"Dynamic Design to Cost for Value (DDtCV):

copes with imposed deadlines and fixed prices."

Tom Gilb and Kai Gilb
gilb.com
@ImTomGilb
Workshop at 'Smidig' (Agile) Conference, Oslo Monday 2 November 2015, 13:15-14:00
tom@gilb.com, kai@gilb.com

Our Column http://tinyurl.com/AGILEMYTHS
http://tinyurl.com/AGILEMYTHS

Free Workshops at #smidig15

Oslo 16-17 Dec.
http://www.meetup.com/Oslo-Software-Architecture/events/225774527/

London 10-11 Nov. BCS

Startup Planning for Entrepreneurs, Startups, Innovator
http://www.bcs.org/category/10136

an Oslo version of this is being planned with Peter Skeide, Skalar

The highest priority for human survival is:

- Water
- Air
- Food

Critical Body Priorities

Dynamic prioritization, the human body method, is a pretty **smart** prioritization method, and keeps you **alive** in **changing** conditions.

We could do worse than to use this **dynamic and logical** method
for management planning.

?

Value Decision Tables

		
Product Value 1		
Product Value 2		
Resources		

Value Decision Tables

		
Product Value 1		
Product Value 2		
Resources		

Value Decision Tables

			
Product Value 1			
Product Value 2			
Resources			

Value Decision Tables

			
Taste			
Resources			

Value Decision Tables

			
Taste			
Nutrition			
Resources			

Value Decision Tables

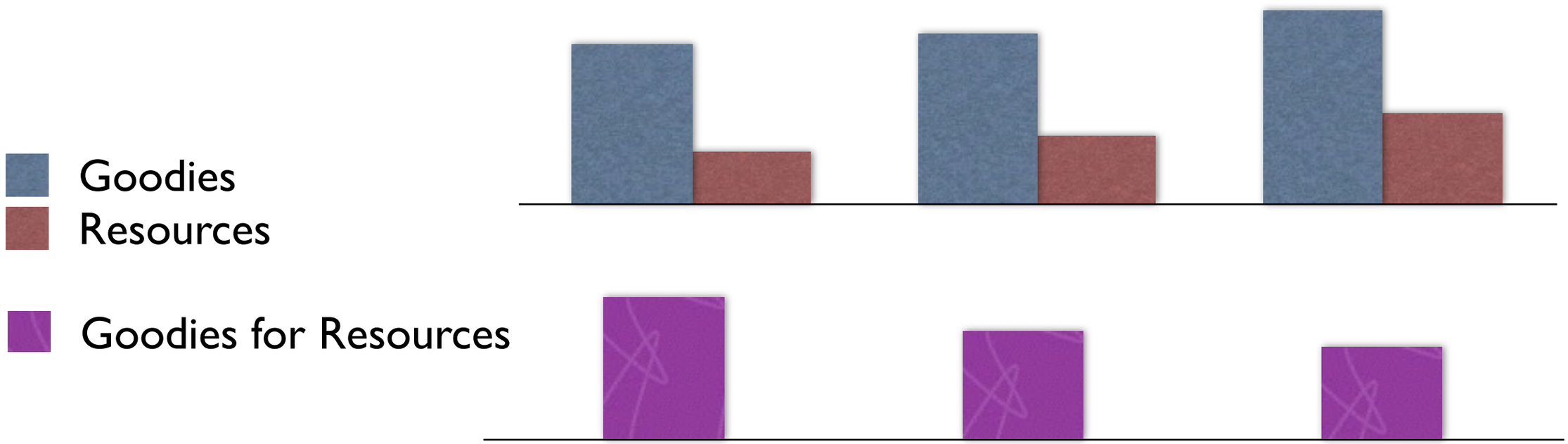
			
Taste			
Nutrition			
Shelf Life			
Resources			

Value Decision Tables

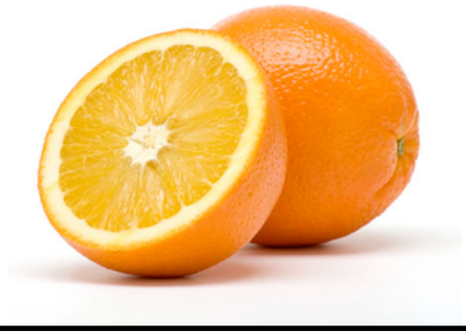
			
Taste			
Nutrition			
Shelf Life			
Sum Goodies			
Resources			

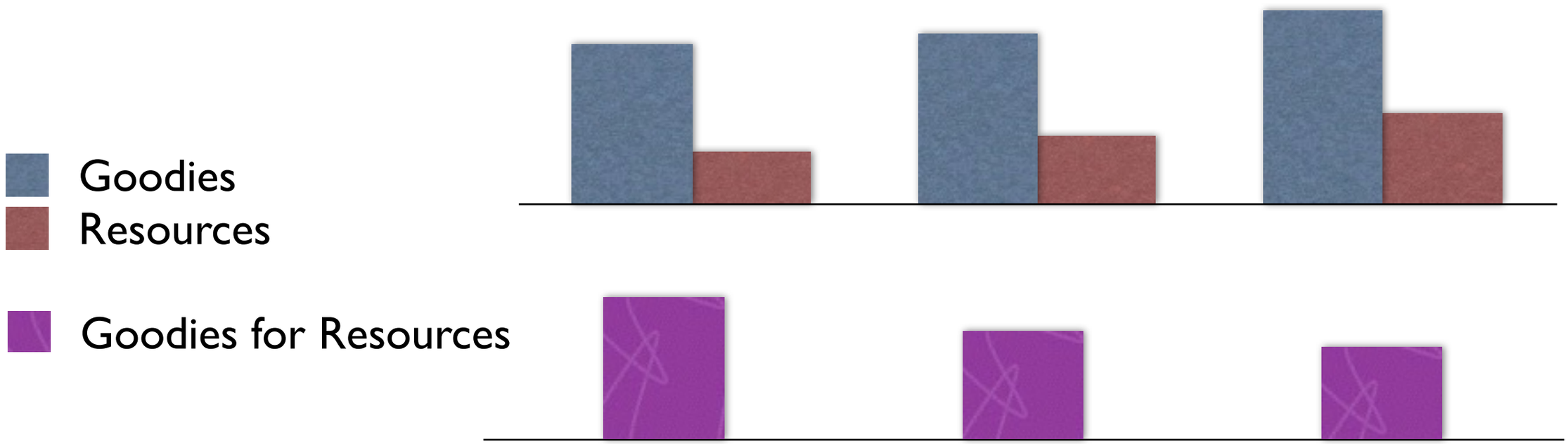
Value Decision Tables

			
Taste	0,2	0,5	0,9
Nutrition	0,3	0,7	0,9
Shelf Life	0,8	0,3	-0,1
Sum Goodies	1,3	1,5	1,7
Resources	0,4	0,6	0,8



Value Decision Tables

			
Taste	0,2	0,5	0,9
Nutrition	0,3	0,7	0,9
Shelf Life	0,8	0,3	-0,1
Sum Goodies	1,3	1,5	1,7
Resources	0,4	0,6	0,8



Confirmit: Results

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec

Confirmit: Results

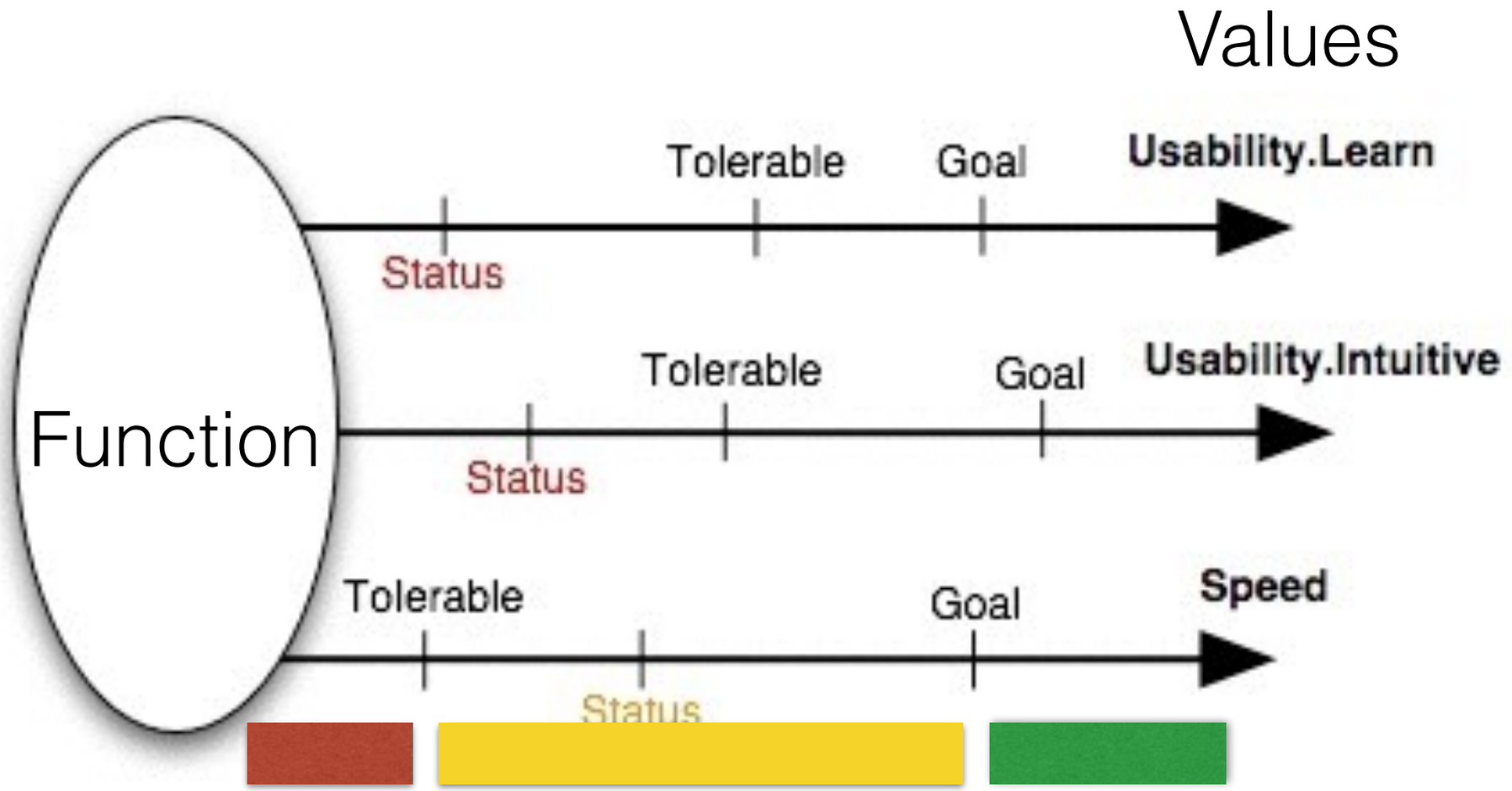
Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000

