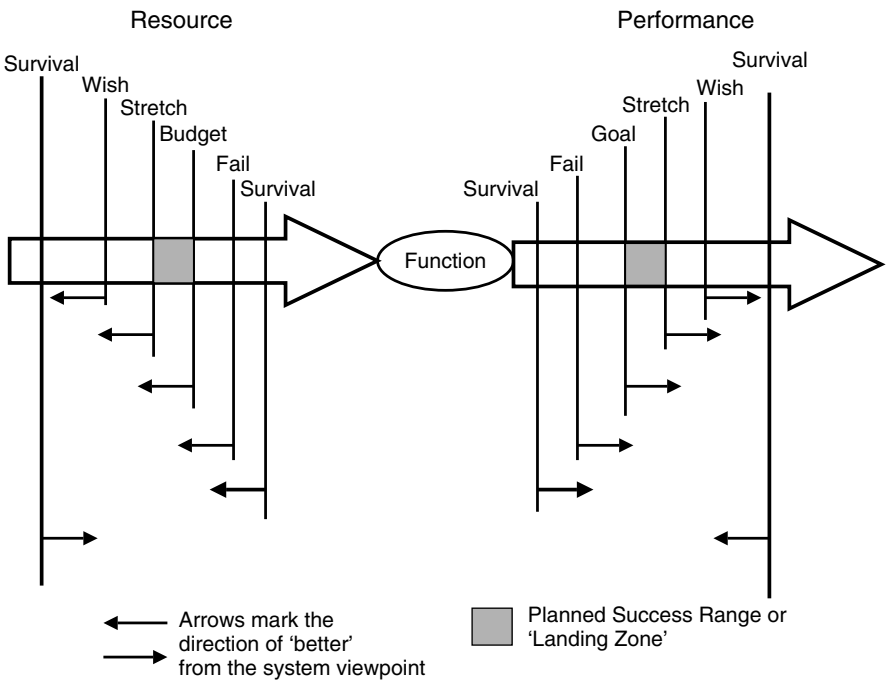


Figure 4.8
Implicit direction for 'better' along a Scale.

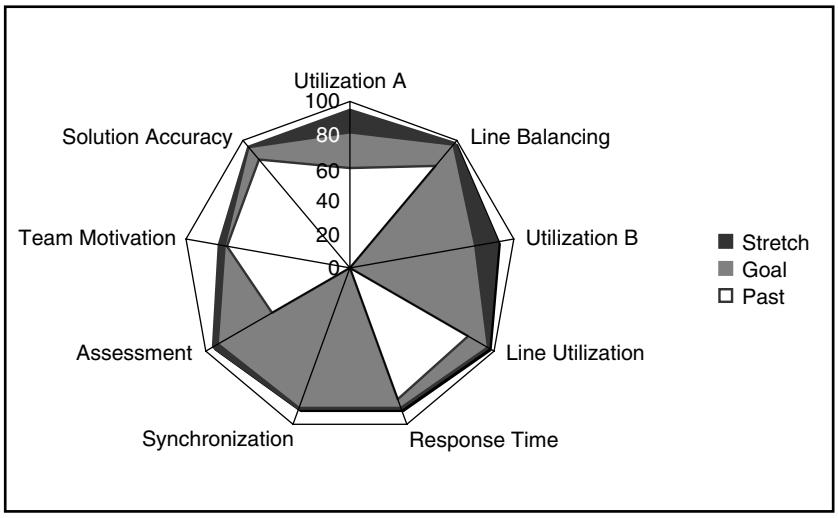


Figure 4.9
A real case study diagram (slightly modified to preserve anonymity) showing a net of multidimensional performance scales of measure. It shows a snapshot of a system at a specific time. The areas show the Past [At Current Time], the gap from Past to the Goal targets, and the gap from the Goal targets to the Stretch targets. This is a powerful graphical way of displaying scalar data.
Note: Resources are not shown and the Performance scalar arrows are spread through 360 degrees.

constraints that apply under the same set of conditions have to be taken into account as well: the stakeholder wants *all* these requirements at the same time. By specifying the Goal level, the stakeholder is providing the information about what they consider the *minimum* performance level for *success* in the light of the other requirements.

