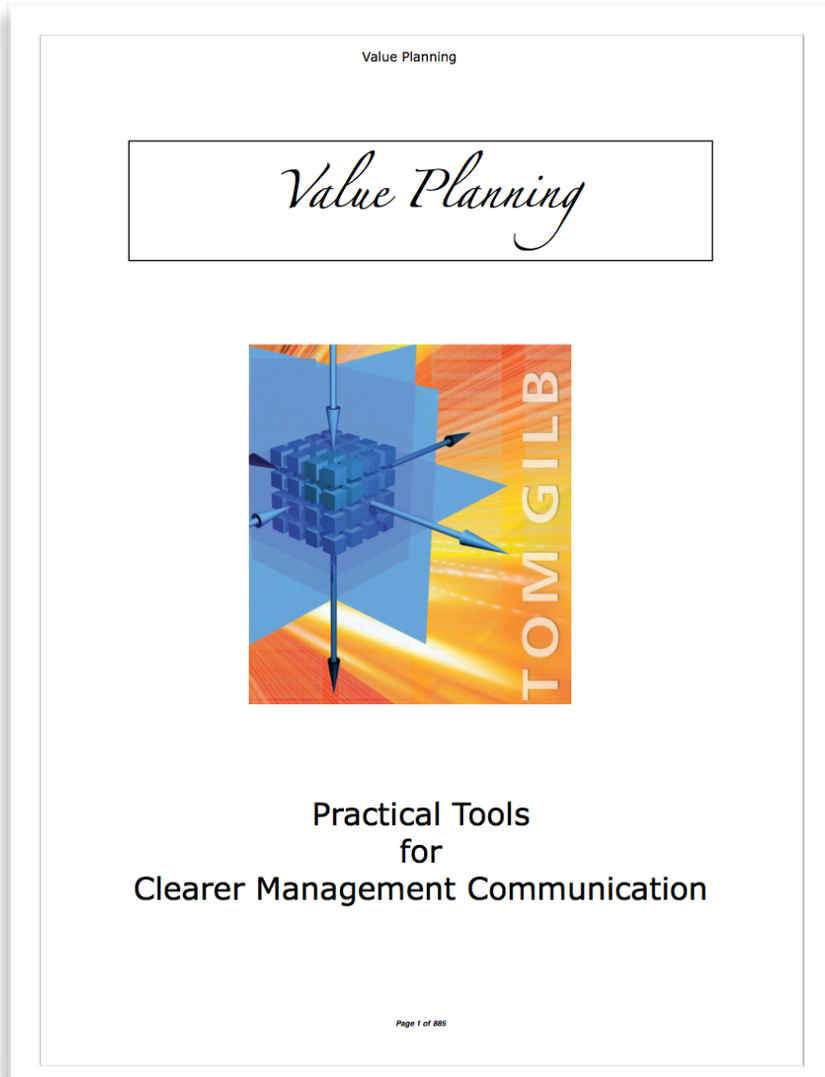
Can we ‘measure’ (Quantify, Test) agility? Measurement Tools for practitioners.

Tom Gilb
at Katowice
Masterclass, 2018
https://nowy.me/gilb/

For the workshop session

THURSDAY 23 May 2019, 10:00 to 13:00
Location: Polna 11, Warsaw

full agenda of the event yet
https://sektor3-0.pl/en/festival/agenda/



10:00 TO 10:50
Measuring the Agility Process:
how to manage your organizational process.

- **Measuring the Agility Process**
  - **<u>Bad idea</u> (Scrum)**
    - **Focussing on 'Velocity'**
    - **without measuring stakeholder value delivery and costs**

- **<u>Better Idea</u>**
  - **Measure delivery of prioritized stakeholder values**
  - **Measure costs and resources**
    - **not just velocity**
    - **but Capital cost**
    - **Technical Debt**
    - **Delivery of value within deadlines**

Measuring the Agility Process: Some Unusual but Useful Examples of Measures

- **Development Process Measures**
  - **% Planned Values Delivered by Deadline**
  - **Values to Costs Ratio (Efficiency)**
  - **Value Payments**
- **Operational Process Measures**
  - **Technical Debt**
    - **Maintainability**
  - **Help Requests**
  - **Problem Reports**
  - **Scalability**
- **Organizational Maturity Measures**
  - **Organizational Learning (DPP. [Raytheon Case below].)**

https://resources.collab.net/blogs/agile-metrics-measuring-what-actually-matters

Bad idea (Scrum)

- **Focussing on ‘Velocity’**
  - **without measuring stakeholder value delivery and costs**
  - **assuming that code = value**
- **Why?**
  - **total lack of control of real value delivered**
  - **guaranteed 19% total project failure rate and worse (J. Sutherland)**

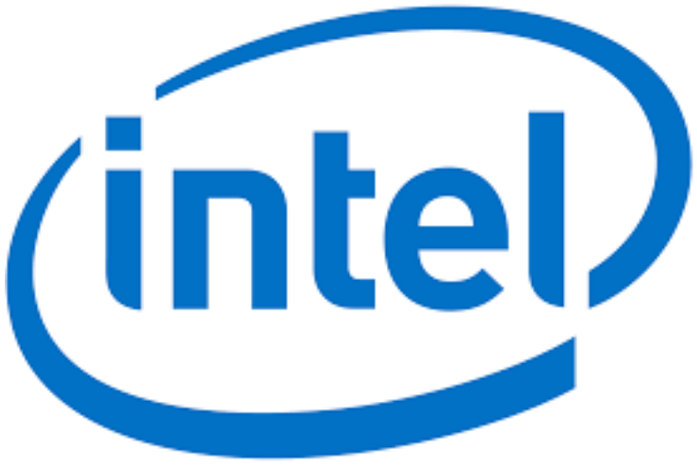
Better Ideas

- **Measure delivery of prioritized stakeholder values**
  - **Sprints = 2% of budget, Value Delivery steps**
  - **Attempt to maximize priority stakeholder values**
  - **Attempt to minimize value delivery costs**
  - **Look at the System (stakeholders, values, people, hardware, dataware, maybe even some ‘code’.**
  - **Look at top 10 multiple critical values, at same time**
  - **Contract for payment, reward, keep going depending on successful value delivery**

Measure costs and resources

- **not just velocity**
- **but Capital cost**
- **Technical Debt**
- **Delivery of value within multiple prioritized deadlines**

Intel Measures of Gilb Methods 2013



The Impact of Requirements on Software Quality across Three Product Generations

John Terzakis

Intel Corporation, USA
john.terzakis@intel.com

***Abstract*—In a previous case study, we presented data demonstrating the impact that a well-written and well-reviewed set of requirements had on software defects and other quality indicators between two generations of an Intel product. The first generation was coded from an unorganized collection of requirements that were reviewed infrequently and informally. In contrast, the second was developed based on a set of requirements stored in a Requirements Management database and formally reviewed at each revision. Quality indicators for the second software product all improved dramatically even with the increased complexity of the newer product. This paper will recap that study and then present data from a subsequent Intel case study revealing that quality enhancements continued on the third generation of the product. The third generation software was designed and coded using the final set of requirements from the second version as a starting point. Key product differentiators included changes to operate with a new Intel processor, the introduction of new hardware platforms and the addition of approximately fifty new features. Software development methodologies were nearly identical, with only the change to a continuous build process for source code check-in added. Despite the enhanced functionality and complexity in the third generation software, requirements defects, software defects, software sightings, feature commit vs. delivery (feature variance),**

ays from project
he second to the

requirements
quality, multi-

ort paper [1] that

II. Product Backgrounds

The requirements for Gen 1 that existed were scattered across a variety of documents, spreadsheets, emails and web sites and lacked a consistent syntax. They were under lax revision and change control, which made determining the most current set of requirements challenging. There was no overall requirements specification; hence reviews were sporadic and unstructured. Many of the legacy features were not documented. As a result, testing had many gaps due to missing and incorrect information.

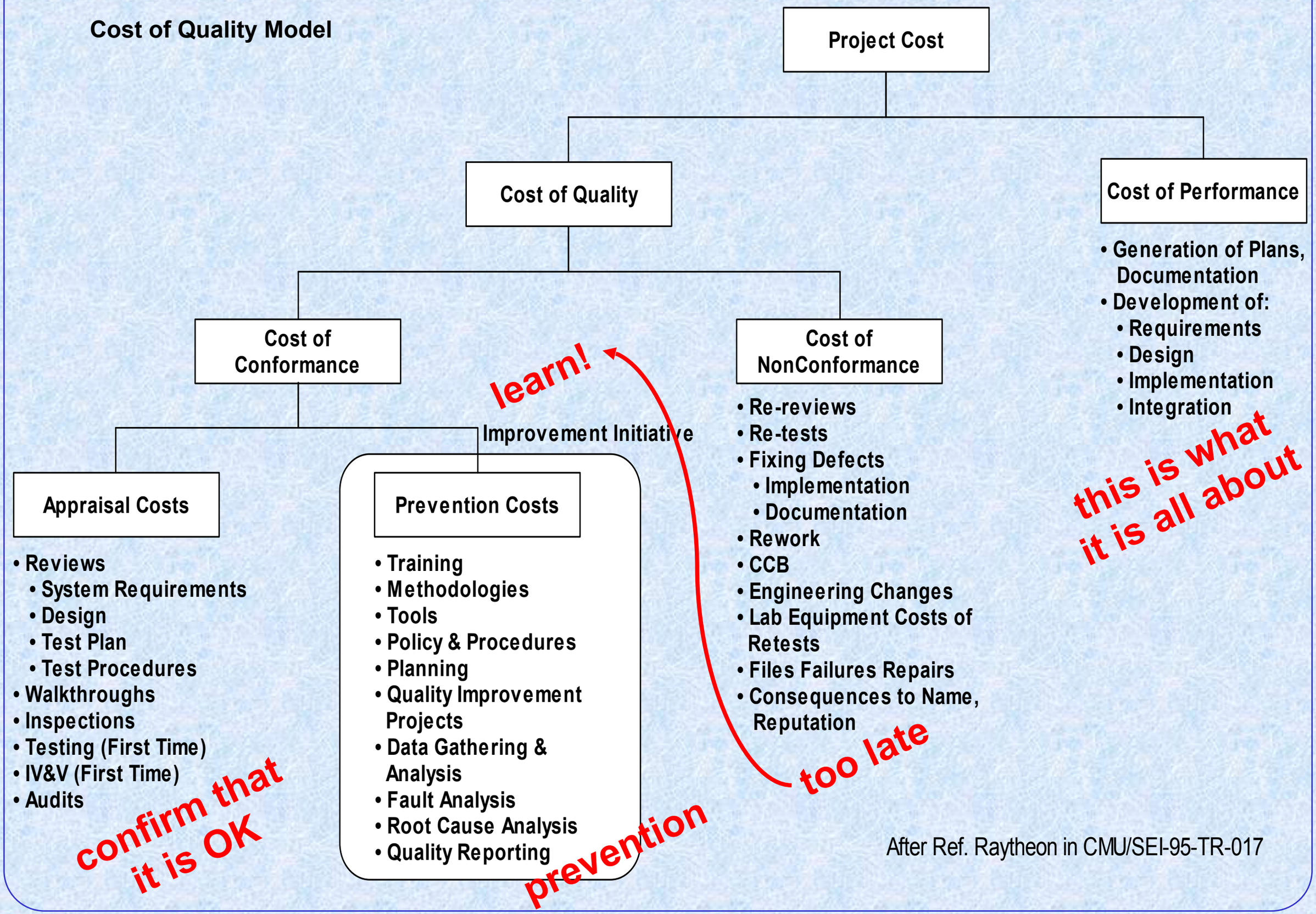
The Gen 1 product was targeted to run on both desktop and laptop platforms running on an Intel processor (CPU). Code was developed across multiple sites in the United States and other countries. Integration of the code bases and testing occurred in the U.S. The Software Development Lifecycle (SDLC) was approximately two years.

After analyzing the software defect data from the Gen 1 release, the Gen 2 team identified requirements as a key improvement area. A requirements Subject Matter Expert (SME) was assigned to assist the team in the elicitation, analysis, writing, review and management of the requirements for the second generation product. The SME developed a plan to address three critical requirements areas: a central repository, training, and reviews. A commercial Requirements Management Tool (RMT) was used to store all product requirements in a database. The data model for the requirements was based on the Planguage keywords created by Tom Gilb [2]. The RMT was configured to generate a formatted Product Requirements Document (PRD) under revision control. Architecture specifications, design documents and test cases were developed from this PRD. The SME provided training on best practices for writing requirements, including a standardized syntax, attributes of well written requirements and Planguage to the primary authors (who were

