WICKED PROBLEMS ARE POSSIBLY SIMPLE; IF YOU KNOW HOW

Tom’s Personal View of Evil
23 June 2016, GilbFest

BASED ON T GILB PAPER

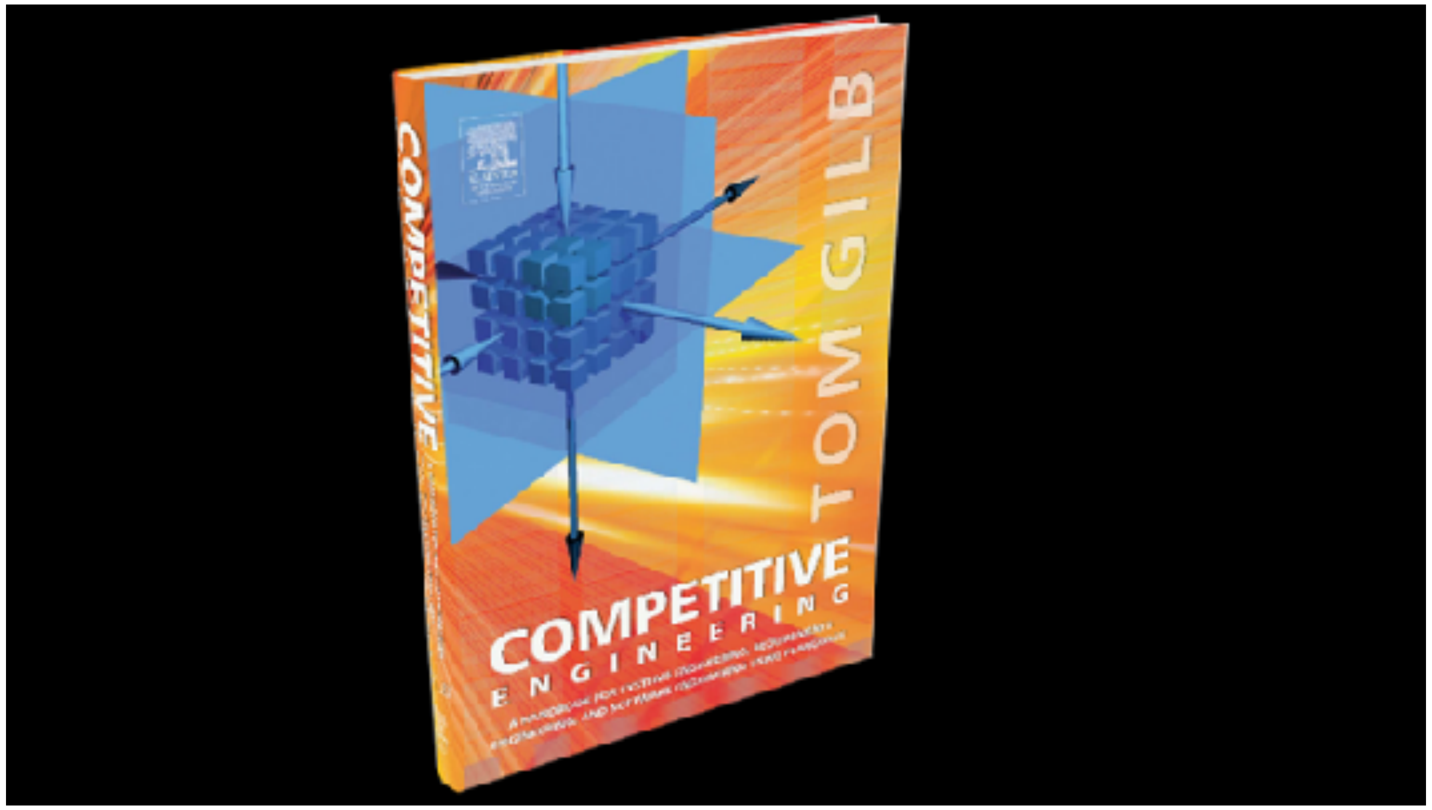
“CONFRONTING WICKED PROBLEMS:

and some *Planguage* Tools to deal with them”

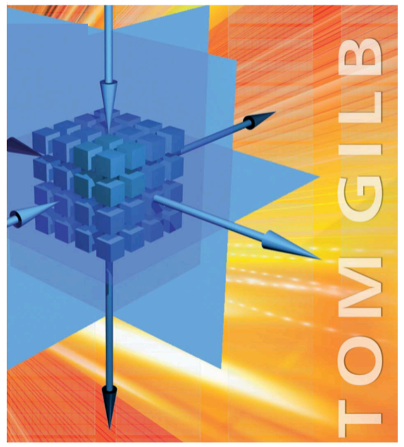
10 January 2016,
gilb.com/dl866

How Planguage Tools

Help Whack Wickedness.



Value Planning



Practical Tools
for
Clearer Management Communication

W1. There is no *definitive formulation* of a wicked problem.

Planguage (The Planning Language) does not need or expect a 'definitive formulation' of a problem.

Planguage allows you to specify any set of *problem statements* (value objectives, constraints, assumptions, constrained strategies, budgets, deadlines, stakeholders) that might be useful.

They can all be modified at any time. They can be versioned. They can be officially sanctioned or approved, until further notice. They can be quality controlled. The quality , relevance, correctness and usefulness of any class of problem statement can be gradually enhanced.

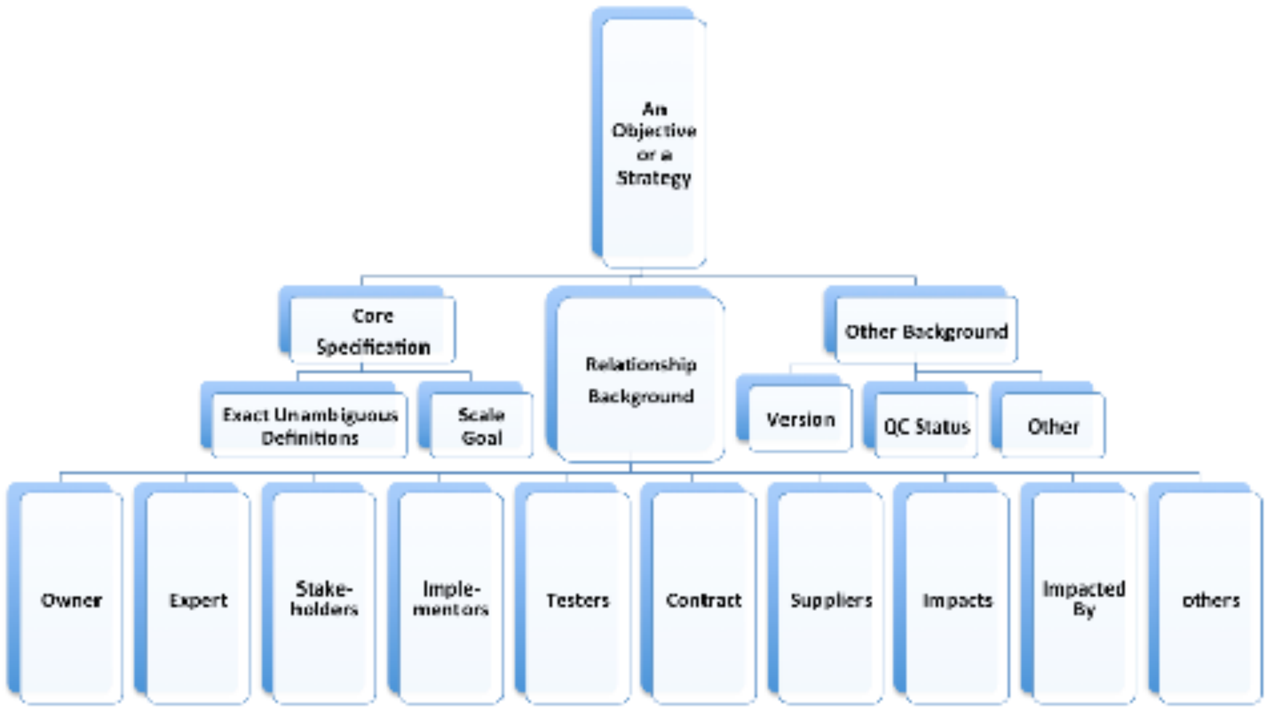
Problem statements can easily be integrated with each other, and their relationship to solutions (strategies, designs, architectures) can easily and automatically be mapped and tracked.

Problem statements can be directly and measurably related to emerging value delivery, and costs: giving real time feedback to the planning model.

A rich set of *background* specifications is expected and encouraged for all problem statements. These include such items as issues, assumptions, constraints, sources, evidence, risks, stakeholders, and very much more [VP 2.2, 3.1, 4.2 for detail].

The *background* specification for a problem statement (like an 'objective') does not change the *core problem specification*. But it enables us to sense the larger and more complex relationships involved (for example multiple *risks* and *stakeholders* for every single problem statement).

Background specification triggers and motivates us to analyze deeper and improve our view or model of the problem space.



W2. Wicked problems have no stopping rule.

Planguage makes no assumptions about stopping a development, or the existence of a 'final state'.

Planguage assumes that the systems it is planning for, already have a life in the real world, and will continue to have a life for the foreseeable future.

Planguage is all about high-priority incremental improvement, towards the current long-term objectives, using resources actually available.

The Planguage planning process is merely a tool for keeping track of concerns, and solution ideas.

The tool is used to keep track of the current state of the system, from any interesting, and all useful emerging, multiple viewpoints.

Planguage assumes conflict and change are normal, and natural: and tries to make the best decisions in that light.

The nearest thing we have to 'stopping rules' (knowing when to quit planning, or investing in change) are locally formulated policies, such as:

Stop when the next cycle of change is not profitable enough (VP 4.7, 5.9). Stop when no credible solutions are on the table. (VP 4.9, 4.8, 5.8)
Stop when planned results have repeatedly not been delivered (VP 4.5).

W E Deming taught me that the Plan Do Study Act cycle (PDSA) was expected to continue, 'as long as there is competition'. We do not think in terms of any 'big stop': just focus on smart prioritization.