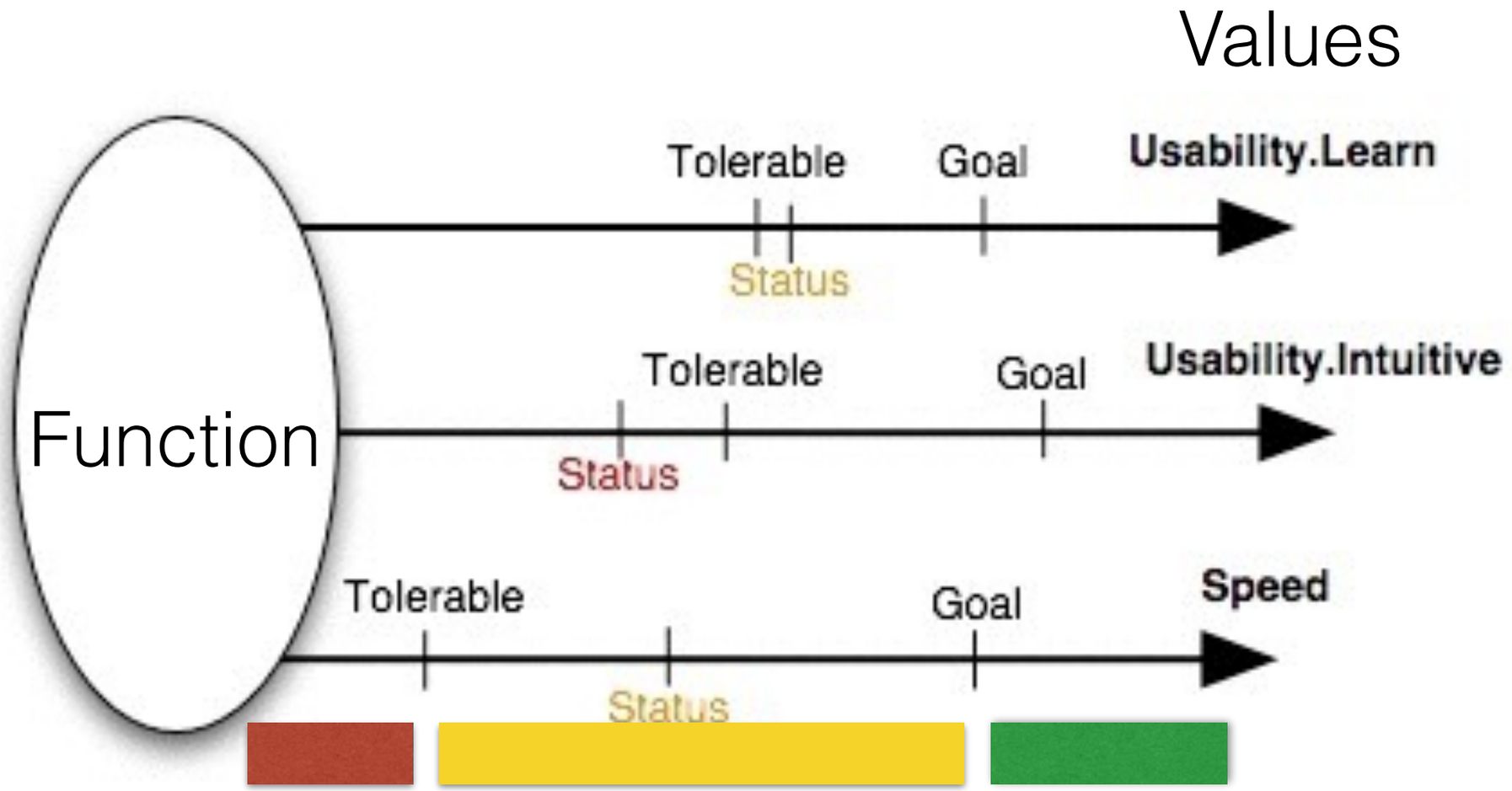
Confirmit
Snapshot End Week 9 of 12

Current Status	Improvements		Goals			Step9 Recoding Estimated impact		Step9 Recoding Actual impact	
Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
			Usability.Replacability (feature count)						
1,00	1,0	50,0	2	1	0				
			Usability.Speed.NewFeaturesImpact (%)						
5,00	5,0	100,0	0	15	5				
10,00	10,0	200,0	0	15	5				
0,00	0,0	0,0	0	30	10				
			Usability.Intuitiveness (%)						
0,00	0,0	0,0	0	60	80				
			Usability.Productivity (minutes)						
20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
			Development resources						
	101,0	91,8	0		110	4,00	3,64	4,00	3,64

Confirmit
Snapshot End Week 9 of 12

Current Status	Improvements		Goals			Step9 Recoding Estimated impact		Actual impact	
Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
			Usability.Replacability (feature count)						
1,00	1,0	50,0	2	1	0				
			Usability.Speed.NewFeaturesImpact (%)						
5,00	5,0	100,0	0	15	5				
10,00	10,0	200,0	0	15	5				
0,00	0,0	0,0	0	30	10				
			Usability.Intuitiveness (%)						
0,00	0,0	0,0	0	60	80				
			Usability.Productivity (minutes)						
20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
			Development resources						
	101,0	91,8	0		110	4,00	3,64	4,00	3,64





Confirmit
Snapshot End Week 9 of 12

Current Status	Improvements		Goals			Step9 Recoding Estimated impact		Actual impact	
Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
			Usability.Replacability (feature count)						
1,00	1,0	50,0	2	1	0				
			Usability.Speed.NewFeaturesImpact (%)						
5,00	5,0	100,0	0	15	5				
10,00	10,0	200,0	0	15	5				
0,00	0,0	0,0	0	30	10				
			Usability.Intuitiveness (%)						
0,00	0,0	0,0	0	60	80				
			Usability.Productivity (minutes)						
20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
			Development resources						
	101,0	91,8	0		110	4,00	3,64	4,00	3,64

Confirmit
Snapshot End Week 9 of 12

Dynamic Priority Metric

Weekly Progress

Constraint

Target

Current Status	Improvements		Goals			Step9 Recoding			
						Estimated impact		Actual impact	
Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
			Usability.Replacability (feature count)						
1,00	1,0	50,0	2	1	0				
			Usability.Speed.NewFeaturesImpact (%)						
5,00	5,0	100,0	0	15	5				
10,00	10,0	200,0	0	15	5				
0,00	0,0	0,0	0	30	10				
			Usability.Intuitiveness (%)						
0,00	0,0	0,0	0	60	80				
			Usability.Productivity (minutes)						
20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
			Development resources						
	101,0	91,8	0		110	4,00	3,64	4,00	3,64

Estimates

Weekly Testing

Confirmit

4 product areas were attacked in all: **25 Qualities** concurrently

Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportReport (minutes		
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewReport (seconds)		
1,0	14,0	100,0	15	3	1
			Development resources		
203,0			0		191

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	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

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

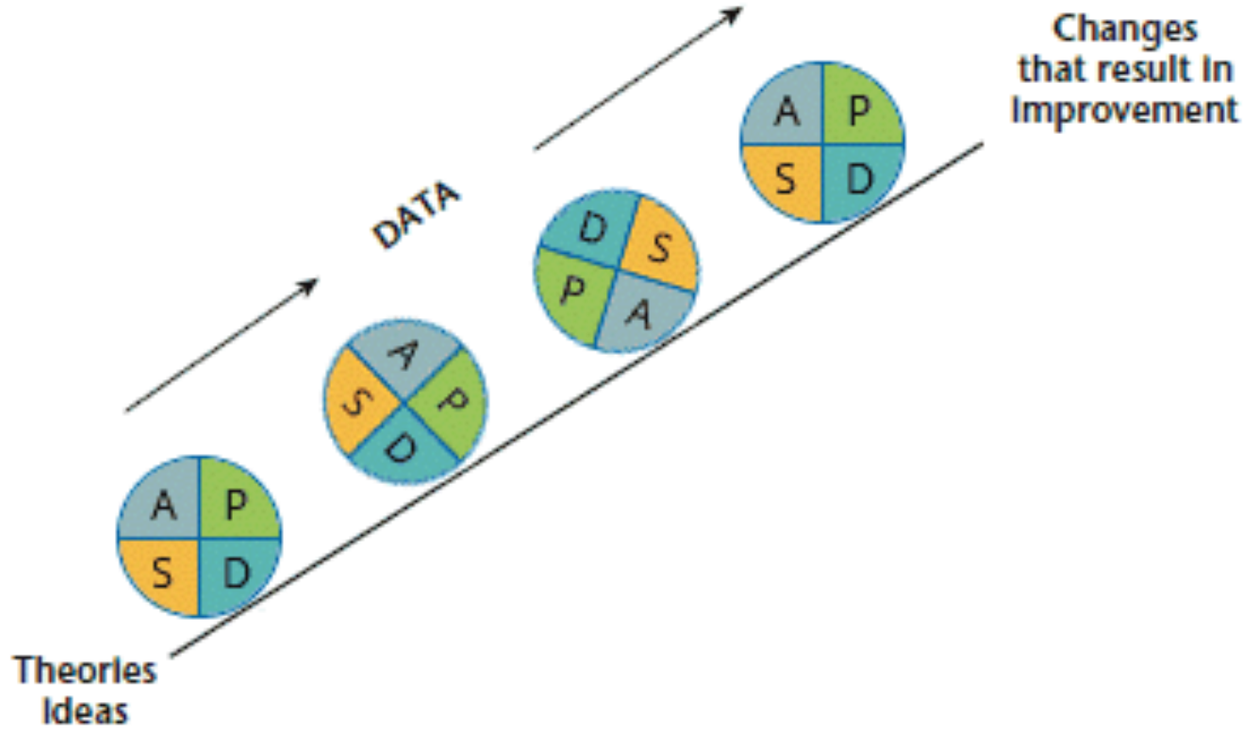
Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	#######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		48

#NoEstimates

- "Estimation: A Paradigm Shift Toward Dynamic Design-to Cost and Radical Management"
  - Volume 13 Issue 2 of SQP journal - the March 2011 version.
    - Software Quality Professional, USA
    - The American Society for Quality (ASQ)
  - http://www.gilb.com/tiki-download_file.php?fileId=460
  - Slides: For BCS SPA, London
    - http://www.gilb.com/tiki-download_file.php?fileId=470

The basic process: DDtCV

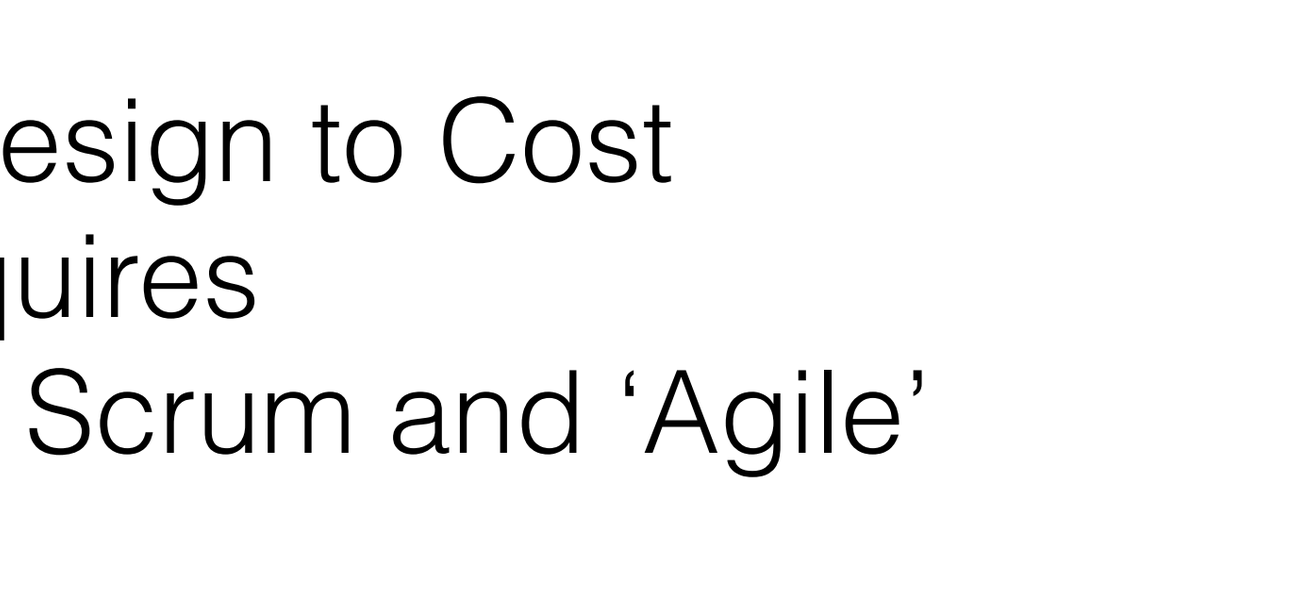
- **If all is 'on track'**
  - x% value**s**, for
  - X% cost**s**
- **Do a new value delivery cycle**
- **If not on track, then '*change something*'; to get back on track**



PDSA: Plan Do Study Act
Deming Cycle

Dynamic Design to Cost
requires
things absent in Scrum and ‘Agile’

- Multiple resource constraints
  - deadline, money, people, space
- Multiple measurable values
  - qualities, savings
- Cycle Decomposition by Value
- Measurement of Value each cycle
- Design to cost



Attributes of Dynamic Design to Cost (DDC)

- **Ability to deliver on time**
- **Ability to deliver to budget**
- **Ability to delivery to multiple ambitious quality targets**
- **Ability to learn what works early**
- **Ability to experiment with high promise architecture, at low risk**
- **Ability to experiment, low risk, with development processes**
- **Fits a no cure no-pay contracting model**
  - **flexiblecontracts-com**

Dynamic Design to Cost as a defence against arbitrary budgets and deadlines.

in 4.5 VP

‘Dynamic design to cost’ as a **management process**, is particularly interesting to understand,
when you do not have the luxury *to estimate how much you need or want, for your own scheduling and funding purposes.*
You are not *asked*, you are ***told*** **the costs and deadlines.**

The government client, or other powerful forces, set a deadline for you; and they allocated a fixed-cost budget.
Your salespeople ‘happily’ won, as low bidder of a fixed-price contract.
You, however, are then stuck with **the problem of ‘making it happen’, on time, under budget.**

Principle 6.2

DYNAMIC PRIORITY

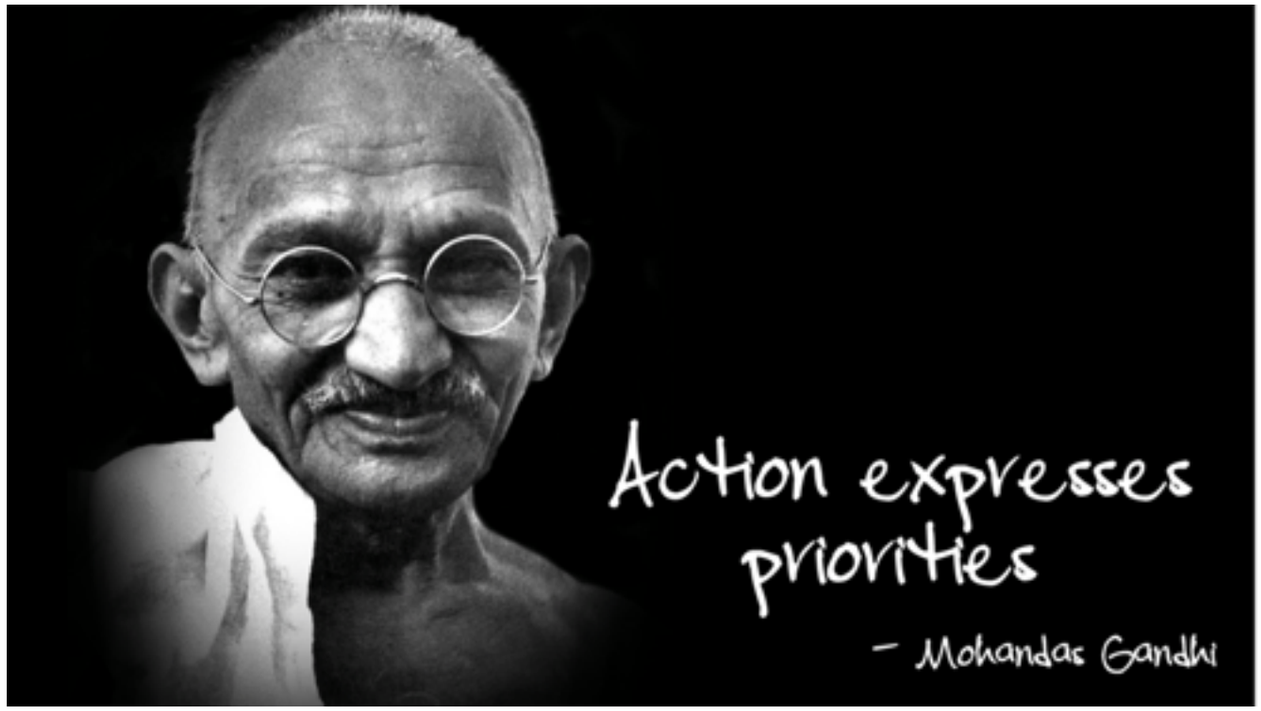
(VP book):