Exactly what 'or better' means in numeric terms depends on your Scale definition. A stakeholder wants *more* performance and to use *less* resource (see Figure 4.8). However, the Scale finally dictates the 'direction' of the numeric value and, therefore, the numeric interpretation of 'better.' For example, 'better' performance can mean a reduction in the time taken to carry out a task – a numeric level would therefore be expected to *reduce* over time as performance improved along the Scale.

4.8 Further Example/Case Study: Performance Specification for a Water Supply

Here is a real example of specifying Norwegian Church Aid's performance requirements (objectives) for improving the water supply in Eritrea.

Function: Supplying Water [Eritrea] <- Norwegian Church Aid (NCA).

We began by capturing the immediate objectives:

EXAMPLE

Operation and Maintenance

Local Control:

Ambition: Strengthen conditions for local management of Operation and Maintenance.

Scale: % of Water Supply Pumps which <function> more than 23 hours out of each 24-hour period.

Meter: A <status report> from the Local Water Committees every quarter year.

Past [Eritrea, Four Years Ago]: 65 ± 5% <- Survey conducted by NCA's health co-ordinator.

Goal [Eritrea, By End of this Year]: 80%,

[Eritrea, By End of Next Year]: 90% <- NCA Planning Committee [May Last Year].

Pump Availability:

Ambition: No single Water Supply Pump shall be <out of order> for <a long period of time>.

Scale: % of year Water Supply Pumps <function>.

Meter: Faults reported by the Local Water Committees and the Water Supply Projects.

Past [Eritrea, Four Years Ago]: 60 ± 40% <- ?

Goal [Eritrea, By End of This Year]: 90 ± 10% <- ?,

[Eritrea, By End of Next Year]: 95 ± 5%.

Water Supply Efforts

Well Rehabilitation:

Ambition: Rehabilitation of earlier water supply projects and efforts.

Scale: Number of Water Supply Pumps put into operation anew each year, which satisfy the <minimum need>.

Meter: Reports by Local Water Committees every quarter year.

Past [Eritrea, Four Years Ago]: 30 ± 5%. "of a total of 300."

Goal [Eritrea, By End of This Year]: 40 ± 5% <- ?,

[Eritrea, By End of Next Year]: 35 ± 5%.

New Wells:

Ambition: Make newly drilled wells when other alternatives are not feasible.

Assumptions: {1. New Wells are only to be drilled when other alternatives are impossible. 2. Institutional responsibility and participation from the local village shall be defined and accepted in advance.} <- NCA Policy.

Scale: Number of New Wells completed by agreed dates and according to the Contract between the Drilling Team and the Employer.

Meter: Reports by Local Water Committees every quarter year.

Past [Eritrea, Seven Years Ago]: 66,

[Eritrea, Six Years Ago]: 17.

Goal [Eritrea, By End of This Year]: 10,

[Eritrea, By End of Next Year]: 9.

Alternative Sources:

Gist: Alternatives to drilled wells will be developed whenever the situation permits it.

Scale: Number of efforts per year, which result in <alternative water supplies>.

Meter: Reports by Local Water Committees, Aid Partners or Aid Projects every quarter year.

Past [Eritrea, Five Years Ago]: 20,

[Eritrea, Four Years Ago]: 19.

