Chapter
5

Scales of Measure

How to Quantify

Glossary Concepts
Scale
Meter

5.1 Introduction

Scales of measure are fundamental to Planguage. They are central to the definition of all scalar attributes, that is, to all the performance and resource attributes.

You should learn the art of developing your own *tailored* scales of measure for the performance and resource attributes, which are important to your organization or system. You cannot rely on being 'given the answer' about how to quantify. You would soon lose control over your current vital concerns if you waited for that!

Finding and Developing Scales of Measure and Meters

The basic advice for identifying and developing scales of measure and meters (practical methods for measuring) for scalar attributes is as follows:

1. Try to 'reuse' previously defined Scales and Meters. *See Figure 5.3, Examples of Scales of Measure.*
2. Try to 'modify' previously defined Scales and Meters.
3. If no existing Scale or Meter can be reused or modified, use common sense to develop innovative home-grown quantification ideas.
4. Whatever Scale or Meter you start off with, you must be prepared to learn. Obtain and use early feedback, from colleagues and from field tests, to redefine and improve your Scales and Meters.

See also Section 5.5, 'Process Description: Scale Definition.'

Reference Library for Scales of Measure

'Reuse' is an important concept for sharing experience and saving time when developing Scales. You need to build reference libraries of your 'standard' scales of measure. Remember to maintain details supporting each standard Scale, such as Source, Owner, Status and Version (Date). If the name of a Scale's designer is also kept, you can probably contact them for assistance and ideas.

EXAMPLE

Tag: <Assign a tag to this Scale>.
Type: Scale.
Version: <Date of the latest version or change>.
Owner: <Role/email of person responsible for updates/changes>.
Status: <Draft, SQC Exited, Approved>.
Scale: <Specify the Scale with defined [qualifiers]>.
Alternative Scales: <Reference by tag or define other Scales of interest as alternatives and supplements>.

Embedded Scale Qualifiers: <Define the scale parameters, list options>.
Meter Options: <Suggest Meter(s) appropriate to the Scale>.
Known Usage: <Reference projects & specifications where this Scale was actually used in practice with designers' names>.
Known Problems: <List known or perceived problems with this Scale>.
Limitations: <List known or perceived limitations with this Scale>.

This is a draft template with hints for specification of scales of measure in a reference library.

Reference Library for Meters

Another important standards library to maintain is a library of 'Meters.' Meters (*as discussed in Chapter 4*) support scales of measure by providing practical methods for actually measuring the numeric Scale values. 'Off the shelf' Meters from standards' reference libraries can save considerable amounts of time and effort; they are already developed and are 'tried and tested' in the field.

It is natural to reference suggested Meters within definitions of specific scales of measure (as in the template above). Scales and Meters belong intimately together.

EXAMPLE

Tag: Ease of Access.
Type: Scale.
Version: <*version date*>.
Owner: Rating Model Project (Bill).
Scale: Speed for a defined [Employee Type] with defined [Experience] to get a defined [Client Type] operating successfully from the moment of a decision to use the application.
Alternative Scales: None known yet.
Embedded Scale Qualifiers:
Employee Type: {Credit Analyst, Investment Banker, . . . }.
Experience: {Never, Occasional, Frequent, Recent}.
Client Type: {Major, Frequent, Minor, Infrequent}.
Meter Options:
Test all frequent combinations of parameters at least twice. Measure speed for the combinations.
Known Usage: Rating Model Project.
Known Problems: None recorded yet.
Limitations: None recorded yet.

Example of a 'Scale' specification for a reference library.

Managing 'What' You Measure

It is a well-known paradigm that you can manage what you can measure. If you want to achieve something in practice, then quantification, and later measurement, are essential first steps for making sure you get it. If

you do not make it measurable, then it is likely to be less motivating for people to find ways to deliver it (they have no clear targets to work towards and there are not such precise criteria for judgment of failure or success).

On Quantification

- No matter how complex the situation, good systems engineering involves putting value measurements on the important parameters of desired goals and performance of pertinent data, and of the specifications of the people and equipment and other components of the system.
- It is not easy to do this and so, very often, we are inclined to assume that it is not possible to do it to advantage.
- But skilled systems engineers can change evaluations and comparisons of alternative approaches from purely speculative to highly meaningful.
- If some critical aspect is not known, the systems experts seek to make it known. They go dig up the facts.
- If doing so is very tough, such as setting down the public's degree of acceptance among various candidate solutions, the perhaps the public can be polled.
- If that is not practical for the specific issue, then at least an attempt can be made to judge the impact of being wrong in assuming the public preference.
- Everything that is clear is used with clarity: what is not clear is used with clarity as to the estimates and assumptions made, with the possible negative consequences of the assumptions weighed and integrated.
- We do not have to work in the dark, now that we have professional systems analysis.

by Simon Ramo

Figure 5.1
A quote by Simon Ramo of TRW (Ramo and St. Clair 1998 Page 81).

5.2 Practical Example: Scale Definition

'User-friendly' is a popular term. Can you specify a scale of measure for it?

Here is my advice on how to tackle developing a definition for this.

If we assume there is no 'off-the-shelf' definition that could be used:

1. Be more specific about the various aspects of the quality 'user-friendly' that are to be tackled. There are many, but decide on about 5 to 15 in practice that are key to your environment. For this example, let's select 'environmentally friendly' as the one of many aspects that we are interested in, and we shall work on this below as an example. (There are many other elementary aspects of the complex requirement, 'User Friendly', which we could also have chosen.)
2. Invent and specify a Tag: 'Environmentally Friendly' is sufficiently descriptive. Ideally, it could be shorter, but it is very descriptive left as it is.

EXAMPLE Tag: Environmentally Friendly.

Note, we usually don't explicitly specify 'Tag:'.

3. Check there is an Ambition statement, which briefly describes the level of requirement ambition.

EXAMPLE Ambition: A high degree of protection, compared to competitors, over the short-term and the long-term, in near and remote environments for health and safety of living things.

4. Ensure there is general agreement by all the involved parties with the Ambition. If not, ask for suggestions for modifications or additions to it. Here is a simple improvement to my initial Ambition statement. It actually introduces a 'constraint'.

