SCALE-FREE:
Practical Scaling Methods
for Industrial Systems Engineering

Tom Gilb
for ALSSE
Norsec, Oslo
Monday 12 September 2016, minor edit 9March17

Based on a paper:

"Some Advanced Tools and Principles for Scaling Agile Projects - Agile Engineering.
40 practical Engineering ideas for scaling agile development successfully all the time."

A very short pdf paper, supported by references to necessary detail. Not least the The new LeanPub.com/ ValuePlanning book

http://www.gilb.com/dl865

And based on detailed methods in Value Planning book

http://gilb.com/dl853
(free digital copy, for YOU)
Planning book

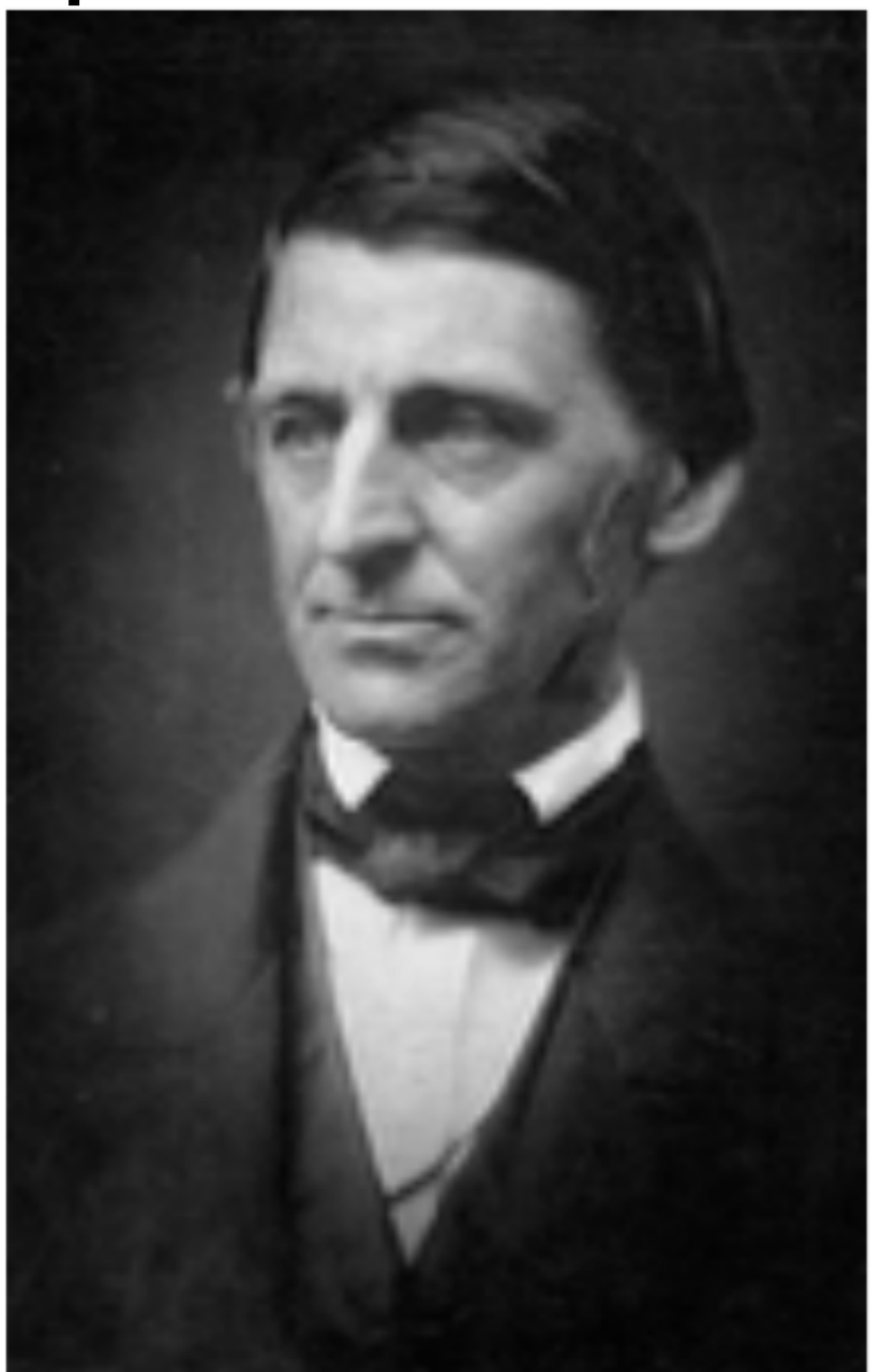
Commercially available
Leanpub.com/ValuePlanning

og Get a Deal at Gilb.com for this and related products like Videos and Courses
My current book manuscript Value Planning
For my friends, use coupon code **VP5** for a €5 discount.

Principles And Methods for ‘Any Scale’ SYSTEMS ENGINEERING Projects

Principles

- *“As to methods, there may be a million and then some, but principles are few.*
- *The man who grasps principles can successfully select his own methods”.*
- - Ralph Waldo **Emerson,**
  - 1803-1882, USA



Erik Simmons, Intel Scaling

On 08 Jan 2016, at 19:30, Simmons, Erik erik.simmons@nucognitive.com wrote:
Just a couple of things come to mind after reading this:

(Gilb:

Beyond Scaling: Scale-free Principles for Agile Value Delivery - Agile Engineering.

http://www.gilb.com//dl865
Version March 14 2016, Modified April 11 2016 (XP)

Cheers,

e

Erik Simmons, Intel Scaling

I've not been a fan of the scaling movement since it started.

There are very **few things that scale well**, and economies of scale are often pursued without adequate understanding of the accompanying **diseconomies of scale.**

SW development does not scale well

- *because* of the **diseconomies of complexity**,
- such as the number of communication pathways,
- cognitive load on programmer brains, etc.
- That is among the core reasons for Brooks' Law.

