Version 0.2 March 10 2017

Quantification Versus Measurement

A course participant (HH, [4]) was concerned that we emphasized the specification of the 'Scale of Measure' [1] for requirements. He was concerned that we downplayed the necessity of careful and credible measurement [1] of delivery of the requirement.

Well, it is true that we believe that the requirement specification process needs to primarily focus on really 'good' value [1] definitions, using scales of measure, not management BS.

Now, just because we prioritize the value quantification, in our teaching and our planning process; that does not mean we do not care about sufficiently good measurement at a later stage. We do.

But our course participant explicitly stated, as many other think and do, that he would consciously choose a scale of measure for a stakeholder critical requirement, 'because it was easy to measure'. This remark caused us to take 'violent' exception to that idea. Never!

Quantification: clear communication and definition of a Value

The 'Scale of measure', is primarily a 'definition'. But it is a special class of definition. Scale is a definition of a variable value or resource. Scale is a tool that guarantees that we can use numbers to communicate ideas about that value's specifications (what is the required level), and in time, about that value's current status (have we actually reached the required level of this value?).

Scales have the *primary purpose* and the property, that we can attach any number of useful numeric values *(current status, state of art, competitive levels, past history, market trends, short term targets, long term targets, targets for high value stakeholders and tasks - for example [1])*, to **help manage** this *critical* value.

For example (in 'Planguage' [1]:

Usability.Intuitiveness:
Type: Top Level Critical Product Quality Requirement.

Scale: The % of [Users] who can Successfully Learn and Complete a [Task].

Goal [End next year, Users = Internet Mobile App Users, Task = Order Product] 97%.

Defined 'Scales', are a *basic tool* to help us *communicate* a 'value requirement' *clearly* and *unambiguously* to all parties concerned.

All stakeholders, all developers, all management. 'Scale' helps defeat the the pervasive and chaotic 'management BS' level ('Much better security', 'Modernization', 'Lean and Agile') where all parties, and all individuals either *make a wild guess* as to the requirement, or *ignore* the critical requirement completely, in favour of building a probably failed IT system, at great cost.

Google 'failed IT projects! (221 million hits today)'.

We teach and practice, that <u>absolutely all</u> critical value variables, can and <u>must always</u> be expressed quantitatively. otherwise they will not be clear and the project will be anarchy and chaos, ending in the all-too-common failure.

Most people do not have this *'quantification of all values'* culture. Most universities do not teach managers and engineers to do this. most people do not know how to do it. many people believe that some values are 'soft', cannot quantified, and must be expressed by nice-sounding words only *('more effective service', 'much higher security', 'enhanced creativity').* They are objectively wrong. They are not stupid people. They are just poorly taught, poorly led and in a poor culture. But since they are not stupid, they can be taught better and led better.

The simplest way for anyone to see the truth of this is to Google the variable, with the word 'metrics' or measures' after them. Try 'usability metrics', 'security metrics', 'creativity metrics', service efficiency metrics' or 'intuitiveness metrics'. Anything you like of values, and you will discover that experienced experts will give you good free advice on how to quantify!

Go on. Try this now on your smartphone. use any value idea you like! there is always plenty of good examples and ideas on the first page, and there ire often millions of hits. There is not called 'no answer', 'never been done'. Assuming we use prevalent terminology, not wholly new artificial words not found in dictionaries. 'Bollicock Jabber metrics' does not work!

Quantification is a paradigm shift for many cultures

So, Scale, is a partial definition, a partial requirement specification. Scale is however a *paradigm shift, a really big culture change,* from the pervasive blah blah blah, or 'Management 'Speak'. But this 'value quantification' is a necessary minimum step if we are to improve the success rate of change projects, and IT change projects in particular.

Note that the value quantification process is not 'because we want to measure'. Not because we want to do maths or statistics, or logic (although that isa possibility it opens up). We quantify as a basic tool for communicating critical stakeholder value variable - clearly and unambiguously all concerned parties. So that, the distributed and complex team can work together towards a common goal - that cannot be misunderstood. In a word, better COMMUNICATION.

For example (in 'Planguage' [1]:

<u>**Usability.Intuitiveness**</u>:
Type: Top Level Critical Product Quality Requirement.

Scale: The % of [Users] who can Successfully Learn and Complete a [Task].

Meter [for a Sprint]: a representative sample of at least 50 users with at least 5 major representative tasks, to be tested on a same day basis.