EXAMPLE Ambition: A high degree of protection, compared to competitors, over the short term and the long term, in near and remote environments for health and safety of living things, **which does not reduce the protection already present in nature**.

5. Using the Ambition description, define an initial Scale that is somehow measurable. Think about what will be perceived by the stakeholders if the level of quality changes. What would be a visible effect if the quality improved? My initial unfinished attempt at finding a suitable Scale captured the ideas of change occurring and of things getting better or worse:

EXAMPLE Scale: The % change in positive (good environment) or negative directions for defined

However, I was not happy with it, so I made a second attempt. I refined the Scale by expanding it to include the ideas of specific things being effected in specific places over given times:

EXAMPLE Scale: % destruction or reduction of defined [Thing] in defined [Place] during a defined [Time Period].

This felt better. In practice, I have added [qualifiers] into the Scale, to indicate the variables that must be defined by specific things, places and time periods whenever the Scale is used.

6. Determine if the term needs to be defined with several scales of measure, or whether one like this, with general parameters, will do. Has the Ambition been adequately captured? To determine what's best, you should list some of the possible sub-components of the term (that is, what can it be broken down into, in detail?). For example:

EXAMPLE Thing: {Air, Water, Plant, Animal}.
Place: {Personal, Home, Community, Planet}.
Thing: = {Air, Water, Plant, Animal}.
Place: Consists of {Personal, Home, Community, Planet}.

The first example means: 'Thing' is defined as the set of things Air, Water, Plant and Animal (which since they are capitalized are themselves defined elsewhere). Instead of just the colon after the tag, the more explicit Planguage parameter 'Consists Of' or '=' can be used to make this notation more immediately intelligible to novices in reading Planguage.

Then consider whether the Scale enables the performance levels for these sub-components to be expressed. You may have overlooked an opportunity and may want to add one or more qualifiers to that Scale. For example, we could potentially add the scale qualifier '. . . under defined [Environmental Conditions] in defined [Countries] . . .' to make the scale definition even more explicit and more general.

Scale qualifiers (like . . . 'defined [Place]' . . .) have the following advantages:

(a) they add clarity to the specifications
(b) they make the Scales themselves more reusable in other projects
(c) they make the Scale more useful in this project: specific benchmarks, constraints and targets can be made for any interesting combination of scale variables (such as, 'Thing = Air').

7. Start working on a Meter (remember, you should first check there is not a standard or company reference library Meter that you could use). Try to imagine a practical way to measure things along the Scale, or at least sketch one out. My example is only an initial rough sketch.

EXAMPLE Meter: {scientific data where available, opinion surveys, admitted intuitive guesses}.

The Meter will help confirm your choice of Scale as it will provide evidence that practical measurements can feasibly be obtained using the Scale.

8. Now try out the Scale. Define some reference points from the past (*benchmarks*) and some future requirements (*targets* and *constraints*):

EXAMPLE Environmentally Friendly:
Ambition: A high degree of protection, compared to competitors, over the short-term and the long-term, in near and remote environments for health and safety of living things, which does not reduce the protection already present in nature.
Scale: % destruction or reduction of defined [Thing] in defined [Place] during a defined [Time Period].
========== Benchmarks ==========
Past [Time Period = Next Two Years, Place = Local House, Thing = Water]: 20% <- intuitive guess.
Record [Last Year, Cabin Well, Thing = Water]: 0% <- declared reference point.
Trend [Ten to Twenty Years From Now, Local, Thing = Water]: 30% <- intuitive. "Things seem to be getting worse."

========== Scalar Constraint ==========
Fail [End Next Year, Thing = Water, Place = Eritrea]: 0%. "Not get worse."
========== Targets ==========
Wish [Thing = Water, Time = Next Decade, Place = Africa]: +20% <- Pan African Council Policy.
Goal [Time = After Five Years, Place = <our local community>, Thing = Water]: < 5%.

Not very impressive, maybe I had better find another, more specific, scale of measure? Maybe use a set of Scales?

Here is an example of a more-specific Scale:

EXAMPLE Scale: % change in water pollution degree as defined by UN Standard 1026.

Here is an example of some alternative and more-specific set of Scales for the 'Environmentally Friendly' example:

EXAMPLE
Environmentally Friendly:
Ambition: A high degree of protection, compared to competitors, over the short-term and the long-term, in near and remote environments for health and safety of living things, which does not reduce the protection already present in nature.
Air: Scale: % of days annually when <air> is <fit for all humans to breath>.
Water: Scale: % change in water pollution degree as defined by UN Standard 1026.
Earth: Scale: Grams per kilo of toxic content.
Predators: Scale: Average number of <free-roaming predators> per square km, per day.
Animals: Scale: % reduction of any defined [Living Creature] who has a defined [Area] as their natural habitat.

'Environmentally Friendly' is now defined as a complex attribute, because it consists of a number of elementary attributes: {Air, Water, Earth, Predators, Animals}. A different scale of measure defines each of these elementary attributes. Using these Scales we can add Meters, benchmarks, constraints and target levels to describe exactly how Environmentally Friendly we want to be.

Level of Specification Detail

How much detail you need to specify, depends on what you want control over and how much effort it is worth. The basic paradigm of Planguage is you should only elect to do what pays off for you. You should not build a more detailed specification than is meaningful in terms of your project and economic environment. Planguage tries to give you sufficient power of articulation to control both complex and simple problems. You need to scale up, or down, as appropriate. This is done through common sense, intuition, experience and organizational standards (reflecting

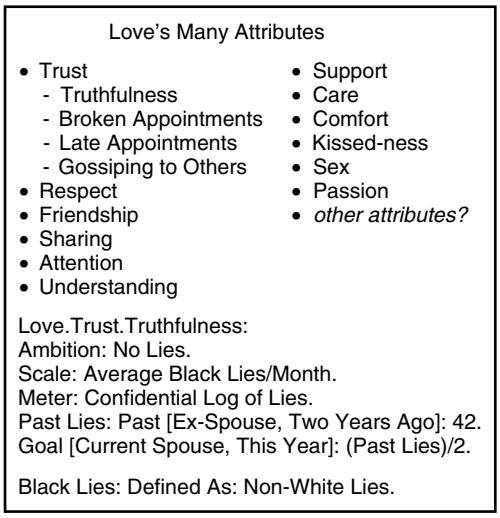


Figure 5.2
Love is a many-splendored thing! Another example of decomposing a complex subject into its component attributes. This is from a classroom exercise, which was done in two stages. First, we decomposed the complex concept, 'Love' into many aspects. Then we took one attribute at random to see if a reasonable quantified specification could be achieved.

experience). But, if in doubt, go into more detail. History says we have tended in the past to specify too little detail about requirements. The result consequently has often been to lose control, which costs a lot more than the extra investment in requirement specification.