TABLE I: GEN 2 REQUIREMENTS DEFECT DENSITY

PRD Revision	# 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%***				

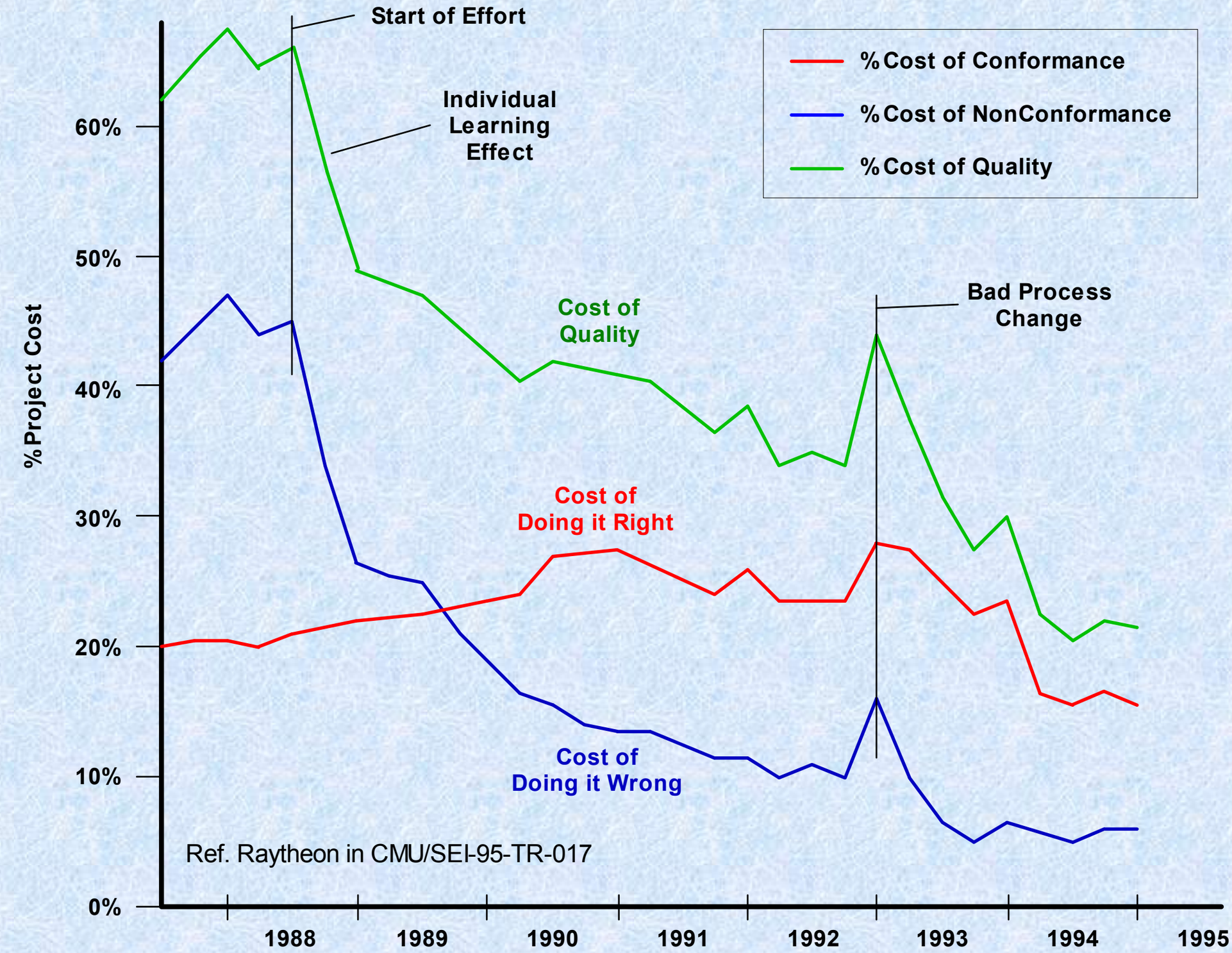
Cost of Quality Model



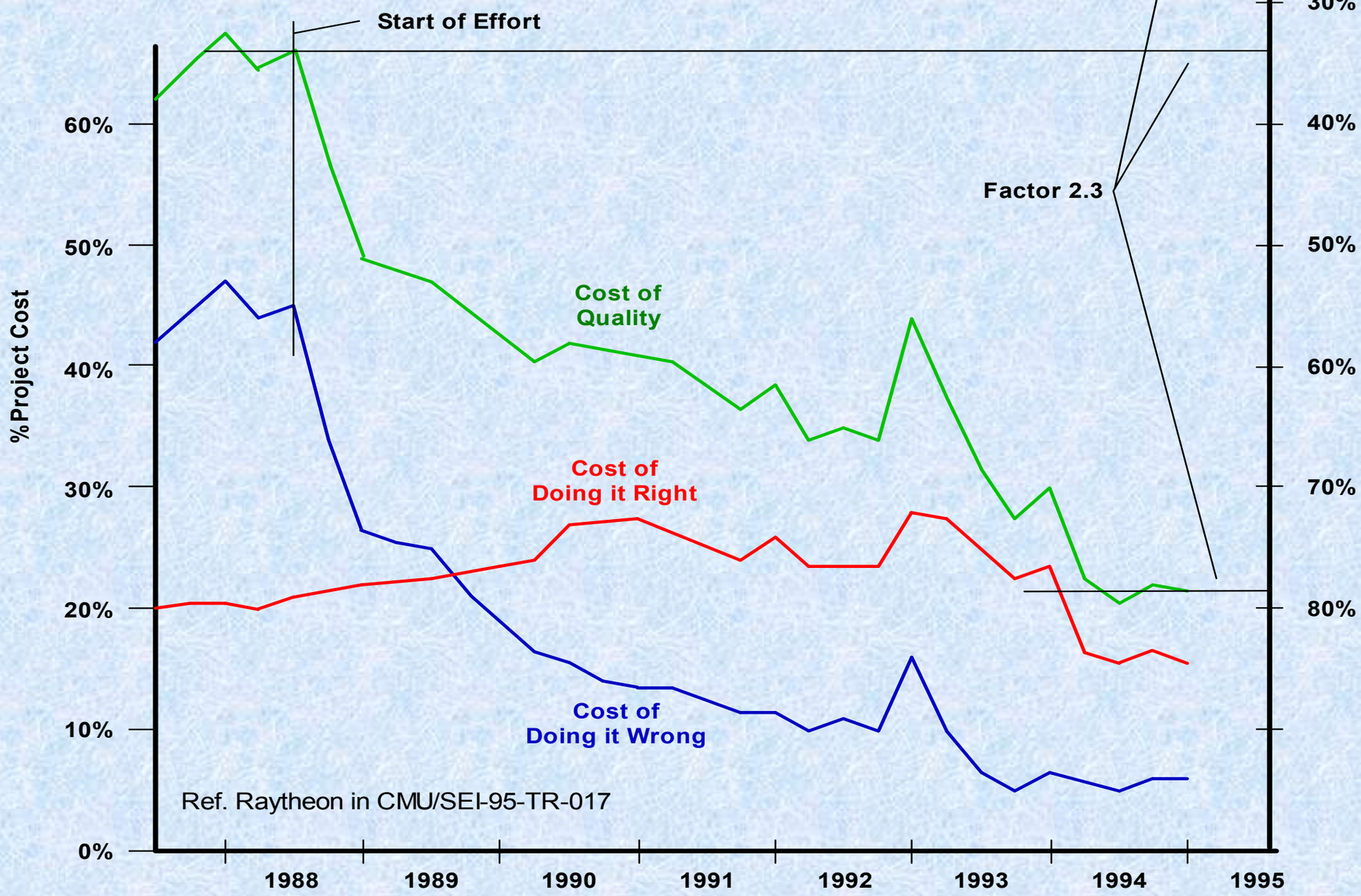
After Ref. Raytheon in CMU/SEI-95-TR-017

Cost of Quality

Raytheon used the Defect Prevention Process
To Incrementally change and improve their organization



Productivity gains



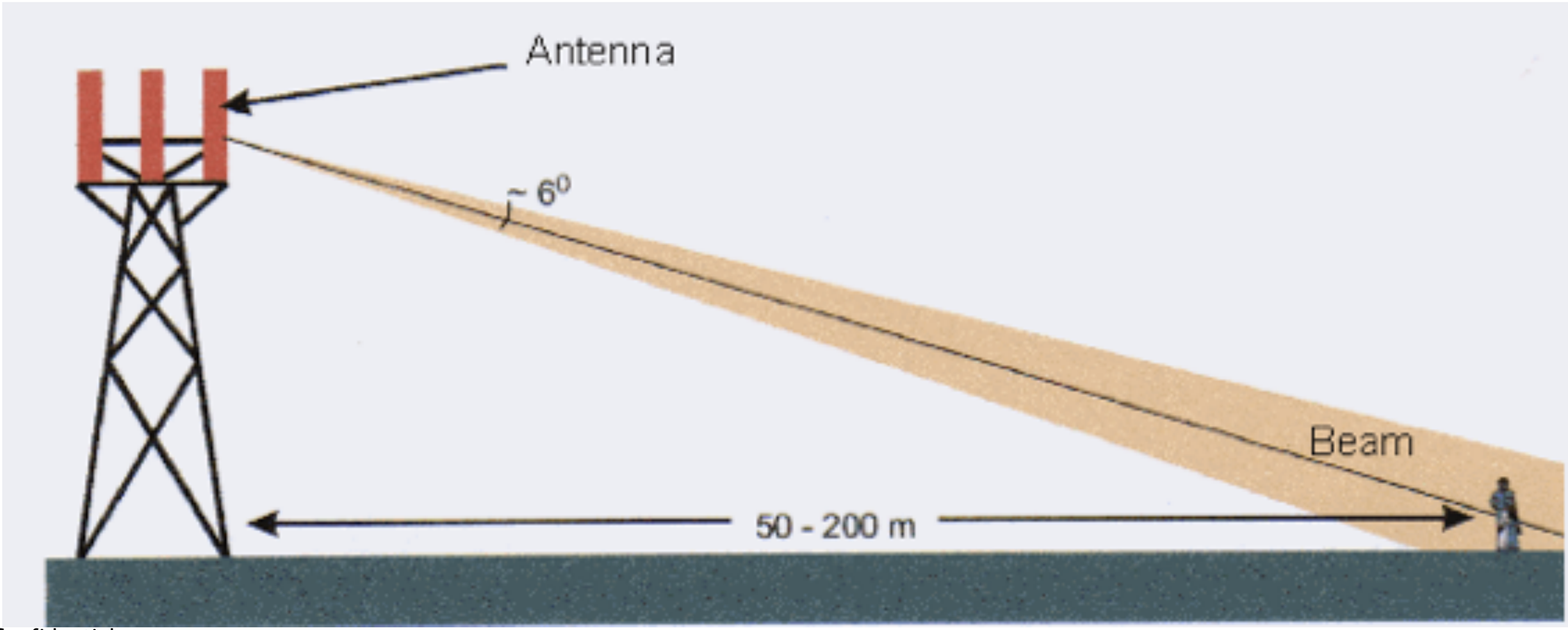
Software Engineering Productivity Study

ERICSSON

An example of setting objectives for process improvement

with 70% software labor development content in products

.



Main beam from a macrocell base station antenna

The problem

- Great Market Growth Opportunities
- Too Few Software Engineers

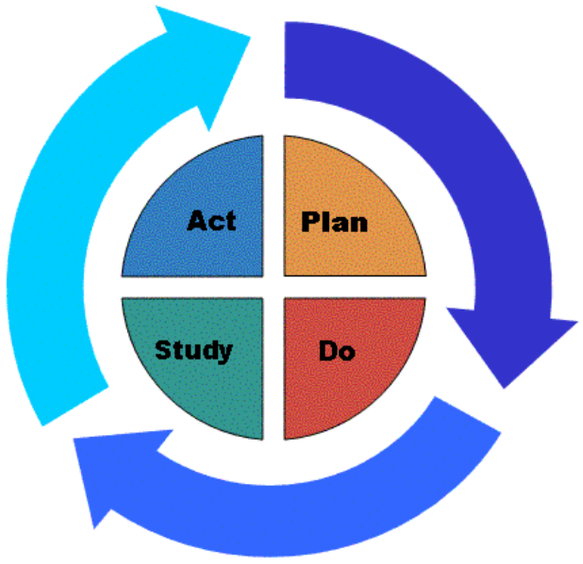
- Solution:
  - Increase productivity of existing engineers

The One Page Top Management Summary

(after 2 weeks planning)

The Dominant Goal

Improve Software Productivity in R PROJECT by 2X by year 20XX



Dominant (META) Strategies

Continual Improvement (PDSA Cycles)

.DPP: Defect Prevention Process

.EVO: Evolutionary Project Management

Long Term Goal [Next 3 Years+]

DPP/EVO, Master them and Spread them on priority basis.

Short Term Goal [Next Weeks]

DPP [RS?]

EVO [Package C ?]

Decision: {Go, Fund, Support}

The Ericsson Quality Policy:

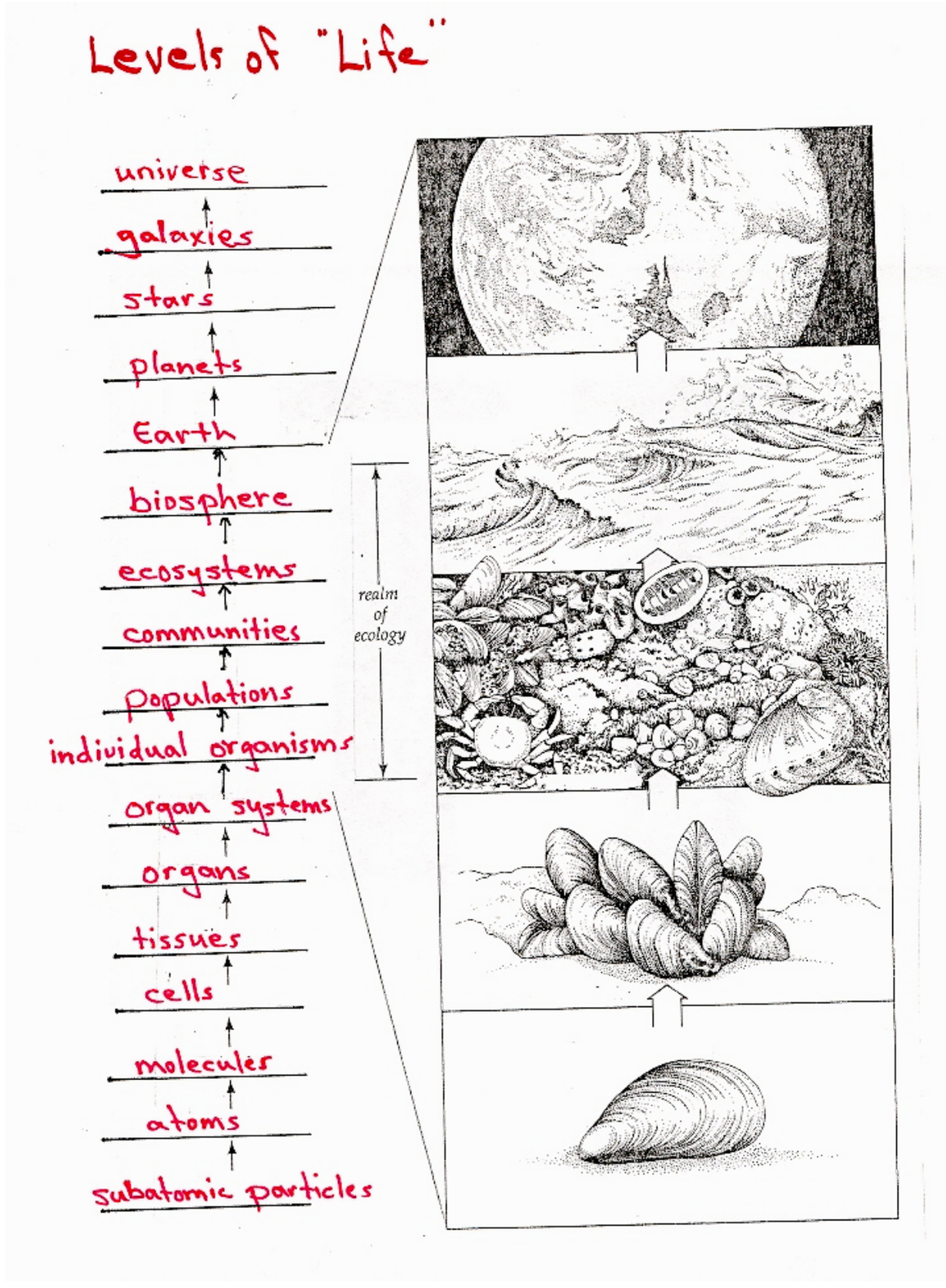
ERICSSON

"every company shall *define*
performance indicators (which) ..
–reflect **customer satisfaction**,
– internal **efficiency**
–and business **results**.

- The performance indicators are used in **controlling** the operation."
- Quality Policy [4.1.3]

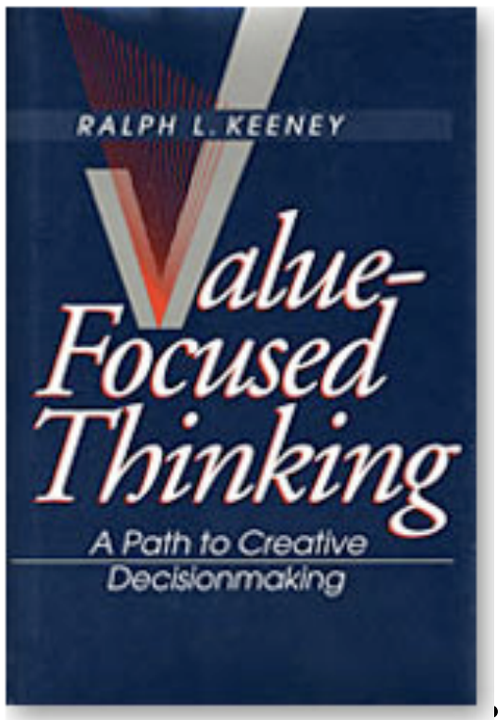
Levels of Objectives.