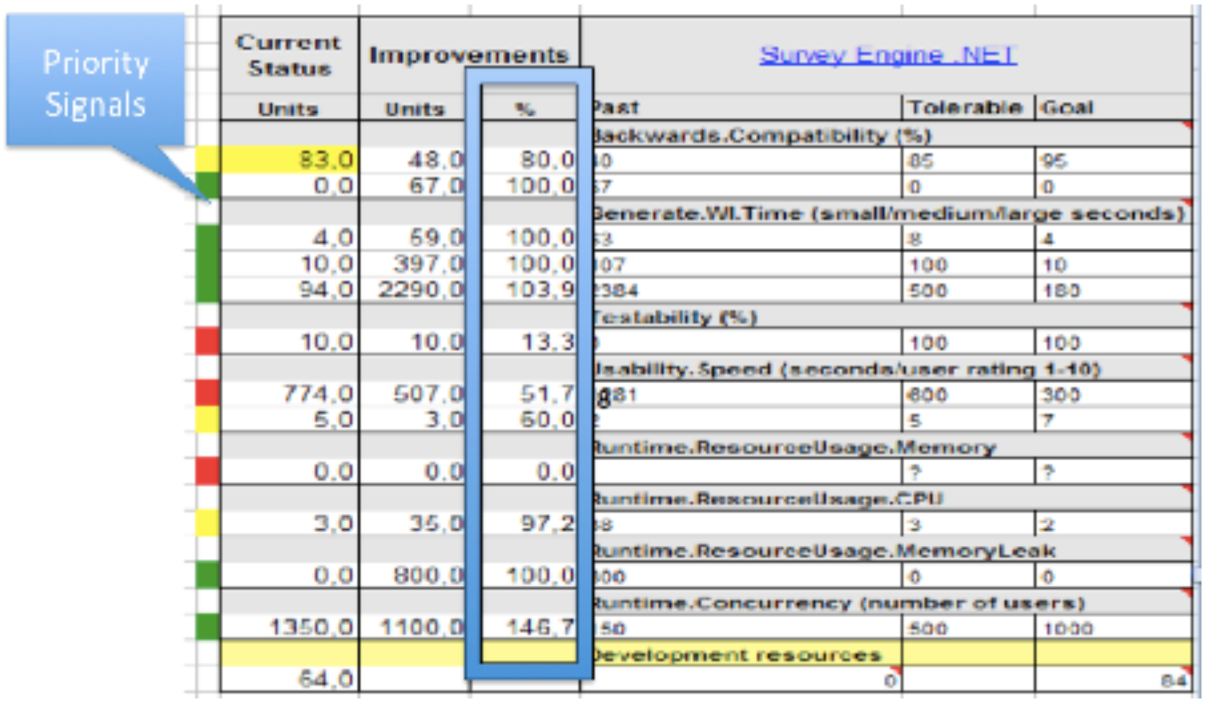
Planguage is unusually quantified regarding problem statements (VP Part 1, Chapter 1). All values and qualities are normally quantified. No management BS allowed [10].

So the quantified worst-case levels,and target levels of a desired value, give us a very specific device to know when to stop: when to stop planning, when to stop inventing, when to stop delivering improvements, when to stop and NOT deliver changes at all.

Planguage has a rich variety of tools and specifications for stopping, when that is appropriate. For a rich variety of reasons and conditions. But it has a 'lust for life' to try to keep delivering value to stakeholders. And it makes that possible by quickly stopping low-priority activity. (VP Chapter 6).

Your own culture needs to decide on your own values and priorities regarding when to stop and go.

Planguage has rich built-in specifications that even automatically point out red lights and green lights. Planguage specifications can compute what to prioritize (Green) and what to stop (Red Light). (VP 6.7)


Priority Signals

	Current Status	Improvements		Survey Engine .NET		
	Units	Units	%	Past	Tolerable	Goal
				Backwards.Compatibility (%)		
🟨	83,0	48,0	80,0	0	85	95
🟩	0,0	67,0	100,0	7	0	0
				Generate.WI.Time (small/medium/large seconds)		
🟩	4,0	59,0	100,0	3	8	4
🟩	10,0	397,0	100,0	07	100	10
🟩	94,0	2290,0	103,9	2384	500	180
				Testability (%)		
🟥	10,0	10,0	13,3		100	100
				Usability.Speed (seconds/user rating 1-10)		
🟥	774,0	507,0	51,7	881	800	300
🟨	5,0	3,0	60,0	2	5	7
				Runtime.ResourceUsage.Memory		
🟥	0,0	0,0	0,0		?	?
				Runtime.ResourceUsage.CPU		
🟨	3,0	35,0	97,2	38	3	2
				Runtime.ResourceUsage.MemoryLeak		
🟩	0,0	800,0	100,0	800	0	0
				Runtime.Concurrency (number of users)		
🟩	1350,0	1100,0	146,7	150	500	1000
				Development resources		
	64,0			0		84

W3. Solutions to wicked problems are not true-or-false, but good-or-bad.

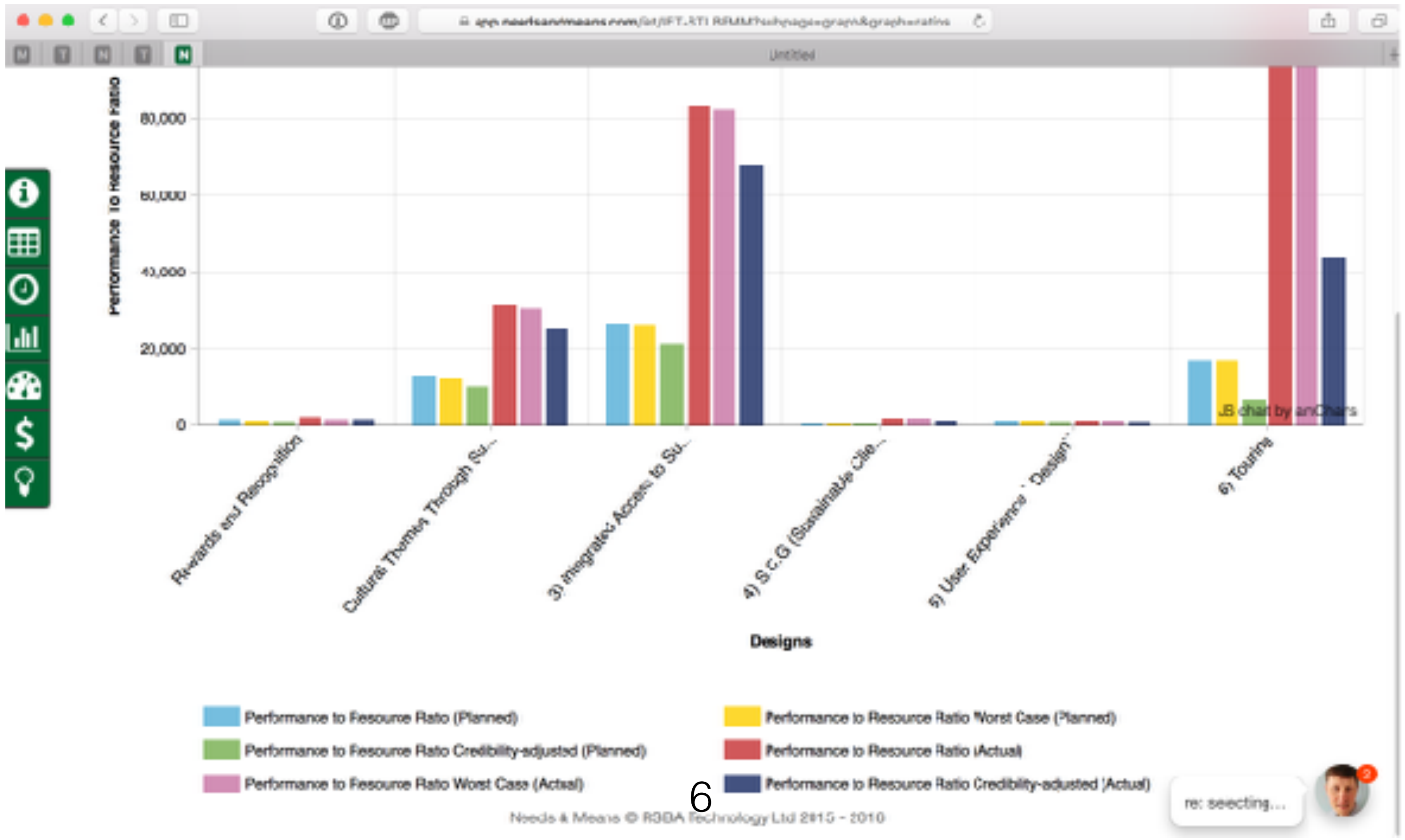
Planguage has no preconceived notion that solutions are 'correct or good' or not.
It *explicitly* recognizes that:
- **Problems** (objectives, constraints) are the best currently available subjective stakeholder compromise.
- Problem specification is subject to constant change pressure.

- **Solutions** (strategies, architecture) are the best available 'hypothesis' as to how to solve a set of problems ('solve' = delivery sufficient [Target level] overall value delivery, within constraints,

at lowest budgeted resource costs; with regard to risk-of-deviation from expectations (estimates). The degree of solution 'goodness' is directly related to the current problem specification.

- The degree of goodness is numerically computed in the Planguage tool 'Impact Estimation table" (sample IET in above VP [8] Figure 6.7). This can be supported by automation as in the example below.

I conclude that **Planguage is well suited to this 'good or bad' aspect of Wicked Problems.**



W4. There is no immediate and no ultimate test of a solution to a wicked problem.

This problem is not particular for Wicked Problems. It applies to all problems, all efforts, all changes. Butterfly Effect: the sensitive dependence on initial conditions in which a small change in one state of a deterministic nonlinear system can result in large differences in a later state. It is tough to make predictions, especially about the future (Yogi Berra et al).

In Planguage, using the Project management subset 'Evo' Value Delivery [7], we do in fact measure, in the short term (typically weekly, as in Confirmit example above [Figure W2], the impacts on all the critical factors that interest us.

In this Confirmit case, we estimate and later measure on a set of critical top level Performance Values, and we estimate and later measure time and effort needed to do the change.[See VP [8] Section 8.6 Getting early short-term feedback.]

Later, for example at quarterly release, addition measurements are made. This takes into account the changes made after the earlier changes. It accounts for the parallel changes made by other teams. it typically is more sophisticated testing and measurement (pre release to world market).