What makes us think that scaling Scrum, which is successful in small teams and projects, is a good idea?

A grown-up is not a scaled baby.

Scaling as a concept is selling a lot of books, consulting, and certifications right now. But **I don't think it is a valuable concept.**

erik.simmons@nucognitive.com

Erik Simmons, Intel Scaling

- Instead, I believe that the **majority of what you have** included for ideas, principles, etc. from CE and VP are in fact **scale-free**.

- They are **not dependent on *project* or *organization* size.**

- They are **good heuristics for almost any project,**

- and **nearly universally applicable**
  - (nearly universal because I hear Koen in my head, and all is heuristic).

- So, CE and VP are not *about* scaling
  - so much as they should be taught and understood as **scale-free.**

- Size is not a reason to choose (or not choose) to use Competitive Engineering, Evo, Planguage, etc.

- As you quoted me in the paper – **this stuff works**.
  - It works on **small** projects. It works on **large** projects.

- Evo on a 5-person team is not really much different than Evo on a 100-person team, except there are more people.

- The principles apply **without alteration** (or “scaling”).

- Anyone who sees a random page of your new paper would probably not guess the topic is scaling (unless you happen to mention that in the text on that particular page).

- **‘Competitive Engineering’ does not scale. It doesn’t need to.**

erik.simmons@nucognitive.com

Erik Simmons, Intel Scaling

- There's no doubt that large projects are *different*.
- There's no doubt that we should approach them *differently*.
- We still don't have a recipe for large projects, and probably never will.

- But all that does *not* lead me to think that the answer to large projects can be found in scaling successful practices for small projects.

- Instead, **it must be found** in use of **principles and practices that are scale-free**,
  - coupled with use of particular practices that are effecting on large projects.

- If something that works on small projects also works on large projects, then I'd propose we call it a **scale-free practice**, not a scaled practice.

erik.simmons@nucognitive.com

Erik Simmons, Intel Scaling

- I'm deeply interested in scale-free practices.
- I'm also interested in specific practices tuned to large, small, complicated, and complex projects,

- but I find **particular power in scale-free practices.**

- Your work for decades has been focused on a very good set of these.
  - **SQC, for example, works on any size specification. It does not (need to) scale.**
  - SQC: (Specification Quality Control).see next slide
-
- BTW, I think the agile principles are also quite scale-free. But most **Scrum practices are definitely not.**
-
- So, perhaps you can chart a better course by advocating for use of scale-free core practices,
  - augmented with a set of specific, tailored practices
  - that are effective for the size of the project in question.

erik.simmons@nucognitive.com

Source Eric Simmons,erik.simmons@nucognitive.com 25 Oct 2011-17
Personal Public Communication

A Recent Example

Application of Specification Quality Control by a SW team resulted in the following defect density reduction in requirements over several months:

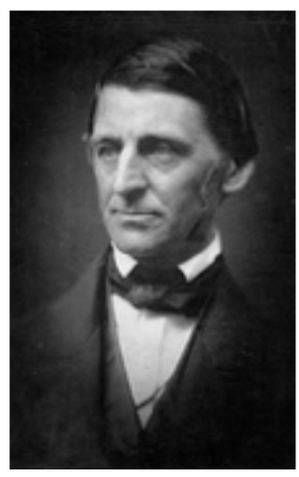
Rev.	# of Defects	# of Pages	Defects/ Page (DPP)	% Change in DPP
0.3	312	31	10.06	
0.5	209	44	4.75	-53%
0.6	247	60	4.12	-13%
0.7	114	33	3.45	-16%
0.8	45	38	1.18	-66%
1.0	10	45	0.22	-81%
Overall % change in DPP revision 0.3 to 1.0:				-98%

Downstream benefits:
- Scope delivered at the Alpha milestone increased 300%, released scope up 233%
- SW defects reduced by ~50%
- Defects that did occur were resolved in far less time on average

The Impact of a Requirements Specification on Software Defects and Other Quality Indicators
http://selab.fbk.eu/re11_download/industry/Terzakis.pdf
see his 2013 IEEE IREC Rio paper for update: see comment in this slide