Static initial prioritization is unrealistic –

things change

Why
Priority
must be
Dynamic

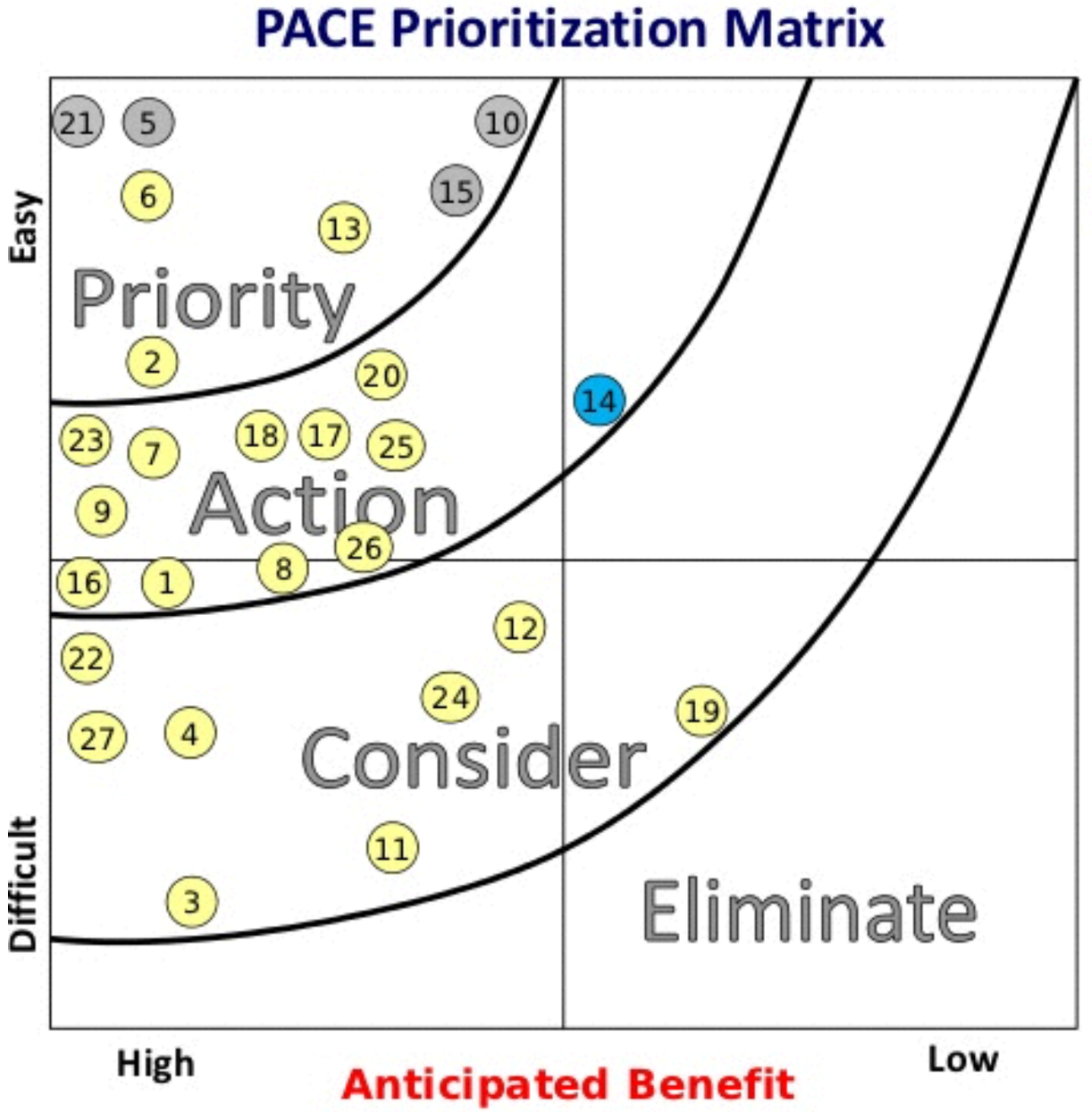
- The **facts needed** to determine your current priority,
  - o are **constantly and arbitrarily changing**
- The facts needed are:
  - o ***remaining limited resources, and remaining distance to Goals***
- Only when these facts are available, can you search for a 'suitable strategy':
  - o *one that will move you towards your Goals as much as possible,*
  - o *within the (weekly) cycle duration,*
  - o *with as little use of other resources, like money, as possible.*
- We can **prioritize** any strategy, which we can find,
  - o that gives best **progress**, towards *residual* **Goal** levels,
  - o at the lowest consumption of residual resources.



Conditions for Logical Prioritization

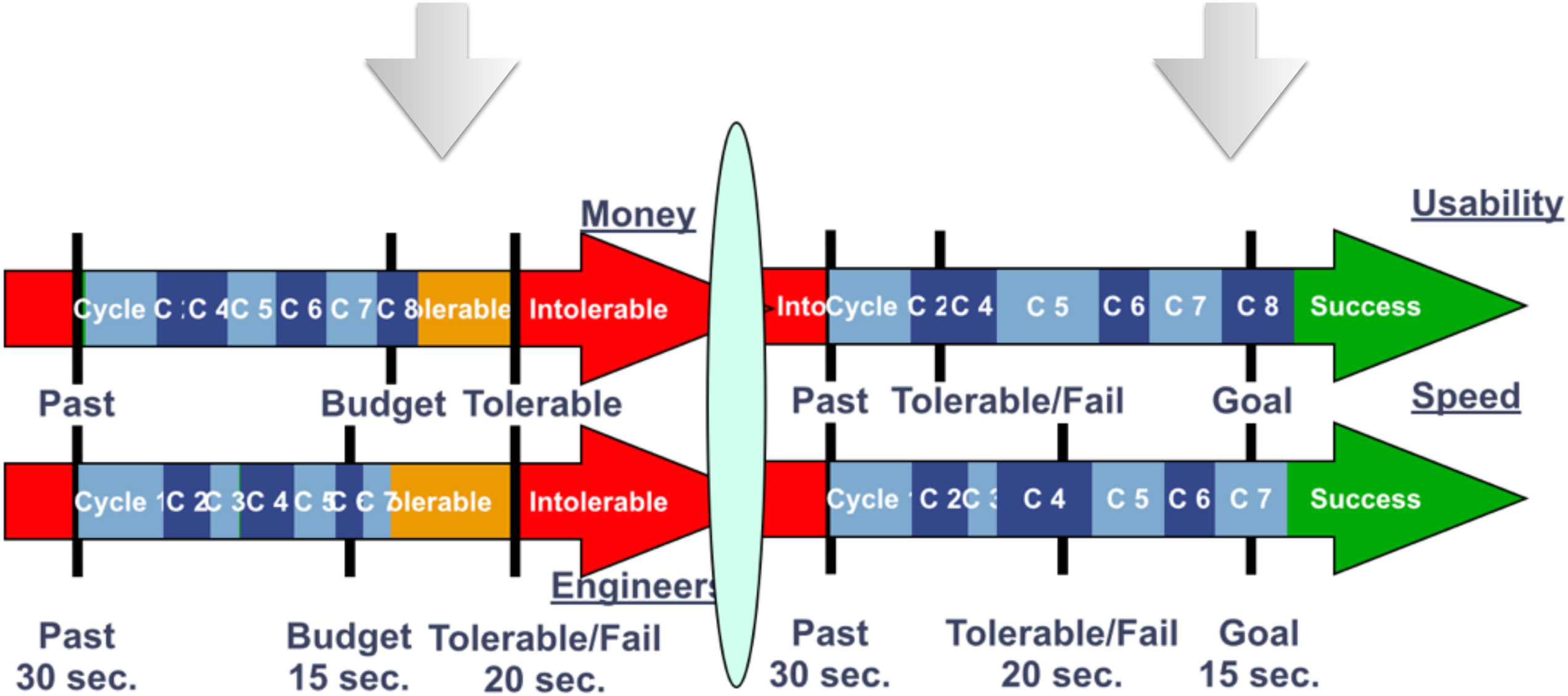
VP 6.8

1. **Critical Objectives identified**
2. **Objectives Quantified**
3. **Constraints ID & Quantified**
4. **Clear detailed strategies**
5. **Estimates of Strategy Impacts & Costs**
6. **Risks and Uncertainties ID**
7. **Policy for deciding what to prioritize (Value / € ?), Risk**

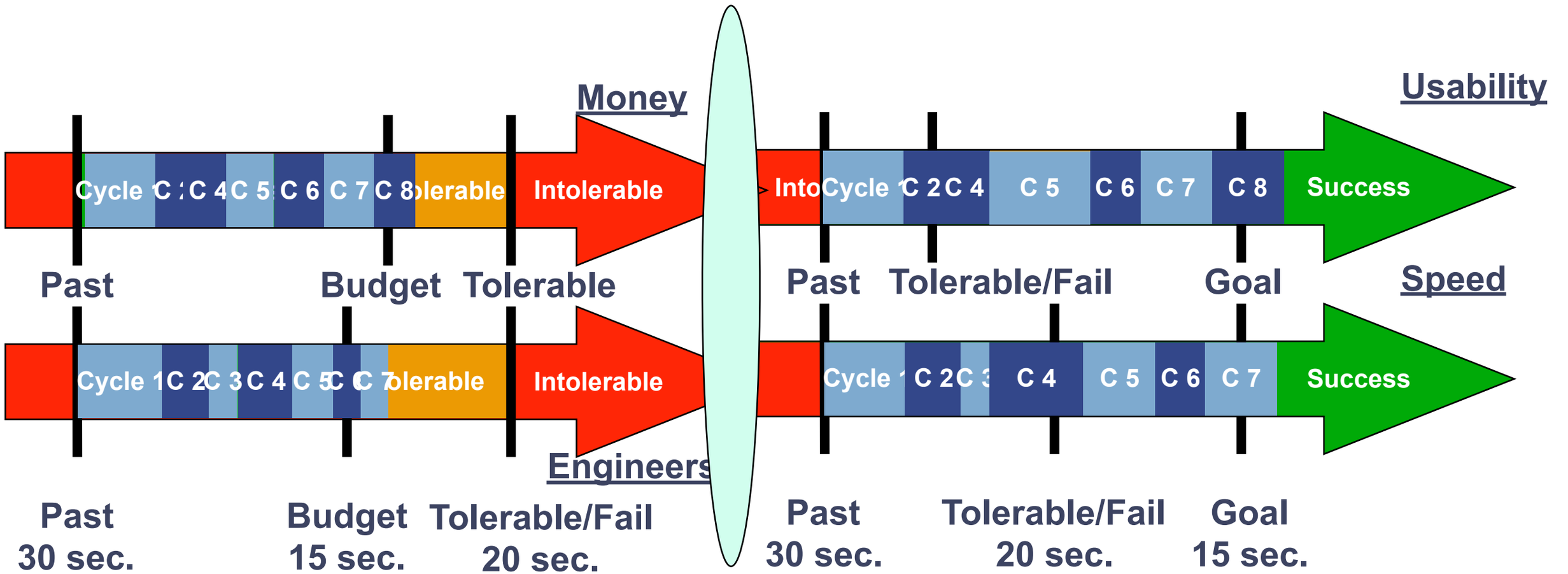


http://www.slideshare.net/KarenMartinGroup/08-232012-value-stream-mapping

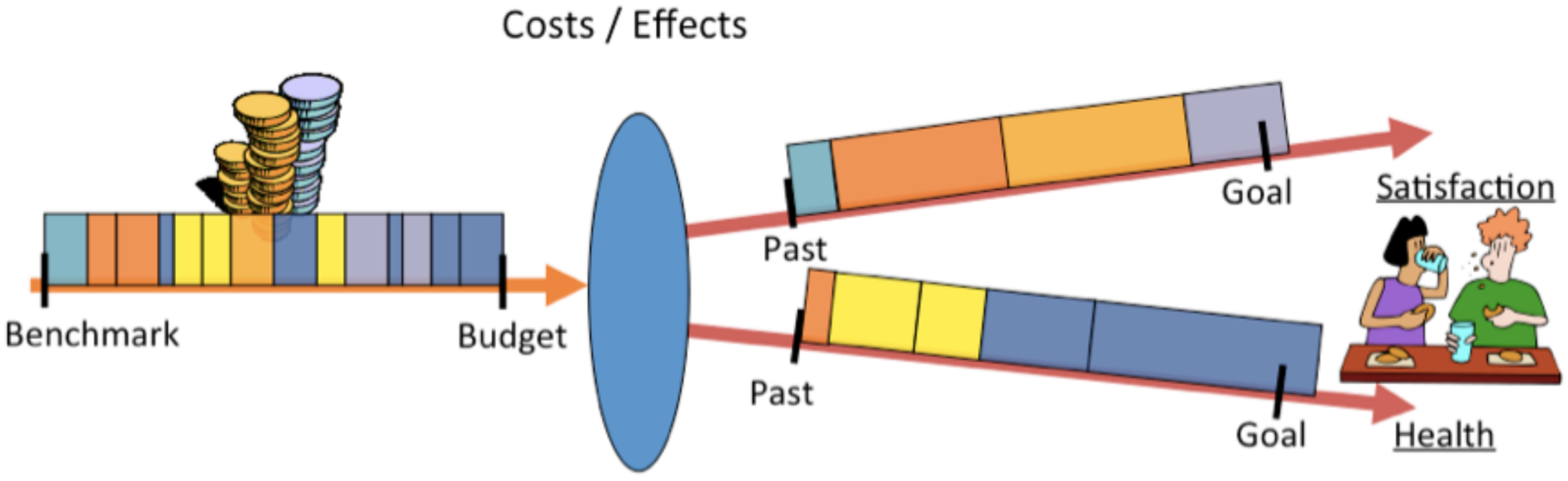
Multiple Constraints and Multiple Objectives (Static)