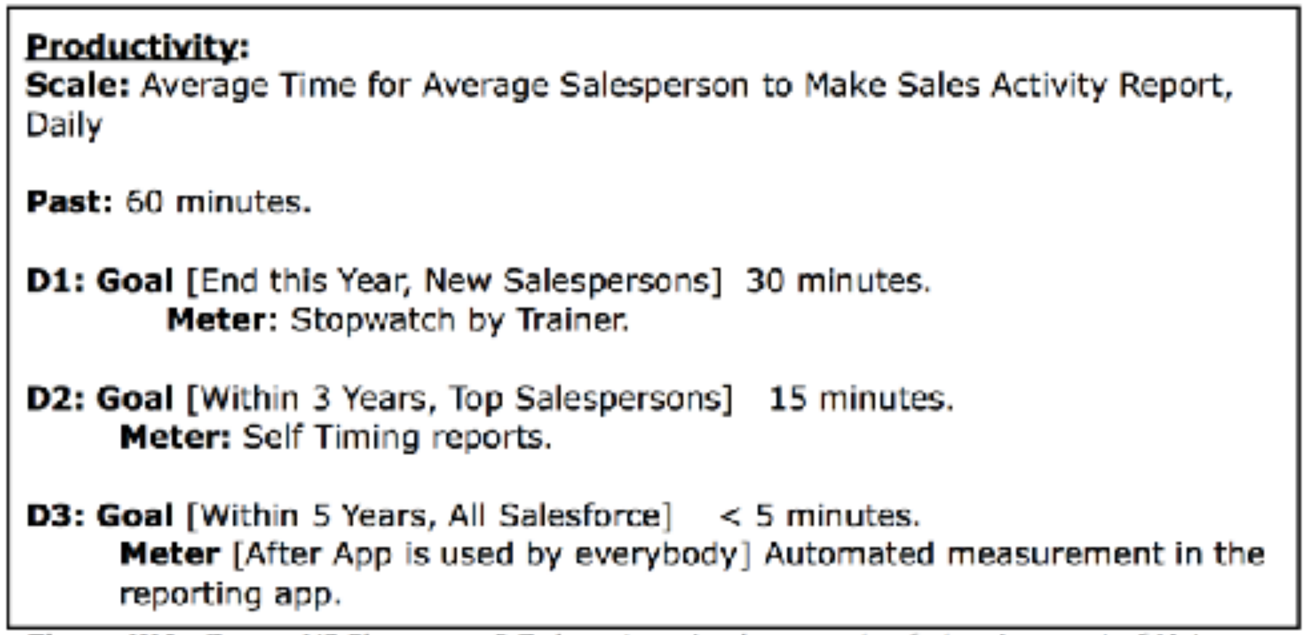
After a release of changes we can *continue* to measure the factors of interest, as they affect real world users of a system. We can certainly expect feedback if they are unhappy!

Finally, in the next round of changes, the critical performance values will *again* be measured, as demonstration that they have held up, or not, over time.

We do not need 'ultimate tests in infinite time'. We need to keep reasonable track of reality in a cost effective manner, and Planguage [Evo] gives us rich numeric opportunity to do so.

All critical problems (of improvement) are always *quantified* in Planguage, or at least 'testable' for presence: that is the basic idea of Planguage.

Reasonable and sufficient measurement and testing is invariably possible.



Productivity:
Scale: Average Time for Average Salesperson to Make Sales Activity Report, Daily

Past: 60 minutes.

D1: Goal [End this Year, New Salespersons] 30 minutes.
    Meter: Stopwatch by Trainer.

D2: Goal [Within 3 Years, Top Salespersons] 15 minutes.
    Meter: Self Timing reports.

D3: Goal [Within 5 Years, All Salesforce] < 5 minutes.
    Meter [After App is used by everybody] Automated measurement in the reporting app.

***Figure W4**. Source VP Planguage 6.7: here is a simple example of planning a set of Meters. Meters are a process for measuring in practice, along a single defined Scale of measure. The Meter statement is usually a rough outline only. The detailed 'test' planning to be done by others, such as system testers. Notice we are dealing with short term and long term measurements here at the same planning specification.*

W4 (continued) I believe one central reason that 'Wicked Problems' ***appear* to be** so wicked, is because we have such a ***poor culture of quantification*** of critical factors.

Words and 'poetry' ('state of the art competitiveness', 'end world hunger') substitute for clear thinking and clear problem specification.

This quantification, and background clarification, does not itself, and alone solve the problem central to Wicked problems (the very complex and voluminous nature of real systems).

But lack of quantification of critical system performance problems makes even short-term and real- time understanding of the problem impossible. But that is NOT a Wicked Problem; it is simply our professional incompetence.

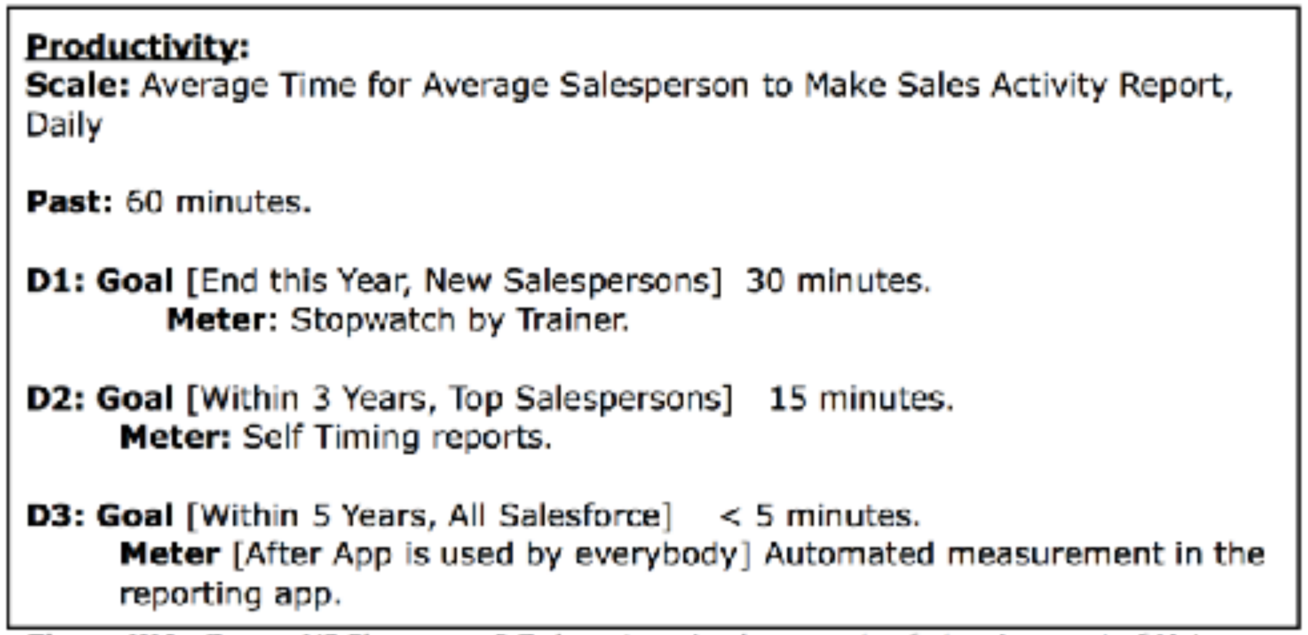
We then *falsely* blame our lack-of-understanding on the 'system complexity': when we in fact have *not even taken very basic steps* to clear the fog in front of our faces (to quantify critical variables).

In conclusion:

1. we can normally get immediate and continuous, tests and measurements, of solutions, in
   relation to clear problem statements, if we want them.

2. we do not need to worry about *unrealistic ideas* like 'ultimate test' of a solution.

'Ultimate tests' would be nice, of course, but they are not necessary, and they are never possible in the real world.



Productivity:
Scale: Average Time for Average Salesperson to Make Sales Activity Report, Daily

Past: 60 minutes.

D1: Goal [End this Year, New Salespersons] 30 minutes.
Meter: Stopwatch by Trainer.

D2: Goal [Within 3 Years, Top Salespersons] 15 minutes.
Meter: Self Timing reports.

D3: Goal [Within 5 Years, All Salesforce] < 5 minutes.
Meter [After App is used by everybody] Automated measurement in the reporting app.

***Figure W4**. Source VP Planguage 6.7: here is a simple example of planning a set of Meters. Meters are a process for measuring in practice, along a single defined Scale of measure. The Meter statement is usually a rough outline only. The detailed 'test' planning to be done by others, such as system testers. Notice we are dealing with short term and long term measurements here at the same planning specification.*

W5. Every solution to a wicked problem is a "one-shot operation"; because there is no opportunity to learn by trial-and-error, every attempt counts significantly.

This is another example of a possibly ‘artificial’ problem which is not inevitably inherent in complex systems.

It might be, but another possibility is that the planner has **simply not learned** to decompose ‘big strategies’ into smaller, deliverable and possibly retractable ‘experiments.

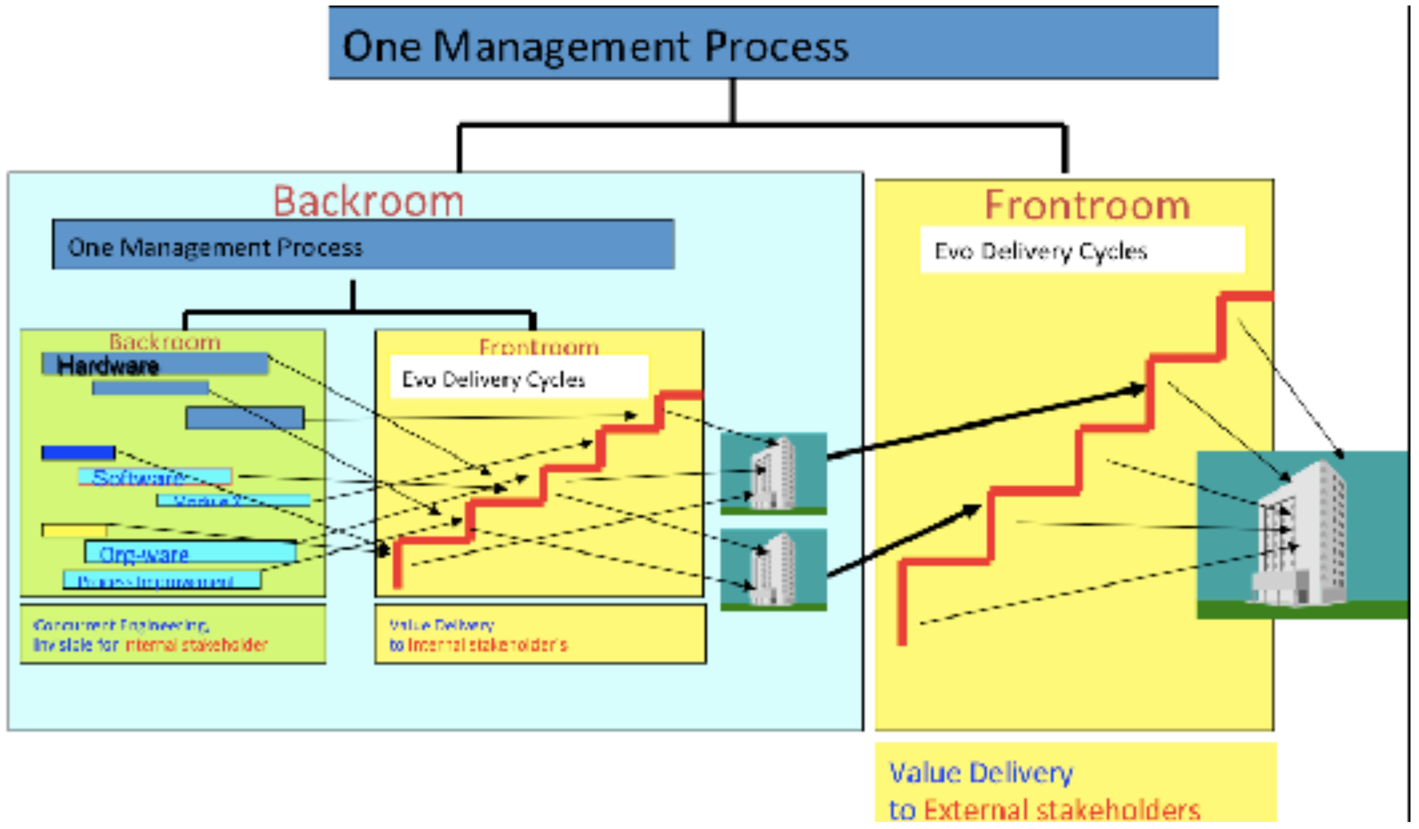
I view this widespread inability do decompose big strategies as ‘professional incompetence’. The incompetence is caused by lack of knowledge, and training, in decomposition.