Meter [for a Quarterly Customer Release]: thorough regression testing, Beta testing for at least a week. Full test set runs. Automated Regression test after all changes.QA Manager authority to release.

Goal [End next year, Users = Internet Mobile App Users, Task = Order Product] 97%.

Measurement of the Value Delivered: Real World Feedback

Now, having made our point about ***quantification*** of all values, let us turn to the issue of *<u>measurement</u>*. We should separately specify, one or more suitable '**Meter:**' s (Test Processes), for *each* Scale we defined. See the example above.

We do not *have to* do this *initially*. But we ultimately do have to do it, if we want to to verify that we are *moving towards*, and *reaching* our '*planned levels* of the critical value requirement'.

If we are in *Waterfall* mode, we could delay this test planning until towards the end of the project. But, if we are in an *Agile* mode, with many early and frequent 'Sprints' (value delivery steps), then the Meter must be suitable for that *short* step. If the value delivery step is 'a week', then the test process has to fit in to the week's value-delivery-step framework.

One of our clients (<u>confirmit.com</u>) decided to arbitrarily limit step measurement to '30 minutes per week'. Or, 'no more than overnight' - which was practical; because Microsoft Usability Labs offered to measure overnight for free, all 10 usability sub-requirements (like Intuitiveness, Intelligibility, see [1] chapter 5 for detail on these). This was *sufficient for purpose*. It gave indications that the value was more-or-less delivered, or not.

They had more-serious acceptance testing every 12th weekly cycle, before quarterly release of 12 increments, to the worldwide marketplace.

The defined Scale is a primary and critical specification, that will determine *almost everything about the project.* This includes Architecture, detailed requirements, rest planning, contracting, progress reporting, customer documentation, marketing, cost and time estimation.

The selection of test processes (Meter spec) is useful, but nowhere near as critical as the choice of Scale. Test processes need to be sufficient for purpose, and after that as little costly in time, money and people as possible.

The notion of a ‘sufficient’ test process depends on the *multiple objectives and constraints* of the test or other quality control process. The exact set of the multiple requirements (objectives and constraints) depends on the development or maintenance stage, stakeholder requirements (including laws and contracts), corporate politics and culture (think NASA, or Intel) and many other factors.

The actual determination, and final selection of a suitable test process can, in an advanced engineering environment (like Boeing, Intel, NASA) be viewed as an engineering problem in itself.

A set of maybe 10 or so quality and performance requirements, together with 2 to 5 resource limitations, and perhaps other constraints (legal, contractual, policy) are the inputs to the design process. These test values and test resources can be all quantified, like any other values [1]. Then a test design engineer will find one or more test or other QC processes (like Inspections, Reviews) that alone or in combination might satisfy the multiple value objectives, while not exceeding the resource limits, or violating other constraints (like legal, contractual).

The set of test designs that gives sufficient value and then ‘best value for costs with regard to risks and uncertainties’ should be selected for trial implantation. A planning device such as an Impact Estimation Table [1] can be used to numerically evaluate all these factors.

Some Useful Attributes of a Test or Measurement Process

The measurable and quantifiable quality and other performance factors that might be included, if they are considered useful or critical to evaluate could include: (I brainstorm freely). Examples of Quantification (Scales of measure suggestions, or templates, or patterns) of many of these will be found in [1].

1. accuracy
2. repeatability or replicability
3. automation
4. credibility
5. scalability
6. usability
7. security
8. safety
9. maintainability
10. setup speed
11. execution speed
12. degree of automated analysis of results
13. comprehensiveness (coverage)
14. legality
15. Technical debt incurrence
16. Portability
17. Intellectual Property rights
18. and more