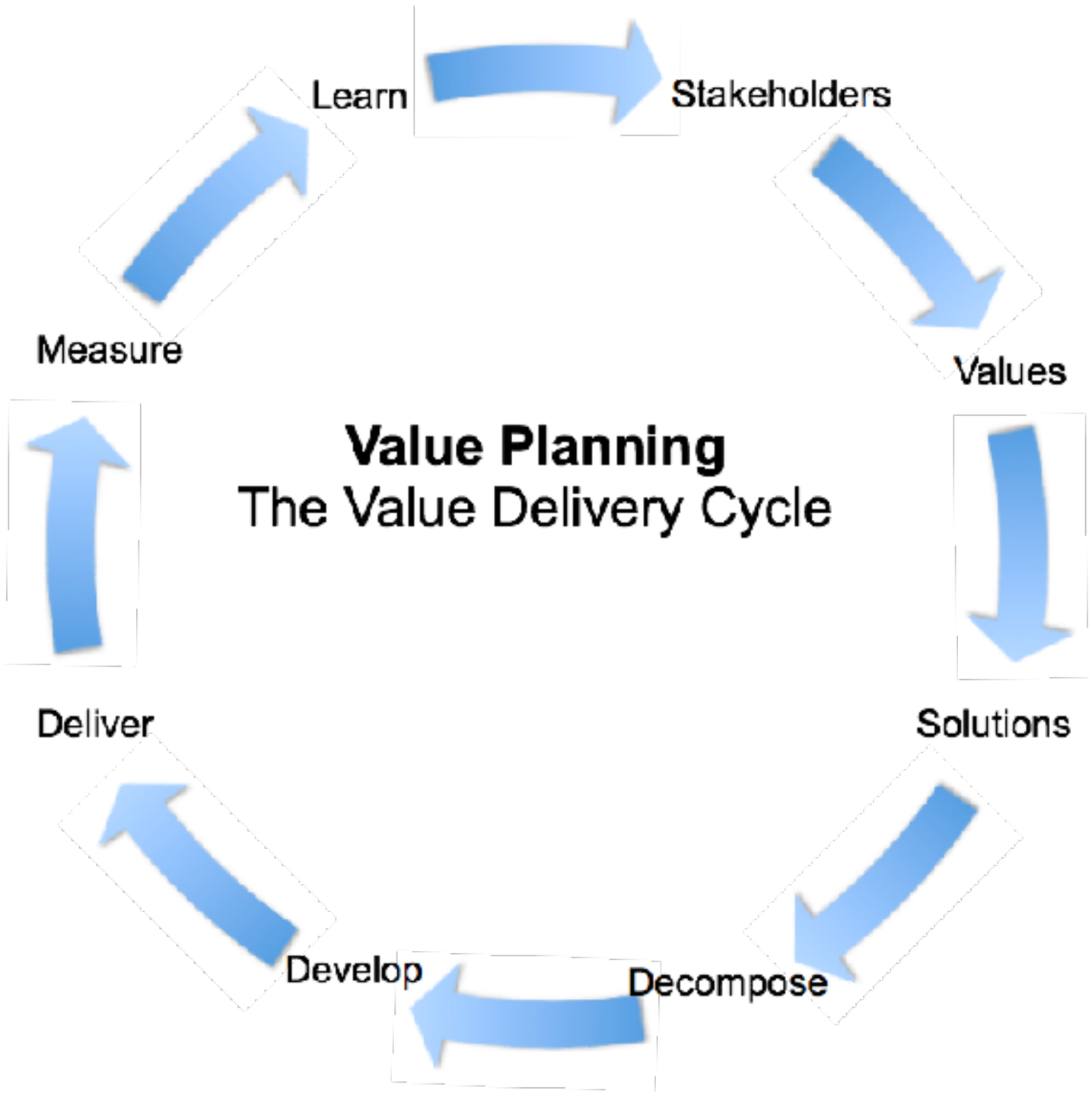
Scale-free Principles

1. Keep focus on measurable delivery of critical values and their costs. [3, 4, 5, 6, 9, 10, 12, VP (20) Part 1**, VP 10.6**]
2. Deliver value early, quickly and regularly: in roughly 2% increments. [14, 11, **VP Ch.4,** 2**,** 5]
3. Do NOT focus on code delivery; focus on overall system value and costs. [**VP Ch.4,** 10D, 10F, 13, VP 3.4, VP 2.10, VP 9.8, 4, 12]
4. Focus on quantified *critical stakeholder* values. [**19**, VP 3.4, VP 3.7, VP 3.9, VP 3.10 VP 4.2, 10]
5. Synchronize all teams in terms of measurable value delivery. [VP 3.3, VP 3.4, VP Part 1, VP 3.6, VP 3.8, VP 8.4 , 11, 12, 13]
6. Solve big problems through ingenious architecture; not through coding faster. [VP 4.5, VP 5.1, VP 5.3, VP 7.2, 15]
7. Decompose the large problems by incremental value deliveries: not code deliveries. [**7**, VP Ch. 5, VP 5.1, VP 5.6 , 10, 11, 13, 15]
8. The software component needs to be integrated into the total system of hardware, data, people, culture. [VP 5.2, 10]
9. If your team cannot deliver small increments of real value early, frequently, and predictably; they are incompetent and need to be abandoned for those who can deliver. [7, VP 2.8, 10]
10. Never commit to contacts for *work done* or *code delivered* alone: there must always be a sufficiently large contractual protection, of paying for measurable value delivered. [12, 15].



Methods

1. Quantification of Values [10, VP 1.1].
2. Quantification of short term and long term costs [VP 3.4, VP 4.5, VP 6.7].
3. Design to Cost: Top Level Architecture [VP 7.9, 10].
4. Dynamic Design to Cost: Each Delivery Cycle [12 C, VP 4.5, VP 2.5, VP 2.3, 5, 10, 12].
5. Quality Control of Plans, Contracts, Code and all written artifacts [VP Part 2, VP Part 4, VP 7.7].
6. Flexible Contracting [12, VP 4.5].
7. **Value delivery Cycle Measurable Feedback, Learning and Change [4, VP 7.3, VP 9.8, VP 6.7, VP 8.6, 2, 9, 10, 11, 14].**
8. **Value Decision Tables (Impact Estimation Tables) [9, VP 2.3, VP 4.4, VP 5.3, 13].**
9. Risk Management in all aspects of planning and Management [VP Ch. 7], 12.
10. Intelligent Prioritization Policies: for short term and long term [VP Ch. 6, 12, 13, 14].



Value Planning Cycle = ‘Evo’
Cycle of Value delivery of any size project

7. Value delivery Cycle Measurable Feedback, Learning and Change [4, VP 7.3, VP 9.8, VP 6.7, VP 8.6, 2, 9, 10, 11, 14].

PERSINSCOM: Value Decision Table

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**	**12:1**	**29.5:1**	

Top Level View of Any Size Project
a Model of Relation between
Requirements and **Architecture**

8. Value Decision Tables (Impact Estimation Tables) [**9**, VP 2.3, VP 4.4, VP 5.3, 13].

Engineering Tools

1. The Planning language: ‘Planguage’ [22, VP, 8, 9].
2. The 111111 Decomposition Method [7B, 7C, 3].
3. Flexible Contracts [12].
4. **The ‘Needs and means Planning’ tool [16, 9].**
5. Quantification of Values processes: Scales, Meters, Past, Tolerable, Wish, Goal. [VP 10.7].
6. The Agile Spec QC measurement process, Exit Processes, Rules [VP 10.4, VP Part 4].
7. Multiple Relationship Management technology [9, VP Ch.3, VP Ch. 6, 13].
8. Continuous Architecture adjustment based on delivery cycle feedback (Cleanroom) [5, 14, 8].
9. Graphic Visibility of Values, Costs, and Risks [16].
10. Design to Cost Practices: initially and continuously [14, 12 C, VP 4.5, VP 2.5, VP 2.3, 5].