Notice that decomposing solutions into simple experimental components is fundamental to both scientific experiment and to engineering. And there are some very big hairy problems they tackle. Think ‘Space’ and ‘Universe’.

Planguage tries to deal with this problem of decomposition, at length, with constructive and teachable methods. [8, Evo. and VP Chapter 5. Decomposition (by value, by responsibility) page 363 to p. 415].

Imagination, intelligence, experience, motivation will allow professionals to figure out how to decompose. I had to learn it by practical experience over decades. But most professionals have not learned such methods explicitly. Half of them are in illogical denial (it ‘cannot’ be composed). So it is time to teach the methods, rather than hope people will figure it out in a few decades, personally.

I conclude that some **problems appear more ‘Wicked’ than they really are, because people are not trained in decomposition methods**, which would allow us to avoid the ‘every attempt counts significantly’ problem.



W6. Wicked problems do not have an enumerable (or an exhaustively describable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan.

Well, with this point of view, absolutely all real life problems, about people, culture and technology are 'Wicked'. Again this is an *unnecessary* and *unrealistic* expectation ('exhaustively describable') to real world problem solving.

Unrealistic and Unnecessary:

Even in chess, where the solution space is theoretically exhaustively describable using a computer, there is a time limitation, and even a constraint n real players not using computers in real play. There is too often far too many combinations of play. And this is irrelevant, as long as you either win, or sometimes 'draw'. You do not need all possible solutions, you need a 'pretty good' or 'good enough', **on time**, to meet your deadline (chess clock).

Planguage as a planning tool has a large number of tools to support this concept of 'good enough, on time'. We will explain a few, as a sample of the toolset.

The first concept is what we call a 'scalar constraint'. It is used in problem formulation. For example "The room temperature must be at least 15 degrees C". One Planguage term we use is to call this a Tolerable Level of theValue.

So if at least one potential solution, to the temperature problem are estimated to give us 'at least 15 degrees C', then the solution is theoretically sufficient. There is not need to look for 1,000+ other possible solutions. This logic is built into the Impact Estimation table in Planguage. And you se it in Figure W4 above. If the level delivered is at or above the Tolerable level, we get a 'yellow' light signal. The solution is 'sufficient', to meet minimum requirements.

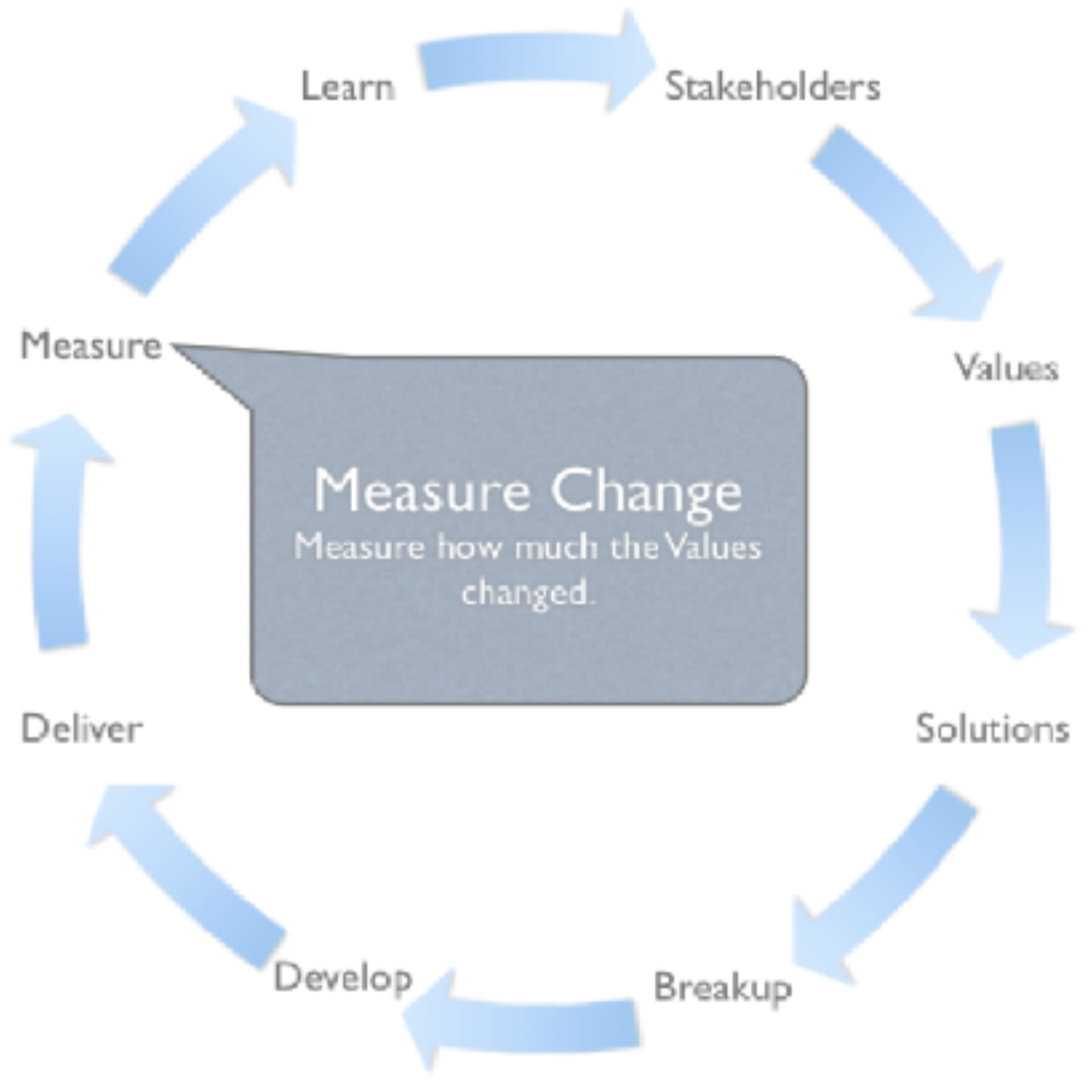
In the next stage of deciding we have enough solutions, we ask in Planguage is the solutions will potentially (later when applied, 'really reach the target levels) reach our Target levels (a formal numeric definition of success and sufficient problem solving). If any set of solutions will reach our success, sufficiency, levels that there is hotpoint is considering the entire solution space exhaustively. That would cost far more than any benefit. it would delay delivery of benefits to the real world in good time. it is silly to even hint that this is 'necessary', to exhaust the solution space at all. Can we use common sense here , please?

of course the above explanation is a simplification, to show the principles involved. Even fairly simple (not especially Wicked) problems require us to think about many other factors, when considering if we have explored the solution space sufficiently. For example costs, interaction between solutions, changes in the stakeholder space, poor implementation of otherwise theoretically good solutions, and much more. I can assure you that these factors are all systematically considered, and we have tools for them built into basic Planguage.

See for yourself. VP Part 1 to 5 (50 pages, free book sample [8,]) and really most detailed in the larger books [8, 7]

Wicked Problems

I conclude that this **'Enumerable Solutions' Wicked Problem characteristic [W6] is artificial, academic theory, of no practical use in the real world. A waste of time to worry about at all. The real problem is finding sufficient for defined purpose solutions.**



W7.1 Every wicked problem is essentially *unique*.

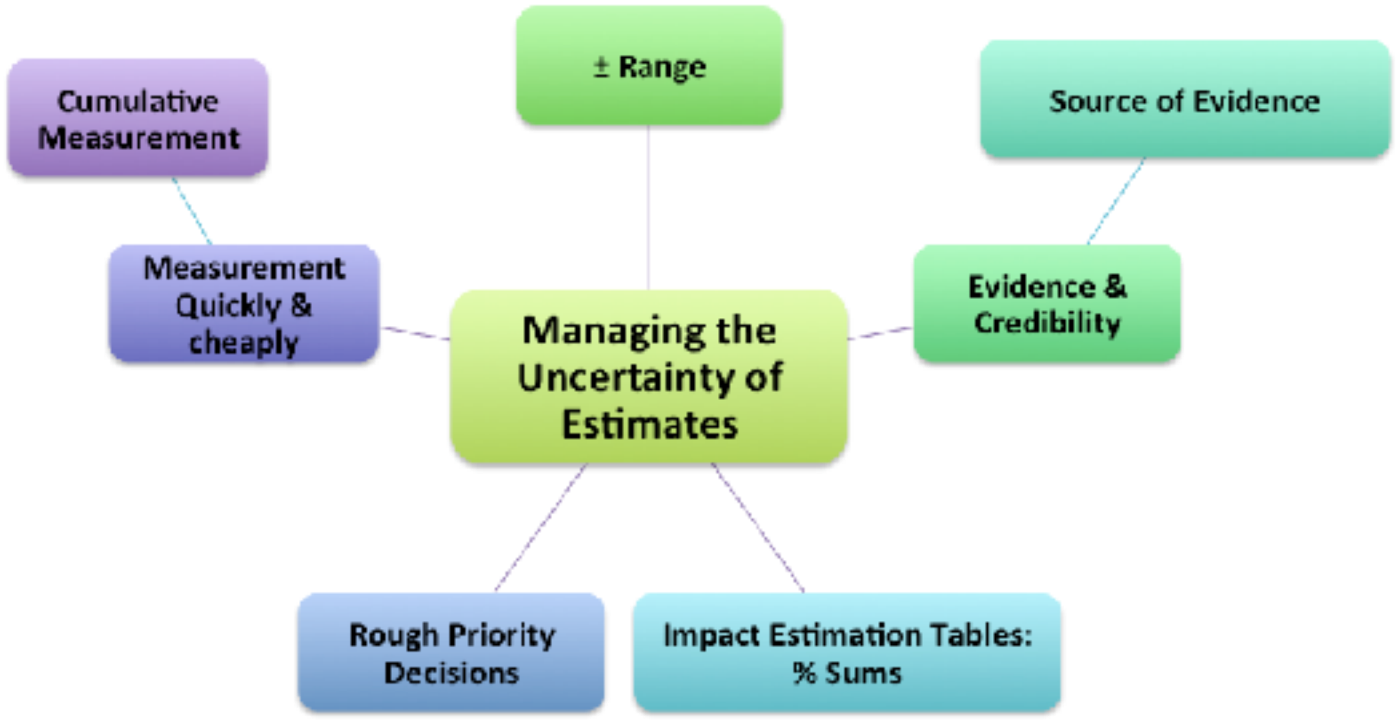
‘Wickedness’: needs an improved definition.