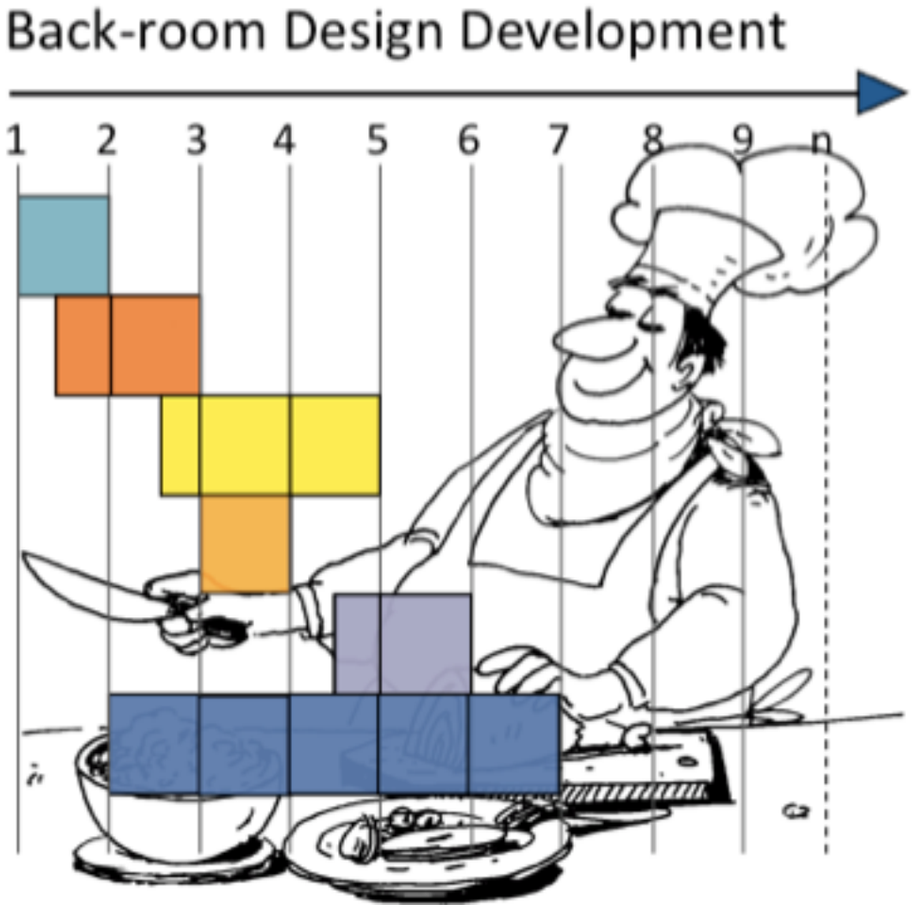


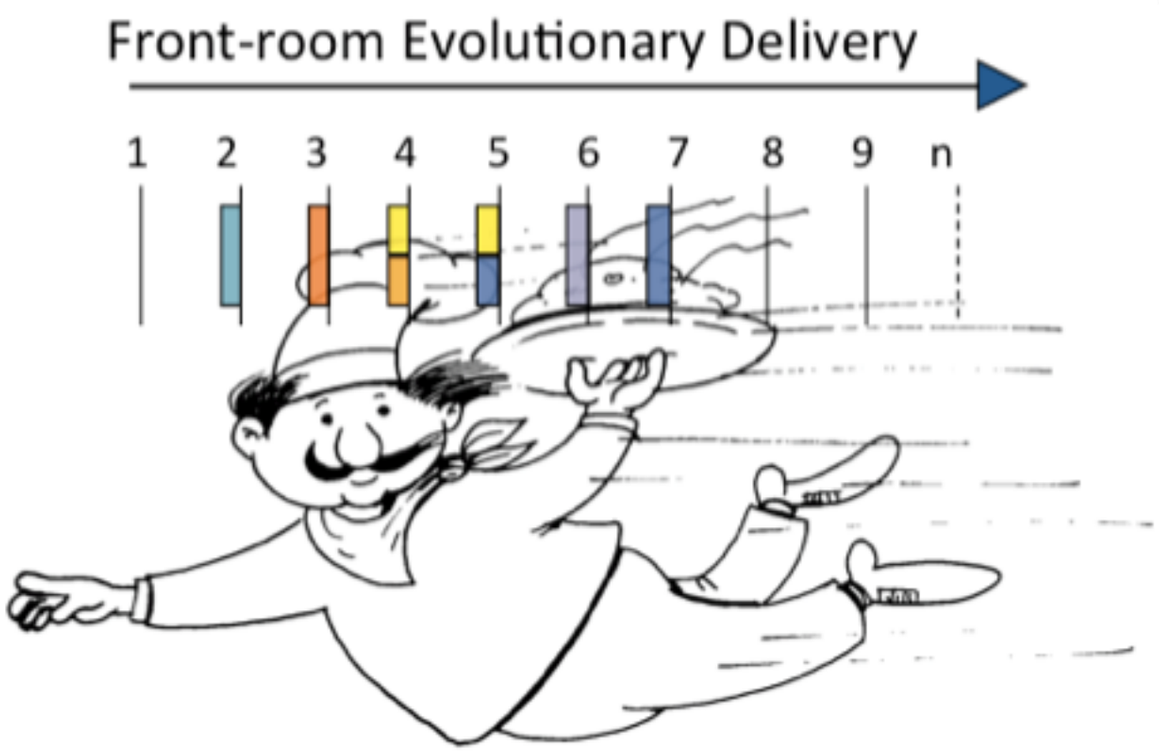
Each Evolutionary Cycle uses a constrained budget of Development Resources



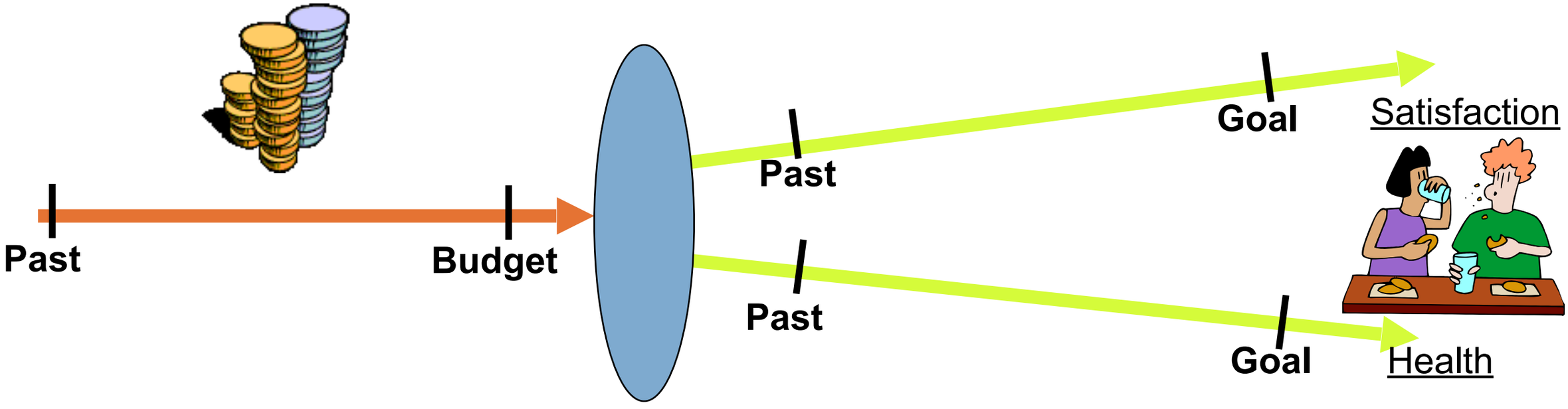
Dynamic ‘Restaurant’ Prioritization (Static)