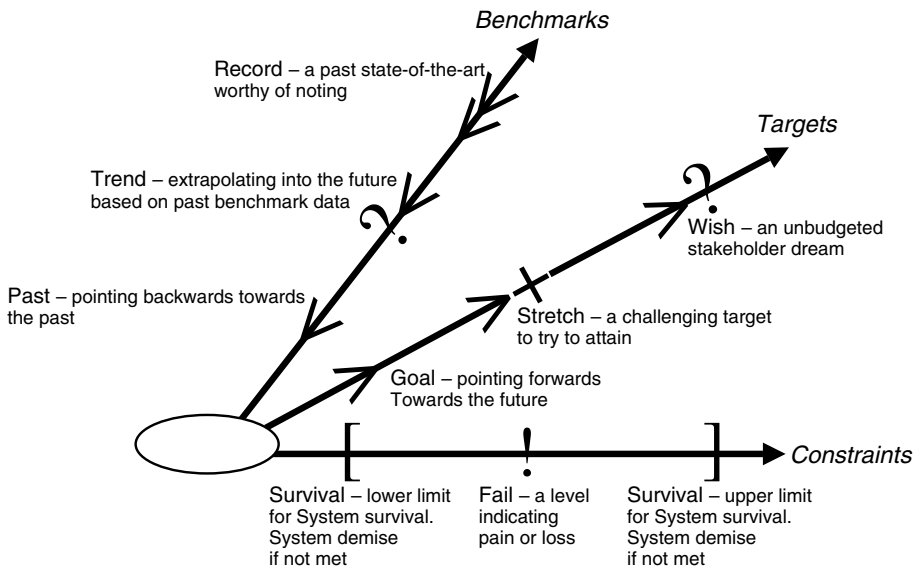
Goal [Eritrea, By End of This Year]: 30,

[Eritrea, By End of Four Years]: 46.

Once we had captured these objectives, we were pleased that we had a clear statement of the requirements that could easily be used for planning purposes and could readily be monitored. However, we soon realized that these goals were *not* directly specifying people's needs; for example, improvement in health, clean water and ease of getting the water to where it should go. Suggestions were consequently made for improved goal setting with a series of new scales. For example, 'average time to pick up the water' and '% of people that die/get sick due to unclean water.'

The major result of the specification was the recognition that the high-level aims of the water projects needed better definition, and that the water projects needed to be seen in that light.

4.9 Diagrams/Icons: Scalar Attribute Requirements



Note: A Scale icon is drawn as a line with an arrowhead, connected to a function oval symbol. Performance scales are to the right from the function oval (O→), and resource scales are at the left of the oval with arrowhead connected to the oval (→O). The performance and resource attribute icons must both include a function icon (an oval) to distinguish them from each other. The arrow in a performance attribute points away from the function oval. For a resource attribute, the arrow points towards the function oval.

Figure 4.10
Three graphical performance attributes showing the icons for scalar performance attribute levels: three analytical benchmarks, three future requirement targets and two future requirement constraints, respectively. Usually an attribute would have a mix of whatever benchmark, target and constraint levels were relevant.

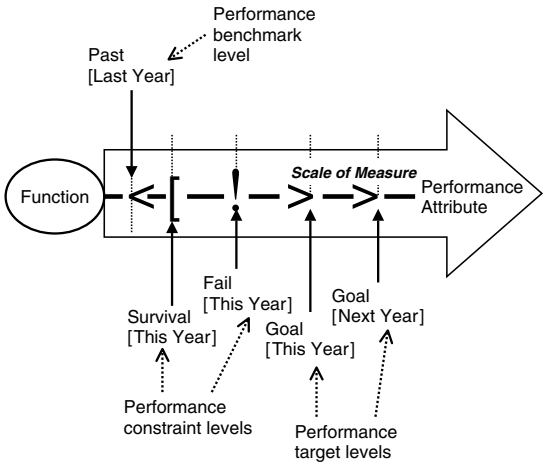


Figure 4.11
Example of using some of the scalar icons: two performance target levels and two constraint levels compared to one benchmark level.

Table 4.3 Icons for scalar attribute requirements.

Planguage Term *Attribute Definition*	*Icon*
Gist	Σ
Ambition	@.Σ
Scale	-|-|-
Meter	-|?|-
Targets	
Goal or Budget	>
Stretch	>+
Wish	>?
Constraints	
Fail	!
Survival	[ ]
System Space Conditions	
Time, Place and Event	[qualifier conditions]
Supporting Information	
Source	<-
Comment	"text."
Benchmarks	
Past	<
Record	<<
Trend	?<

Elementary scalar requirement template <with hints>

Tag: <Tag name of the elementary scalar requirement>.