- **Fundamental Objectives**
- **Strategic Objectives**
- **Means Objectives:**
-
- **Organizational Activity Areas.**
  - Pre-study.
  - Feasibility Study.
  - Execution.
  - Conclusion.
- **Generic Constraints**
  - Political Practical
  - Design Strategy Formulation Constraints
  - Quality of Organization Constraints
  - Cost/Time/Resource Constraints



Keeney's: Levels of objectives.

- **1. Fundamental Objectives**
  - **(above us)**
- **2. Generic Constraints**



(our given framework)

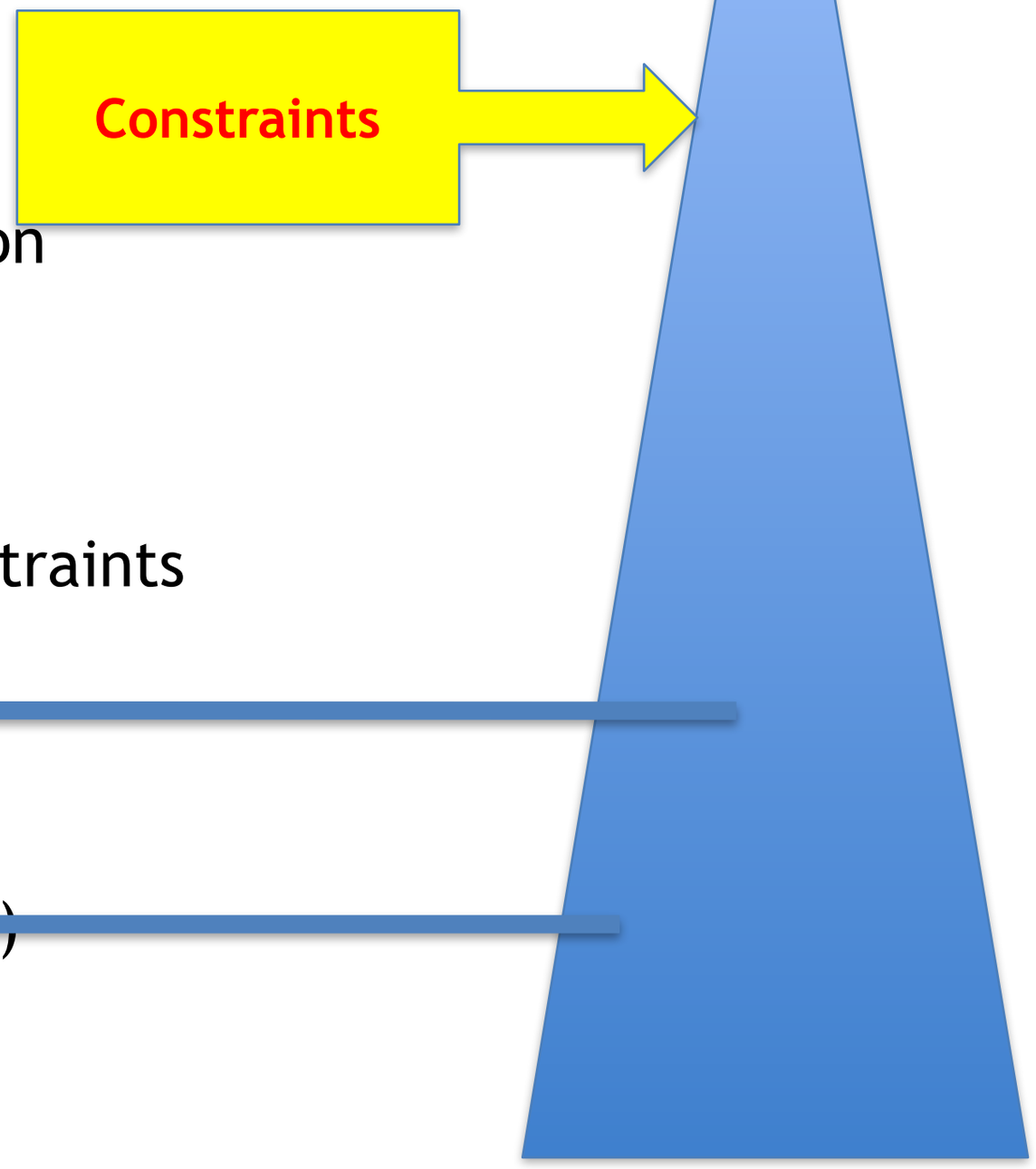
Political Practical

Design Strategy Formulation Constraints

Quality of Organization Constraints

Cost/Time/Resource Constraints

- **3. Strategic Objectives**
  - **(objectives at our level)**
- **4. Means Objectives:**
  - (*supporting* our objectives)



The Strategic Objectives (CTO level)

- Support
  - the **Fundamental** Objectives (Profit, survival)
  - **Software Productivity:**
    - **Lines of Code Generation Ability**
  - **Lead-Time:**
  - **Predictability.**
  - **TTMP: Predictability of Time To Market:**
  - **Product Attributes:**
  - **Customer Satisfaction:**
  - **Profitability:**

‘Means’ Objectives:

- Support the **Strategic** Objectives
  - *Complaints:*
  - *Feature Production:*
  - *Rework Costs:*
  - *Installation Ability:*
  - *Service Costs:*
  - *Training Costs:*
  - *Specification Defectiveness:*
  - *Specification Quality:*
  - *Improvement ROI:*

"Let no man turn aside,
ever so slightly,
from the broad path of honour,
on the plausible pretence
that he is justified by the goodness
of his end.
All good ends can be worked out
by good means."
Charles Dickens

Strategies: (total brainstormed list)
'Ends for delivering Strategic Objectives'

- –Evo [Product development]:
- –DPP [Product Development Process]: Defect Prevention Process.
- –Inspection?
- –Motivation.Stress-Management-AOL
- –Motivation.Carrot
- –DBS
- –Automated Code Generation
- –Requirement -Tracability
- –Competence Management
- –Delete-Unnecessary -Documents
- –Manager Reward:?
- –Team Ownership:?
- –Manager Ownership:?

- •Training:?
- •Clear Common Objectives:?
- •Application Engineering area:
- •Brainstormed List (not evaluated or prioritized yet)?
- •Requirements Engineering:
- •Brainstormed Suggestions?
- •Engineering Planning:
- •Process Best Practices:
- •Brainstormed Suggestions?
- •Push Button Deployment:
- •Architecture Best Practices:
- •Stabilization:
- •World-wide Co-operation?

Principles for Prioritizing Strategies

- **They are well-defined**
  - **Not vague**
- **The have some relevant predictable numeric experience**
  - **On main effects**
  - **Side effects**
  - **Costs**
  - **Risks - Uncertainty**
- **Not huge spread of experience**

"Software Productivity" =

Lines of Code Generation Ability

—*"Software Engineering net production in relation to corresponding costs."*

—*Ambition: Net lines of code successfully produced per total working hours needed to produce them. A measure of the*

— *efficiency ('effective production/cost of production') of the organization in using its software staff.*

•Scale: [Defined Volume, kNCSS or kPlex] pe

•Software Development: Defined:

•Productivity calculations include Work-Hours

• Meter : <PQT Database and EPOS, CPAC>

—*Comment: we know that real software proc* *his measure as it is available in our current cul*

—P1: Past [1997, ERA/AR] < to be calculate

• Past-R PROJECT: Past [1997, R PROJECT] rs >

• Past-EEI: Past [1997, Ireland, Plex] ___??__ kPLEX / W

•<add more like LuleÂ>

Scale: [Defined Volume, kNCSS or kPlex] per Software Development Work-Hour.

•Fail [end 1998, R PROJECT, Same Reliability] 1.5 x Past-R PROJECT <- R PROJECT AS 3 c " by 50%".

—*"50% better useful code productivity in 1.5 years overall"*

•Same Reliability: State: The Software Fault Density is not worse than with comparable productivity. Use official The Company Software Fault Density measures <- 1997 R PROJECT Balanced Scorecard (PA3).

•Goal [Year=2000, R PROJECT, Same Reliability] 2 x Past-R PROJECT,

— [Year=2005, RPL, Same Reliability] 10?? x Past-R PROJECT

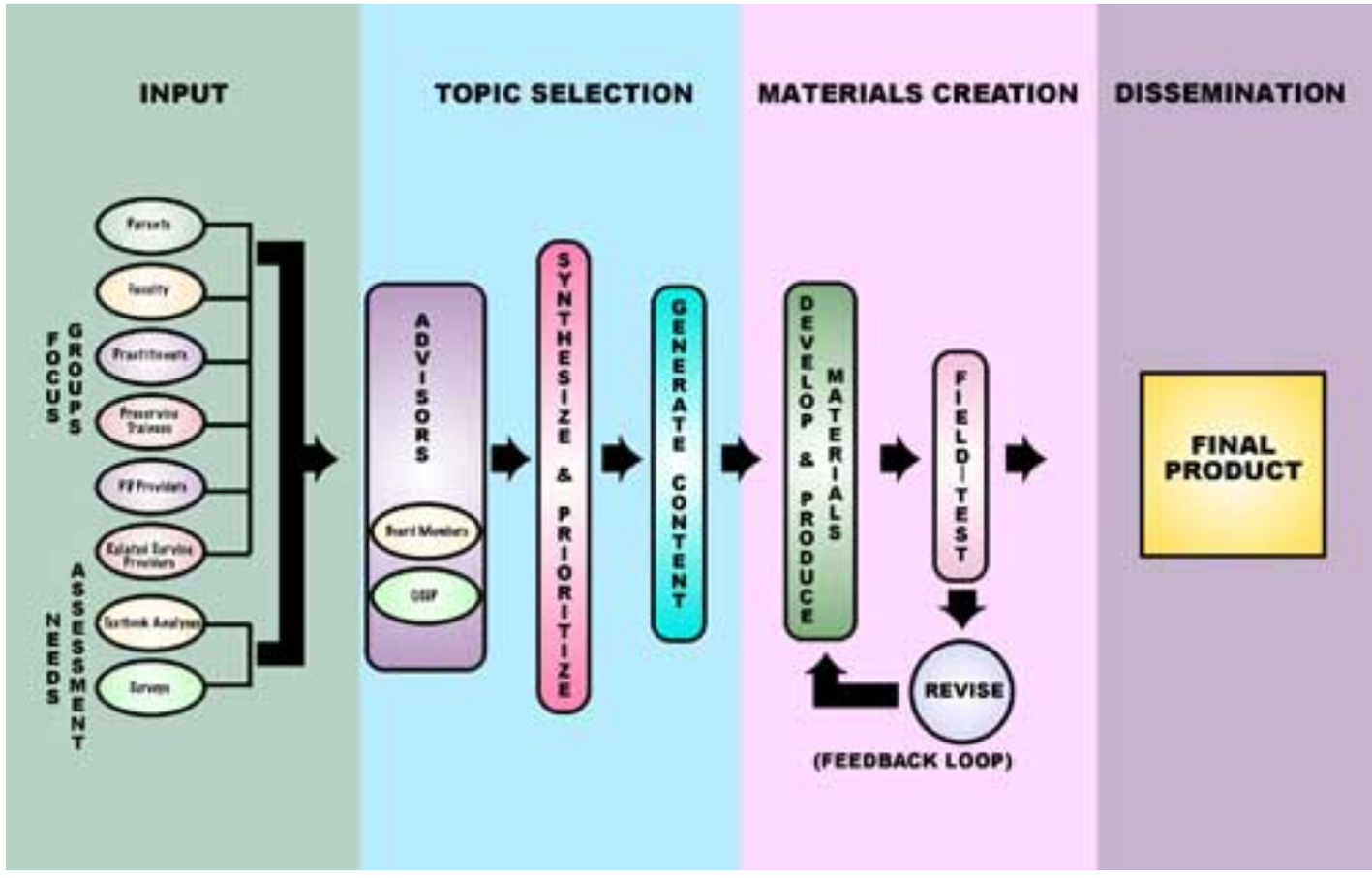
•Wish [Long term, vs. D pack.] 10 x Past-R PROJECT "times higher productivity" <- R PROJECT 96 1.1 c

•Wish [undefined time frame] 1.5 x Past-R PROJECT <- R PROJECT AS 3 c " by 50%"

—*Comment: May 13 1997 1600, We have worked a lot on the Software Productivity objectives (all day) and are happy that it is in pretty good shape. But we recognize that it needs more exposure to other people.*

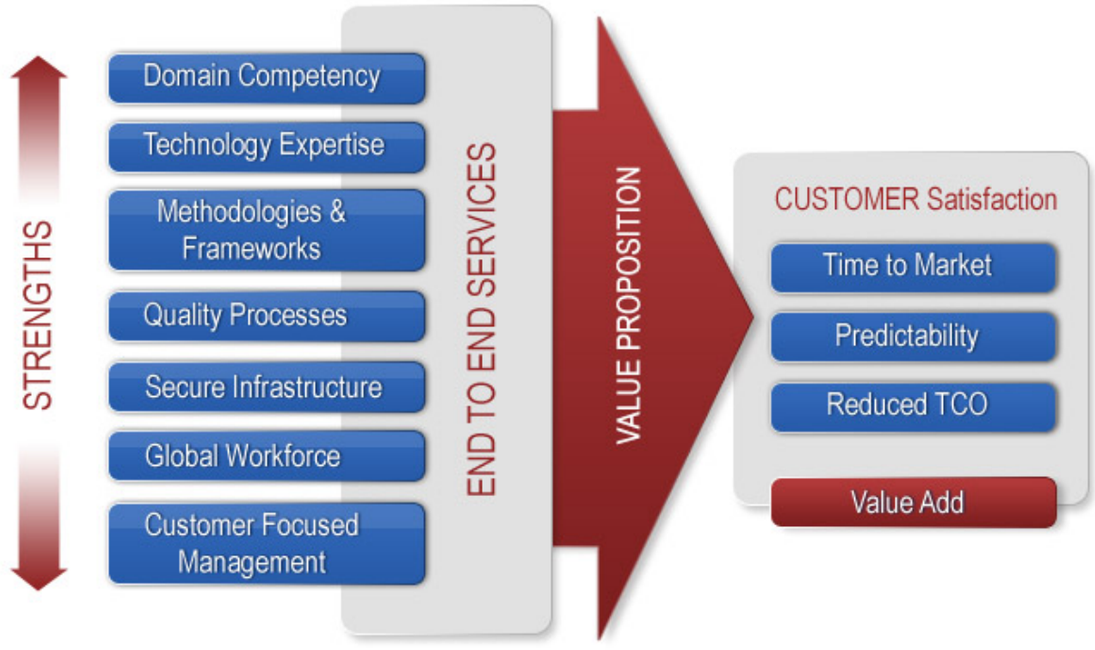
Lead-Time:

- <u>Lead-Time:</u>
  - *"Months for major Packages"*
- ***Ambition**: decrease months duration between major Base Station package release.*
- **<u>Scale</u>: Months from TG0, to successful first use for**
  - **major work station package.**
  - *Note: let us make a better definition. TG*
- <u>Past</u> [C Package, 1996?] 20? Months?? <-guess tg
- <u>Goal</u> [D-package] 18 months <- guess tg
- <u>Goal</u> [E-package and later] 10.8 Months <- R PROJECT 96 1.1 a "40% > D"
- <u>Goal</u> [Generally] ??? <- R PROJECT AS 3a
  - *"10% Lead-Time reduction compared to any benchmark".*