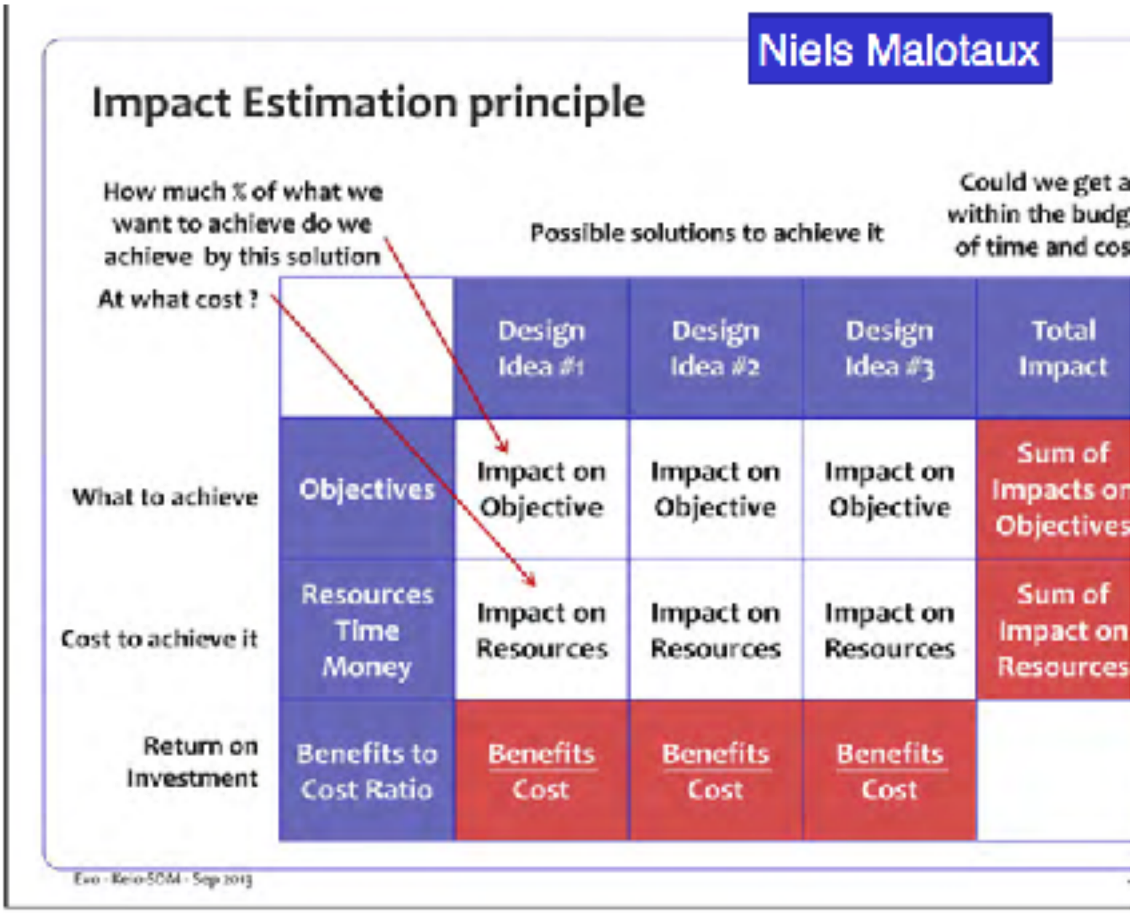

Figure 3: The structure of an Impact estimation table for evaluating test designs against test objectives and resources. Courtesy of N Malotaux.