5.3 Language Core: Scale Definition

This section builds on the specification ideas presented in Chapter 4. It discusses the specification of Scales with qualifiers.

Specifying Scales

The Central Role of a Scale within Scalar Attribute Definition

A scale of measure (Scale) is the heart of a scalar specification and essential to support all the targets, constraints and benchmarks. The specified Scale of an elementary scalar attribute is used (*reused!*) within all the scalar parameter specifications of the attribute (that is, within all the Goal, Budget, Stretch, Wish, Fail, Survival, Past, Record and Trend parameters).

Each time a scalar parameter is specified, the Scale dictates what has to be defined. And then, later, each time a scalar parameter

definition is read, the Scale 'interprets' its meaning. So the Scale is truly central to the definition of any scalar parameter. Well-defined scales of measure are well worth the small investment to define and refine them.

Specifying Scales using Qualifiers

The scalar attributes (performance and resource) are best measured in terms of specific times, places and defined conditions. If we fail to do this, they lose meaning. People wrongly guess other times, places and conditions than you intend, and cannot relate their experiences and knowledge to your numbers. If we don't get more specific by using qualifiers, then performance and resource continues to be a vague concept and there is ambiguity (which times? which places? which events?).

Further, it is important that the set of *different* performance and resource *levels* for different specific time, places and defined conditions are identified. It is likely that the levels of the performance and resource requirements will differ across the system depending on such things as time, location, role and system component.

Decomposing complex performance and resource ideas, and finding market-segmenting qualifiers for differing target levels, is a key method of competing for business.

Embedded Qualifiers within a Scale: A Scale specification can set up useful qualifiers by declaring embedded scale qualifiers, using the format 'defined [<qualifier>]'. It can also declare default qualifier values that apply by default if not overridden, 'defined [<qualifier>: default: <User-defined Variable or numeric value>]'. For example, [. . . default: Novice].

Additional Qualifiers: However, embedded scale qualifiers should not stop you adding any other useful additional qualifiers later, as needed, during specification. But, if you do find you are adding the same type of parameters in almost all specifications, then you might as well design the Scale to include those qualifiers. A Scale should be built to ensure it forces the user to define the critical information needed to understand and control a critical performance or resource attribute.

Here is an example of how user-defined terms (additional qualifiers) can make a quality more specific. Note also, how a requirement can be made conditional upon an event. If the event is not true, the requirement does not apply.

First, some *basic definitions* are required:

EXAMPLE Assumption A:
Basis [This Financial Year]: Norway is still not a full member of the European Union.
EU Trade:
Source: Euro Union Report [EU Trade in Decade 2000–2009].
Positive Trade Balance:
State [Next Financial Year]: Norwegian Net Foreign Trade Balance has Positive Total to Date.

Now we apply those definitions below:

EXAMPLE Quality A:
Type: Quality Requirement.
Scale: % of Goods Delivered, by <value>, which are Returned for Repair or Replacement by Consumers.
Meter [Development]: Weekly samples of 10, [Acceptance]: 30 day sampling at 10% of representative cases, [Maintenance]: Daily sample of largest cost case.
Fail [European Union, Assumption A]: 40% <- European Economic Members.
Goal [EU and EEU members, Positive Trade Balance]: 50% <- EU Trade.

The Fail and the Goal requirements are now defined partly with the help of qualifiers. The Goal to achieve 50% (or more, is implied) is only a valid plan if 'Positive Trade Balance' is true. The Fail level requirement of 40% (or worse, less, is implied) is only valid if 'Assumption A' is true.

5.4 Rules: Scale Definition

Tag: Rules.SD.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: Rules for Scale Definition.

Note: These rules are concerned with the use of scales of measure and also specification of scalar parameters, including specification of numeric values. They complement Rules.SR.

Base: These rules are to be used in addition to the rules for Scalar Requirement Specification (Rules.SR).

R1: **Standard**: The Scale and/or Meter must, wherever possible, be derived from a standard version (held in named files or referenced sources) and the standard shall be source referenced in the specification. For example, Scale: . . . <-Corporate Scale 1.2.

R2: **Notify Owner**: If a Scale or Meter is not standard, a notification must be sent to the appropriate Library Owner to inform them about the availability this new case. ''Note sent to <Library Owner>'' will be included as a specification comment to confirm this act.

R3: **Scale Definition**: Each scale definition in a specification is part of an elementary attribute (that is, the associated elementary requirement definition must have a unique tag, and appropriate set of parameters, such as Past and Goal). The scale definition must define the units of measure so that benchmarks, constraints and targets can be set clearly and consistently.

R4: **Elementary Attribute**: An elementary attribute must only have one Scale.

R5: **Differentiate**: A distinction will be made, by using qualifiers, between those system components which must have significantly higher performance levels than others, and components which do not require such levels. ''The most ambitious level [across an entire system] can cost too much.''[1]

EXAMPLE Goal [Operating System Core]: 99.98%, [Online Internet Components]: 99.90%, [Offline Components]: 99%.

R6: **Uncertainty**: Whenever there is known uncertainty in the precise level for a specified numeric value, its upper and lower boundaries should be explicitly stated. Expressions, such as {60 ± 20, 60 to 80, 60?, 60??}, can be used.

R7: **Scalar Priority**: No artificial 'weights'. Use scalar priority. The relative 'static' (initial) priority of a scalar requirement (its 'claim on limited resources) is initially given by means of the target and constraint statements {Goal, Stretch and Wish, Fail and Survival levels} and, also by the complementary information given by qualifiers, Source and Authority statements. It is unnecessary and 'corrupting' to add any other priority information (such as weights or relative priority by tag name).

The final real 'dynamic' priority of meeting a scalar requirement is a matter for systematic engineering tradeoff later, when the total real impacts and costs of design ideas are better understood during design analysis or system development.