Type:
<{Performance Requirement: {Quality Requirement,
Resource Saving Requirement,
Workload Capacity Requirement},
Resource Requirement: {Financial Requirement,
Time Requirement,
Headcount Requirement,
others}}>.

========== Basic Information ==========
Version: <Date or other version number>.
Status: <{Draft, SQC Exited, Approved, Rejected}>.
Quality Level: <Maximum remaining major defects/page, sample size, date>.
Owner: <Role/e-mail/name of the person responsible for this specification>.

Stakeholders: <Name any stakeholders with an interest in this specification>.

Gist: <Brief description, capturing the essential meaning of the requirement>.
Description: <Optional, full description of the requirement>.
Ambition: <Summarize the ambition level of *only the targets* below. Give the overall real ambition level in 5–20 words>.

========== Scale of Measure ==========
Scale: <Scale of measure for the requirement (States the units of measure for all the targets, constraints and benchmarks) and the scale qualifiers>.

========== Measurement ==========
Meter: <The method to be used to obtain measurements on the defined Scale>.

========== Benchmarks ========== "Past Numeric Values" ==========
Past [<when, where, if>]: <Past or current level. State if it is an estimate> <- <Source>.
Record [<when, where, if>]: <State-of-the-art level> <- <Source>.
Trend [<when, where, if>]: <Prediction of rate of change or future state-of-the-art level> <- <Source>.

========== Targets ========== "Future Numeric Values" ==========
Goal/Budget [<when, where, if>]: <Planned target level> <- <Source>.
Stretch [<when, where, if>]: <Motivating ambition level> <- <Source>.
Wish [<when, where, if>]: <Dream level (unbudgeted)> <- <Source>.

========== Constraints ========== "Specific Restrictions" ==========
Fail [<when, where, if>]: <Failure level> <- <Source>.
Survival [<when, where, if>]: <Survival level> <- <Source>.

========== Relationships ==========
Is Part Of: <Refer to the tags of any supra-requirements (complex requirements) that this requirement is part of. A hierarchy of tags (For example, A.B.C) is preferable>.
Is Impacted By: <Refer to the tags of any design ideas that impact this requirement> <- <Source>.
Impacts: <Name any requirements or designs or plans that are impacted significantly by this>.

========== Priority and Risk Management ==========
Rationale: <Justify why this requirement exists>.
Value: <Name [stakeholder, time, place, event]: Quantify, or express in words, the value claimed as a result of delivering the requirement>.
Assumptions: <State any assumptions made in connection with this requirement> <- <Source>.
Dependencies: <State anything that achieving the planned requirement level is dependent on> <- <Source>.
Risks: <List or refer to tags of anything that could cause delay or negative impact> <- <Source>.
Priority: <List the tags of any system elements that must be implemented before or after this requirement>.
Issues: <State any known issues>.

Figure 4.12
A scalar requirement template with hints.

4.10 Summary: Performance Requirements

The basic initial step to get control over the primary 'drivers' for plans and resulting projects is to have a clear specification of what we want.

Consider:

- Performance requirements are often 'hidden' in undefined requirement terms, such as 'increased adaptability'.
- Performance requirements may be hidden in designs and plans that have been inadvertently specified amongst the requirements. For example 'Flexible Contracts' is a design idea seeming to imply that there is some (undefined) form of 'flexibility' required, but what is it?
- Performance requirements need to be *numeric* and to be *qualified by conditions*, so we can specify *exactly* what stakeholders want and the [time, place and event] conditions that we must meet.
- Performance requirements must be specified in such a way that they are testable.
- Performance levels are variable; they change from project to project and vary within a project over time, place and events.

Performance requirements are the key statements of expected and necessary critical stakeholder benefits for a project. Performance requirements are the main reason why projects are funded at all. So it is critical that they are done well and managed well.