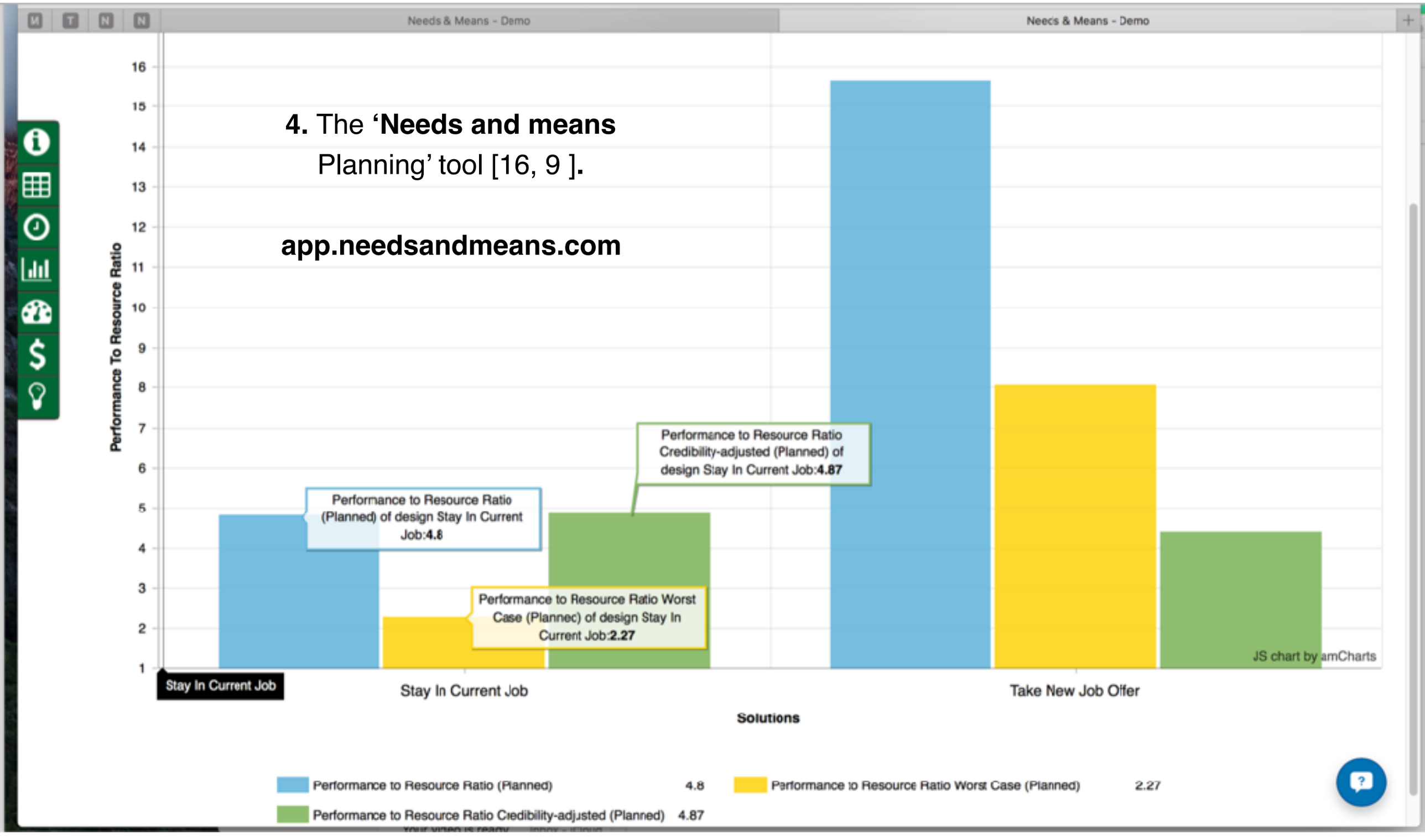
4. The ‘**Needs and means** Planning’ tool [16, 9].