(*Note: Function requirements can however state 'simple priority' directly. They have no scalar mechanisms for determining priority based on unfulfilled Goal attainment. See Rules.FR:R5: Function Priority.*)

[1] I once participated in 'saving' a German telecoms project, which had run about 3,000 work years over financial budget and two to three calendar years late, mainly as a result of applying the highest quality levels across the entire system (n fact, only the core software warranted such levels).

Table 5.1 Examples of Scales of Measure.

Performance	*Effect of Change in Performance*	*Scale of Measure*
Customer Satisfaction	Fewer letters of complaint	Number of letters complaining about a defined [Product] received within a defined [Time Period]
Customer Satisfaction	Fewer returned goods	Percentage of defined [Product] returned within defined [Time Period after Purchase] with defined [Customer Issue]
Environmentally Friendly	Improved rating as measured on international standard	Number of defined [Product Type] failing defined [Test] within a defined [Time Period]
User-friendly	Fewer errors made	Percentage of defined [Transaction Type] with defined [Error] input by defined [User Type]
User-friendly	Faster time for completion of transactions	Time in minutes for a defined [Transaction] to be carried out to <satisfactory> completion
Restful Ambience	Calming, relaxing effect	Percentage of users of defined [User Type] agreeing that defined [Room Space] was <restful>
Reliability	Fewer breakdowns	Mean Time Between Repair (MTBR)
Staff Satisfaction	Lower rate of staff turnover	Number of staff of defined [Job Description Response]
Predictability	Less variance in time to initial response	Percentage of service calls of defined [Service Type] exceeding <initial response> within defined [Time Period]

5.5 Process Description: Scale Definition

Process: Scale Definition

Tag: Process.SD.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: Determining a Scale of Measure.

Note: The procedure steps cannot simply be done sequentially. Iteration is needed to evolve realistic scales of measure.

Entry Conditions

E1: The Generic Entry Conditions apply. Input documentation includes contracts, marketing plans, product plans and requirements specification. The relevant rules should also be available: the generic

specification rules (Rules.GS), the requirement specification rules (Rules.RS), the rules for scalar requirement specification (Rules.SR) and, the rules for scale definition (Rules.SD).

E2: Do not enter this procedure if company files or standards already have adequate quantification devices. Preferably use the existing Scales and Meters found in the standards' libraries.

Procedure

P1: Ensure that you have derived an elementary attribute (from a complex requirement), and that you are not trying to use a complex requirement, which needs decomposition into its elementary attributes. (*Trying to find a single Scale for a complex (multi-Scale) requirement doesn't work well. It is usually the cause of trouble when people fail to find a suitable Scale.*)

If you find you do indeed have a complex requirement, then decompose it and try to find Scales for its components. You might well find that further (second-level and more) decomposition is required!

P2: Ensure that the elementary attribute that you are developing a Scale for has a suitable tag and a Gist or Ambition parameter that adequately describes the concept in outline terms.

P3: Using the Gist or Ambition, analyze how a 'change' of degree in the scalar attribute level would be expressed. What would a user experience or perceive? For some examples, see Table 5.1, 'Examples of Scales of Measure'.

Sometimes you can keep things simple, and 'make do', by controlling the details at a higher level of abstraction:

- *by deciding to use one dominant Scale only, and consciously ignoring the potential other scales.*
- *by aggregating several scales of measure to express one summary scale of measure.*
- *by defining a complex attribute as the 'set' of other Scales and definitions.*

P4: Specify the critical [time, place, event] qualifiers to express different benchmarks, constraints and target levels.

P5: If there is no appropriate standard Meter (or test), start working on a Meter. Try to imagine a practical way to measure things along the Scale, or at least sketch one. Try thinking about any measures that are currently being carried out (this could even help you start developing ideas for scales of measure). Also, think about whether any current system could be modified, or have its settings changed, to perform additional measurement.

P6: Try out the Scale. Define some reference points from the past (*benchmarks*) and then, on the basis of benchmarks, specify future requirements (*targets* and *constraints*).

P7: Repeat this process until you are satisfied with the result. Try to get approval for your Scale from some of the stakeholders. Does it quantify what they really care about?

P8: Consider putting embedded parameters into the Scale definition. Rationale: To enable a Scale to be reused both within a project and in other projects.

EXAMPLE Scale: Time needed to do defined [Task] by defined [User] in defined [Environment].
Goal [Task = Get Number, User = <Novice>, Environment = <Noisy>]: 10 minutes.

P9: Once you have developed a useful Scale, ensure it is made available for others to use (on your intranet, or a web site, or in course materials, or your 'personal' glossary of Scales[2]). Offer the Scale to the owner of the 'Scales' library within your organization.

Exit Conditions

X1: The Generic Exit Conditions apply.

X2: Alternatively, consensus is obtained on trying out the Scale in practice, and exit condition X1 is temporarily waived.

Rationale [X2, Tryout]: The intent being to gain experience, or to obtain opinions concerning the quantification, so it can be refined ready for <official use>.

5.6 Principles: Scale Definition

1. **The Principle of 'Defining a Scale of Measure'**
   If you can't define a scale of measure, then the goal is out of control.
   Specifying any critical variable starts with defining its units of measure.
2. **The Principle of 'Quantification being Mandatory for Control'**
   If you can't quantify it, you can't control it.[3]
   If you cannot put numbers on your critical system variables, then you cannot expect to communicate about them, or to control them.

[2] See (Gilb 2004, Requirement Slides) for further examples of scales of measure.
[3] Paraphrasing a well-known old saying.

3. **The Principle of 'Scales should control the Stakeholder Requirements'**
   Don't choose the easy Scale, choose the powerful Scale.
   Select scales of measure that give you the most direct control over the critical stakeholder requirements. Choose the Scales that lead to useful results.

4. **The Principle of 'Copycats Cumulate Wisdom'**
   Don't reinvent Scales anew each time – store the wisdom of other Scales for reuse.
   Most scales of measure you will need will be found somewhere in the literature or can be adapted from existing literature.

5. **The Cartesian Principle**
   Divide and conquer said René – put complexity at bay.
   Most high-level performance attributes need decomposition into the list of sub-attributes that we are actually referring to. This makes it much easier to define complex concepts, like 'Usability', or 'Adaptability,' measurably.