Again, I am getting a bit tired of the fact that these Wickedness characteristics are not especially for ‘Wicked’ problems. Maybe I need to define ‘Wicked’ to my *own* satisfaction?

One immediate thought is that ‘Wickedness’ is *not about the problem itself*. It is the *combination* of ‘the problem’ and ‘the methods-we-know-about; and are willing to use to deal with the problem set’.

And maybe, we need to include, some other factors like resources, constraints and motivations.

The essential ideas are that it is about our real-life current *ability to solve the problem*; not about the problem *itself*. Maybe Wicked projects’, or ‘***Wicked processes***’ (eternal cycles) better capture what we are dealing with here.


± Range
Cumulative Measurement
Source of Evidence
Measurement Quickly & cheaply
Managing the Uncertainty of Estimates
Evidence & Credibility
Rough Priority Decisions
Impact Estimation Tables: % Sums

W7.2 Every wicked problem is essentially *unique*.

‘ Uniqueness is the norm. ‘Identity’ is an impractical

ideal.

Let us bring in ‘obvious common sense’ again.

Surely **absolutely every problem we humans deal with is in some senses uniq**

So what. Absolutely identical problems are not really very interesting. The implicatio of W7 is that if the problem were identical, we might know the solution. So what?

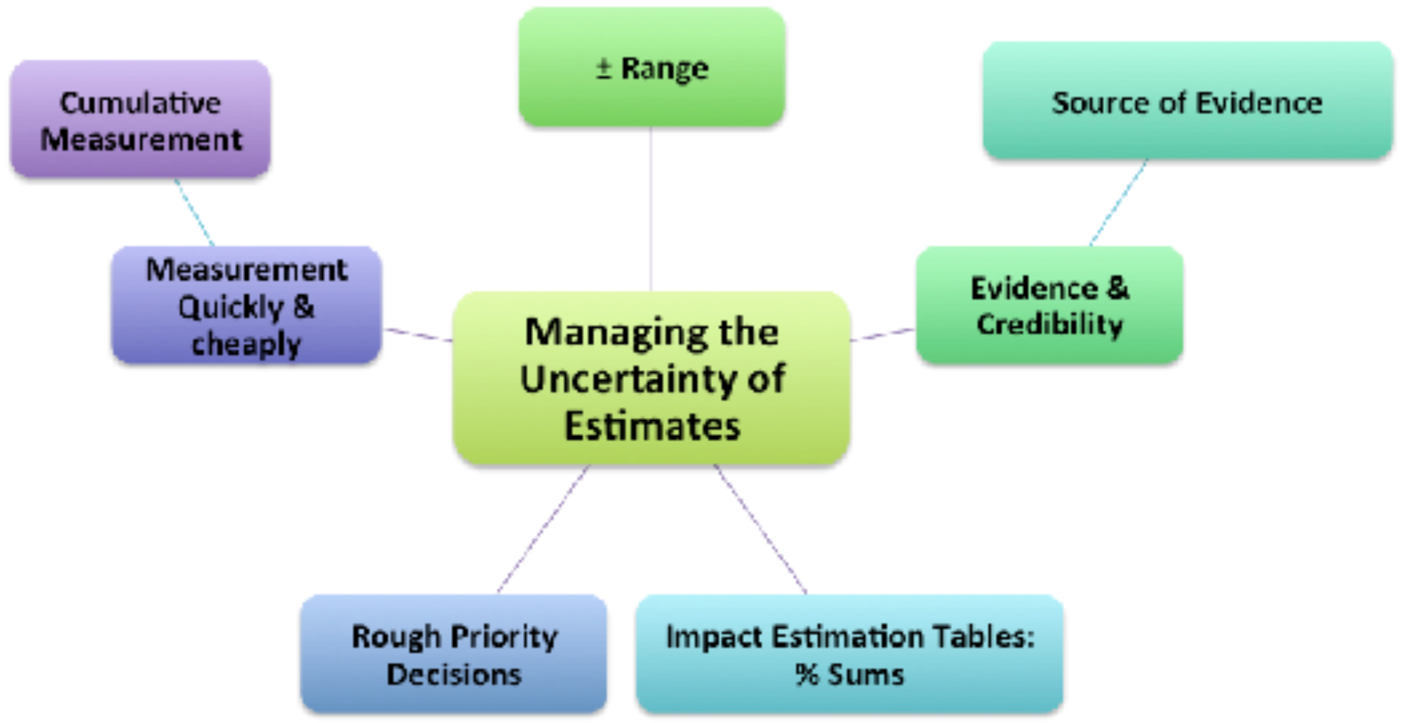
Identical problems, that have already been successfully solved, do not guarantee th the solution used is known, or knowable to us - in *time*. An earlier solution may be secret, hidden, undocumented, or even misunderstood (what the real solution was, opposed to a publicly documented solution).

Anyway, *nothing* is really identical. And we need to find workable solutions if possibl anyway. And it does not really matter if there *was* another known solution that worke once. it may be easier for us to just ‘get something to work’, and move on, than to research, at unknown costs and success of finding it, the ‘real solution used in the past by someone’.

Planguage has no such notion as identical problems, and corresponding solutions. Why waste time asking if there is an ‘identical problem’, anyway. Focus on solving tl problem.

Planguage does have well-articulated concepts of asking for *evidence* of past values and costs for any proposed solution [VP Part 2, 4.4]. The solutions and problems are never identical. We know that. But they do not have to be identical. Just *good enough*.

We are not trying to be identical, but we are trying to improve the probability that we will discover, prioritize, use, and measure - *pretty good* solutions quickly.


Cumulative Measurement
± Range
Source of Evidence
Measurement Quickly & cheaply
Managing the Uncertainty of Estimates
Evidence & Credibility
Rough Priority Decisions
Impact Estimation Tables: % Sums

W7.3 Every wicked problem is essentially *unique*.

In conclusion:

'Unique problems' is **not a *useful* concept.**

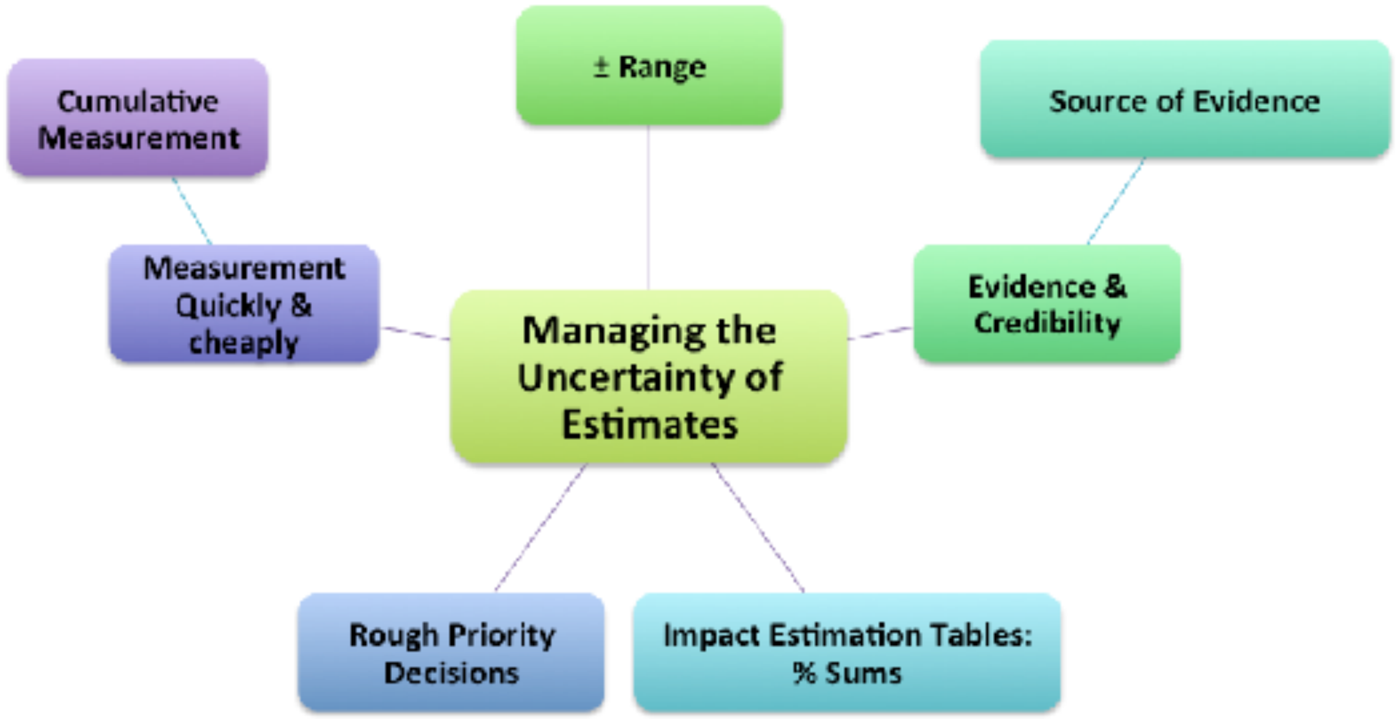
It is not a clear and useful distinguishing characteristic of a problem.