app.needsandmeans.com

Requirements	Stay In Current Job	Take New Job Offer	Sum
Forecast 36 Month Income Status: 350000 → Wish: 540000 £ GB Pounds *No qualifiers* 2016	=:390000 ± 10000 £ 0 Δ: 40000 £ Δ%: **21** ± 5 % 21 %: *11 %* (x 0.5)	=:450000 ± 10000 £ 0 Δ: 100000 £ Δ%: **53** ± 5 % 74 %: *11 %* (x 0.2)	ΣΔ%: **74** ± 10 %
Mortgage Kill Likelihood Status: 5 → Wish: 20 % Percentage [New Qualifier 1 = Qualifier Value] 2016	=:5 ± 0 % 0 Δ: 0 % Δ%: **0** ± 0 % 0 %: *0 %* (x 1.0)	=:105 ± 30 % 0 Δ: 100 % Δ%: **667** ± 200 % 667 %: *67 %* (x 0.1)	ΣΔ%: **667** ± 200 %
Likely Retirement Age Status: 55 → Wish: 50 Years Old Age at Retirement *No qualifiers* 2016	=:55 ± 0 Years Old 0 Δ: 0 Years Old Δ%: **0** ± 0 % 0 %: *0 %* (x 1.0)	=:45 ± 5 Years Old 0 Δ: 10 Years Old Δ%: **200** ± 100 % 200 %: *40 %* (x 0.2)	ΣΔ%: **200** ± 100 %
Work Life Balance Of Weekends Status: 8 → Wish: 0 Hours Hours Working *No qualifiers* 2016	=:12 ± 1 Hours 0 Δ: -4 Hours Δ%: **-50** ± 13 % -50 %: *-25 %* (x 0.5)	=:10 ± 2 Hours 0 Δ: -2 Hours Δ%: **-25** ± 25 % -75 %: *-3 %* (x 0.1)	ΣΔ%: **-75** ± 38 %
Work Life Balance Of Weekd... Status: 6 → Wish: 0 Hours Hours Working *No qualifiers*	=:9 ± 1 Hours 0 Δ: -3 Hours Δ%: **-50** ± 17 % -50 %: *0 %* (x 0.0)	=:7 ± 0.5 Hours 0 Δ: -1 Hours Δ%: **-17** ± 8 % -67 %: *-2 %* (x 0.1)	ΣΔ%: **-67** ± 25 %
Role Stretch Status: 5 → Wish: 40 Percent Percentage *No qualifiers* 2016	=:5 ± 0 Percent 0 Δ: 0 Percent Δ%: **0** ± 0 % 0 %: *0 %* (x 0.8)	=:45 ± 30 Percent 0 Δ: 40 Percent Δ%: **114** ± 86 % 114 %: *46 %* (x 0.4)	ΣΔ%: **114** ± 88 %
Respect For Direct Boss Status: 0.5 → Wish: 5 Days No. Days *No qualifiers* 2016	=:2 ± 1 Days 0 Δ: 1.5 Days Δ%: **33** ± 22 % 33 %: *17 %* (x 0.5)	=:5 ± 2 Days 0 Δ: 4.5 Days Δ%: **100** ± 44 % 133 %: *80 %* (x 0.8)	ΣΔ%: **133** ± 88 %
Environmental Fit Status: 90 → Wish: 100 % Percentage *No qualifiers* 2016	=:100 ± 0 % 0 Δ: 10 % Δ%: **100** ± 0 % 100 %: *80 %* (x 0.8)	=:100 ± 5 % 0 Δ: 10 % Δ%: **100** ± 50 % 200 %: *20 %* (x 0.2)	ΣΔ%: **200** ± 50 %
Leadership Level Status: 2 → Wish: 3 Role Scale (1 - Head of, 2 - Directo... *No qualifiers* 2016	=:2 ± 0 0 Δ: 0 Δ%: **0** ± 0 % 0 %: *0 %* (x 0.5)	=:3 ± 1 0 Δ: 1 Δ%: **100** ± 100 % 100 %: *100 %* (x 1.0)	ΣΔ%: **100** ± 100 %
EVO Practice Challenge Status: 3 → Wish: 5 1 - Very Low Challenge, 2 - Low Chal... *No qualifiers* 2016	=:3 ± 0 0 Δ: 0 Δ%: **0** ± 0 % 0 %: *0 %* (x 1.0)	=:5 ± 1 0 Δ: 2 Δ%: **100** ± 50 % 100 %: *80 %* (x 0.8)	ΣΔ%: **100** ± 50 %
Access To Mentors			

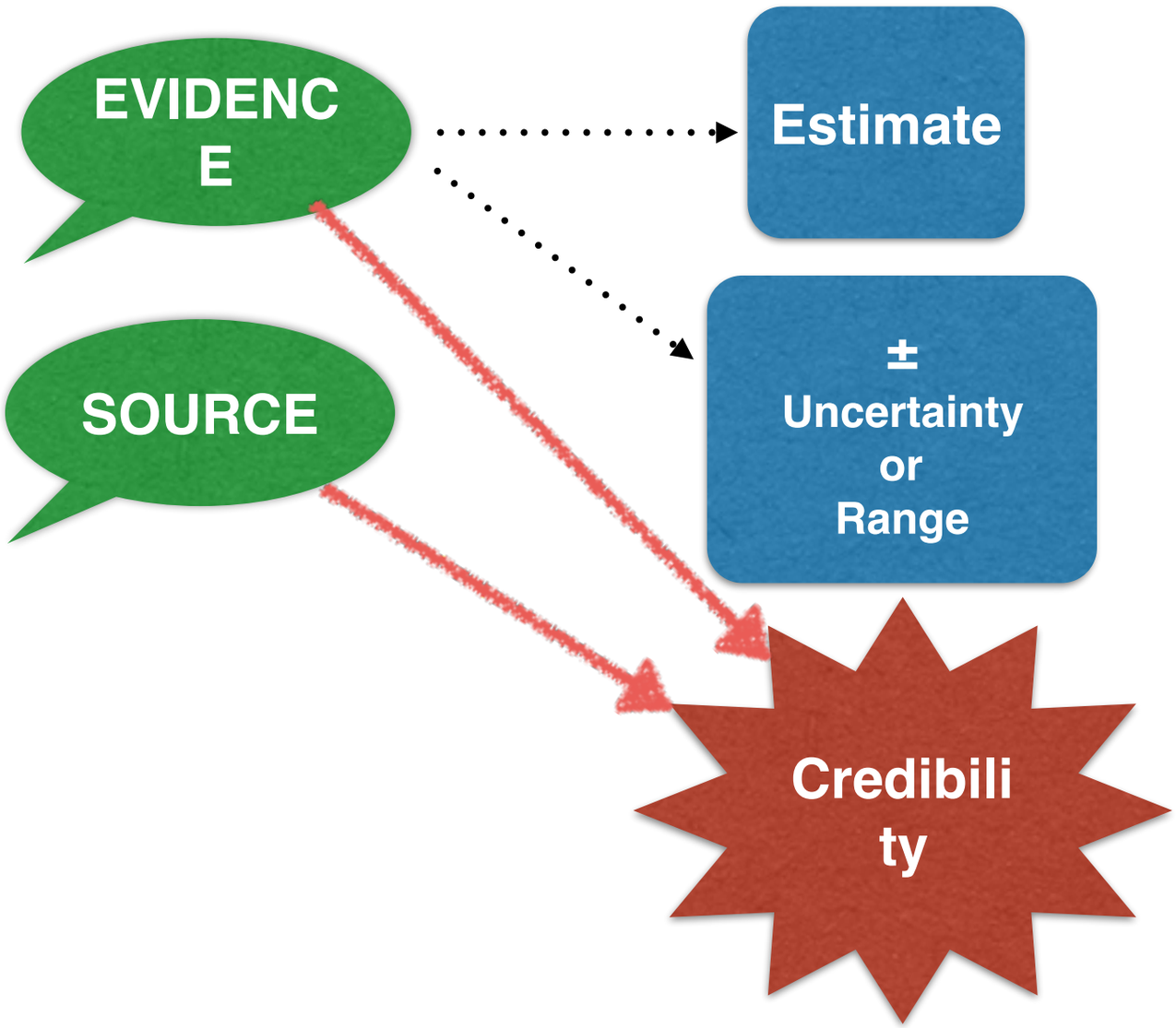
4. The ‘**Needs and means** Planning’ tool [16, 9].