6. **The Principle of 'Quantification is not Measurement'**
   You don't have to measure in order to quantify!
   There is an essential distinction between quantification and measurement. "I want to take a trip to the moon in nine picoseconds" is a clear requirement specification without measurement."
   The well-known problems of measuring systems accurately are no excuse for avoiding quantification. Quantification allows us to communicate about how good scalar attributes are or can be – before we have any need to measure them in the new systems.

7. **The Principle of 'Meters Matter'**
   Measurement methods give real world feedback about our ideas.
   A 'Meter' definition determines the quality and cost of measurement on a scale; it needs to be sufficient for control and for our purse.

8. **The Principle of 'Horses for Courses'**[4]
   Different measuring processes will be necessary for different points in time, different events, and different places.[5]

9. **The Principle of 'The Answer always being "42"'**[6]
   Exact numbers are ambiguous unless the units of measure are well-defined and agreed.
   Formally defined scales of measure avoid ambiguity. If you don't define scales of measure well, the requirement level might just as well be an arbitrary number.

---

[4] 'Horses for courses' is a UK expression indicating something must be appropriate for use, fit for purpose.

[5] There is no universal static scale of measure. You need to tailor them to make them useful.

[6] Concept made famous in Douglas Adams, *The Hitchhiker's Guide to the Galaxy*, Macmillan, 1979, ISBN 0-330-25864-8.

10. **The Principle of 'Being Sure About Results'**
    If you want to be sure of delivering the critical result – then quantify the requirement.
    Critical requirements can hurt you if they go wrong – and you can always find a useful way to quantify the notion of 'going right.'

5.7 Additional Ideas: Generic Hierarchies for Scalar Attributes

You can decompose many scalar attributes into arbitrarily large or small sets of more specific 'elementary' scalar attributes. The selection of exactly which elementary attributes to define is a practical matter of knowing your domain well enough to decide which of them will give you best control over your critical success factors. At best we make reasonable guesses with some effort to begin with. Then we learn some hard lessons, usually about what we forgot to exercise control over.

Having said this, we have found that templates for performance and resource/cost attributes are helpful to most people. So, we will give some basic performance attributes in this section. Remember, in any real system they will need to be used *selectively*: they will need to be *tailored* to your local purpose. (*See Figure 5.4 for an overview of these attribute definitions.*)

Note all these template ideas build upon the templates originally presented in Gilb (1988 Chapter 19).

They are organized into multilevel hierarchies of attributes. This does not imply that any one hierarchical organization is best or correct. But they are useful. The essential idea is to get control over those elementary attributes that determine your success or failure. A flat list of the right ones works as well as any hierarchy. Hierarchies are mainly useful groups for human convenience, but are not a reality for the system.

Basic Usability Model

Performance: 'Useful values deliverable to stakeholders.'
Includes: {Quality, Resource Savings, Workload Capacity}.

1. **Quality**: 'How well a system performs.'
   Includes: {Availability, Adaptability, Usability, Other}.

   1.1 **Availability**: 'The readiness of a system to do its work.'
   Gist: Availability is the measure of how much a system is usefully (not merely technically) available to perform the work that it was designed to do.

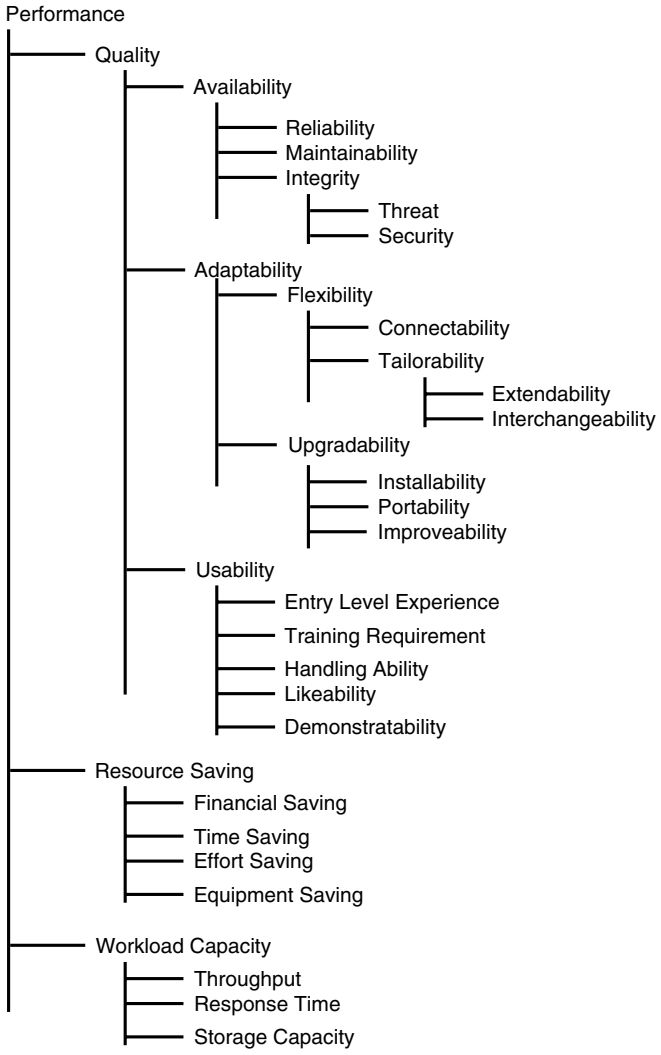


Figure 5.3
One decomposition possibility for performance attributes with emphasis on the detail of the quality attributes.

Availability: Type: Elementary Quality Requirement.
Scale: % of defined [Time Period] a defined [System] is available for its defined [Tasks]

Availability: Type: Complex Quality Requirement.
Includes: {Reliability, Maintainability, Integrity}.

1.1.1 **Reliability**: 'A system performs as it is intended.'
Gist: Reliability is a measure of the degree to which a system performs as it was designed to do, as opposed to doing something else (like producing a wrong answer or providing

no answer). Definitions of Reliability will therefore vary according to the definition of what the system is supposed to do. In general, if a system is in an unreliable state then it is 'unavailable' for its intended work tasks.