'Uniqueness of identical problems' is not a helpful concept.

We need to

focus on finding a solution stream,

cumulating to a useful potential set, of solutions:



Cumulative Measurement
± Range
Source of Evidence
Measurement Quickly & cheaply
Managing the Uncertainty of Estimates
Evidence & Credibility
Rough Priority Decisions
Impact Estimation Tables: % Sums

W8.1 Every wicked problem can be considered to be a symptom of another problem.

Again, we do not need to bring in ‘Wicked’ at all.

Every problem is a symptom of another problem.

That is just the way things are at any level of complexity.

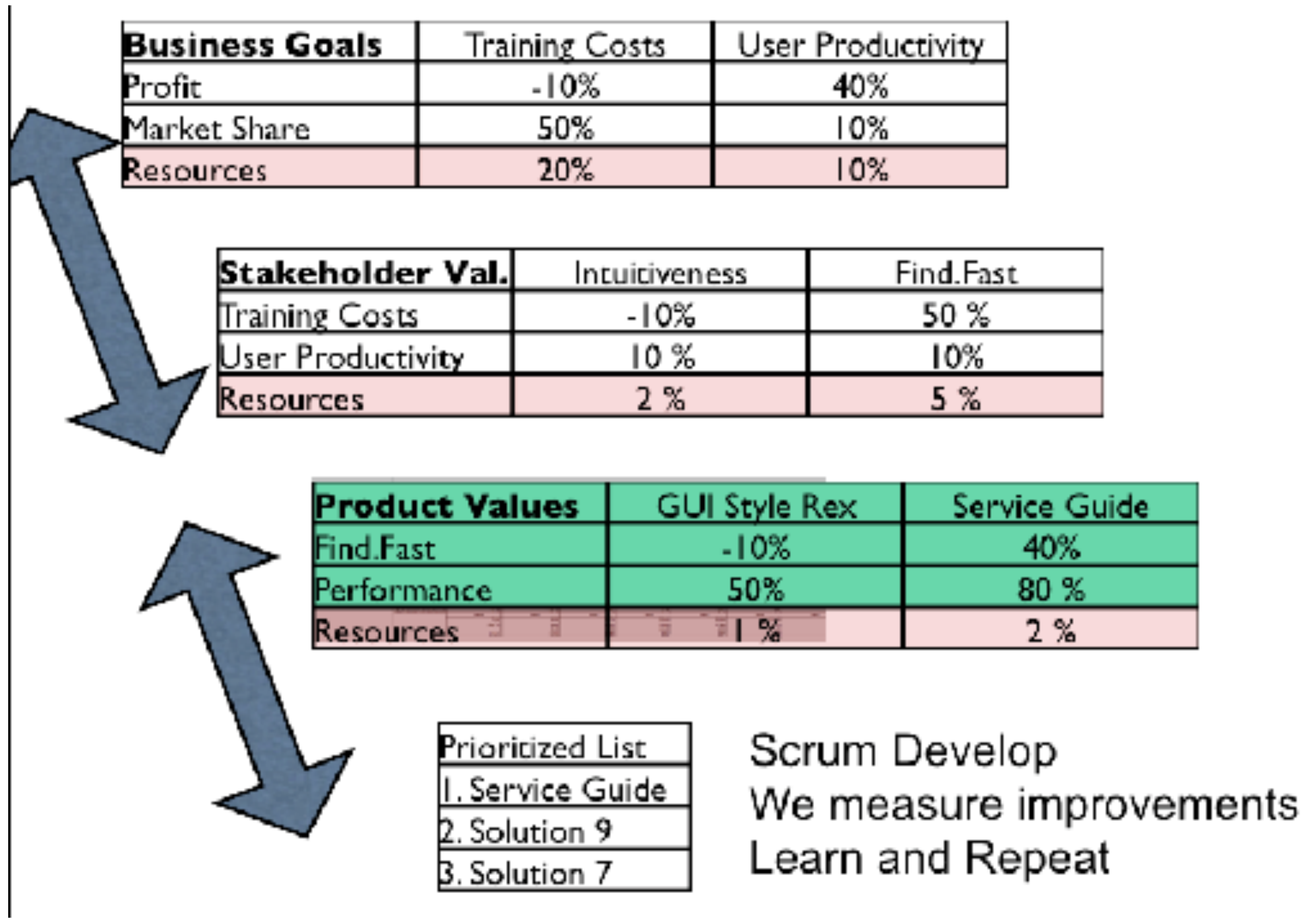
My boss’ solution becomes *my* problem, and my solution becomes my *teams* problem.

Planguage **explicitly acknowledges *many* related levels of concern.**

Stakeholder levels.

Planguage ties these related stakeholder levels *together*, in a variety of ways.

The most interesting method in Planguage is using **Impact Estimation tables to model, quantitatively, the relationships between any set of problems:** any above, any below, or any sideways stakeholder levels.



Business Goals	Training Costs	User Productivity
Profit	-10%	40%
Market Share	50%	10%
Resources	20%	10%

Stakeholder Val.	Intuitiveness	Find.Fast
Training Costs	-10%	50 %
User Productivity	10 %	10%
Resources	2 %	5 %

Product Values	GUI Style Rex	Service Guide
Find.Fast	-10%	40%
Performance	50%	80 %
Resources	1 %	2 %

Prioritized List
1. Service Guide
2. Solution 9
3. Solution 7

W8.2 Every wicked problem can be considered to be a symptom of another problem.

1practical application of this was by Kai Gilb at the Transport Company ‘Bring’ [11].

Figure W8. [11] in order to save a large IT Scrum project that failed initially, (the new system drastically killed sales!). Kai modelled the (obviously, ‘it failed’) ‘wicked system’. He built one Impact Estimation Table (aka Value Decision Table) for the top level of the Bring (Norwegian Post Office essentially) organization. This succeeded to resurrect the system, because it mapped the connection between technology and the higher levels of organizational objectives. The IT Development team was then instructed to focus on developing things that led to business (sales!) success. An extremely *simplified* example is above [For more detail see 11].

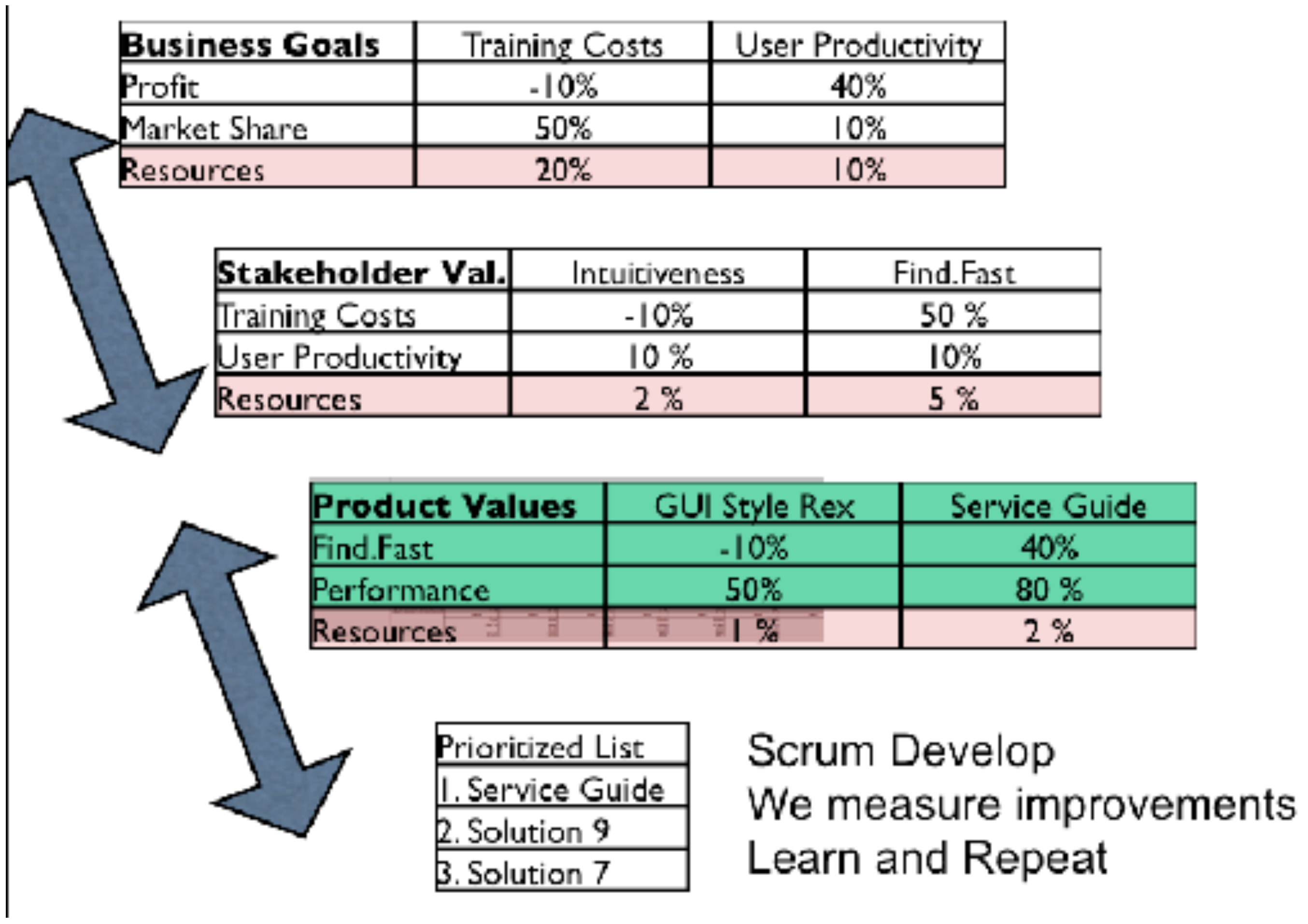
Business Goals: The top management stakeholder level has problems, like *Increase Profit* and *Market Share*. Solutions have been identified (reduce *Training Costs*, and improve *User Productivity*). The expected, estimated, impact of these solutions on the (elsewhere, see Figure W4 for ‘how it looks’) *quantified* Problems, is given by the numbers estimated (later ‘measured as a result) at their intersection. For example Training Costs reduction, if the solution works as expected, promised to move us 50% of the way towards our Market Share objective (the Problem,

Stakeholder Value: These solutions become the the Problem at the next level. The Stakeholder level. Think of these as the 30 or so individual transport companies that had been bought and merged to form Bring. It looks like the Solution named ‘Intuitiveness’ is estimated to contribute 10% of the progress we need towards the User Productivity problem objective. All objectives are, of course, quantified, elsewhere.