app.needsandmeans.com



Management Policies for any scale

1. We will primarily manage critical stakeholder value improvements [VP Ch. 3, 8, 19].
2. We will simultaneously manage the short term and long term resources [VP 2.5, VP 9.9, VP 10.6,].
3. We will contract for measurable values for money, rather than 'work done' [**12**, VP 8.4].
4. We will manage all basic system qualities in a quantified engineering manner. [VP Part 1, 3, 4, 8, 10, 17].
5. We will prioritize delivery of measurable value, early, frequently, predictably [VP 1.2, **VP Ch. 6**, 6, 12].
6. We will not lock ourselves into investments or expenditures of any kind that cannot be reversed if they do not produce expected value for money [VP Ch. 7, VP 8.10].
7. **We will make the risks of all strategies, designs, actions, and relationships visible numerically; and make decisions with regard to worst-case risks [VP Ch. 7, 12, 17].**
8. We will empower the 'troops' to make real-time project decisions, based on current numeric feedback, about real values and real costs [**VP 8.1, VP 8.2**, 10, 11, 13].
9. Every team or set or related teams will be judged by their ability to deliver a measurable, predictable value improvement stream [VP 8.7].
10. Decisions will not be made on badly-defined-package costs: decisions will be made continuously, and if necessary retracted, on provable values for costs, with regard to risks. [VP 7.2, 3, 6, **17**]



Home / Value Tables / BRexit Value Decision Table

BRexit Value Decision Table

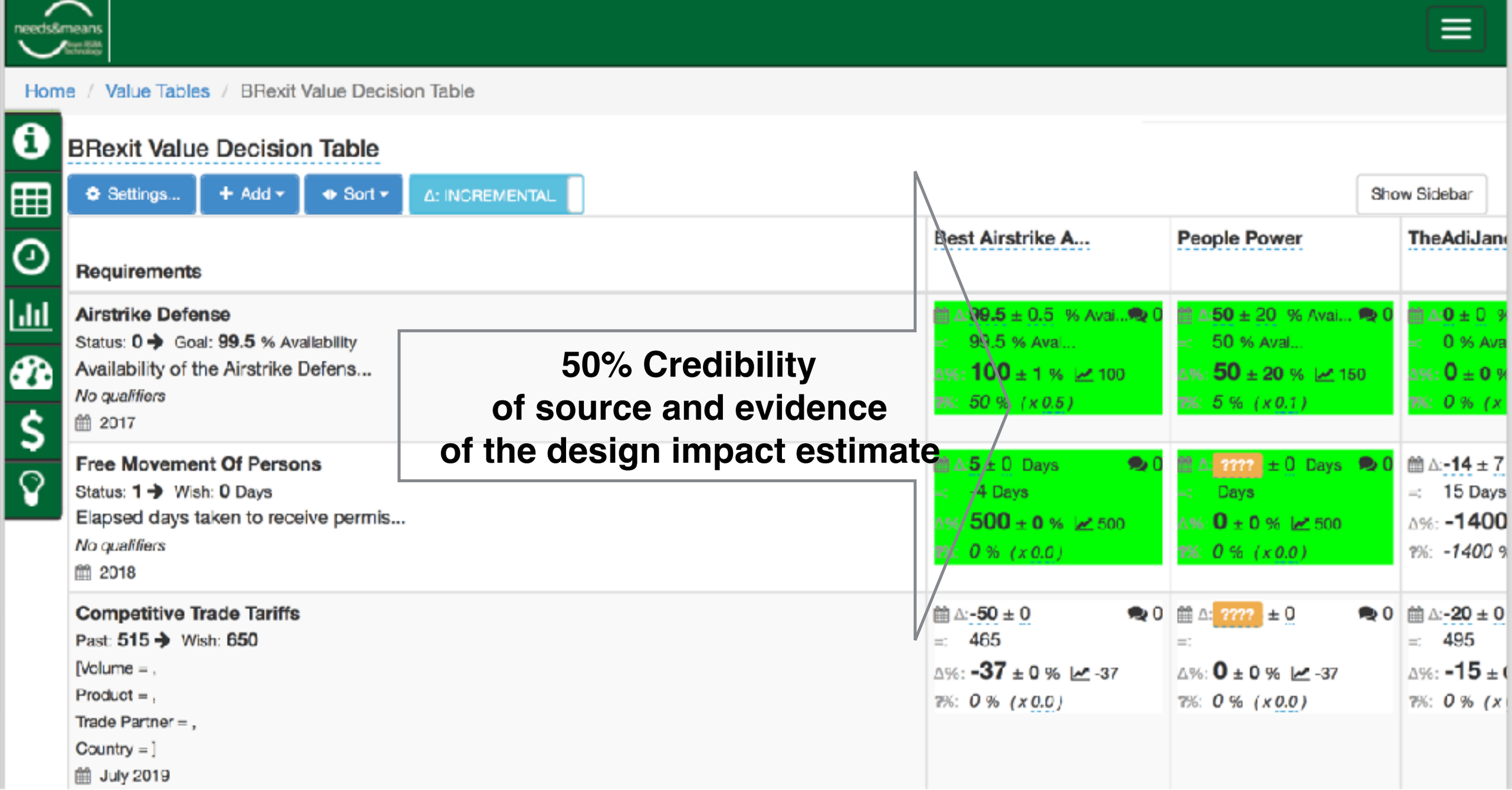
Settings... | + Add | Sort | Δ: INCREMENTAL | Show Sidebar