Scale: Mean time for a defined [System] to experience defined [Failure Type] under defined [Conditions].

1.1.2 **Maintainability**: 'Resource required to repair an unreliable system.'
Gist: Maintainability is a measure of how quickly an unreliable system can be brought to a reliable state. In general, this covers not only the actual repair of the fault, but also recovery from any effects of the fault and, quality control and test of the repair.
Conventionally, maintenance is concerned with the process of fault handling, rather than for improvement of a faultless system. However, the difference between what is a fault and what is a system improvement can be subjective! (*See also later definition for 'Adaptability'*).

Maintainability: Type: Elementary Quality Requirement.
Scale: Mean time to carry out a defined [Type of Repair] to a defined [System] using defined [Repair Method] under defined [Conditions].

1.1.3 **Integrity**: 'The ability of the system to survive attack.'
Gist: Integrity is a measure of the confidence that the system has suffered no harm: its security has not been breached and, its use has resulted in no 'corruption' or impairment to it. An attack on the Integrity of a system can be accidental or intentional. The Integrity of a system depends on the frequency of *threat* to it and the effectiveness of its *security*.

Integrity: Type: Elementary Quality Requirement.
Scale: Probability for a defined [System] to achieve defined [Coping Action] with defined [Attack] under defined [Conditions].
Coping Action: {detect, prevent, capture}.
Integrity: Type: Complex Quality Requirement.
Includes: {Threat, Security}.

1.2 **Adaptability**: 'The efficiency with which a system can be changed.'
Gist: Adaptability is a measure of a system's ability to change. Since, if given sufficient resource, a system can be changed in almost any way, the primary concern is with the amount of resources (such as time, people, tools and finance) needed to bring about specific changes (the 'cost').

Maintainability:
Type: Complex Quality Requirement.
Includes: {Problem Recognition, Administrative Delay, Tool Collection, Problem Analysis, Change Specification, Quality Control, Modification Implementation, Modification Testing {Unit Testing, Integration Testing, Beta Testing, System Testing}, Recovery}.

Problem Recognition:
Scale: Clock hours from defined [Fault Occurrence: Default: Bug occurs in any use or test of system] until fault officially recognized by defined [Recognition Act: Default: Fault is logged electronically].
Administrative Delay:
Scale: Clock hours from defined [Recognition Act] until defined [Correction Action] initiated and assigned to a defined [Maintenance Instance].
Tool Collection:
Scale: Clock hours for defined [Maintenance Instance: Default: Whoever is assigned] to acquire all defined [Tools: Default: all systems and information necessary to analyze, correct and quality control the correction].
Problem Analysis:
Scale: Clock time for the assigned defined [Maintenance Instance] to analyze the fault symptoms and be able to begin to formulate a correction hypothesis.
Change Specification:
Scale: Clock hours needed by defined [Maintenance Instance] to fully and correctly describe the necessary correction actions, according to current applicable standards for this.
Note: This includes any additional time for corrections after quality control and tests.
Quality Control:
Scale: Clock hours for quality control of the correction hypothesis (against relevant standards).
Modification Implementation:
Scale: Clock hours to carry out the correction activity as planned. "Includes any necessary corrections as a result of quality control or testing."
Modification Testing:
Unit Testing:
Scale: Clock hours to carry out defined [Unit Test] for the fault correction.
Integration Testing:
Scale: Clock hours to carry out defined [Integration Test] for the fault correction.
Beta Testing:
Scale: Clock hours to carry out defined [Beta Test] for the fault correction before official release of the correction is permitted.
System Testing:
Scale: Clock hours to carry out defined [System Test] for the fault correction.
Recovery:
Scale: Clock hours for defined [User Type] to return system to the state it was in prior to the fault and, to a state ready to continue with work.

Source: The above is an extension of some basic ideas from Ireson, Editor, Reliability Handbook*, McGraw Hill, 1966* (Ireson 1966).

Figure 5.4
A more detailed view of Maintainability.

Adaptability: Type: Elementary Quality Requirement.
Scale: Time needed to adapt a defined [System] from a defined [Initial State] to another defined [Final State] using defined [Means].

Adaptability: Type: Complex Quality Requirement.
Includes: {Flexibility, Upgradeability}.

1.2.1 **Flexibility**:
Gist: This concerns the 'in-built' ability of the system to adapt or to be adapted by its users to suit conditions (without any fundamental system modification by system development).
Type: Complex Quality Requirement.
Includes: {Connectability, Tailorability}.

1.2.1.1 **Connectability**: '*The cost to interconnect the system to its environment.*'
Gist: The support in-built within the system to connect to different interfaces.

1.2.1.2 **Tailorability**: '*The cost to modify the system to suit its conditions.*'
Type: Complex Quality Requirement.
Includes: {Extendability, Interchangeability}.

1.2.1.2.1 **Extendability**:
Scale: The cost to add to a defined [System] a defined [Extension Class] and defined [Extension Quantity] using a defined [Extension Means].
"In other words, add such things as a new user or a new node."
Type: Complex Quality Requirement.
Includes: {Node Addability,
Connection Addability,
Application Addability,
Subscriber Addability}.

1.2.1.2.2 **Interchangeability**: '*The cost to modify use of system components.*'
Gist: This is concerned with the ability to modify the system to switch from using a certain set of system components to using another set.
For example, this could be a daily occurrence switching system mode from day to night use.

1.2.2 **Upgradability**: '*The cost to modify the system fundamentally; either to install it or change out system components.*'
Gist: This concerns the ability of the system to be modified by the system developers or system support in planned stages (as opposed to unplanned maintenance or tailoring the system).
Type: Complex Quality Requirement.
Includes: {Installability, Portability, Improveability}.

1.2.2.1 **Installability**: '*The cost to install in defined conditions.*'
This concerns installing the system code and also, installing it in new locations to extend the system

coverage. Could include conditions such as the installation being carried out by a customer or, by an IT professional on-site.

1.2.2.2 **Portability**: '*The cost to move from location to location.*'
Scale: The cost to transport a defined [System] from a defined [Initial Environment] to a defined [Target Environment] using defined [Means].
Type: Complex Quality Requirement.
Includes: {Data Portability,
Logic Portability,
Command Portability,
Media Portability}.