Product Val.: At the third level (Product Values), ‘Find.Fast’ (one of the Stakeholder solutions, is considered an IT System objective (a problem statement).

It looks like ‘Service Guide’ is a solution that is expected to contribute 40% towards the ‘Find.Fast’ Problem solution. And ‘Service Guide’ *also* is expected to contribute 80% towards a Performance problem.

Scrum Level: The Service Guide solution will be developed and implemented by the Scrum Team. Hopefully its impact will be approximately as expected, and will impact several levels up towards the Business Goals.



Business Goals	Training Costs	User Productivity
Profit	-10%	40%
Market Share	50%	10%
Resources	20%	10%

Stakeholder Val.	Intuitiveness	Find.Fast
Training Costs	-10%	50 %
User Productivity	10 %	10%
Resources	2 %	5 %

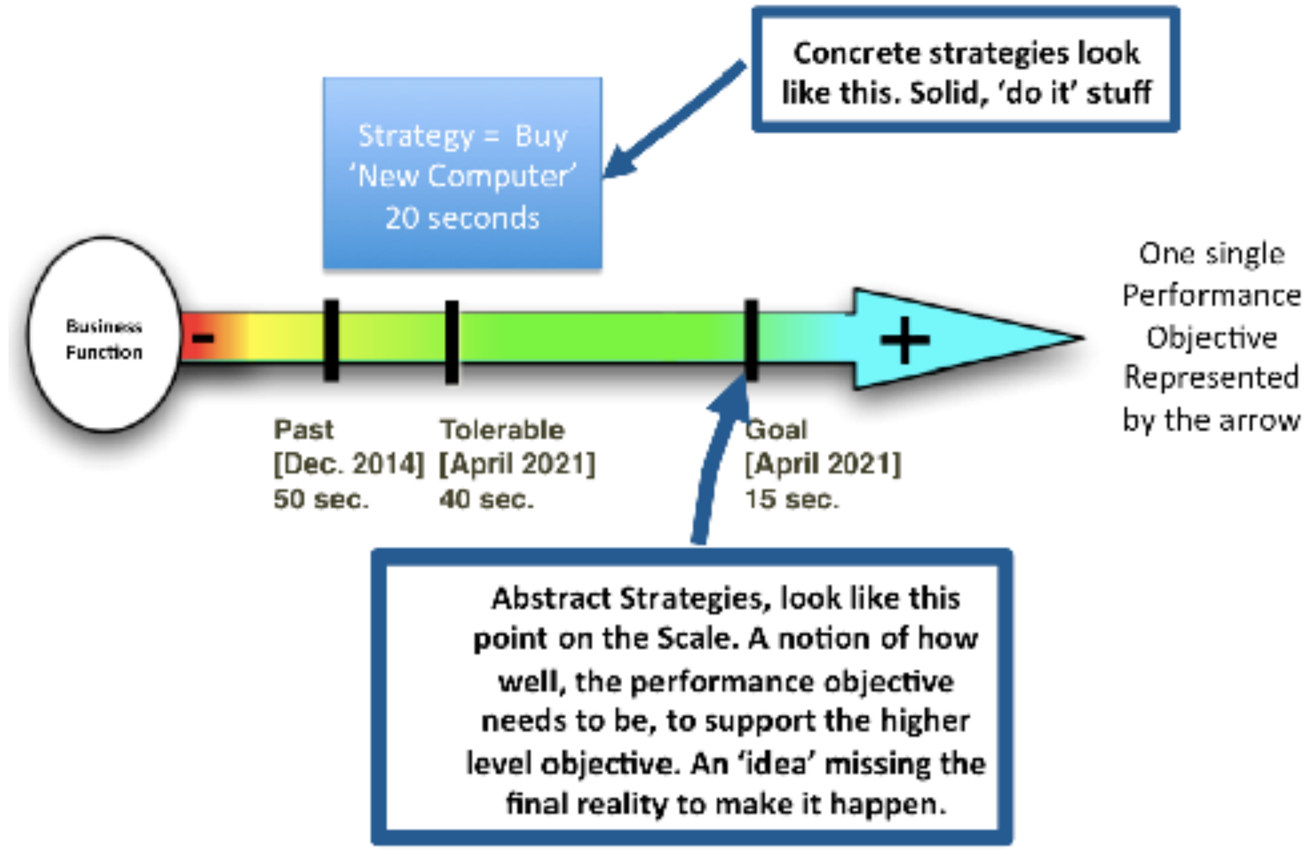
Product Values	GUI Style Rex	Service Guide
Find.Fast	-10%	40%
Performance	50%	80 %
Resources	1 %	2 %

Prioritized List
1. Service Guide
2. Solution 9
3. Solution 7

Scrum Develop
We measure improvements
Learn and Repeat

W9.1 The existence of a discrepancy in representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution.

This is confusingly written up in the literature [1].
Let me try to suggest what it means.


Concrete strategies look like this. Solid, 'do it' stuff
Strategy = Buy 'New Computer' 20 seconds
Business Function
-
+
One single Performance Objective Represented by the arrow
Past [Dec. 2014] 50 sec.
Tolerable [April 2021] 40 sec.
Goal [April 2021] 15 sec.
Abstract Strategies, look like this point on the Scale. A notion of how well, the performance objective needs to be, to support the higher level objective. An 'idea' missing the final reality to make it happen.

If there is more than one way any people can identify, to solve a problem,

<u>that *alone*</u> allows you to classify the problem as 'wicked'. (W9 says)

The actual choice of solution, to a Wicked Problem is arbitrary,

and based on the point of view of the planner.

Normal scientific methods of evaluating

W9.2 The existence of a discrepancy in representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution.

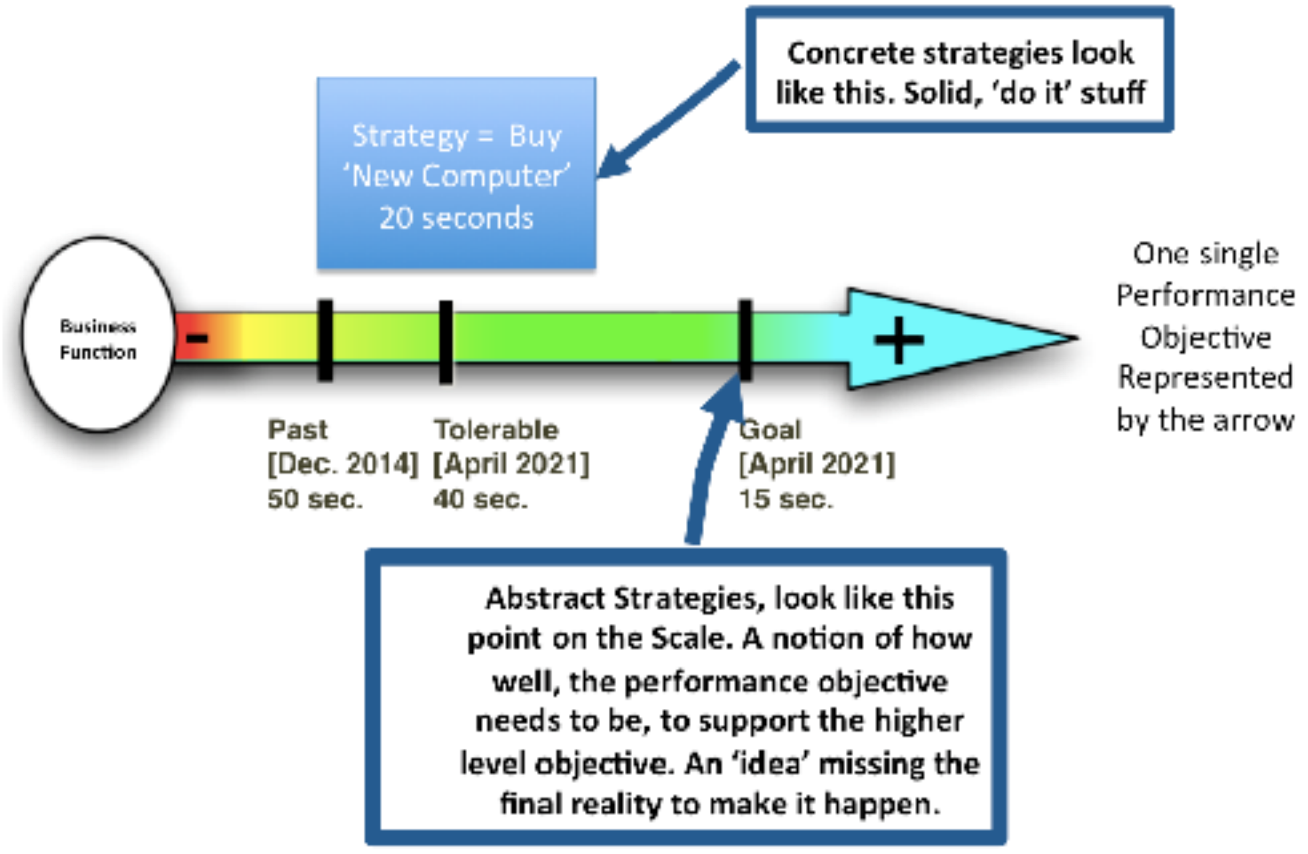
In Planguage:

1. **any solution that works, delivers value for money, and does not violate any constraints, is 'acceptable'. It does not matter that it is one of many possibilities, or that it is a subjective, comfortable, choice by an arbitrary planner.**
2. **we are happy to document the points of view (stakeholders, sources), and to analyze their 'credibility'. But, if it is legal and it works, we will use it.**
3. **there are many notions of 'priority'. This in clouds value for money, cultural power, riskiness, credibility, and pleasing other people. We can make these priority explicit, or documented and accepted. The important thing is to aware of acceptable and official priorities. And to be able to question and change priorities, because of other priorities [12].**

Conclusion:

This characteristic does not give me any useful insight.

But that could be because I do not understand it yet.



Concrete strategies look like this. Solid, 'do it' stuff
Strategy = Buy 'New Computer' 20 seconds
Business Function
-
+
One single Performance Objective Represented by the arrow
Past [Dec. 2014] 50 sec.
Tolerable [April 2021] 40 sec.
Goal [April 2021] 15 sec.
Abstract Strategies, look like this point on the Scale. A notion of how well, the performance objective needs to be, to support the higher level objective. An 'idea' missing the final reality to make it happen.

W10. **The ‘planner’ (designer) e wrong’.**

My interpretation, based on [1].