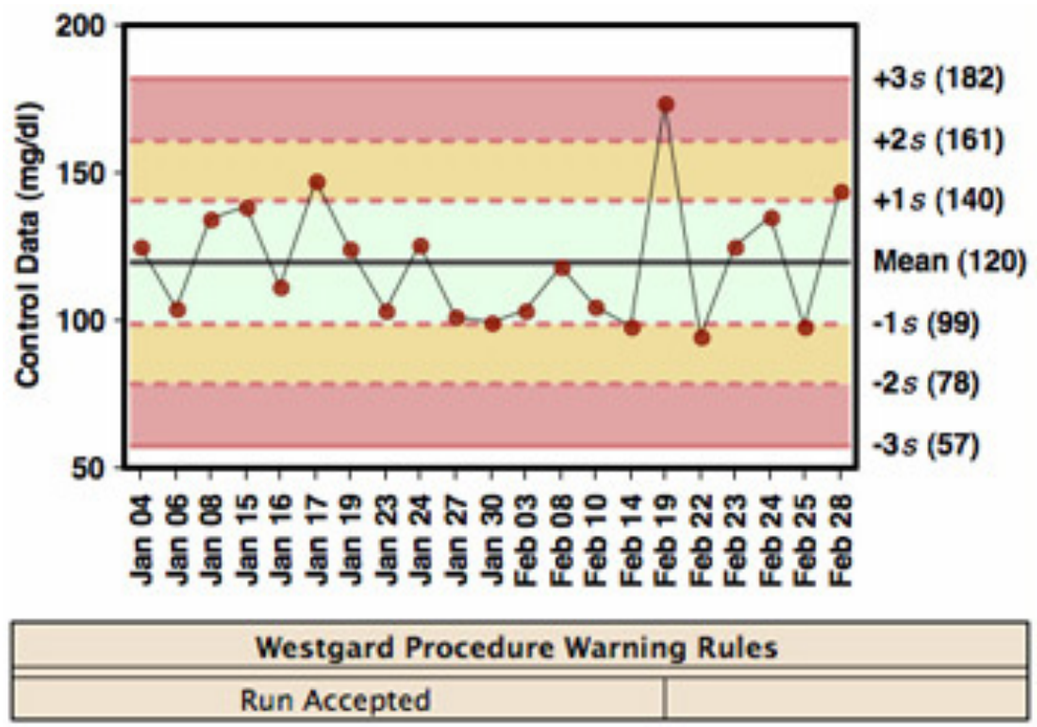
Predictability of Time To Market:

- **TTMP: Predictability of Time To Market:**
  - *Ambition: From Ideas created to customers can use it. Our ability to meet agreed specified customer and self-determined targets.*
  - **Scale: % overrun of actual Project Time compared to planned Project Time**
    - **Project Time: Defined:** time from the date of Toll-Gate 0 passed, or other Defined Start Event, to, the Planned- or Actually- delivered Date of All [Specified Requirements], and any set of agreed requirements.
    - **Specified Requirements: Defined:** written approved Quality requirements for products with respect to Planned levels and qualifiers [when, where, conditions]. And, other requirements such as function, constraints and costs.
  - **Meter: Productivity Project or Process Owner will collect data from all projects, or make estimates and put them in the Productivity Database for reporting this number.**
  - **Past [1994, A-package] < 50% to 100%> <- Palli K. guess. [1994, B-package] 80% ?? <- Urban Fagerstedt and Palli K. guess**
  - **Record [IBM Federal Systems Division, 1976-80] 0% <- RDM 9.0 quoting Harlan Mills in IBM SJ 4-80**
  - ***"all projects on time and under budget"***
  - **[Raytheon Defense Electronics, 1992-5] 0% <- RDE SEI Report 1995 Predictability.**
  - **Fail [All future projects, from 1999] 5% or less <- discussion level TG**
  - **Goal [All future projects, from 1999] 0% or less <- discussion level TG**



Product Attributes:

- <u>Product Attributes</u>:
  - *"Keeping Product Promises."*
  - *Ambition: Ability to meet or beat agreed targets, both cost, time and quality. (except TTMP itself, see above)*
- ***Scale*: % +/- deviation from [defined agreed attributes with projects].**
- ***Past*** **[1990 to 1997, OUR DIVISION] at least <u>100%</u> ???**
  - **<- Guess. Not all clearly defined and differences not**
    - **tracked. TSG**
- *Goal* [Year=2000, R PROJECT] **near <u>0%</u>** negative deviation <- TsG for discussion.



Customer Satisfaction

Customer Satisfaction:
"Customer Opinion of Us"

Scale: average survey result on scale of 1 to 6 (best)

Meter: The Company Customer Satisfaction Survey

Past [1997] **4**

Goal [1998-9?] **5** <- R PROJECT 96 1.1 b

Profitability

- **Profitability:**
  - ***"Return on Investment."***
  - ***<u>Ambition</u>: Degree of <u>saleable</u> product ready for installation.***
  - **<u>Scale</u>: Money Value of Gross Income derived by**
    - **[All R PROJECT Production OR**
    - **defined products] for**
    - **[Product Lifetime OR**
    - **a defined time period]**
  - **Goal: <we did not complete this>**

‘Means Objectives’ Samples Same *definition* process as higher level objectives

Means Objectives

- *"support Strategic Objectives"*

- Summary:
  - '**Means Objectives**' are
    - not our major Strategic Objectives (above),
    - but each one represents areas which if improved
      - will normally help us achieve our Strategic Objectives.
  - Means Objectives have a lower priority than Strategic Objectives.
  - They must never be 'worked towards'
    - to the point where they reduce our ability to meet Strategic Objectives.

Complaints

Complaints:

"Customer complaint rate to us"

Ambition:

Means Goal: for Customer Satisfaction (Strategic).

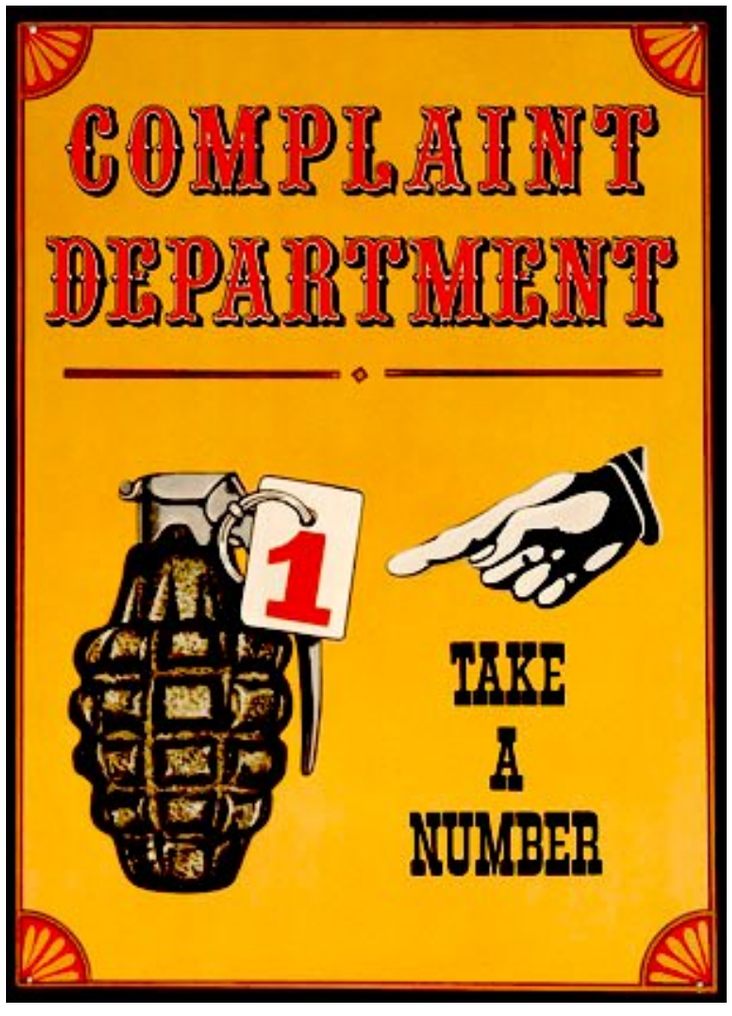
Scale: number of complaints per customer in [defined time into <operation>]

Past [Syracuse Project , 1997] ?? <bad> <- ML

Goal [Long term, software component, in first 6 months in Operation] **zero complaints** <- R PROJECT 96 1.1 b

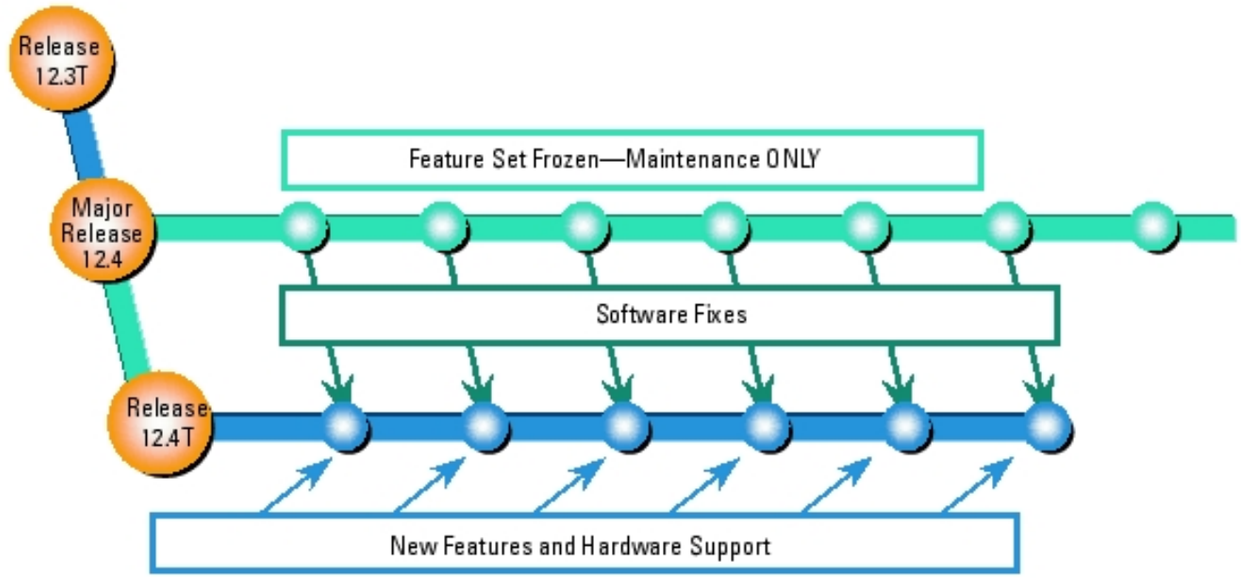
"zero complaints on software features"

Impacts: <one or more strategic objectives>



Feature Production:

- **Feature Production:**
  - *"ability to deliver new features to customers"*
  - ***Ambition**: reverse our decreasing ability to deliver new features <- R PROJECT AS 1.1*
  - **Scale**: Number of new prioritized <Features> delivered successfully to customer per year per software development engineer.
  - Too Little: **Past** [1997] ?? "estimate needed, maybe even definition of feature"
  - **Goal** [1998-onwards] **Too Little + 30% annually**?? <-For discussion purposes TsG.
  - "we need to drastically change our ability to effectively develop SW" <- R PROJECT AS 1.1



Note: Technology releases are those Cisco IOS Software releases that introduce new features, functionality, and hardware support.

Improvement ROI:

Improvement ROI:

"Engineering Process Improvement Profitability"

Ambition: Order of magnitude return on investment in process improvement.

Scale:

The average [annual OR defined time term] Return on Investment in Continuous Improvement as a ratio of [Engineering Hours OR Money]

Note: The point of having this objective is to remind us to think in terms of real results for our process improvement effort, and to remind us to prioritize efforts which give high ROI. Finally, to compare our results to others. <-TsG

Record

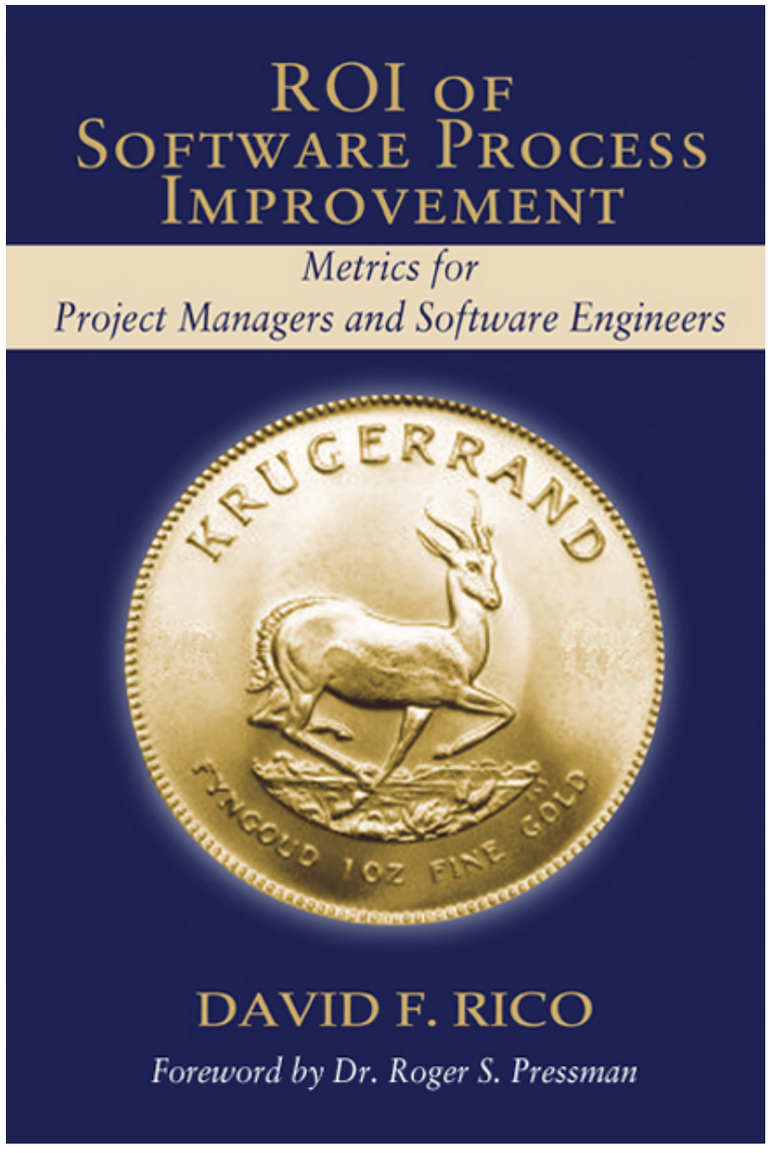
[Shell NL, Texas Instruments , Inspections] 30:1 <- Independently published papers TsG

Past

[IBM RTP, 1995, DPP Process] 13:1 <- Robert Mays, Wash DC test conference slides TsG