1.2.2.3 **Improveability**: 'The cost to enhance the system.'
Gist: The ability to replace system components with others, which possesses improved (function, performance, cost and/or design) attributes.
Scale: The cost to add to a defined [System] a defined [Improvement] using a defined [Means].

1.3 **Usability**: '*How easy a system is to use.*'
Scale: Speed for defined [Users] to correctly accomplish defined [Tasks] when given defined [Instruction] under defined [Circumstances].
Note: This is a generic scale for Usability, which you can use if you want to simplify matters and deal with Usability at an elementary level. It is however more usually declared as 'complex' and then defined in a more specific manner; for example, by using the sub-attributes below. There are of course, many different possible decompositions of Usability.
Type: Complex Quality Requirement.
Includes: {Entry Level Experience, Training Requirement, Handling Ability, Likeability, Demonstratability}.

1.3.1 **Entry Level Experience**:
Scale: The defined [Level of Knowledge] required to receive training or to use a defined [System].

1.3.2 **Training Requirement**:
Scale: The degree of training required for a defined [User Type] to achieve a defined [Degree of Proficiency] with a defined [System].

1.3.3 **Handling Ability**:
Scale: A defined [Degree of Proficiency] with a defined [System] by a defined [Class of User].

1.3.4 **Likeability**:
Scale: The degree to which defined [Users] declare that they are pleased with defined [Aspects] of a defined [System].

1.3.5 **Demonstrability**:
Type: Complex Quality Requirement.
Includes: Elementary Quality Requirement {Customer Self-Demonstrability, Sales Demonstrability}.

Some Alternative Models for Usability:

The point of these three alternative models to the basic Usability model (above) is to emphasize that there is NOT one 'correct model.' All major projects need highly tailored models. I also want to show some specific instances of Usability sub-scales as a checklist or stimulant to reades when building their own models.

EXAMPLE

Usability:
Type: Complex Quality Requirement.
Includes: {Entry Level Experience, Training Requirement, Handling Ability, Likeability, Demonstrability}. "Only one of the many possible decompositions of Usability."
Demonstrability:
Type: Complex Quality Requirement.
Includes: Type: Elementary Quality Requirement {Customer Self-Demonstrability}.
Customer Self-Demonstrability:
Ambition: Ability of Customer to solo self-demonstrate a Product is to be <high>.
Scale: Probability of <successful completion> of self-demonstration within one hour.
Past [Last Year, All Products]: < 5%.
Fail: 90% to 95% <- Corporate Quality Policy.
Goal: 95%.

EXAMPLE

Usability:
Type: Complex Quality Requirement.
Device Swapability:
Scale: Minutes to swap over a defined [Input Device or Output Device].
Training Need: Scale: Hours in <training mode> until capable of defined [Tasks].
User Productivity: Scale: % User Time lost due to Product Fault or <Bad Design>.
User Error Rate: Scale: % of User Actions, which they correct or change.
User Minimum Qualification Level:
Scale: Average % of correct answers to a defined [Qualifying Test] by a defined [User Type].
Userlessness: Scale: % of Tasks, which can run <unattended>.
Coherence:
Scale: % of User Interface Elements, which are perceived as consistent with our Product Image.

User Opinion:
Scale: % of defined [User Type] who express <positive feeling> after using defined [Product Component(s)].
Customer Self-Demonstratability:
Scale: % Probability of successful <self demonstration> of defined [Product or Product Component] by defined [User Type] within defined [Time Span] of attempt to use it.

EXAMPLE

Usability:
Type: Complex Quality Requirement.
Includes: Type: Elementary Quality Requirement {Entry Conditions, Training Requirement, Computer Familiarity, Web Experience Level, Productivity, Error Rate, Likeability, Intuitiveness, Intelligibility}.
Entry Conditions:
Scale: <Grade Level of User>.
Training Requirement:
Scale: Time needed to read <any instructions> or get <any help> in order to perform defined [Tasks] successfully.
Computer Familiarity:
Scale: Years of <experience with computers>.
Web Experience Level:
Scale: Years of <experience with using the web>.
Productivity:
Scale: Ability to correctly produce defined [Work Units: Default: Completed Transactions].
Error Rate:
Scale: Number of Erroneous Transactions requiring correction each <session>.
Likeability:
Scale: Option of <pleasure> of using the system on scale of −10 to +10.
Intuitiveness:
Scale: Probability that a defined [User] can intuitively figure out how to do a defined [Task] correctly (without any errors needing correction).
Intelligibility:
Scale: Probability in % that a defined [User] will correctly interpret defined [Messages or Displays].

2. **Resource Savings**:
Gist: How much resource savings a new system produces compared to some defined benchmark system.
Type: Complex Performance Requirement.
Includes: {Financial Saving, Time Saving, Effort Saving, Equipment Saving}.

- **Financial Saving**: "Financial Cost Reduction"
  Scale: Net Financial Saving planned or achieved compared to a defined [Benchmark Amount].
- **Time Saving**: "Processing Time Reduction, Elapse Time Reduction, Time To Market"

Scale: Net Time Saving planned or achieved compared to a defined [Benchmark Amount].

- **Effort Saving**: ''Reduction in the Person-Hours required''
  Scale: Net Effort saving planned or achieved compared to a defined [Benchmark Amount].
- **Equipment Saving**: ''includes room space!''
  Scale: Net Space saving planned or achieved compared to a defined [Benchmark Amount].

3. **Workload Capacity**: '*The raw ability of the system to perform work.*'
   Type: Complex Performance Requirement.
   Includes: {Throughput, Response Time, Storage Capacity}.

- **Throughput**:
  Gist: Throughput is a measure of the ability of the system to process work. *For example, the average number of telephone sales orders, which can be dealt with by an experienced telephone sales operator, in an hour.*
  Scale: The average quantity of defined [Work Units], which can be successfully handled in a defined [Time Unit].
- **Response Time**: ''Retrieval Timing, Transaction Timing''
  Scale: The mean average speed to perform a defined [Reaction] on receiving a defined [Impulse].
- **Storage Capacity**: 'The ability of the system to increase in size'
  Gist: This is the capability of a component part of the system to store units of some defined kind. For example, number of registered users, lines of code, photographs and boxes.
  Scale: The capacity to store defined [Units] under defined [Conditions].