- **A *scientist* can live with a wrong hypothesis, if the refutation process leads to greater knowledge and truth.**

- **A Planner cannot afford the luxury of this scientific process.**

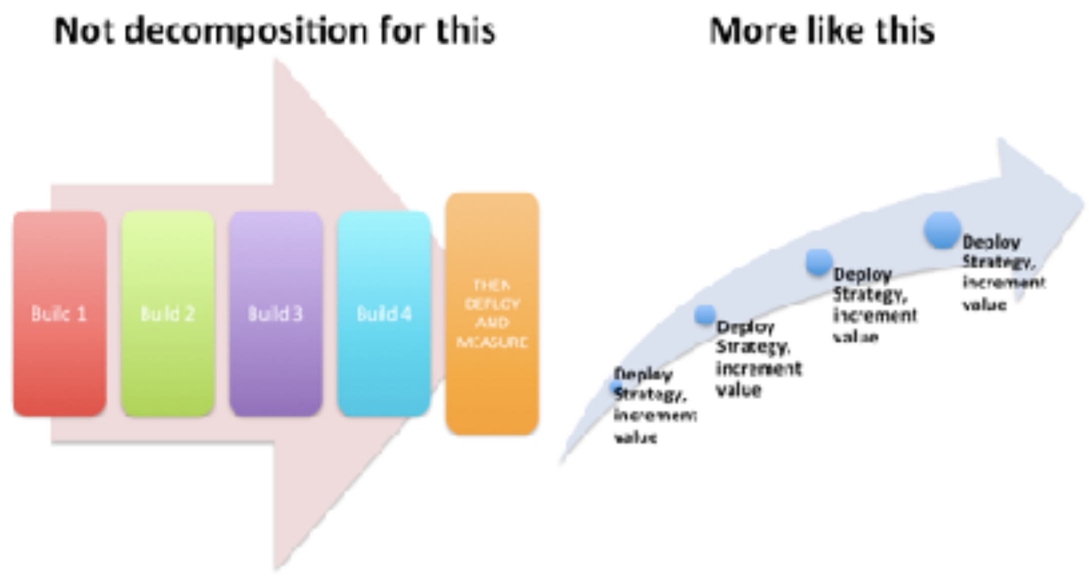
- **Planning is not about ‘finding the truth’**

- **Planning is about making thing better for people.**

- **The planning consequences of a ‘bad’ hypothesis has real, and possibly very negative, impacts on real people.**

- **So, planners cannot ethically have ‘philosophical fun’ with possibly bad hypothesis.**

- **They have to get their solution (and problem) right enough to do no damage, and hopefully right enough, to do good, for people.**


Not decomposition for this
Build 1
Build 2
Build 3
Build 4
THEN DEPLOY AND MEASURE
More like this
Deploy Strategy, increment value
Deploy Strategy, increment value
Deploy Strategy, increment value
Deploy Strategy, increment value

W10. **The ‘planner’ (designer) has no ‘right to be wrong’.**

. Planguage very much supports this process, do-gooding, rather than truth-finding.

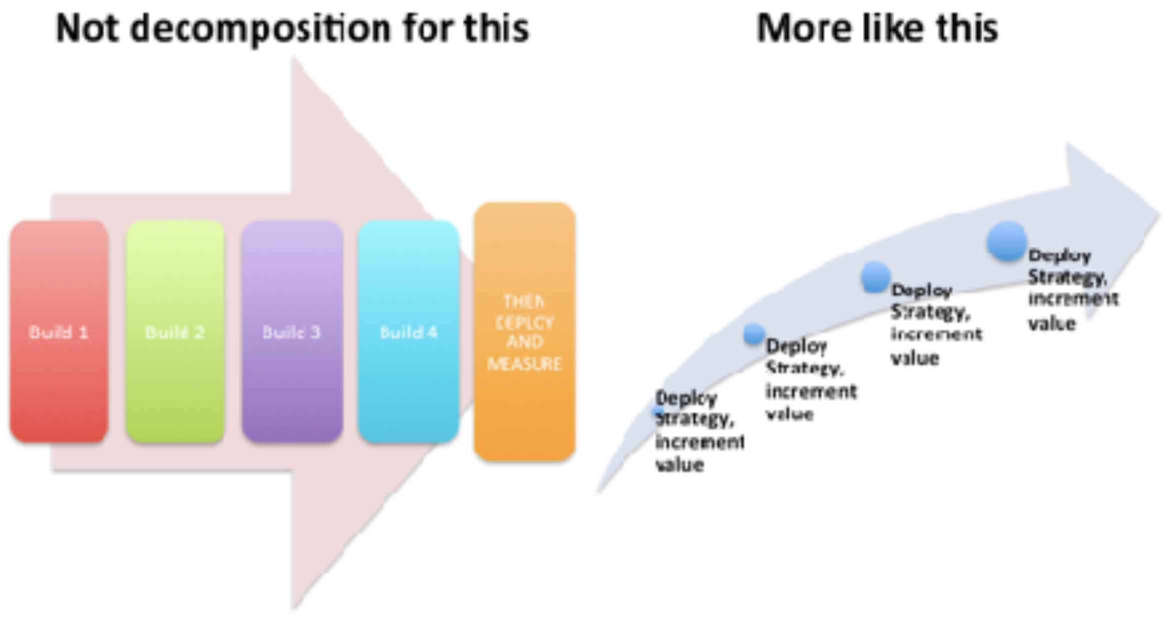
It does so in a large number of large-and-small tools, principles, methods, and processes.

One of many examples of this is

- **the primary Evo process**

  of trying to deliver the largest possible stream of value improvement as early and continuously as possible,

while learning through feedback how to improve on this process itself.


Not decomposition for this
More like this
Build 1
Build 2
Build 3
Build 4
THEN DEPLOY AND MEASURE
Deploy Strategy, increment value
Deploy Strategy, increment value
Deploy Strategy, increment value
Deploy Strategy, increment value

Talk Summary

I think the 'Wicked Problem" ideas are more misleading than useful.

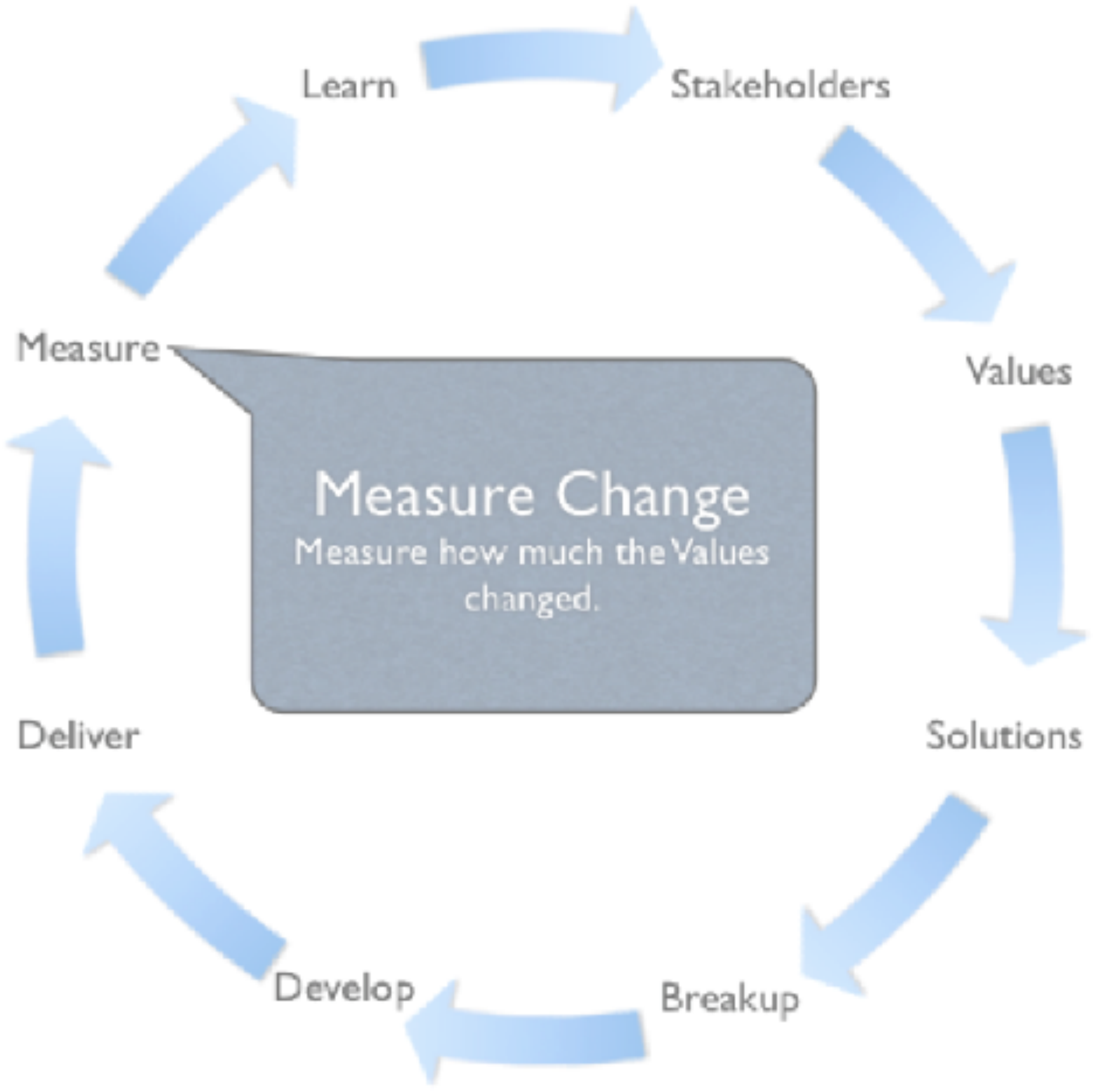
There are a wide variety of methods for handling large and complicated systems in reasonable ways, in addition to the ones I have presented [5 is constructive], here and in my books.

Most intelligent professionals that I encounter, do not seem trained in these methods, and are not aware of the many tools they can use to tackle complicated ('Wicked') systems.

I think we need to focus our attention on mastering a variety of methods for delivering stakeholder value. We are nowhere near good enough, with extremely high failure rates. Failure rates which should shame any professionals with responsibility and pride.

The conditions telling us that we are good enough, or much better are:

- **more than 95% of our projects result in the value improvements we have promised, on time and within budget. We already have the knowledge to do that. Do you ? [F1]**
- **no excuses about 'Wicked Problems'**


Learn
Stakeholders
Values
Solutions
Breakup
Develop
Deliver
Measure
Measure Change
Measure how much the Values changed.