± deviation

Requirements	Best Airstrike A...	People Power	TheAdiJan
Airstrike Defense Status: **0** → Goal: **99.5** % Availability Availability of the Airstrike Defens... *No qualifiers* 2017	Δ: 99.5 ± 0.5 % Avai... 0 =: 99.5 % Ava... Δ%: **100** ± 1 % 100 ?%: *50 % (x 0.5)*	Δ: 50 ± 20 % Avai... 0 =: 50 % Avai... Δ%: **50** ± 20 % 150 ?%: *5 % (x 0.1)*	Δ: 0 ± 0 % =: 0 % Ava Δ%: 0 ± 0 % ?%: 0 % (x
Free Movement Of Persons Status: **1** → Wish: **0** Days Elapsed days taken to receive permis... *No qualifiers* 2018	Δ: 5 ± 0 Days 0 =: -4 Days Δ%: **500** ± 0 % 500 ?%: *0 % (x 0.0)*	Δ: ???? ± 0 Days 0 =: Days Δ%: **0** ± 0 % 500 ?%: *0 % (x 0.0)*	Δ: -14 ± 7 =: 15 Days Δ%: -1400 ?%: -1400 %
Competitive Trade Tariffs Past: **515** → Wish: **650** [Volume = , Product = , Trade Partner = , Country =] July 2019	Δ: -50 ± 0 0 =: 465 Δ%: **-37** ± 0 % -37 ?%: *0 % (x 0.0)*	Δ: ???? ± 0 0 =: Δ%: **0** ± 0 % -37 ?%: *0 % (x 0.0)*	Δ: -20 ± 0 =: 495 Δ%: -15 ± ?%: 0 % (x

Large Scale Problem (Brexit) with risks visible

Management Policy 7. We will **make the risk**s of all strategies, designs, actions, and relationships visible **numerically**; and make decisions with regard to worst-case risks [VP Ch. 7, 12, 17].



50% Credibility
of source and evidence
of the design impact estimate

Large Scale Problem (Brexit) with risks visible

Management Policy 7. We will **make the risk**s of all strategies, designs, actions, and relationships visible **numerically**; and make decisions with regard to worst-case risks [VP Ch. 7, 12, 17].

Why do these scaling ideas work?

1. **Value quantification** allows us to focus on the stakeholder results, the main objectives of any project. All other activity, below this level should be contributing to delivery of the planned values. This means we can delegate the activity to any combination of specialist teams of any size and complexity: yet we can judge whether things are 'working'. We keep our eyes on measured value delivery. We can judge whether both our organization and our architecture are delivering as expected and needed. If not we can adjust (**dynamic design to cost**) and go with things that are actually delivering necessary value.
2. **Contracting for value** relates to the above explanation, with the added benefit that outside contractors are now motivated to focus on value delivery, not just 'doing work', or 'programming'. It does not matter so much about the underlying complexity. That underlying complexity either works (delivers contracted value measurably) or not. If not, we change it until it does, or give up if we cannot change to satisfy value delivery needs.
3. **Decomposition by small 2% deliverable value architecture components**: this is a very basic attack on large size and consequent complexity. We can see the incremental impact of each step on the whole system, regarding both value delivery and costs. If it is not good enough we try new ideas. If we run out of ideas that work, we need to stop.
4. **Risk Management:** our methods, including 1-3 above, are really all about managing the risk of failing to deliver value for money, on time. In addition we have suggested a number of additional risk management ideas. For example estimating the ± uncertainty of a design impact on values and costs [9]. For example asking for specific evidence [9] that any given design, or strategy will deliver the values and costs we need. The more engineering effort we put in to planning for risk up front, the less likely we are to get nasty surprises later (and then blame them on 'project size and complexity'; rather than our own lack of decent engineering planning).
5. **Delegation of decision-making [23]. Delegating the power to make decisions to a grass roots level, and in addition to do so incrementally while keeping any eye of their level of concern (in terms of value and costs), should obviously help us make better decisions, in an evidence-based situation.**

I have personally used these methods, with remarkable success, on projects involving for example 1,000 programmers and 1,000 hardware engineers (example HICOM (which was in total failure mode after 2 years, at Siemens. Boeing Aircraft projects [thousands of employees involved. To mention just a couple of many). There is no doubt for me that they work, and why they work.