5.8 Further Example/Case Study: Scale Definition

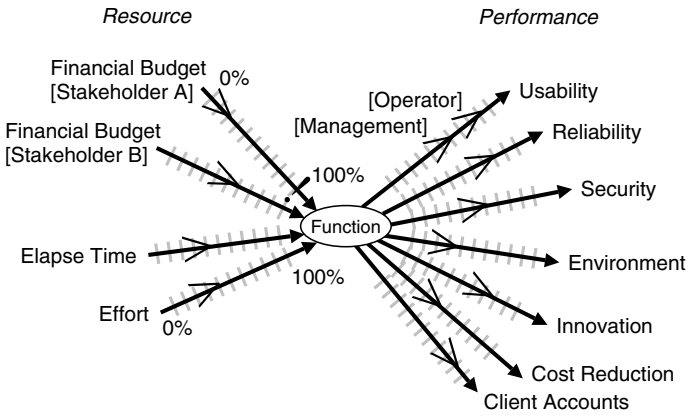
This is part of a quality definition done for the airborne command and control system, which was discussed previously in Section 3.8. It is a first draft (there are lots of things to be refined later) and it is only a sample of the requirement specification we actually worked out. We chose to work on 'Usability' as it was defined as 'the key competitive system quality'. This system is now operational.

EXAMPLE

Usability:
Ambition: Operator ease of learning & doing tasks under <all conditions> should be maximum possible ease & speed of performance with minimum training & minimum possibility of <unchecked error(s)>.
Usability.**Intuitiveness**:
Ambition: High probability that an operator will within a specified time from deciding the need to perform a specific task (without reference to handbooks or help facility) find a way to accomplish their desired task.
Scale: Percentage Probability that a defined [Individual Person: Default: Trained Operator] will find a way to perform a defined [Task Type] without reference to any written instructions, other than the help or guidance instructions offered by the immediate system screen (that is, no additional paper or on-line system reference information), within a defined [Time Period: Default: Within one second from deciding that it is necessary to perform the task].
Comment [Intuitiveness:Scale]: "I'm not sure if one second is acceptable or realistic, it's just a guess" <- MAB.
Meter: To be defined. Not crucial this 1st draft <- TG.
Past [System R]: 80%? <- LN.
Record [Mac User Interface]: 95%? <- TG.
Fail [**Trained Operator, Rare Tasks** [{<1/week, <1/year}]]: From 50% to 90%? <- MAB.
Goal [**Tasks Done** [<1/week (but more than 1/Month)]]: 99%? <- LN,
[**Tasks Done** [<1/year]]: 20%? <- JB,
[Turbulence, **Tasks Done** [<1/year]]: 10% ? <- TG.
========== User Defined Terms ==========
Trained Operato: Defined As: Command and Control Onboard Operator, who has been through approved training course of at least 200 hours duration.
Rare Tasks: Defined As: Types of tasks performed by an Onboard Operator less than one a week on average.
Tasks Done: Defined As: Distinct tasks carried out by Onboard Operator.
Usability.**Intelligibility**:
Ambition: High ability for an operator to <correctly> interpret the meaning of given information.
Scale: Percentage Probability of <objectively correct> interpretation(s) of a defined [Set of <Inputs>] by a defined [Individual Person: Default: Trained Operator] within a defined [Time Period].
Meter [Acceptance]: Use about 10 Trained Operators, and use about 100 <representative sets of information per operator within 15 minutes?> - MAB.
Comment [Meter]: "Not sure if the 15 minutes are realistic" <- MAB.
Comment [Meter]: "This is a client & contract determined detail" <- MAB.
M1: Past: [XXX, 20 Trained Operators, 300 <data sets>, 30 minutes]: 99.0% <- Acceptance Test Report from XXX, MAB.
Record [XXX]: 99.0%. "None other than XXX known by me" <- MAB.
Fail [First Delivery Step]: 99.0%? <- MAB.
Fail [Acceptance]: 99.5%? <- MAB.
Goal [XXX, 20 Trained Operators, 300 <data sets>, 30 minutes]: 99.9% <- LN.

========== More User Defined Terms ==========
Acceptance: Defined As: Formal Acceptance Test as defined by our contract with Customer XXX.
First Delivery Step: Defined As: By end of November this year (The results of the first evolutionary result cycle will be integrated into the system and will be producing useful results).

5.9 Diagrams/Icons: Scale Definition



Note: The keyed icon for scale is '-I-I-'.

Figure 5.5
A representation of multiple performance and resource attributes showing goal and budget levels respectively. The 'point' of the icon goal and budget symbols indicates the level (reference needs to be made to the Scale to interpret the numeric value). One constraint, a Fail level, is shown on the resource attribute for Financial Budget [Stakeholder A]. The lines of the arrows represent the scales of measure (divisions along the scales are also marked).

5.10 Summary: Scales of Measure

Quantification of all performance and resource concepts must be taken seriously. Ideally, you need to have a corporate policy that all such ideas will be expressed quantitatively at all times. Nothing less will satisfy 'the need to be the best' in a fast-changing competitive world.

Here is a summary of the key ideas about *scales of measure*:

- you can and should always define a scale of measure for any system critical variable performance or resource attribute

- defining a scale of measure is a teachable practical process
- specification of a scale can be done using embedded qualifiers, which makes it more immediately powerful and also reusable in other projects
- most scales of measure are tailored variations of a generally applicable set of scales (like Usability and Maintainability). Once you have learned the general set, it becomes much easier to generate useful scales as needed for variations
- qualities do not have to be expressed 'qualitatively' (for example, using words like 'high security') – they should be quantified for serious competitive engineering
- an organization should make a library of useful scales of measure for its area of interest
- really good scales of measure are tailored – truly general scales (like 'volts') are not likely to be what you need for best competitiveness
- scales of measure in requirements are the foundation of understanding any design or architecture impact on that requirement – both when it is being considered, and then when it is being implemented in practice.

> If you think you know something about a subject, try to put a number on it. If you can, then maybe you know something about the subject. If you cannot then perhaps you should admit to yourself that your knowledge is of a meagre and unsatisfactory kind.
>
> *Lord Kelvin, 1893*