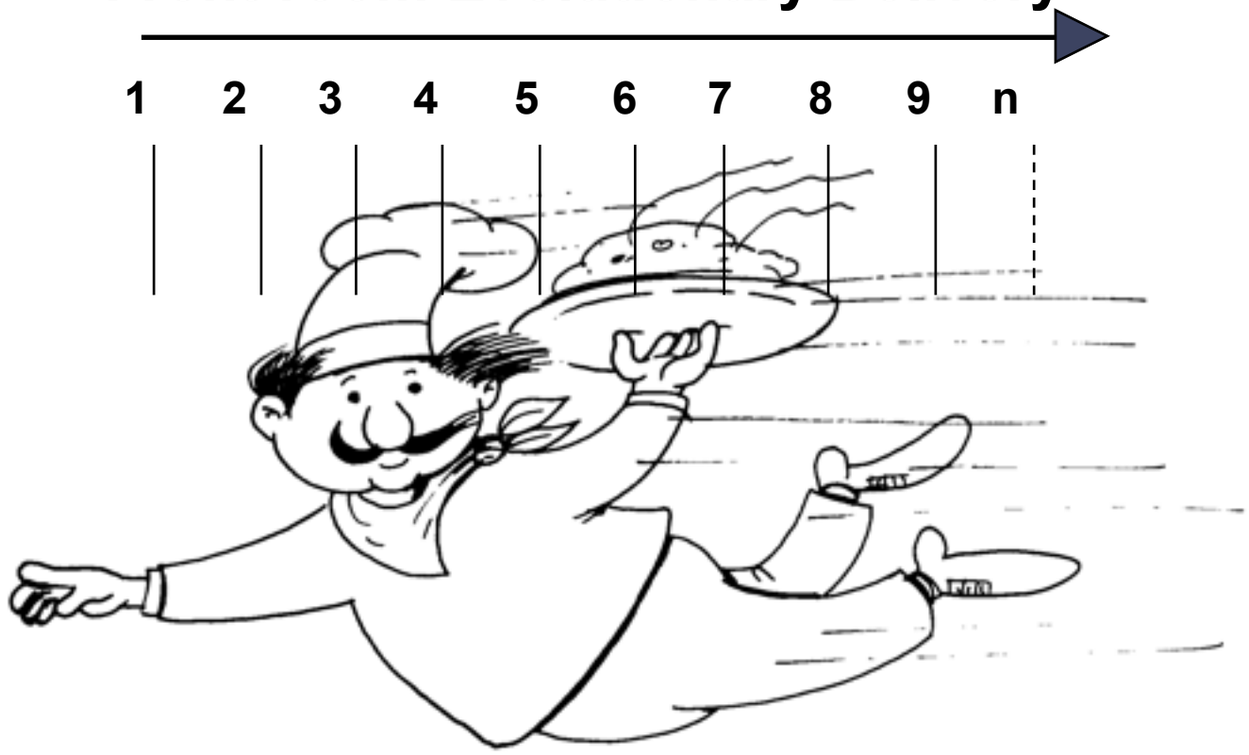


Costs / Effects

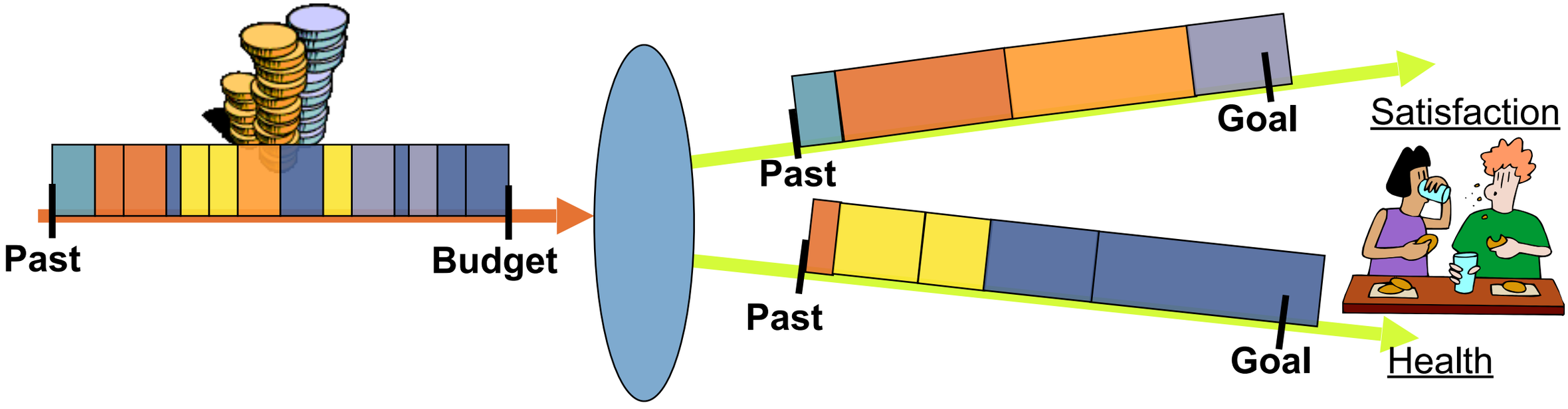


Back-room Design Development

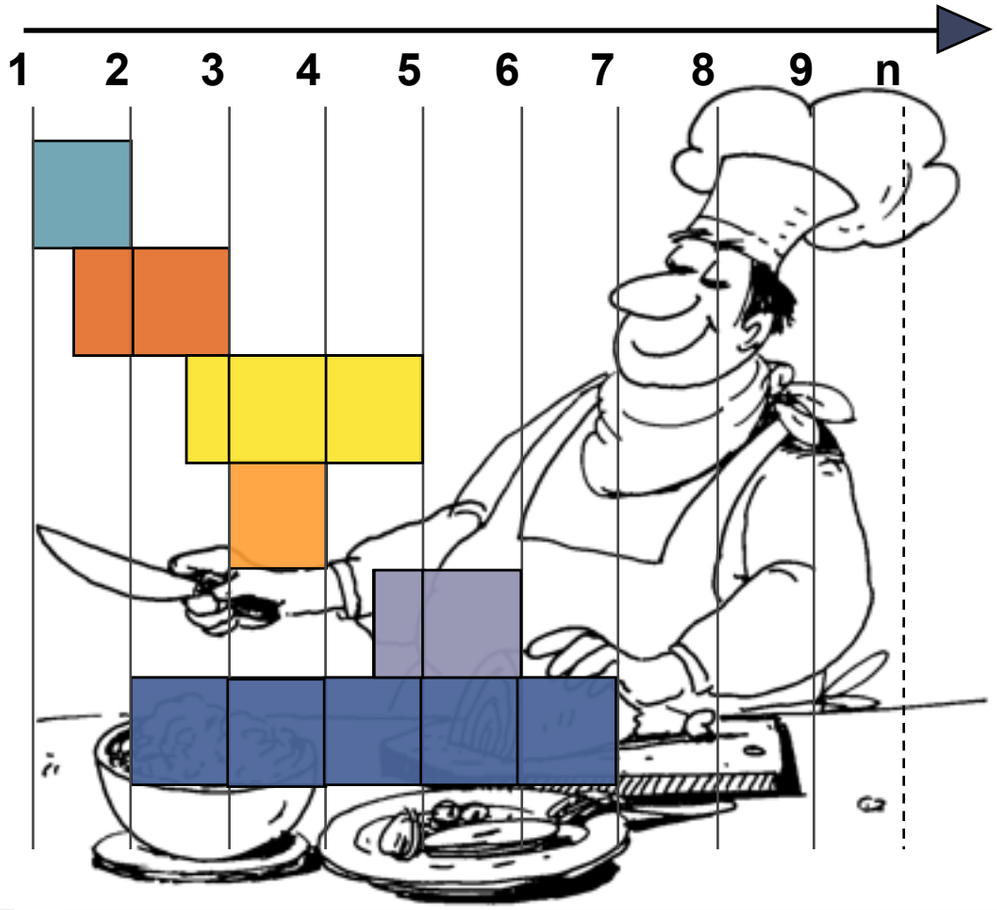
Front-room Evolutionary Delivery



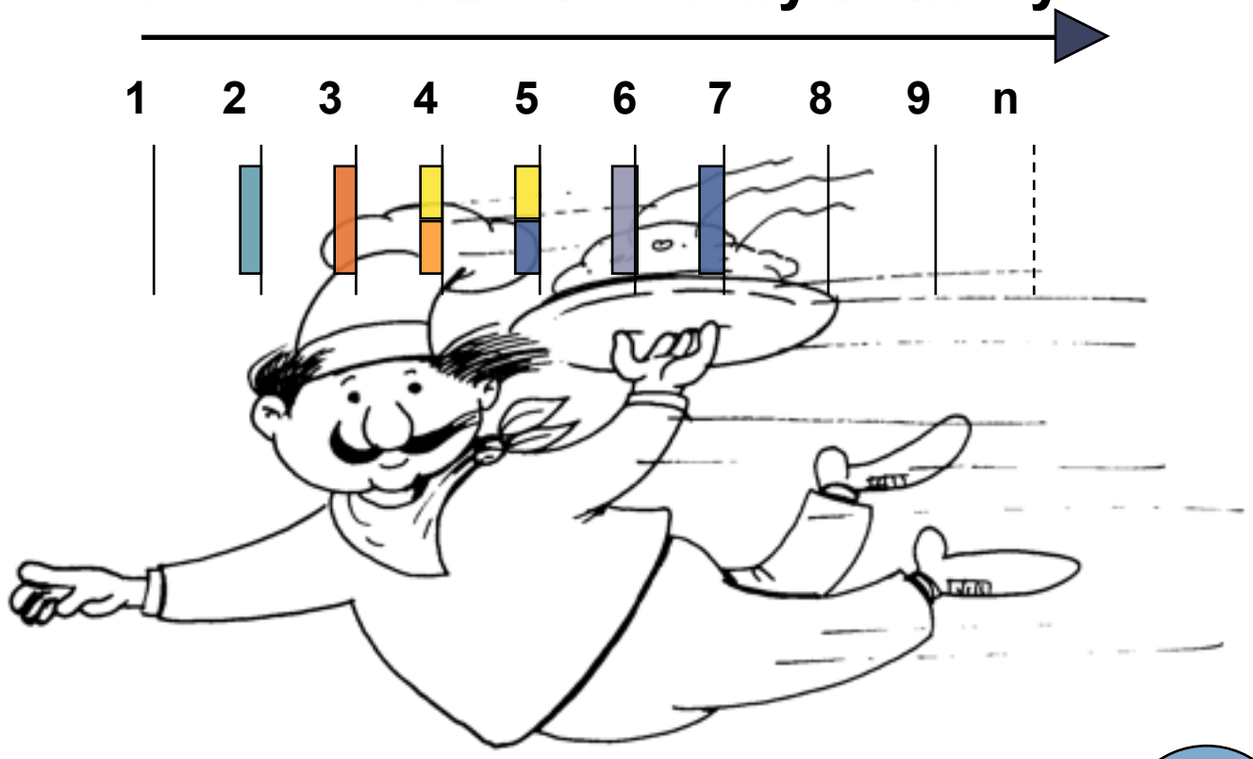
Costs / Effects



Back-room Design Development



Front-room Evolutionary Delivery



Impact Table with highly varied costs, for ‘same impact’ on requirements

Illustration for Talk 2 Nov 2015

Settings... | + Add to table | Sort designs | Show Sidebar

View Impact table

Requirements	D1	D2	D3	D4	Sum
Past: 0 → Wish: 100 %	50 % 0 Δ%: 50 %	50 % 0 Δ%: 50 %	50 % 0 Δ%: 50 %	50 % 0 Δ%: 50 %	ΣΔ%: 200 %
Sum Of Performance:	Σ%: 50 %	Σ%: 50 %	Σ%: 50 %	Σ%: 50 %	
Cost Past: 0 → Wish: 100 %	1 % 0 Δ%: 1 %	10 % 0 Δ%: 10 %	100 % 0 Δ%: 100 %	1000 % 0 Δ%: 1000 %	ΣΔ%: 1111 %
Sum Of Resources:	Σ%: 1 %	Σ%: 10 %	Σ%: 100 %	Σ%: 1000 %	
Performance To Cost:	50.00	5.00	0.50	0.05	