Delegation Examples

My Own Project Development Process: delegation of design to implementors and programmers

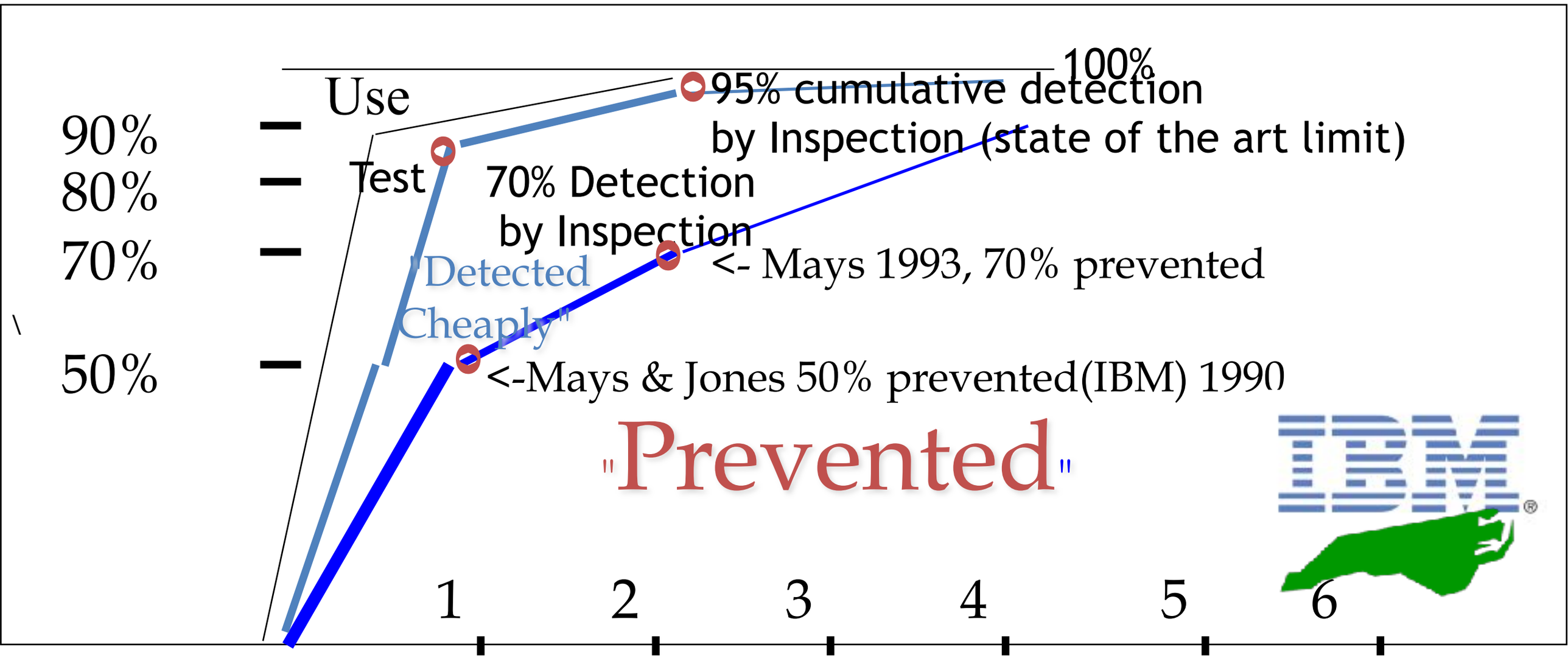
Competitive Engineering: Book 2005
http://tinyurl.com/CEset2015
Is a dropbox set of Full Glossary
Chapter By Chapter pdfs and full pdf

- Make *developers* **responsible**
  - **for delivery of the 'quantified' critical requirements levels**
    - (Performance, Qualities, cost, deadline)
- Give them the **freedom to decide the 'right' designs**
  - With immediate responsibility to *measure* that they are delivering the results
- Get the 'unprofessional' users and **customers 'off their backs'**
  - Avoid receiving features and stories; avoid 'architecture from managers'.
    - which are usually amateur design, by people who have no overview or responsibility or design ability (users and customers, and managers)
- **Elevate your talent** by becoming a **real 'software ENGINEER'**
  - With coding-expert craftsmanship, as your basic talent

Cases and real examples
'Value Driven Project Management' slides
Includes 'Confirmit' Case, slide 70 on.
http://www.gilb.com/dl152

These slides are at
http://www.gilb.com/dl821

Prevention + Pre-test Detection is the most effective and efficient



- Prevention data based on state of the art prevention experiences (IBM RTP), Others (Space Shuttle IBM SJ 1-95) 95%+ (99.99% in Fixes)
- Cumulative Inspection detection data based on state of the art Inspection (in an environment where prevention is also being used, IBM MN, Sema UK, IBM UK)

These slides are at
http://www.gilb.com/dl821

IBM MN & NC DP Experience

- **2162 DPP Actions implemented**
  - between Dec. 91 and May 1993 (30 months)<-Kan

- RTP about 182 per year for 200 people.<-Mays 1995
  - 1822 suggested ten years (85-94)
  - 175 test related
- RTP 227 person org<- Mays slides
  - 130 actions (@ 0.5 work-years
  - 34 causal analysis meetings @ 0.2 work-years
  - 19 action team meetings @ 0.1work-years
  - Kickoff meeting @ 0.1 work-years
  - TOTAL costs 1% of org. resources

- ROI DPP 10:1 to 13:1, internal 2:1 to 3:1
- Defect Rates at all stages 50% lower with DPP

These slides are at
http://www.gilb.com/dl821

Conclusion

We need to take these **scale-free engineering ideas** seriously

if we are to get **better control over large-scale software and systems engineering projects**.

- The ideas have serious practical international experience,
- and can be tried out one-by-one.
- They can be added to any other practices, that are, or will be successful for you,
- They are free ideas.

‘This stuff works!”
(Erik Simmons, Intel, [22, 25])
Experience 1999 to 2016 for 20,000 engineers

Just do it!

References

Very Detailed references are in the presenter notes in this slide

The Main Reference is the **Value Planning** Book

[20: VP] **"Value Planning: Practical Tools for Clearer Management Communication"**

leanpub.com/ValuePlanning

Free sample (Part 0 to 5), and ridiculously cheap download with bundled tools and updates.

- "Value Planning: Practical Tools for Clearer Management Communication"
- http://**gilb.com/dl853**
  - (Free copy for you alone)

- Email me.

Tom@Gilb.com

Russian & German Translations: Leanpub

Go back

End slide