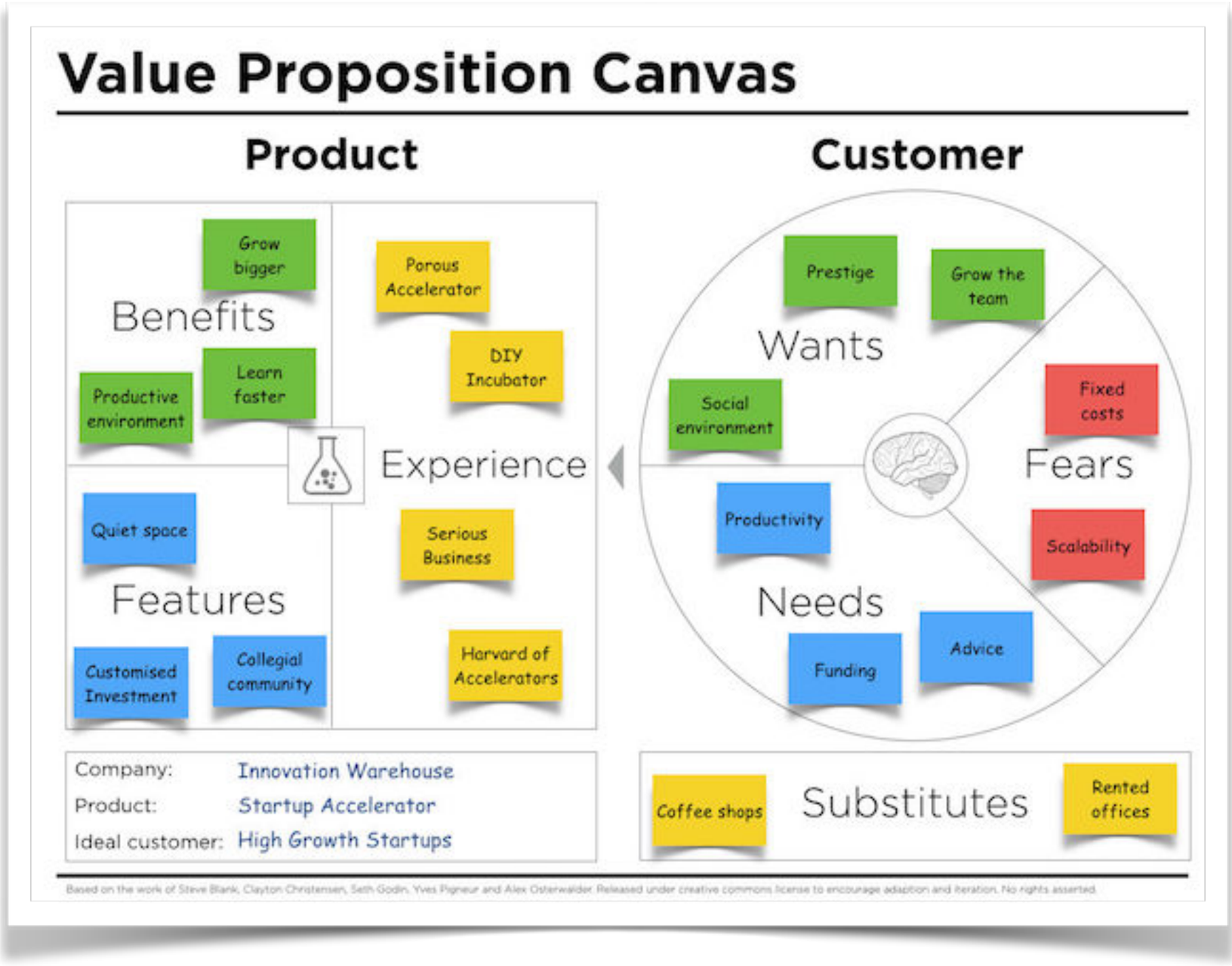
[Raytheon, 1993-5, Inspection & DPP] $7.70:1 <- RDE Report page 51 ($4.48 M$0.58M) Includes detail on how calculated. PK has copy.

[IBM STL, early 1990's] Average 1100% ROI (11:1) <- IBM Secrets pp32. PK has copy. NB Conservative estimate. See Note IBM ROI below.



11:00 TO 11:50
Measuring the Results of Agility in your Product:
How to deal with stakeholder values and qualities.

- **Stakeholders**
- **Values**
- **Resources**
- **Constraints**
- **Priorities (Dynamic)**
- **Decomposition**
- **Risks**



Stakeholders

Definition

A stakeholder is any person, group or object, which has some direct or indirect interest in a defined system.

Stakeholders can exercise control over both the immediate system operational characteristics, as well as over long-term system lifecycle considerations (such as portability, lifecycle costs, environmental considerations, and decommissioning of the system). [4]

Notice:

‘or object’.

This includes laws, regulations, plans, policies, customs, culture, standards. Inanimate. you cannot ask them or discuss with them. But you can analyze them, their priority, the degree of relevance. They can determine if your system is illegal, or acceptable. Determine success or failure.

Gilb's Stakeholder Principles.

1. Some stakeholders are more critical to your system than others.

2. Some stakeholder needs are more critical to your system than others.

3. Stakeholders are undisciplined: they may not know all their needs, or know them precisely, or know their value. But they can be analyzed, coached, and helped to get the best possible deal.

4. Stakeholders may be inaccessible, unwilling, inanimate, oppositional, and worse: but we need to deal with them intelligently.

5. Stakeholders might well ask for the wrong thing, a 'means' rather than their real 'ends'. But they can be guided to understand that. Or their requests can be interpreted in their own real best interests.

6. Stakeholders do not want to wait years, get delays, invest shitloads of money, and then little or no value. They want as much 'value improvement' of their current situation, as they can get, as fast as they can get it. For as little cost as possible,

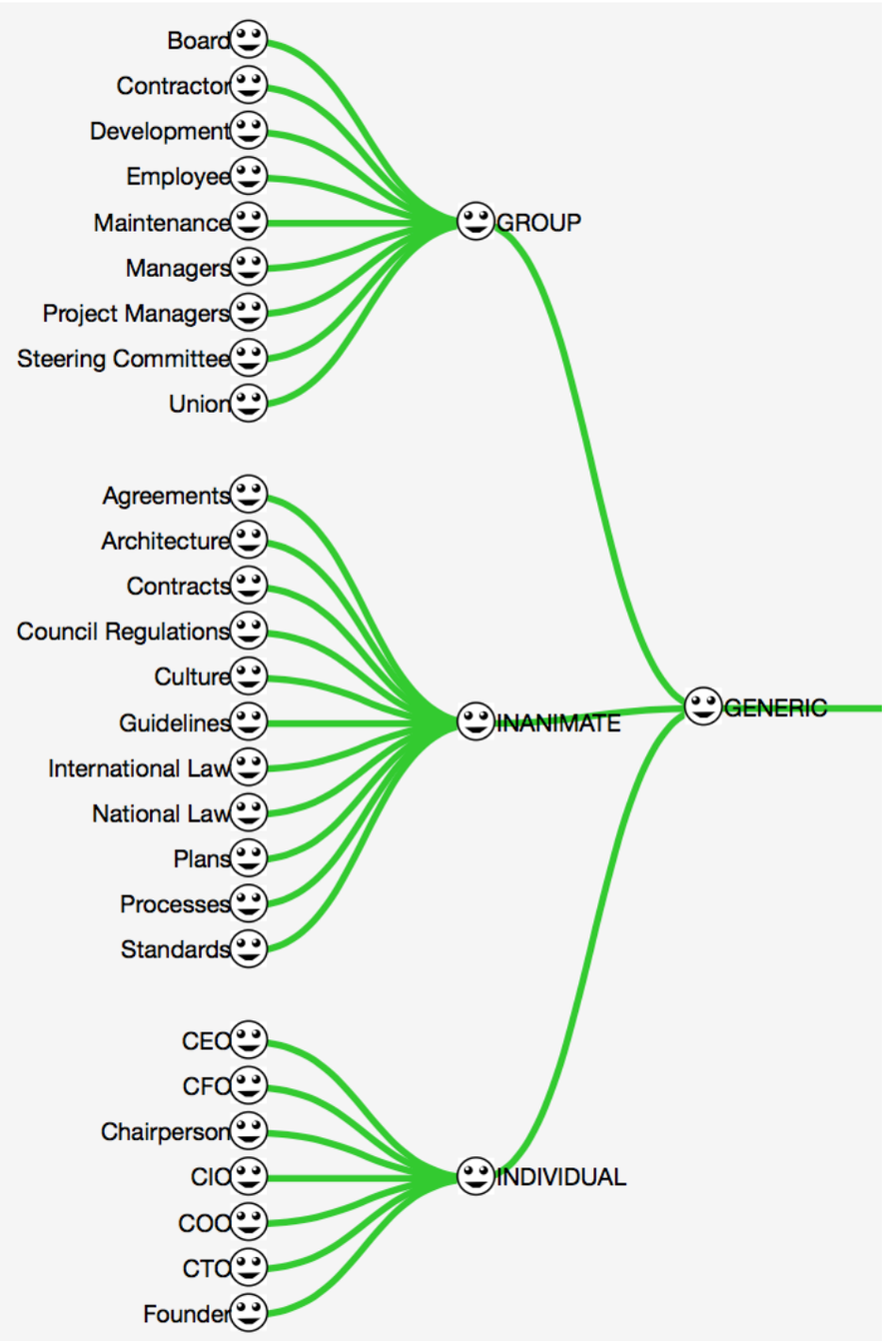
7. Stakeholders cannot have any realistic idea of what their needs and demands will cost to satisfy. So their adopted requirements need to be based on value for costs, not on value alone. Delivering small increments, based on high value-to-cost, is one smart way to deal with this.

8. If you think you have found 'all critical stakeholders', I think you should assume there is at least one more, and when you find that one, …. They will emerge, and they are not all there at the beginning.

9. If you think you have found all critical *needs* of a stakeholder, there will always be *at least one more* need 'hiding'.

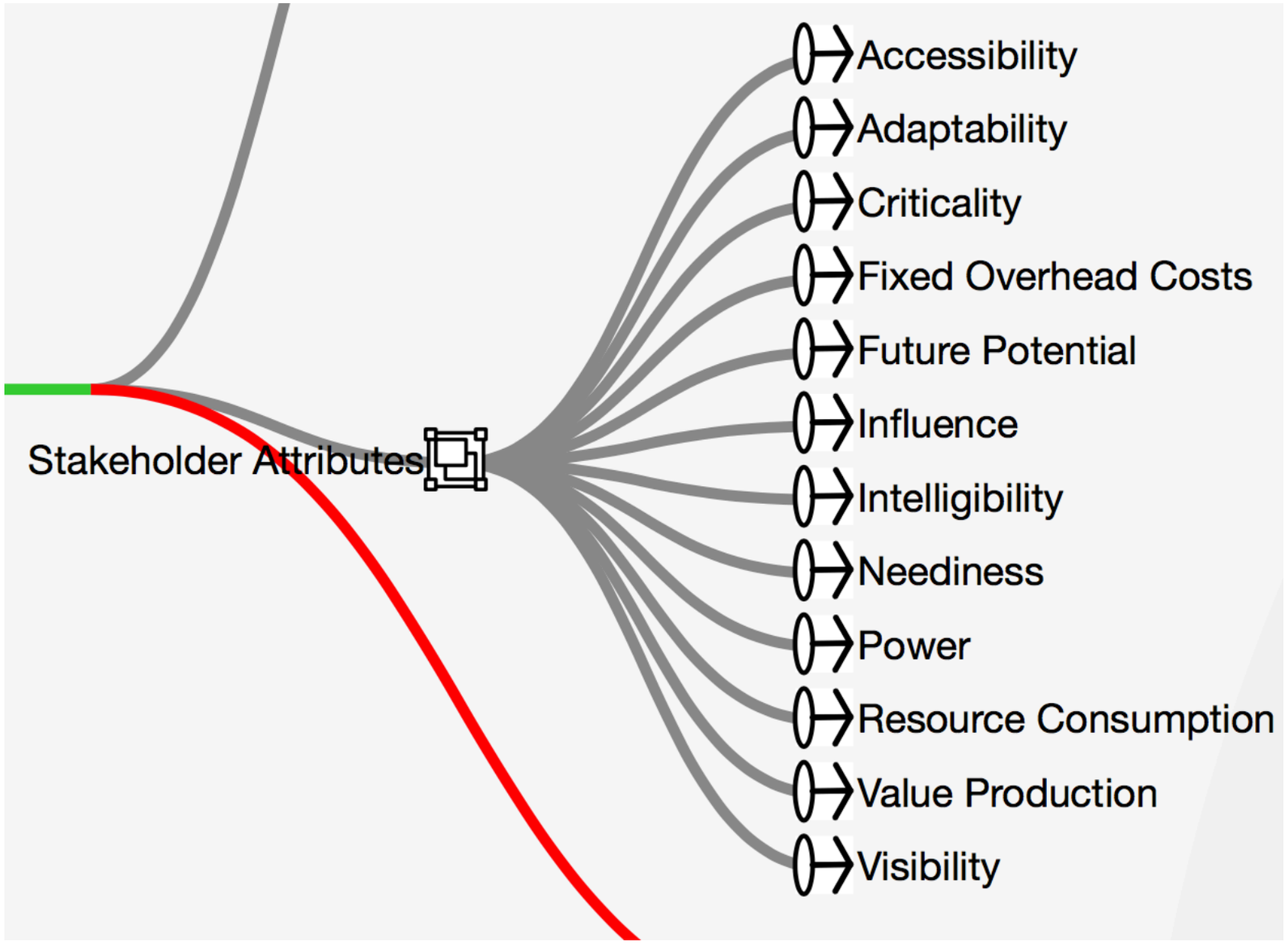
10. If you do not understand, and act on the principles above; you might blame your failure on 'system complexity', and the unexpected and wicked problems. But in reality, it is your own fault and responsibility; deal with it - up front and constantly.

•SOURCE, 2016 Paper
"Stakeholder Power:The Key to Project Failure or Success"
including 10 Stakeholder Principles
http://concepts.gilb.com/dl880 (COPY FEB 2017)
http://concepts.gilb.com/dl872 (FEB 2016)