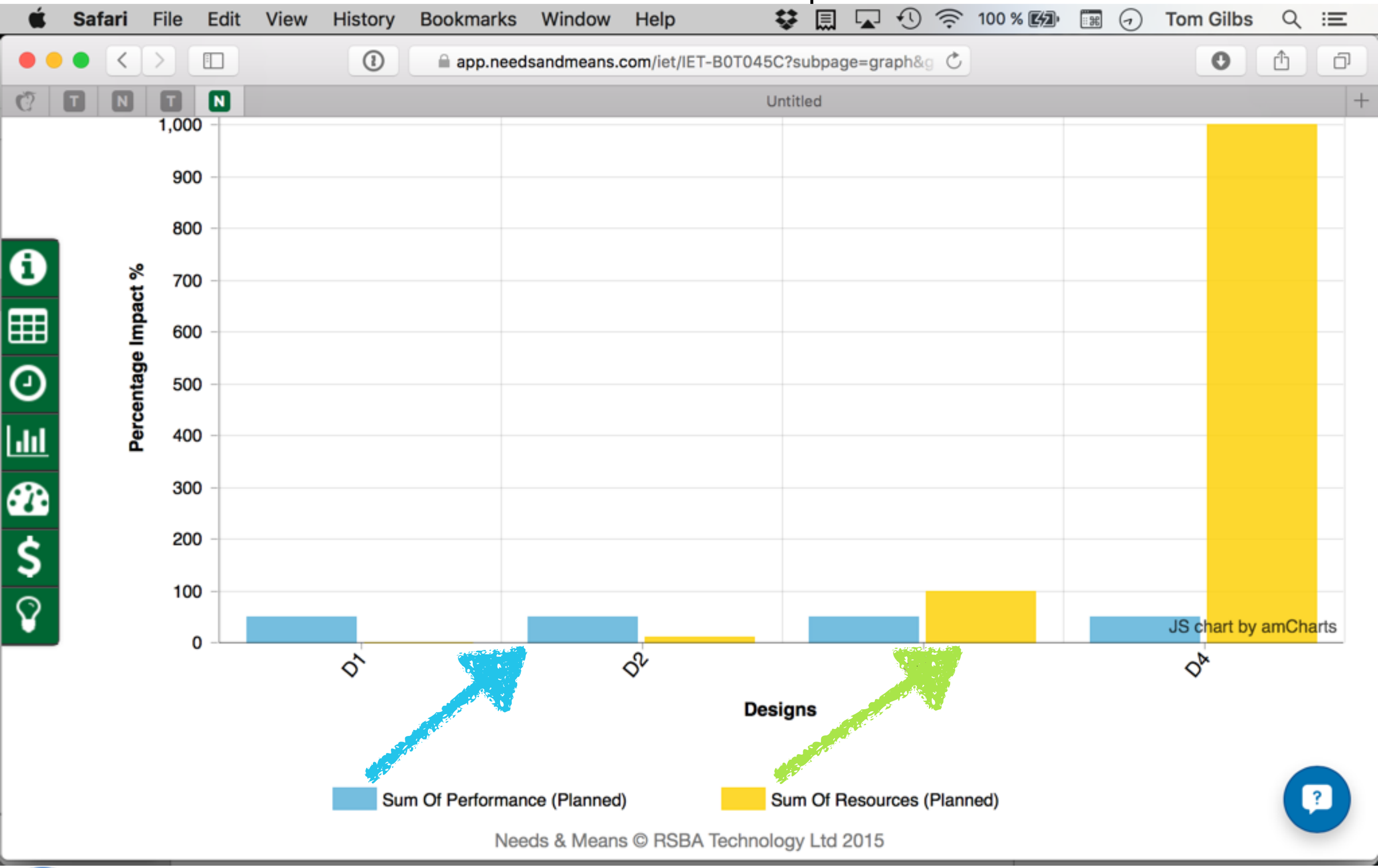
Copy | Excel | CSV | Print table

Comments: 0

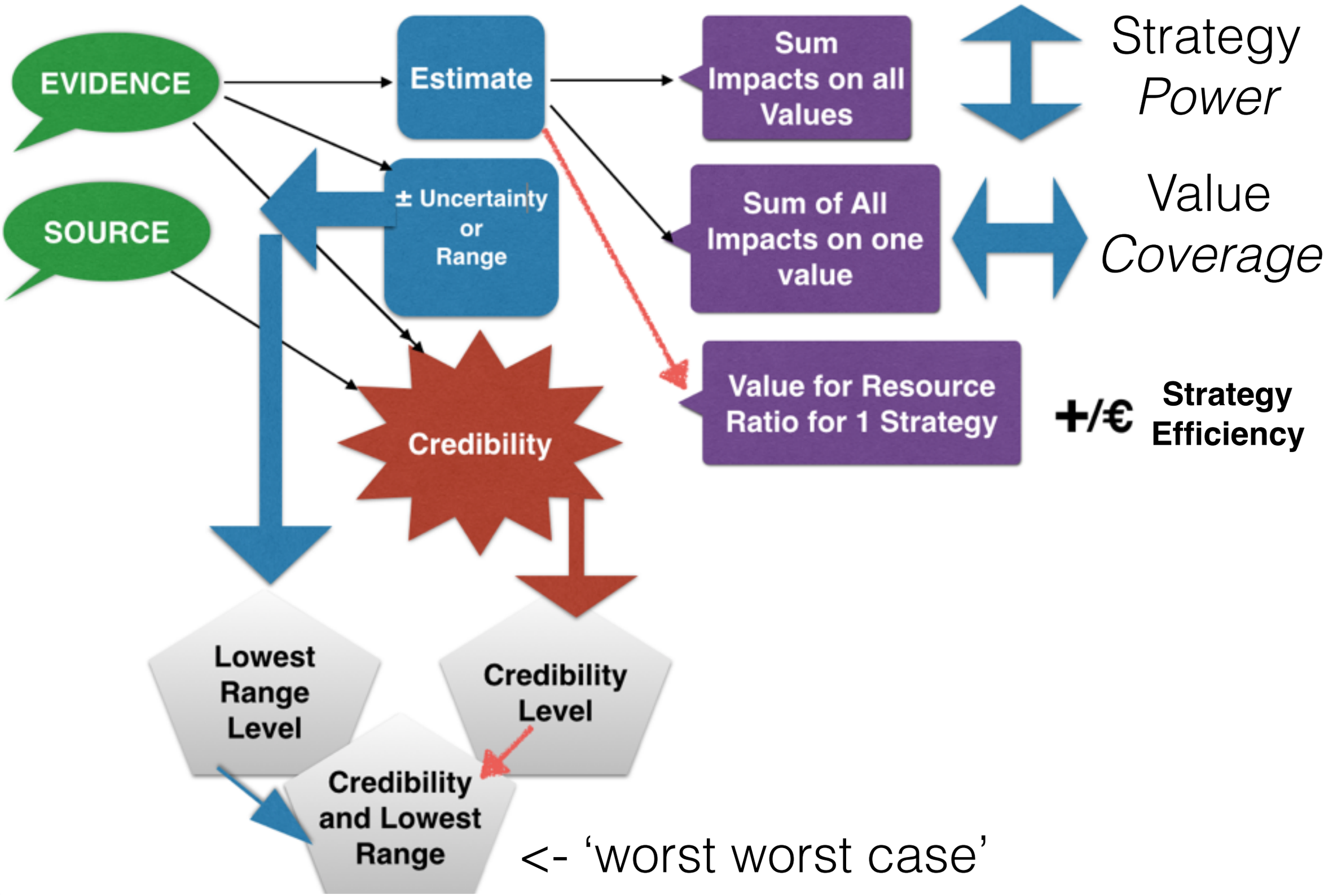
No comments

Add Comment...

Bar Chart from the Impact Table



Dynamic Prioritizing with Risks using IE Table



<- ‘worst worst case’

Impact Table with Risks

Safari File Edit View History Bookmarks Window Help 100 % Tom Gilbs

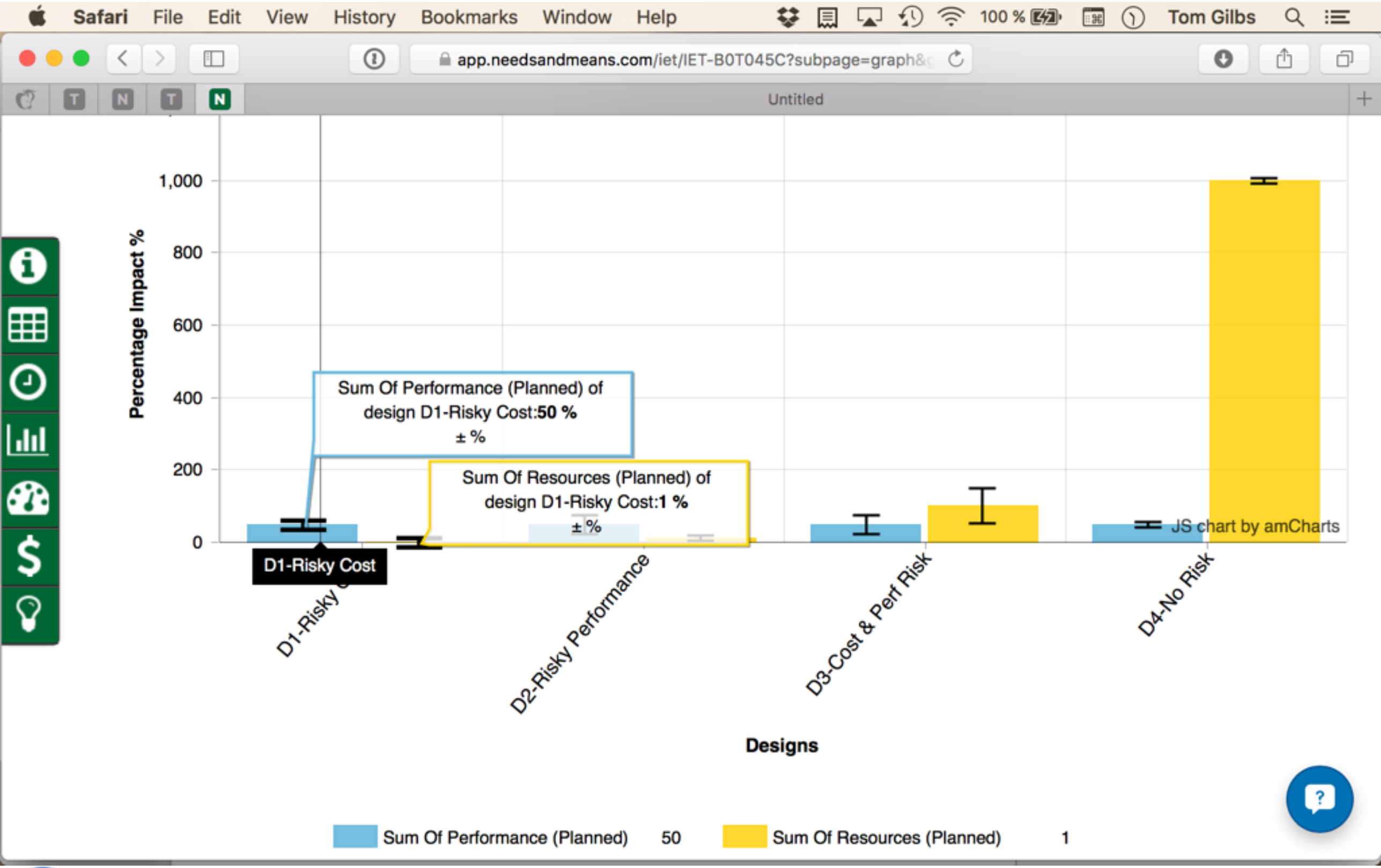
app.needsandmeans.com/iet/IET-B0T045C

Untitled

Settings... | + Add to table | Sort designs | Show Sidebar

Requirements	D1-Risky Cost	D2-Risky Performance	D3-Cost & Pe...	D4-No Risk	Sum
Usability Past: **0** → Wish: **100 %**	50 ± 0 % 0 Δ%: **50 ± 0 %** ?%: *50 % (x 1.0)*	50 ± 49 % 0 Δ%: **50 ± 49 %** ?%: *5 % (x 0.1)*	50 ± 49 % 0 Δ%: **50 ± 49 %** ?%: *5 % (x 0.1)*	50 ± 0 % 0 Δ%: **50 ± 0 %** ?%: *50 % (x 1.0)*	ΣΔ%: **200 ± 98 %**
Sum Of Performance: Credibility - adjusted:	Σ%: **50 ± 0 %** Σ?%: *50 %*	Σ%: **50 ± 49 %** Σ?%: *5 %*	Σ%: **50 ± 49 %** Σ?%: *5 %*	Σ%: **50 ± 0 %** Σ?%: *50 %*	
Cost Past: **0** → Wish: **100 %**	1 ± 0.9 % 0 Δ%: **1 ± 1 %** ?%: *2 % (x 0.1)*	10 ± 0 % 0 Δ%: **10 ± 0 %** ?%: *20 % (x 0.0)*	100 ± 99 % 0 Δ%: **100 ± 99 %** ?%: *190 % (x 0.1)*	1000 ± 0 % 0 Δ%: **1000 ± 0 %** ?%: *1000 % (x 1.0)*	ΣΔ%: **1111 ± 100 %**
Sum Of Resources: Credibility - adjusted:	Σ%: **1 ± 1 %** Σ?%: *2 %*	Σ%: **10 ± 0 %** Σ?%: *20 %*	Σ%: **100 ± 99 %** Σ?%: *190 %*	Σ%: **1000 ± 0 %** Σ?%: *1000 %*	
Performance To Cost:	50.00	5.00	0.50	0.05	
Ratio (Worst Case)	25.00	0.10	0.01	0.05	
Ratio (Cred. - adjusted)	26.32	0.25	0.03	0.05	

Bar Graph of the Impact Table with Risks



The 2 Estimation Elements in 'Design to Cost'.

VP 4.5

1. You estimate, and then re-estimate, repeatedly, based on 'costs to date', you *extrapolate* and say something like 'if we continue with these strategies, then we will run over budget, and past the deadline. So, we must change strategies, and we must do it **now**.'

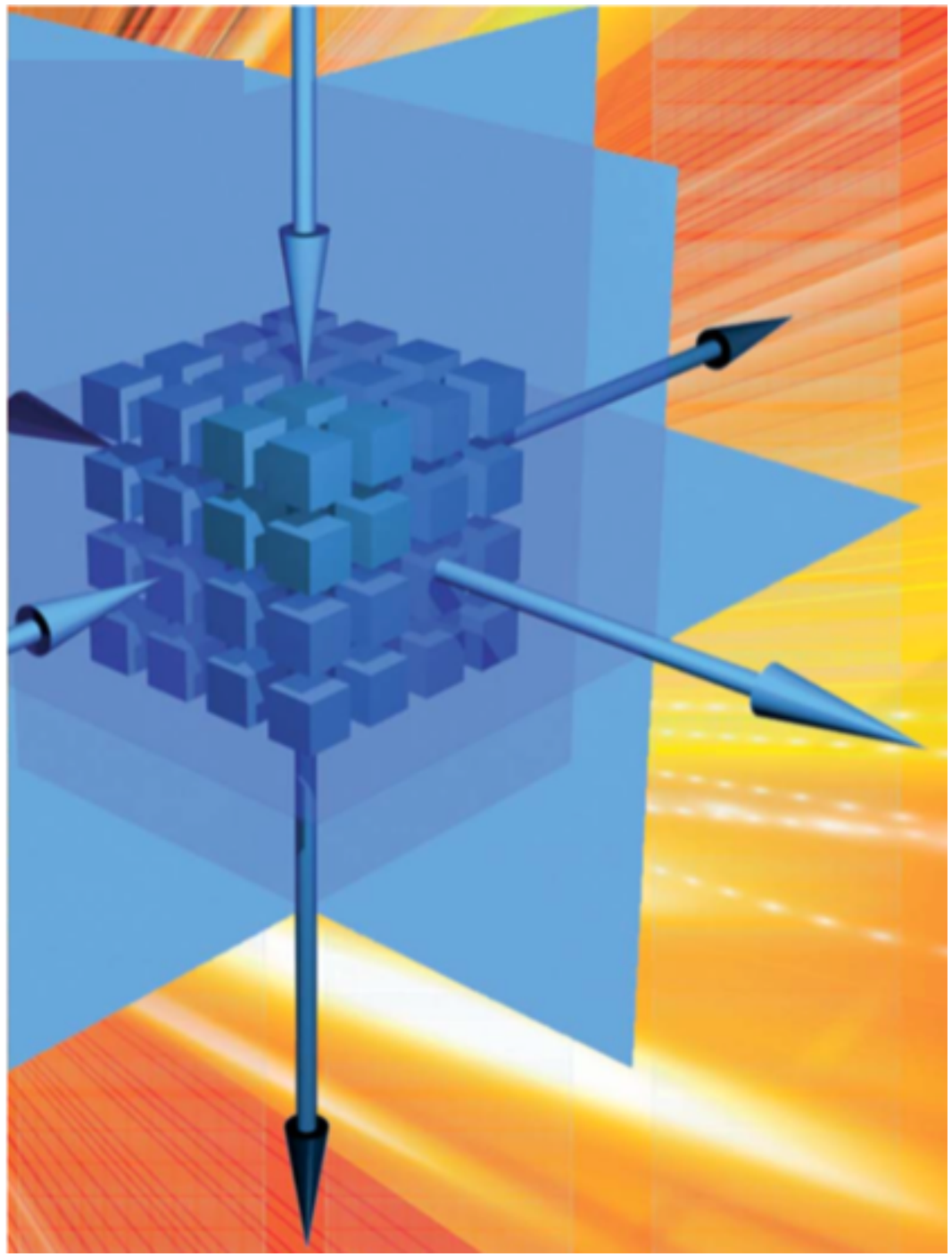
2. In addition to the cost and value extrapolation, based on incremented facts, and on hard credible evidence, we use a *second* sort of estimation:

'what will candidate strategy X cost, in time and/or money?