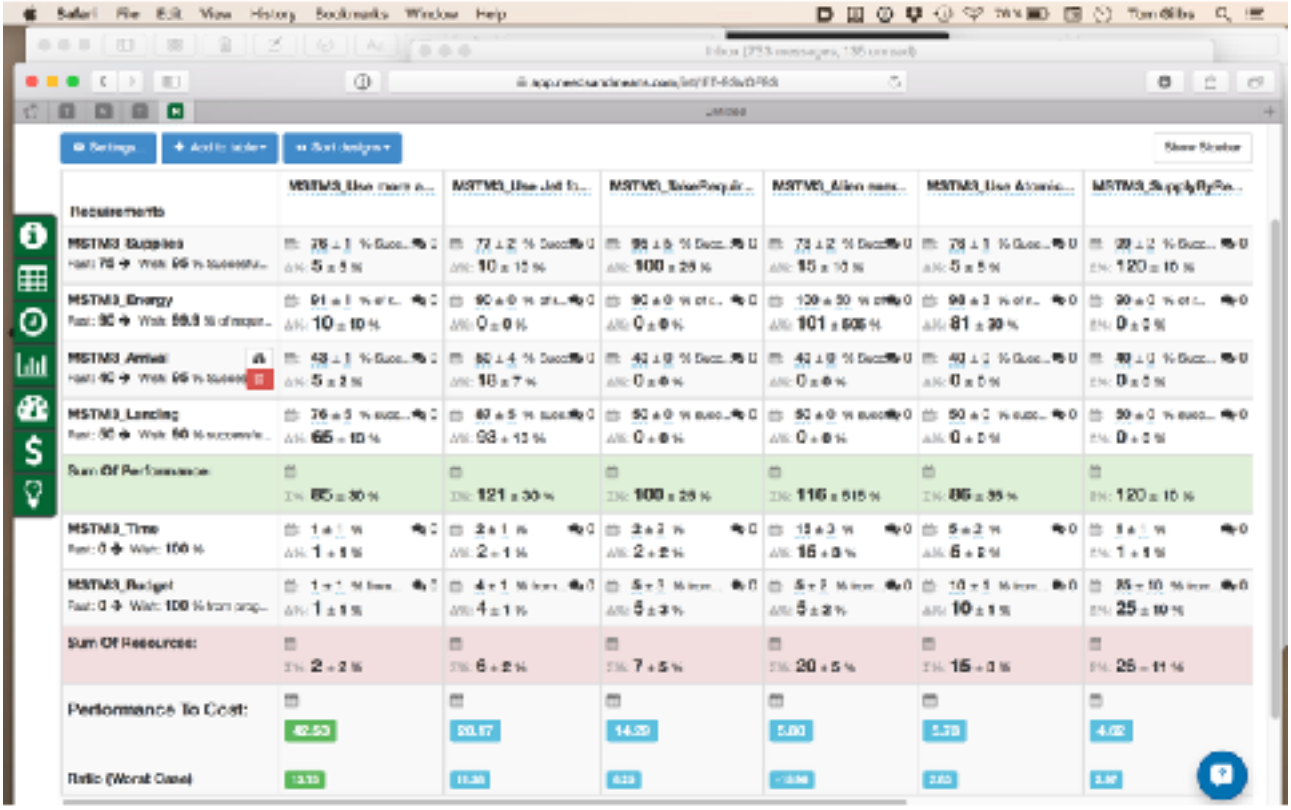

Figure 4: a real Impact Estimation table, using the needsandmeans.com tool. This is intended to more realistically show the format for evaluating test designs that I am suggesting. This format could be used for advanced test planning. This particular instance is from a Gilb student problem, Mars Landing mission, at the Lviv Business School, Ukraine.

There are very few instances of test planning that I know of that go so far as to seriously deal with these factors in a numeric engineering way, as exemplified by use of an Impact Estimation table to keep track of the test planning process. Most test planning is done by culture, rule, tradition intuition and experience, as far as I can see. But certainly software testing as we know it is in very early stages of maturity, and is *notoriously costly*, while being *embarrassingly ineffective* , at the 50% ±20 % level of find problems.(see [2] and Capers Jones many books and papers [3])

No doubt, whatever the engineering evaluation, of a test design initially, we would need to learn incrementally at each agile value delivery step, and modify both our test design, and our objectives and constraints as we go along.

fly me to the moon asap

I like to make the main point here, about the sequence and semi-independence of Scale (quantification) and Meter (test measurement) parameters by the following problem statement.

"I want to get to the moon and back in 1 second'

The quantified requirement is very clear. At least much clearer than, "Fly me to the moon on gossamer wings (old song) asap". (Frank Sinatra Fly Me To The Moon - YouTube https://www.youtube.com/watch?v=mQR0bXO_yI8)

In fact it is so clear, that we all know it is currently well outside the state of that art.

ASAP is possible, but 1 second is not. Not even Beam Me up Scotty works that fast.

Consequently we would delete or modify the requirement, until we were within the state of the art.

But there would be no point in developing Meters or Test Processes to measure if we had actually flown to the moon so quickly.

The point is that quantification, for communicating a value or quality is effective in itself for some useful purposes. Planning the measurement is a *possible* followup; if and when we need feedback on the implementation on of a state of the art architecture or design.

Her is summary of design logic to put the value quantification in a wider context.

The Basic Design Steps Logic: a summary [5, for more detail]

1. Constraints determine scope of environments.

2. Environment Scope helps identify stakeholders.

3. Stakeholders have values and priorities

4. Values have many dimensions

5. Stakeholders determine value levels

6. Design hypotheses should be powerful and efficient ideas, for satisfying stakeholder needs

7. Design hypotheses can be evaluated quantitatively, with respect to all quantified objectives and resources

8. Designs can be decomposed, to find more efficient design subsets, that can be implemented early

9. Designs can be implemented sequentially, and their value-delivery, and resource costs, measured

10. Designs that unexpectedly threaten achievement of objectives, or excessive use of resources, can be removed or modified.