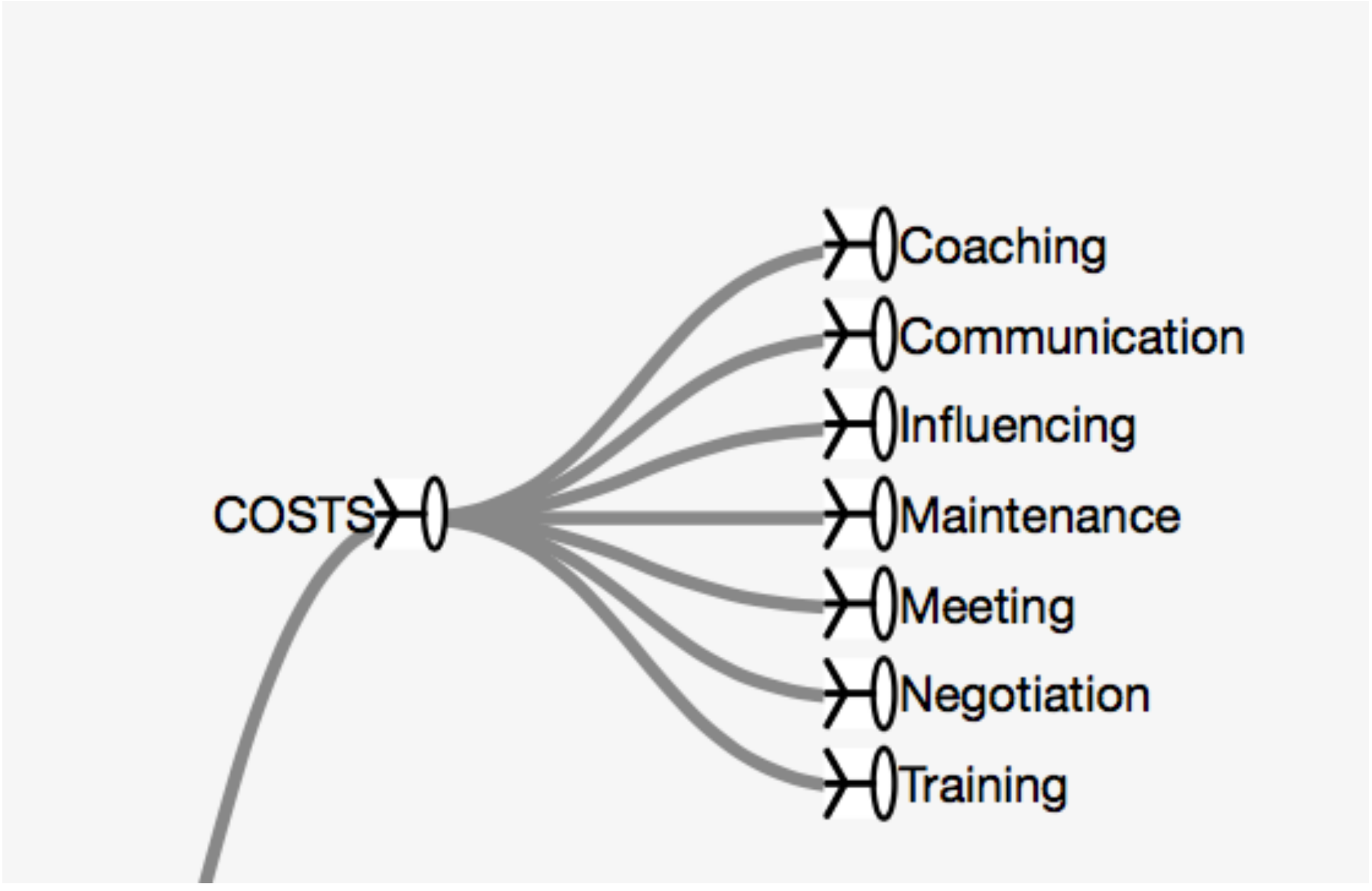


Stakeholder Attributes

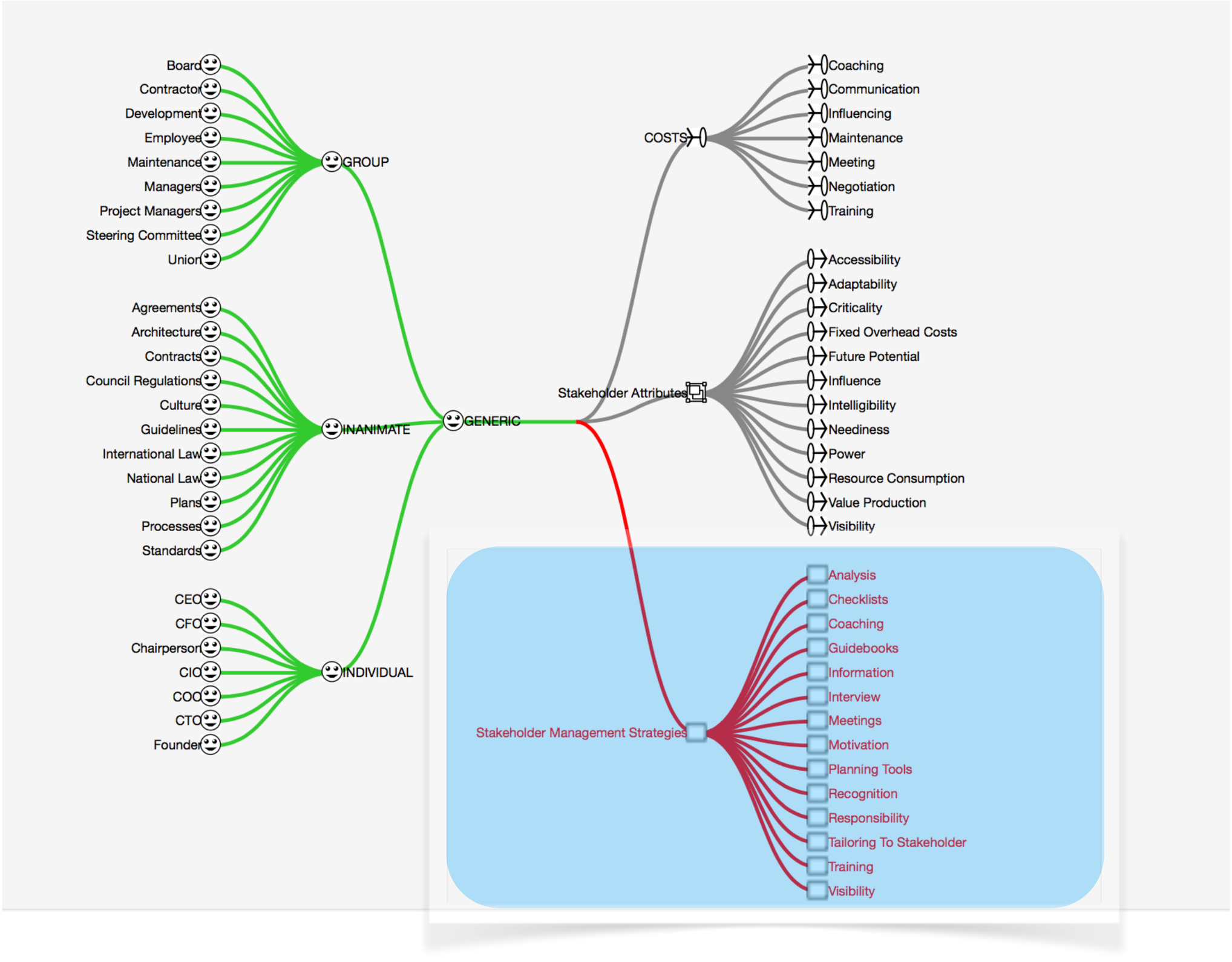
- Some attributes of stakeholders
- which can be defined in more detail,
- and can be quantified
- status estimated
- and potentially *improved*



Stakeholder Costs

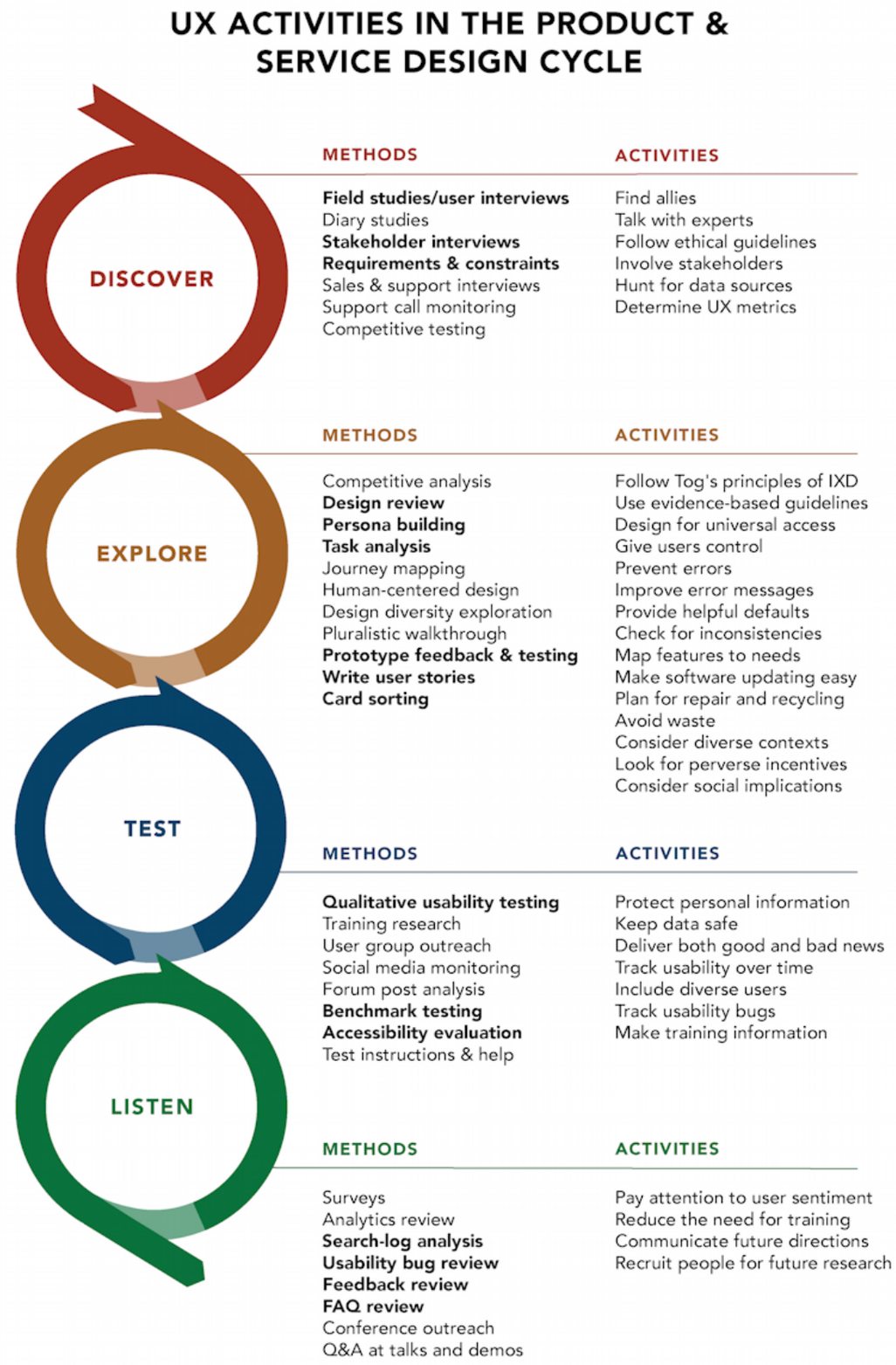


Adding Strategies for Improving Stakeholder Attributes



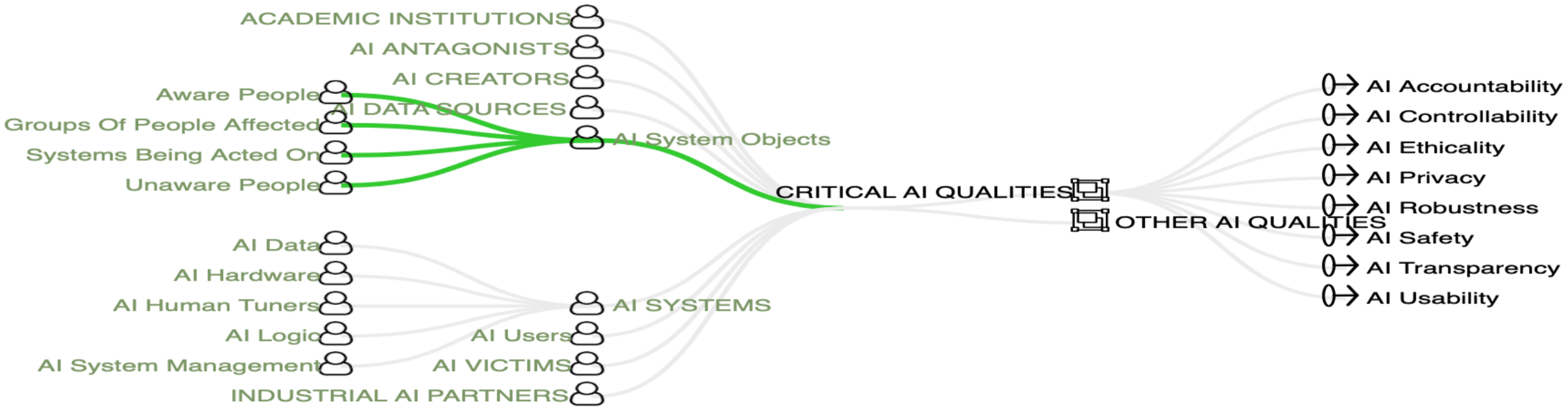
Stakeholder Feedback Types

- **Stakeholders have a variety of ways to feedback, react, and influence the process**
- **gradual measurement of value delivered versus value expected**
- **complaints**
- **‘Sensemaker’ ™ feedback**