Decomposition

Separating out
small stakeholder-delivery
value increments

from your top-level
Architecture/Strategies



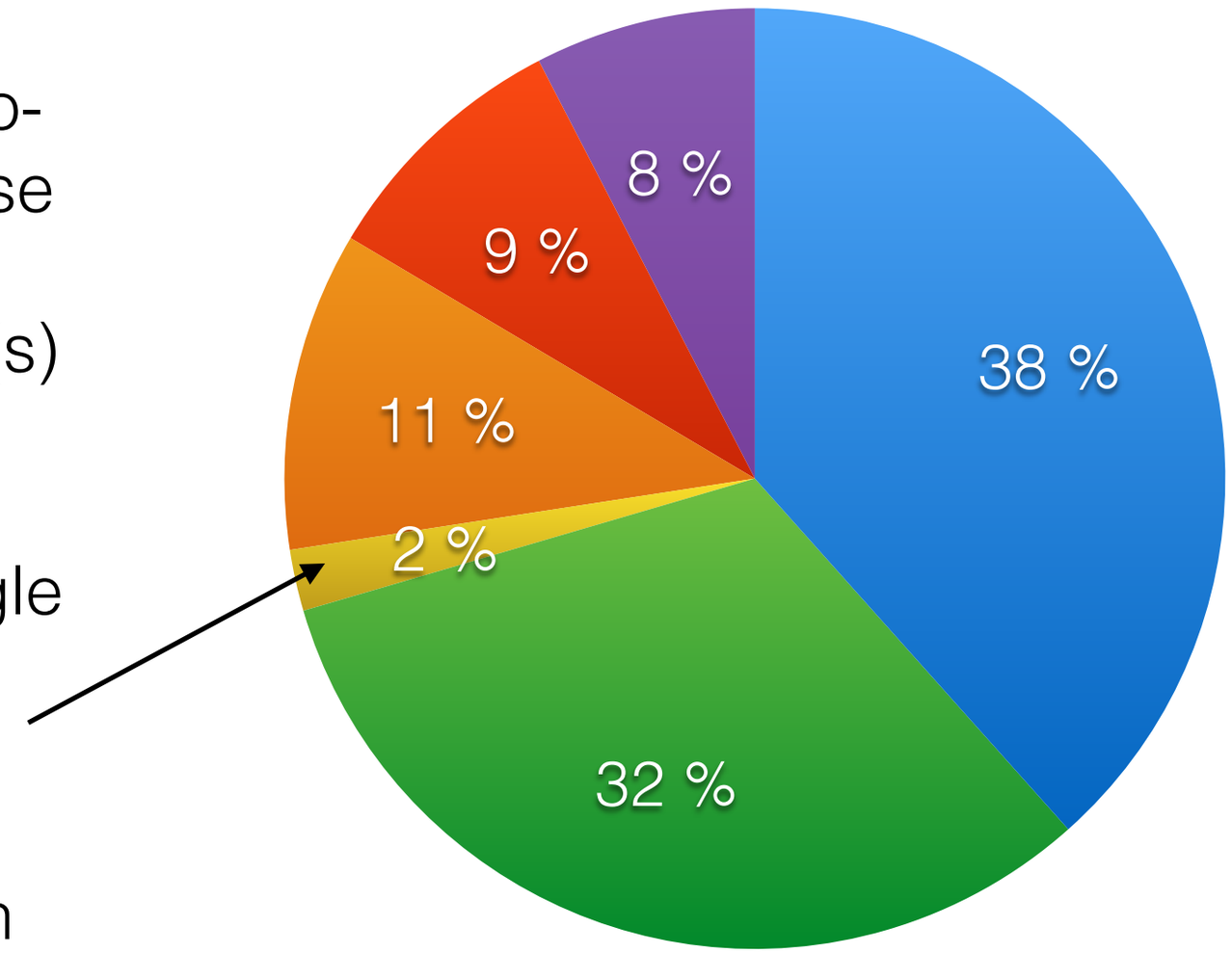
Ideal Separation of a Value-Delivery Step

1. No dependencies, that are not already existing in the to-be-incremented system base

2. Will give measurable value(s) to some stakeholder (s)

3. Can be completed in a single value delivery cycle (2% of time to deadline, a week)

4. Acceptable risk of deviation (±30% ?) from estimated values and costs



Methods for Extraction

1. Just ask: 'what could we do next week to deliver some value' ?

2. Use an Impact Estimation Table to decompose and see high value opportunities

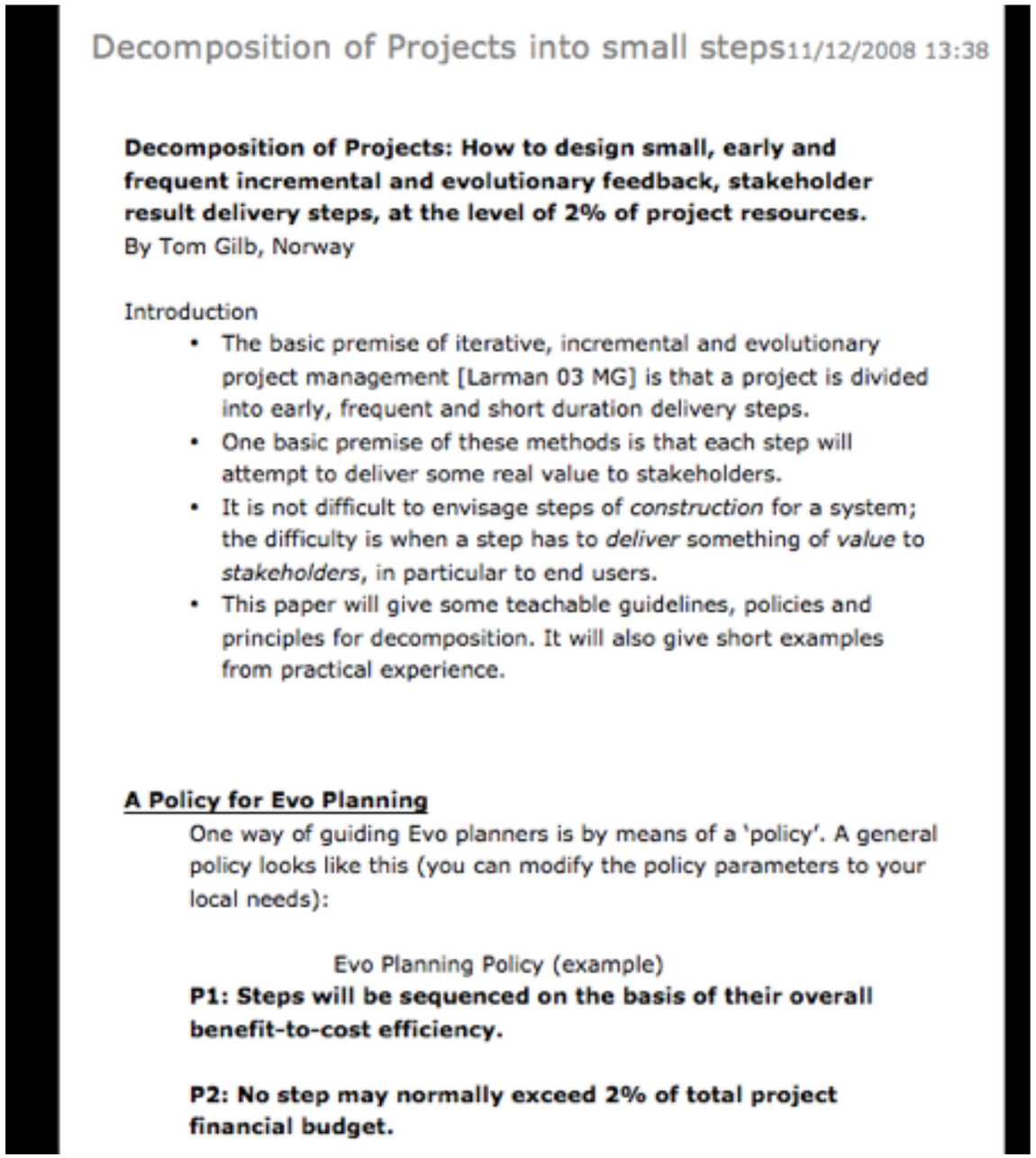
3. Use 20 Principles of Decomposition (CE Ch 10, VP)

US Army Example: PERSINSCOM: Personnel System

STRATEGIES → OBJECTIVES	Technology Investment	Business Practices	People	Empow-erment	*Principles of IMA Management*	Business Process Re-engineering	SUM
Customer Service ?→0 Violation of agreement	50%	10%	5%	5%	5%	60%	185%
Availability 90% → 99.5% Up time	50%	5%	5-10%	0	0	200%	265%
Usability 200 → 60 Requests by Users	50%	5-10%	5-10%	50%	0	10%	130%
Responsiveness 70% → ECP's on time	50%	10%	90%	25%	5%	50%	180%
Productivity 3:1 Return on Investment	45%	60%	10%	35%	100%	53%	303%
Morale 72 → 60 per mo. Sick Leave	50%	5%	75%	45%	15%	61%	251%
Data Integrity 88% → 97% Data Error %	42%	10%	25%	5%	70%	25%	177%
Technology Adaptability 75% Adapt Technology	5%	30%	5%	60%	0	60%	160%
Requirement Adaptability ? → 2.6% Adapt to Change	80%	20%	60%	75%	20%	5%	260%
Resource Adaptability 2.1M → ? Resource Change	10%	80%	5%	50%	50%	75%	270%
Cost Reduction FADS → 30% Total Funding	50%	40%	10%	40%	50%	50%	240%
SUM IMPACT FOR EACH SOLUTION	***482%***	***280%***	***305%***	***390%***	***315%***	***649%***	
Money % of total budget	15%	4%	3%	4%	6%	4%	
Time % total work months/year	15%	15%	20%	10%	20%	18%	
SUM RESOURCES	***30***	***19***	***23***	***14***	***26***	***22***	
BENEFIT/RESOURCES RATIO	***16:1***	**14:7**	**13:3**	**27:9**	**12**		

29.5 to 1

Decomposition Principles
A Teachable Discipline



Decomposition of Projects into small steps11/12/2008 13:38

Decomposition of Projects: How to design small, early and frequent incremental and evolutionary feedback, stakeholder result delivery steps, at the level of 2% of project resources.

By Tom Gilb, Norway

Introduction

- The basic premise of iterative, incremental and evolutionary project management [Larman 03 MG] is that a project is divided into early, frequent and short duration delivery steps.
- One basic premise of these methods is that each step will attempt to deliver some real value to stakeholders.
- It is not difficult to envisage steps of *construction* for a system; the difficulty is when a step has to *deliver* something of *value* to *stakeholders*, in particular to end users.
- This paper will give some teachable guidelines, policies and principles for decomposition. It will also give short examples from practical experience.

A Policy for Evo Planning

One way of guiding Evo planners is by means of a 'policy'. A general policy looks like this (you can modify the policy parameters to your local needs):

Evo Planning Policy (example)

P1: Steps will be sequenced on the basis of their overall benefit-to-cost efficiency.

P2: No step may normally exceed 2% of total project financial budget.

http://www.gilb.com/tiki-download_file.php?fileId=41

How to decompose systems into small evolutionary steps:

some principles to apply:

1• *Believe* there is a way to do it, you just have not *found* it yet!

2• *Identify* obstacles, but don't use them as excuses: use your imagination to get *rid* of them!

3• Focus on *some usefulness* for the user or customer, however small.

4• Do not focus on the design ideas themselves, they are distracting, especially for small initial cycles. Sometimes you have to ignore them entirely in the short term!

5• Think; one customer, tomorrow, one interesting improvement.

6• Focus on the *results* (which you should have defined in your goals, moving toward target levels).

7• Don't be afraid to use temporary-scaffolding designs. Their cost must be seen in the light of the value of making some progress, and getting practical experience.

8• Don't be worried that your design is inelegant; it is results that count, not style.

9• Don't be afraid that the customer won't like it. *If* you are focusing on results *they want*, then by definition, *they* should like it. If you are not,

10• Don't get so worried about "what might happen afterwards" t no practical progress.

11• You cannot foresee everything. Don't even *think* about it!

12• If you focus on helping your customer in practice, *now*, when need it, you will be forgiven a lot of 'sins'!

13• You can understand things much better, by getting *some* pra (and removing *some* of your fears).

14• Do *early* cycles, on willing local mature parts of your user co

15• When some cycles, like a purchase-order cycle, take a long time, initiate them early, and do other useful cycles while you wait.

16• If something seems to need to wait for 'the big new system', ask if you cannot usefully do it with the 'awful old system', so as to pilot it realistically, and perhaps alleviate some 'pain' in the old system.

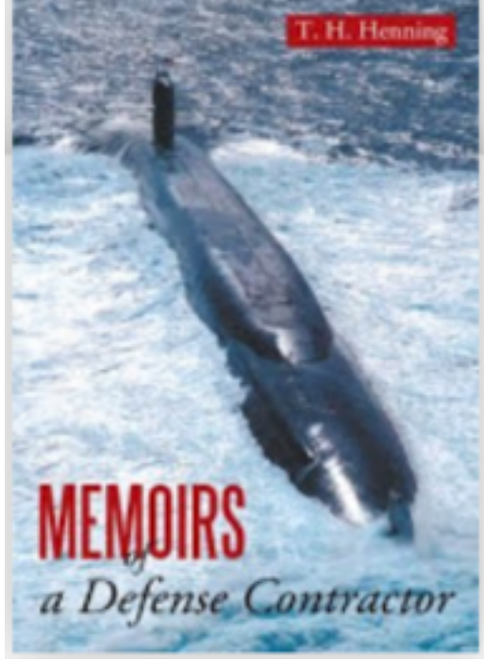
17• If something seems too costly to buy, for limited initial use, see if you can negotiate some kind of 'pay as you really use' contract. Most suppliers would like to do this to get your patronage, and to avoid competitors making the same deal.