11. Designs that have the best set of effects on objectives, for the least consumption of limited resources, should generally be selected for early implementation.

12. A design increment can have unacceptable results, in combination with previous increments, and they, or it, might need removal or modification

13. When all objectives are reached, the process of design is complete: except for possible optimization of operational resources, by even-better design.

14. When deadlined and budgeted implementation-resources are used up, it might be reasonable to negotiate additional resources; especially if the incremental values are worth the additional resources.

METRIC PRINCIPLES:

I would like to summarize these 'quantification' ideas in terms of some 'principles'.

1. Value-Requirement Quantification is a necessary and powerful way of clarifying critical ideas.
2. All improvements can be expressed quantitatively: both benchmarks, constraints and targets.
3. The most important criteria for defining a value with a 'Scale' of measure, is that the defined scale is very closely aligned with the real stakeholder values; and will serve to guide planners and engineers towards really useful value.
4. We should avoid defining a Scale just because it seems easy to measure in practice; you risk getting a wrong value!
5. There are always many useful ways to measure and test, any value level, on any defined Scale of measure; you need to select or design a 'Meter' (measurement process) which is sufficient for purpose, and the lowest-cost idea of those, which is sufficient for purpose.
6. One defined Scale of Measure (or 'quantification scale') can have *any useful number* of defined Meters, for different times and situations.
7. The purposes of any Meter (test process) can be described by a useful number of quantified values (like accuracy, credibility, costs, legality), and this can lead to a systematic engineering process to find , define, and plan one or more testing processes; perhaps using an Impact Estimation Table to model the problem (see [1] Competitive Engineering for IET info).
8. The costs for any testing process should never exceed the total set of values of carrying out the process; do not measure to useless extremes - you are doing *industrial engineering*, not 'academic science for Nobel Prizes'.
9. The path to finding useful scales-of-measure, for any value, are mainly *common sense* and *experience:* but, if you feel something *cannot be quantified* try to Google 'XXXXX metrics' (XXXXX being name of a value like 'Security'), and find *many good patterns* and *practices,* immediately for all conceivable critical values of interest. Nothing new under the sun!
10. *Quantification*, and consequent *measurement* of critical values, is an *engineering approach* to planning; and is a necessity for serious, high-quality, successful projects. If you are satisfied with

the conventional poor quality, unhappy stakeholders and project failure: you do not need these methods.

[References]:

1. T Gilb, "**Competitive Engineering**", 2005. free Digital book download https://www.gilb.com/p/competitive-engineering.
This technical handbook defines our concepts in detail, and is a fairly formal definition of our Planning Language ('Planguage'). See Scale, Meter, Value using a pdf search. An alternative source, at 'management level' is our 2016 book-manuscript 'Value Planning' (see gilb.com)

2. **Lean QA, Lean Quality Assurance. T Gilb.**
Slides for 4-hour Seminar
Jan 21 2016, Quality Week Conference, Vienna
http://www.gilb.com/dl870

3. Capers Jones: http://www.namcook.com (a weath of resources)

Defect Removal Data in Spreadsheet
An *excellent and impressive* fact summary of the effectiveness of a wide range of QA and QC methods, based on huge industrial database of many years.
http://www.gilb.com/dl234

See also
Jones, Capers & Bonsignour, Olivier; The Economics of Software Quality; Prentice Hall, 2011

Jones, Capers; Software Engineering Best Practices; 1st edition; McGraw Hill 2010

Jones, Capers: Applied Software Measurement; 3rd edition; McGraw Hill 2008

Jones, Capers: Estimating Software Costs, 2nd edition; McGraw Hill 2007

Jones, Capers: Software Assessments, Benchmarks, and Best Practices; Addison Wesley, 2000

4. HH = (with permission) haagenhasle@gmail.com, Hågen Hasle. Thank you!

5. Gilb, T., Design Logic paper: The Logic of Design: Design Process Principles.
http://www.gilb.com/dl857   (might be published at BPTRends.com in 2017)

6. This paper is posted at
Quantification versus measurement paper
http://concepts.gilb.com/dl895
March 10 2017