https://www.nngroup.com/articles/ux-research-cheat-sheet/

Values
What Stakeholders Desire



- The Artificial Intelligence Stakeholders, above

- Have these values for AI Sytems

- among many other values

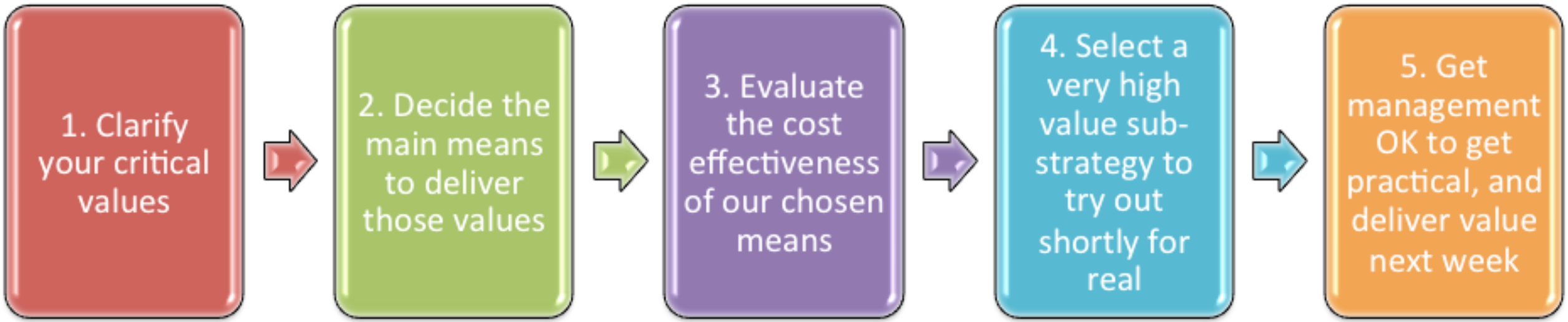
Startup Week: Process

An Agile Project Startup Week

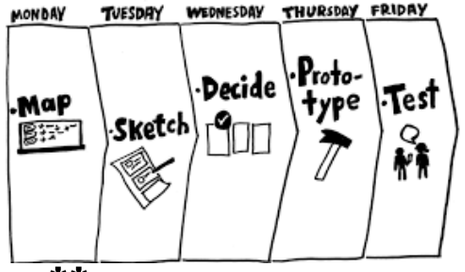
Gilb's Mythodology Column

www.gilb.com/dl568

Startup Week Purposes

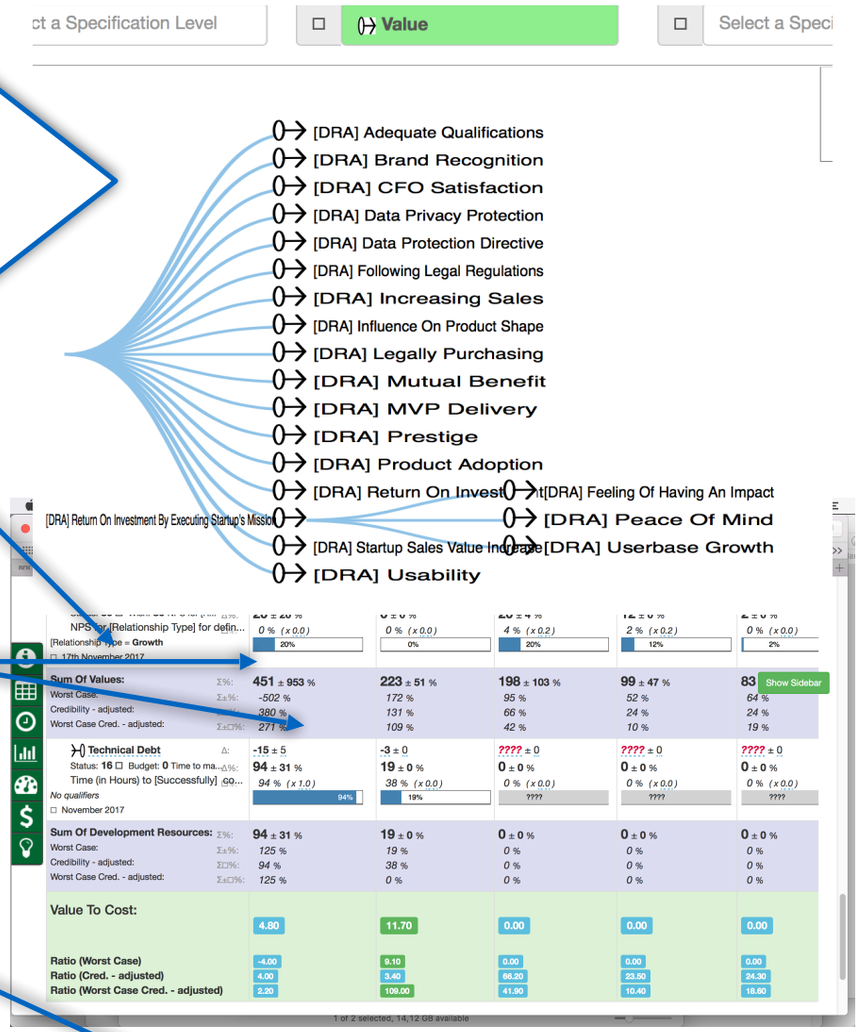


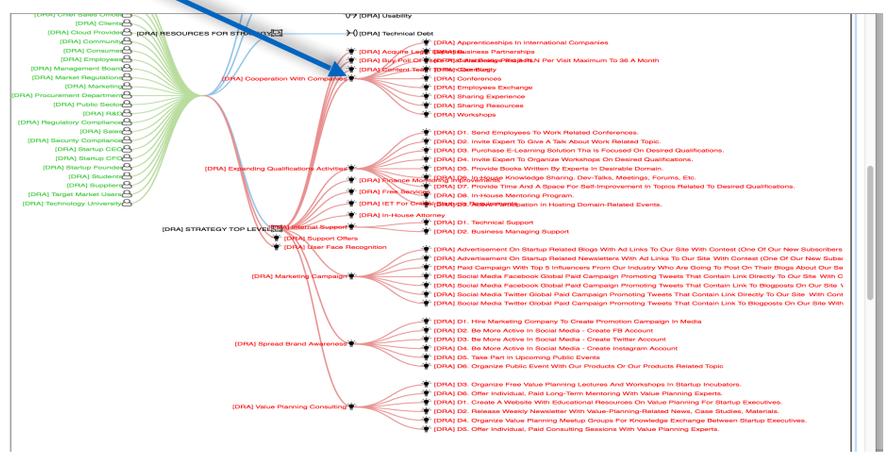
Evo Startup Standard, Jan 12 2013 http://www.gilb.com/dl562

An advanced ‘Design 

Sprint’ for grownups.

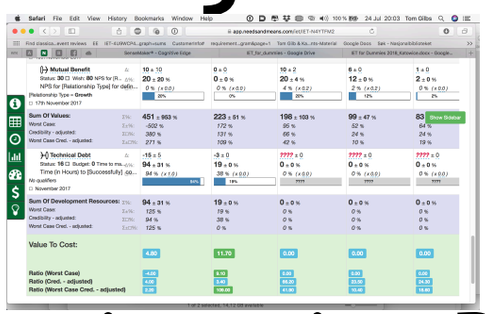
- The Startup Week*. Agile Value Delivery **
- Monday
  - Quantify critical stakeholder values
- Tuesday
  - Identify top 10 strategies or designs to each the values
- Wednesday
  - Rate strategies versus values and costs, and risks on an Impact Table
- Thursday
  - Decompose best strategy, and rate value/costs of details to choose next week’s value delivery
- Friday
  - meet with managers to get OK
- Next week (and every week later)
  - deliver some measurable stakeholder value
  - measure results, costs
  - learn about problems early
  - adjust designs for future



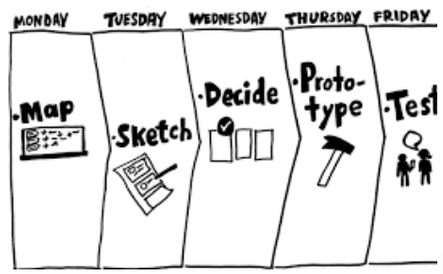


- * source is ‘Polish Export’ examples in ‘Innovative Creativity’ book (gilb.com) chapter 9. Done over 2 days with 60 people in 20 teams. Warsaw, at Startberry (startup Incubator)
- ** http://www.gilb.com/dl812, gilb.com/dl568
- DL812: extensive slides, DL568: short paper, see ‘Presenter Notes ‘in this slide.

Project Startup versus Design Sprint



Planguage
Evo



- Engineering Based
- Systems Applicable (UX)
- All Values Quantified
- Risk Mgt (±.Cred, Prty)
- Scale-Free
- Decades of Experience
- Research Published: HP
- Many publ.Case Studies
- AI Prioritization Val/€
- Design estimates V&€
- Actual incr. measures
- Digital Planning Long Term

- Programming Craft
- Software and UI Limited
- Values Not Quantified
- No Explicit Risk Mgt.
- Not proven large scale
- Hot new idea
- No known research
- Can’t find cases, yet
- Role player decides pri.
- No estimates
- Dodgy Prototype
- Yellow Sticky Culture

See Presenter Notes for references

Design Sprint 'Claimed Benefits' <-Jake
(of course YOU are skeptical, and know this.)

8 incredible Design Sprint benefits for your business

Here are the 8 amazing Design Sprint benefits you get in your business by employing this methodology of Google:

1. Design Sprint helps you save time and money

Design Sprint is designed to work quickly and intensely to get a solution to a business problem through design.

By using Design Sprint you reduce the time you spend on the design process and the process of defining your product, going from months to days

This is a great benefit because you save a lot of time and money and allows you to define a validation plan based on the feedback from your users.

2. Design Sprint Quickly Reduces Product Development Cycles

Derived from the above, development times are dramatically reduced, as Design Sprint work on a connecting problem with the solution. This helps you to test whether an idea works or not, without developing products with very long production cycles (Idea, Design, Approve, Develop, Launch and Validate).

With the Design Sprint you become a more agile organization

Before investing in the development of your product or a new functionality that requires an expensive process you can dedicate 5 days so that the team understands the problem that your company is facing, designing the solutions, creating a functional prototype and validating your ideas in a matter of hours. Becoming a more agile organization.

3. Real feedback with Design Sprint

Knowing the feedback of your product is fundamental to developing successful products. Many times when we get this information is when we have finished the project.

With the Design Sprint, you know firsthand and quickly the real feedback from your customers. This feedback is crucial because it helps you improve your product or service at the same time you design it

On the other hand, your team is actively working on the process, as the production cycle involves different sources of information within your organization.

4. Validate your business ideas with Design Sprint

Without validation, it is difficult for ideas and products to work. That is precisely what you will do on the last day of the Sprint in a very concrete way.

Through Design Sprint you can design the validation plan of the business idea or functionality of your product

Being clear how the process will be, the time you are going to invest and the type of results with which we can continue the process of transferring your product to the market.

5. Generates business and innovation.

Design Sprint gives your team a way of working to solve complex problems in a week.

So you can achieve a new approach to the project that would have taken months, even years

6. Align expectations with your team

Making all departments share knowledge, needs, and strategy so that the result is a solution that satisfies and meets needs.

Being able to make your step to deploy is a cycle of continuous product integration

7. Help you measure

The sprint design uses measurement processes in the different phases that the methodology uses.

What allows you to measure the results obtained at the end of the process, as well as the impact of the same on your business and on the equipment and surplus generated during the process

8. An agile and fast methodology that you can apply to your business

Once you internalize the Design Sprint methodology you can use it and coordinate it with other processes that you already have established in your project or business.

Typically, the first time you make a Sprint Design is tiring and difficult.

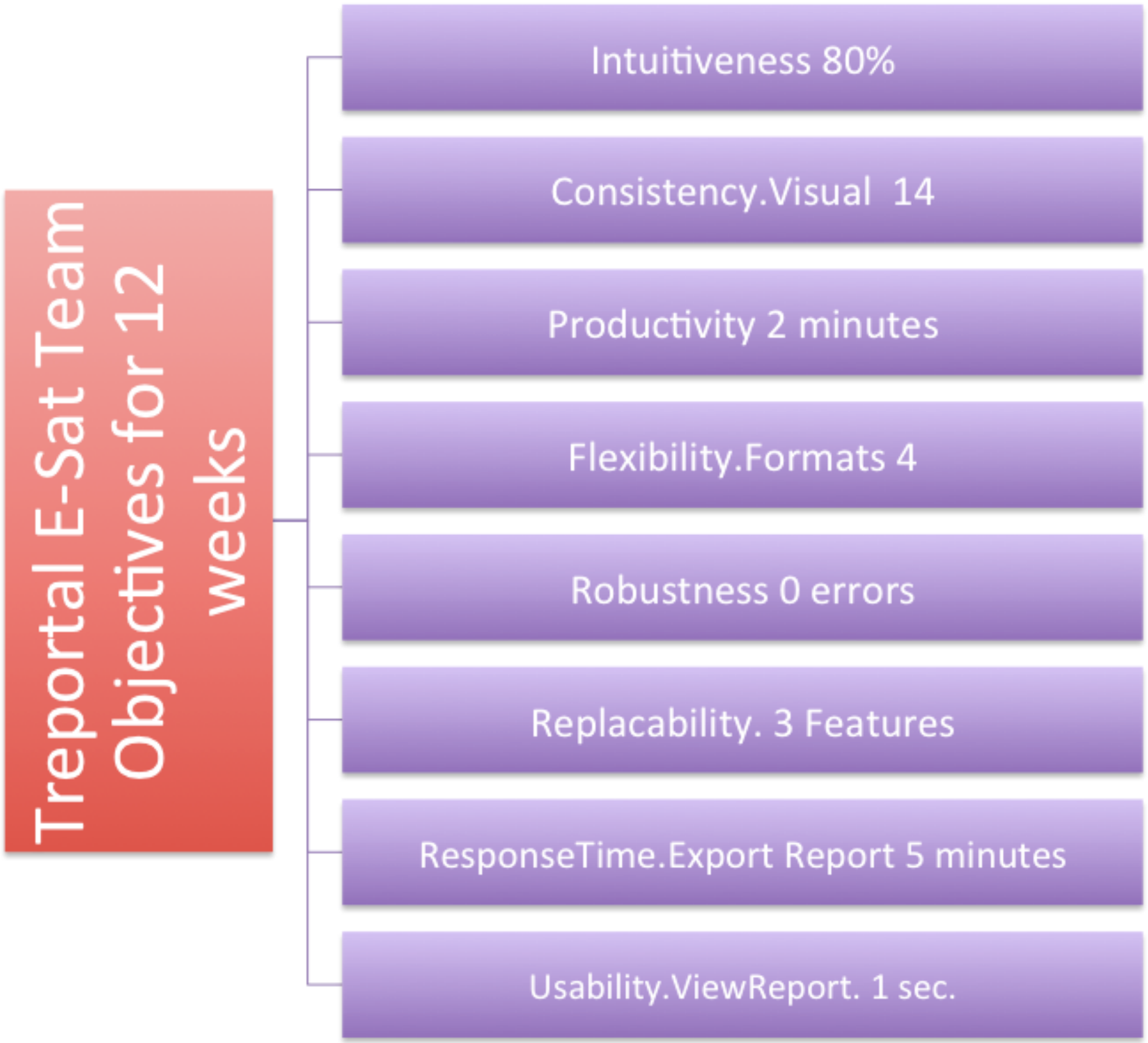
We recommend that you count with the help of a Sprint Master Certified to achieve these incredible results

Skeptical Observations <-TSG

- These claims are made by a **seller** of 'Design Sprint' training and certification service (letshackity.com)
- Most of the terms and concepts have **poor definition**, and are *highly ambiguous* (examples)
  - Design, Align Expectations, Investing (Product Dev), Complex Problems, measure the results, agile methodology, validation, and many more.
- Not one single number is offered to indicate the magnitude of improvements
- No clear baseline (who is going to get improved) is indicated
- No references to real case studies with results, costs, problems
- No comparison with any other known methods
- No links or references to anything
- Lots of causal assertions, none proven
- "This feedback is crucial because it helps you improve your product or service at the same time you design it"
- No indication or example of the types and magnitude of the costs for the individual, the project, and the organization for learning and maintaining the Design Sprint method
- No glowing references from real people or customers
- No information about how things went after the first week, to tell us how good or bad the week was.
- Constant implication: Google is successful, therefore this method is good

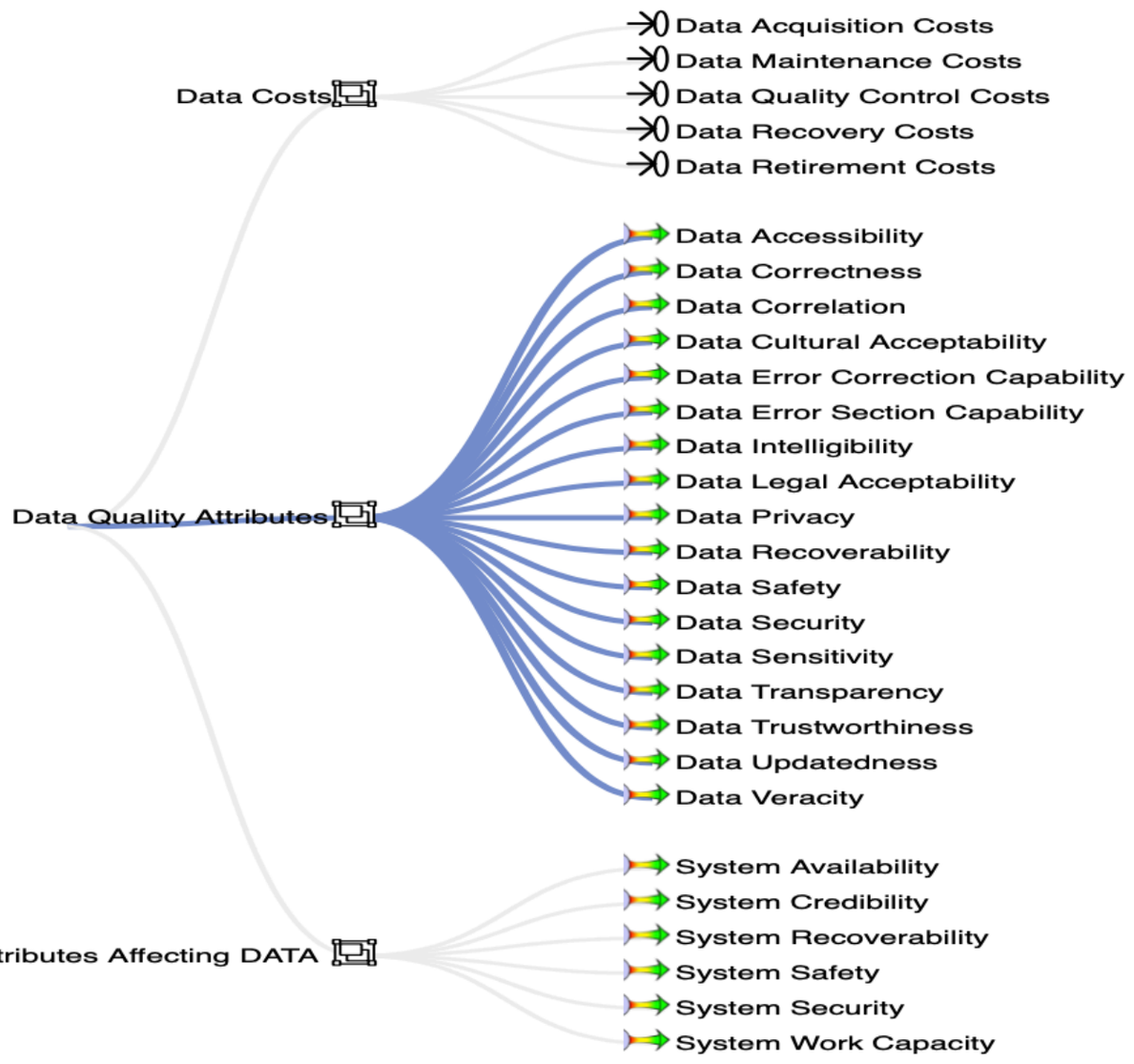
http://www.letshackity.com/en/design-sprint-benefits-business-innovation/