18• If *you* can't think of some useful small cycles, then talk directly with the real 'customer' or end user. They probably have dozens of suggestions.

19• Talk with end users in *any* case, they have insights you need.

20• Don't be afraid to use the old system and the old 'culture' as a launching platform for the radical new system. There is a lot of merit in this, and many people overlook it.

I have never seen an exception in 33 years of doing this with many varied cultures. Oh Ye of little faith!



LAMPS Sub.

Cleanroom Method
Robert Quinnan
uses Dynamic Design to Cost
on 2% (monthly) steps
and result is years of always on time under
budget for 10 years on end.

On Military and Space Projects:
the highest state of art qualities

Cleanroom: IBM FSD, Federal Systems Division
(Agile ‘as it should be’: 1980-1990)
IBM SJ 4/1980, http://trace.tennessee.edu/utk_harlan/18/

Harlan Mills

DESIGN
The first guarantee of quality

“The first guarantee of **quality in design**
is in well-informed, well-educated, and well-motivated **designers**.

Quality must be built into designs, and cannot be inspected in or tested in.

Nevertheless, any prudent development process **verifies quality** through **inspection and testing.**

Inspection by peers in design, by users or surrogates, by other financial specialists concerned with cost, reliability, or maintainability
not only increases confidence in the design at hand,
but also provides designers with valuable lessons and insights to be applied to future designs.

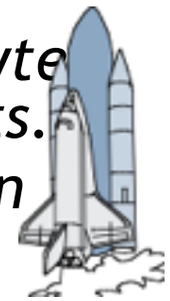
The very fact that **designs face inspections**
<u>motivates</u> even the most conscientious designers
to greater care, deeper simplicities, and more precision in their work.”

inIBM sj 4 80 p.419
In

Mills, H. 1980. The management of software engineering: part 1: principles of software engineering. IBM Systems Journal 19, issue 4 (Dec.):414-420.
Direct Copy
http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan
Library header
http://trace.tennessee.edu/utk_harlan/5/

In the Cleanroom Method, developed by IBM's Harlan Mills (1980) they reported:

- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) "some ten years ago [Ed. about 1970] in a continuing evolution that is still underway:*
- *Ten years ago general management expected the worst from software projects – cost overruns, late deliveries, unreliable and incomplete software*
- *Today [Ed. 1980!], management has learned to expect on-time, within budget deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and a ship in 45 incremental deliveries [Ed. Note 2%!]s. Every one of those deliveries was on time and under budget*
- *A more extended example can be found in the NASA space program,*
- *- Where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million byte of program and data for ground and space processors in over a dozen projects.*
- *- There were few late or overrun deliveries in that decade, and none at all in the past four years."*

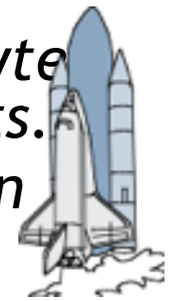


In the Cleanroom Method, developed by IBM's Harlan Mills (1980) they reported:

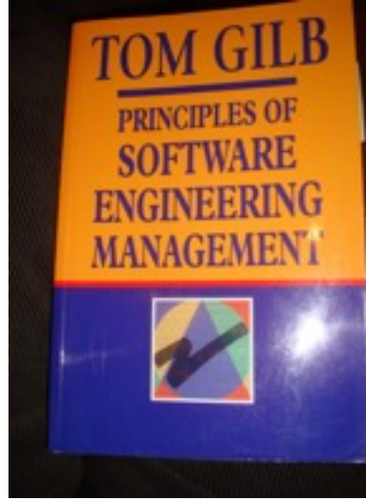
- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, f[illegible] about*
- *T[illegible]cts – cost overruns, late deliveries, unreliable and incomplete software*
- *Today [Ed. 1980!], management has learned to expect on-time, within budget deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over s[illegible]rocessors distrib[illegible]veries [Ed. Note 2[illegible]dget*
- *A mor[illegible]'*
- *- Whe[illegible]-years of softwe[illegible]million byte of pro[illegible]en projects.*
- *- Ther[illegible]ne at all in the po[illegible]*

in 45 incremental deliveries

were few late or overrun deliveries in that decade, and none at all in the past four years



Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. . . yields valid cost plans linked to technical performance. Our practice carries cost management farther by introducing design-to-cost guidance. Design, development, and managerial practices are applied in an integrated way to ensure that software technical management is consistent with cost management. The method [illustrated in this book by Figure 7.10] consists of developing a design, estimating its cost, and ensuring that the design is cost-effective.' (p. 473)

-

He goes on to describe a design iteration process trying to meet cost targets by either redesign or by sacrificing 'planned capability.' When a satisfactory design at cost target is achieved for a single increment, the 'development of each increment can proceed concurrently with the program design of the others.'

'Design is an iterative process in which each design level is a refinement of the previous level.' (p. 474)

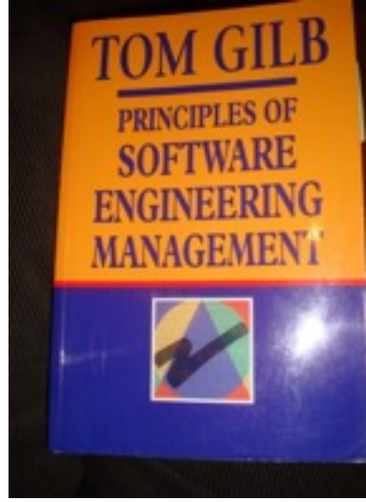
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability of learning from experience, won as each increment develops, and as the true cost of the increment becomes a fact.

'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77

This text is cut from Gilb: The Principles of Software Engineering Management, 1988

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. . . anagement farther by introducing design-t tegrated way to ensure that software technical m k by Figure 7.10] consists of developing a design,

-

_____He goes on to r by sacrificing 'planned capability.' When a sa t of each increment can proceed concurrently with the

'Design is an iterative

of developing a design, estimating its cost, and ensuring that the design is cost-effective

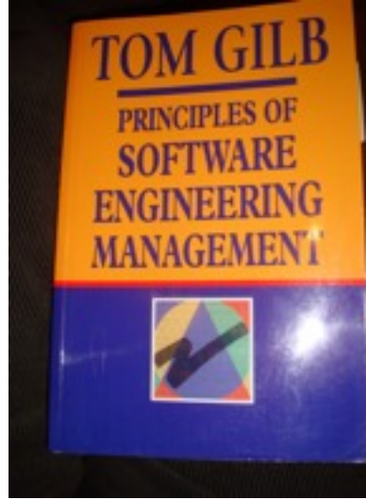
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability of learning from experience, won as each increment develops, and as the true cost of the increment becomes a fact.

'When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.' (p. 474)

Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77

This text is cut from Gilb: The Principles of Software Engineering Management, 1988

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

'Cost management. . . yields valid cost plans linked to technical performance. Our practice carries cost management farther by introducing design-to-cost guidance. Design, development, and managerial practices are applied in an integrated way to ensure that software technical management is consistent with cost management. The method [illustrated in this book by Figure 7.10] consists of developing a design, estimating its cost, and ensuring that the design is cost-effective.' (p. 473)

He goes on to describe a design iteration process trying to meet cost targets by either redesign or by sacrificing 'planned capability.' When a satisfactory design at cost target is achieved for a single increment, the 'development of each increment can proceed concurrently with the

'Design is an iterative

It is clear from **in seeking the appropriate balance between cos** **thus reducing the complexity of the task, and increas** **d as the true cost of the increment becomes a**

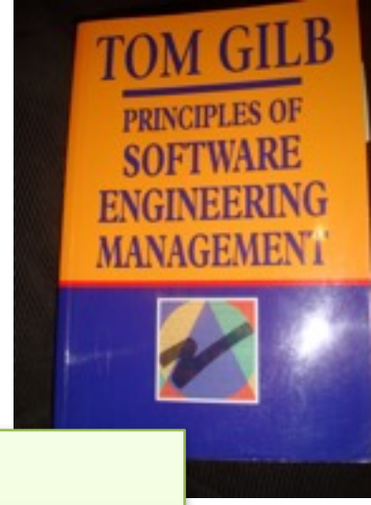
'When the developme **crements is computed.' (p. 474)**

Source: Robert E. Quir 1980, pp. 466~77

This text is cut from G

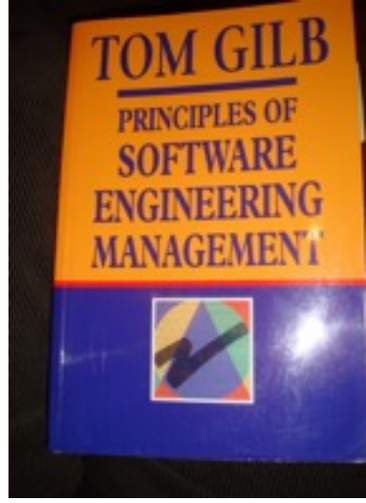
iteration process trying to meet cost targets by either *redesign* or by *sacrificing* 'planned capability'

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Design is an iterative process

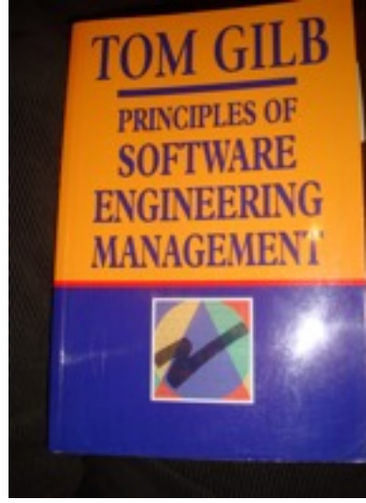
Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

but they iterate through a series of increments,
thus *reducing the complexity of the task*,
and *increasing the probability of learning from experience*

Quinnan: IBM FSD Cleanroom
Dynamic Design to Cost



Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

an estimate to complete the remaining increments is computed.

ed

of

)

Citibank London Case
Using Gilb's Evo & Planguage

Notice that designs that do not work
are immediately swapped
with hopefully better designs

20 Sept, 2015 Report on Gilb Evo method (Richard Smith, Citigroup)

- http://rsbatechnology.co.uk/blog:8
- Back in 2004, I was employed by a large investment bank in their FX e-commerce IT department as a business analyst.
- The wider IT organisation used a complex waterfall-based project methodology that required use of an intranet application to manage and report progress.
- However, it's main failings were that it almost **totally missed the ability to track delivery of actual value improvements to a project's stakeholders**, and **the ability to react to changes in requirements and priority for the project's duration.**
- The toolset generated lots of charts and stats that provided **the illusion of risk control**. but actually provided very little help to the analysts, developers and testers actually doing the work at the coal face.
- The proof is in the pudding;
  - I have **used Evo** (albeit in disguise sometimes) on two large, high-risk projects in front-office investment banking businesses, and several smaller tasks.
  - On the largest critical project, the original business functions & performance objective **requirements document, which included no design, essentially remained unchanged** over the 14 months the project took to deliver,
  - but the detailed **designs** (of the GUI, business logic, performance characteristics) **changed** many many times, guided by lessons learnt and **feedback** gained by delivering a succession of early deliveries to real users.
  - In the end, the new system responsible for 10s of USD billions of notional risk, **successfully went live over one weekend for 800 users worldwide**, and **was seen as a big success by the sponsoring stakeholders.**

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006"

Richard Smith

“ I attended a 3-day course with you and Kai whilst at Citigroup in 2006”

Previous PM Methods:
No ‘Value delivery tracking’.
No change reaction ability

Richard Smith

- “However, (**our old** project management methodology) main failings were that
- it almost **totally missed the ability to track delivery of actual *value* improvements to a project's stakeholders**,
- and **the ability to react to changes**
  - **in requirements and**
  - **priority**
  - **for the project's duration”**

We only had the illusion of control. But little help to testers and analysts

Richard Smith

- "The (old) toolset generated lots of charts and stats
- that provided **the illusion of risk control**.
- But actually provided very little help to the analysts, developers and testers actually doing the work at the coal face."

The proof is in the pudding;

Richard Smith

- “The proof is in the pudding;
- I have **used Evo**
  - *(albeit in disguise sometimes)*
  - on two large, high-risk projects in front-office investment banking businesses,
  - and several smaller tasks. “

Experience: if top level requirements are *separated* from design, the 'requirements' are **stable**!

Richard Smith

- "On the largest critical project,
- the original ***business functions & performance objective*** **requirements document,**
- ***which included*** **no design,**
- essentially remained ***unchanged***
- over the **14 months** the project took to deliver,…."

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006", Richard Smith

Dynamic (Agile, Evo) design testing: not unlike 'Lean Startup'

Richard Smith

- "… but the detailed **designs**
  - **(of the GUI, business logic, performance characteristics)**
- **changed** many many times,
- guided by lessons learnt
- and **feedback** gained by
- delivering a succession of early deliveries
- to real users"

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006", Richard Smith

It looks like the stakeholders liked the top level system qualities, on first try

Richard Smith

- " In the end, the new system responsible for 10s of USD billions of notional risk,
- **successfully went live**
- **over one weekend**
- **for 800 users worldwide**,
- and **was seen as a big success**
- **by the sponsoring stakeholders."**

" I attended a 3-day course with you and Kai whilst at Citigroup in 2006" , Richard Smith

Tom Gilb & Kai Gilb

www.Gilb.com

Our Column
http://tinyurl.com/AGILEMYTHS