Example of Top Ten Critical Objectives (Real Set, Confirmit)



Some Data Attributes

Data Quality	Data Accessibility	Data Costs	Data Trust-worthiness	Data Correlation
Correctness	Security	Acquisition	Transparency	Sensitivity
Updatedness	Privacy	Maintenance	Safety	Veracity
Error Detection	Accessibility	Retirement	Trustworthi-ness	Correlation
Error correction		Quality Control		Cultural Acceptability
Accuracy		Recover-ability		Legal Admissability
Intelligibiity				



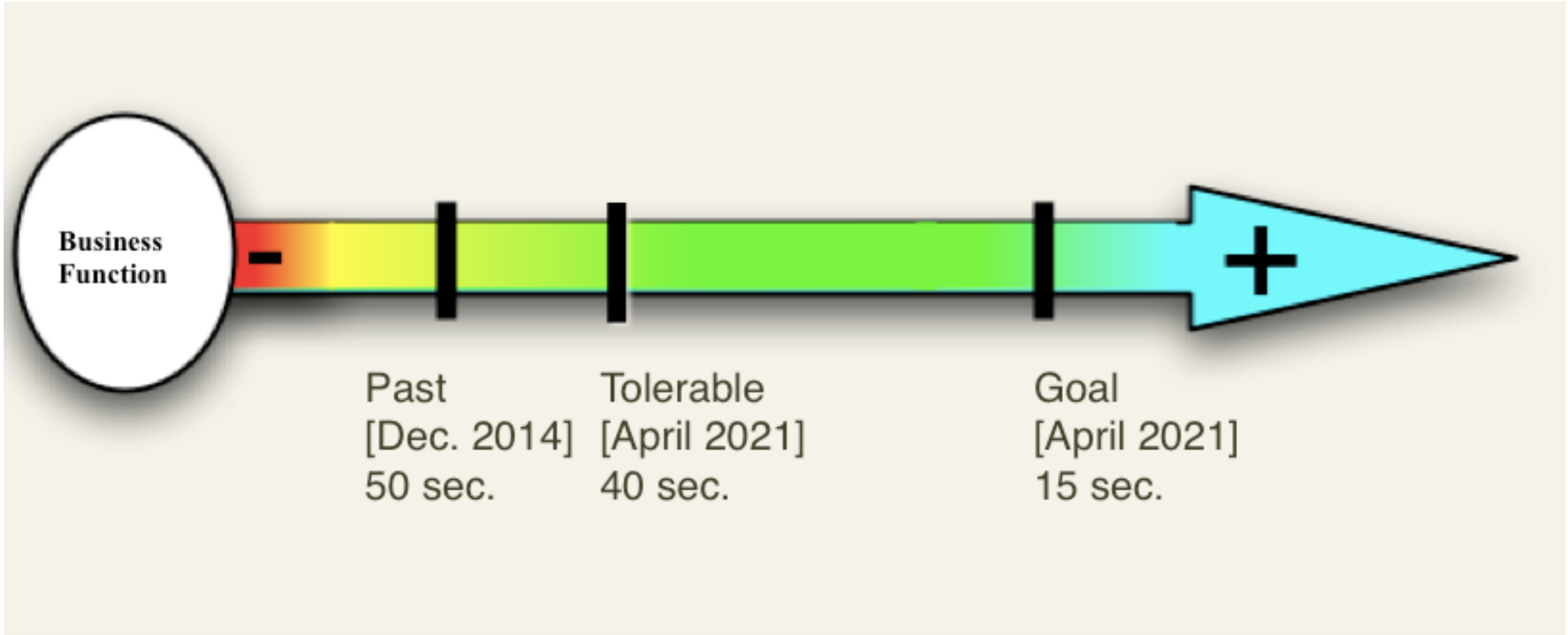
This is Step 1 in Data Engineering
Identification of Critical Values and Costs

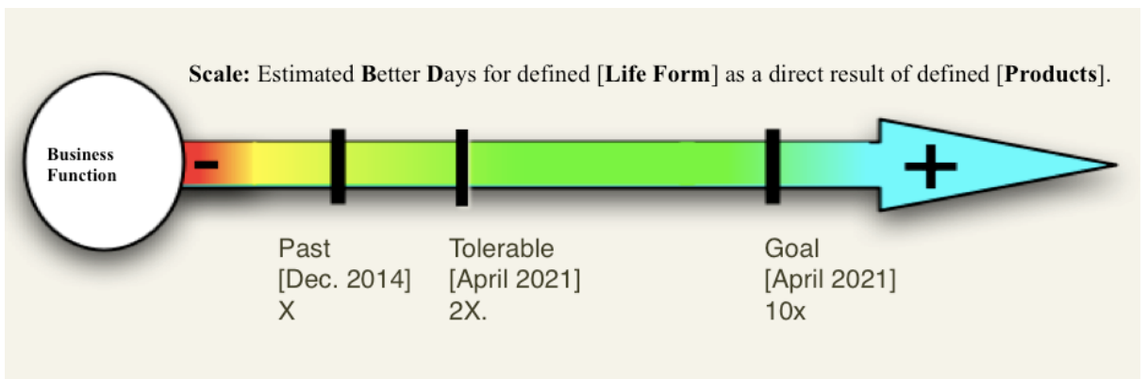
You might want to do a Step 0,

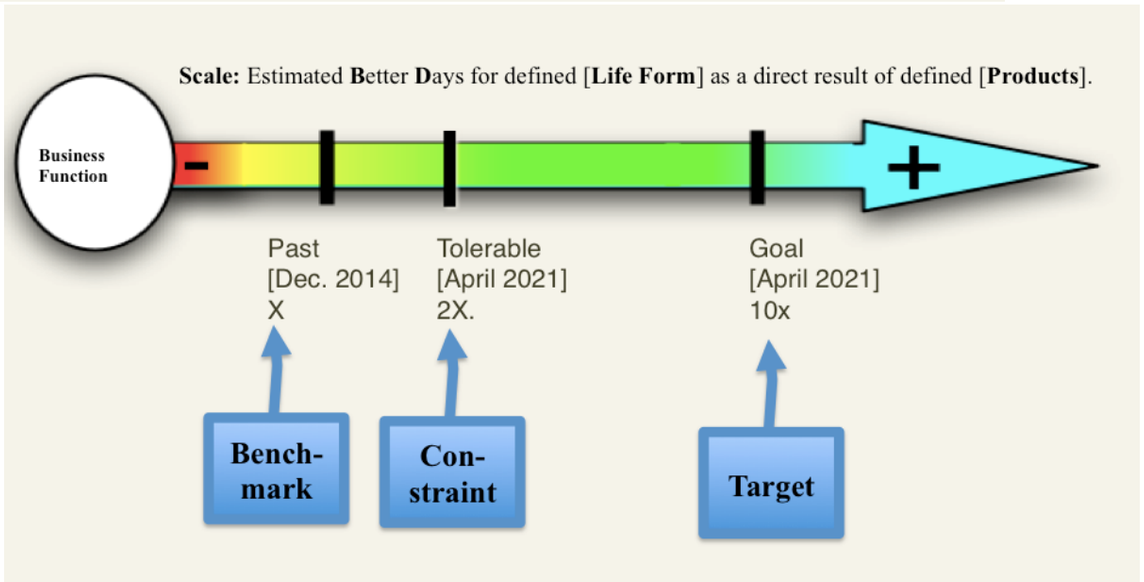
The identification of your Stakeholders
(Example EU -> GDPR - Privacy

THE QUANTIFICATION PRINCIPLE

Performance objectives,
ranging from *core objectives* to 'any' detailed performance objective
– where 'getting better-and-better in time' is implied –
can *always* be defined using 'scales of measure'.







Top 10 Large Bank Project Requirements
Quantifying the most-critical project objectives on day 1, on 1 page

P&L-Consistency&T P&L: **Scale:** total adjustments btw Flash/Predict and Actual (T+1) signed off P&L. per day. **Past 60 Goal**: **15**

Speed-To-Deliver: **Scale**: average Calendar days needed from New Idea Approved until Idea Operational, for given Tasks, on given Markets.
Past [2009, Market = EURex, Task =Bond Execution] **2-3 months** ?
Goal [Deadline =End 20xz, Market = EURex, Task =Bond Execution] **5 days**

Operational-Control: **Scale**: % of trades per day, where the calculated economic difference between OUR CO and Marketplace/Clients, is less than "1 Yen"(or equivalent).
Past [April 20xx] **10%** change this to 90% NH **Goal** [Dec. 20xy] **100%**

Operational-Control.Consistent: **Scale**: % of defined [Trades] failing full STP across the transaction cycle. **Past** [April 20xx, Trades=Voice Trades] **95%**
Past [April 20xx, Trades=eTrades] **93%**
Goal [April 20xz, Trades=Voice Trades] <**95 ± 2%**>
Goal [April 20xz, Trades=eTrades] **98.5 ± 0.5 %**

Operational-Control.Timely.End&OvernightP&L Scale: number of times, per quarter, the P&L information is not delivered timely to the defined [Bach-Run]. **Past** [April 20xx, Batch-Run=Overnight] 1 **Goal** [Dec. 20xy, Batch-Run=Overnight] <**0.5**> **Past** [April 20xx, Batch-Run= **T+1**] **1 Goal** [Dec. 20xy, Batch-Run=End-Of-Day, Delay<1hour] **1**

Operational-Control.Timely.IntradayP&L Scale: number of times per day the intraday P&L process is delayed more than 0.5 sec.

Operational-Control.Timely.Trade-Bookings Scale: number of trades per day that are not booked on trade date. **Past** [April 20xx] **20 ?**

Front-Office-Trade-Management-Efficiency Scale: Time from Ticket Launch to trade updating real-time risk view
Past [20xx, Function = Risk Mgt, Region = Global] **~ 80s** +/- **45s** ??
Goal [End 20xz, Function = Risk Mgt, Region = Global] **~ 50%** better?
Managing Risk – Accurate – Consolidated – Real Time

Risk.Cross-Product Scale: % of financial products that risk metrics can be displayed in a single position blotter in a way appropriate for the trader (i.e. – around a benchmark vs. across the curve).
Past [April 20xx] **0%** 95%. **Goal** [Dec. 20xy] **100%**

Risk.Low-latency Scale: number of times per day the intraday risk metrics is delayed by more than 0.5 sec. **Past** [April 20xx, NA] **1% Past** [April 20xx, EMEA] ??% **Past** [April 20xx, AP] 100% **Goal** [Dec. 20xy] **0%**

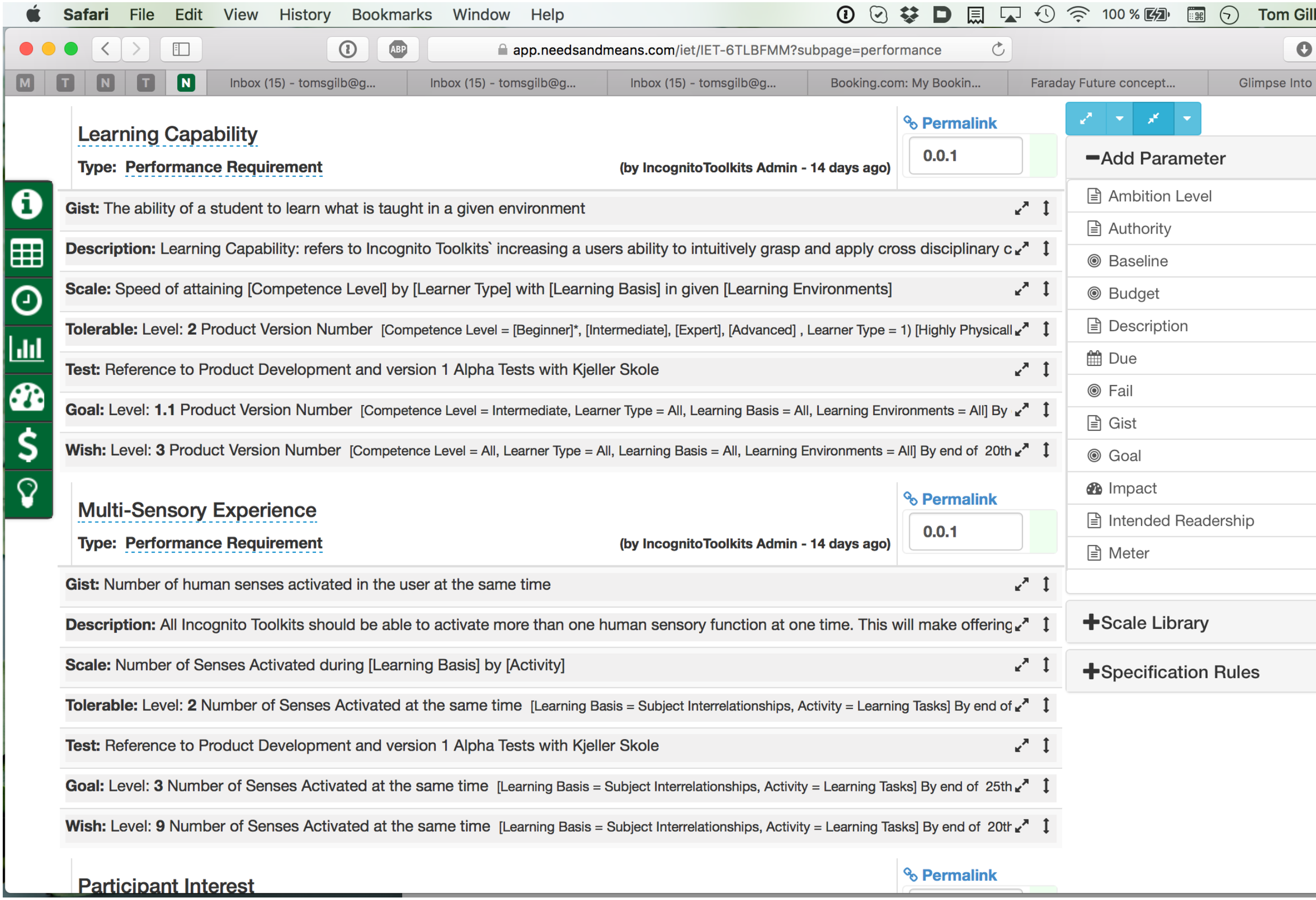
Risk.Accuracy

Risk. user-configurable Scale: ??? pretty binary – feature is there or not – how do we represent?
Past [April 20xx] **1% Goal** [Dec. 20xy] **0%**

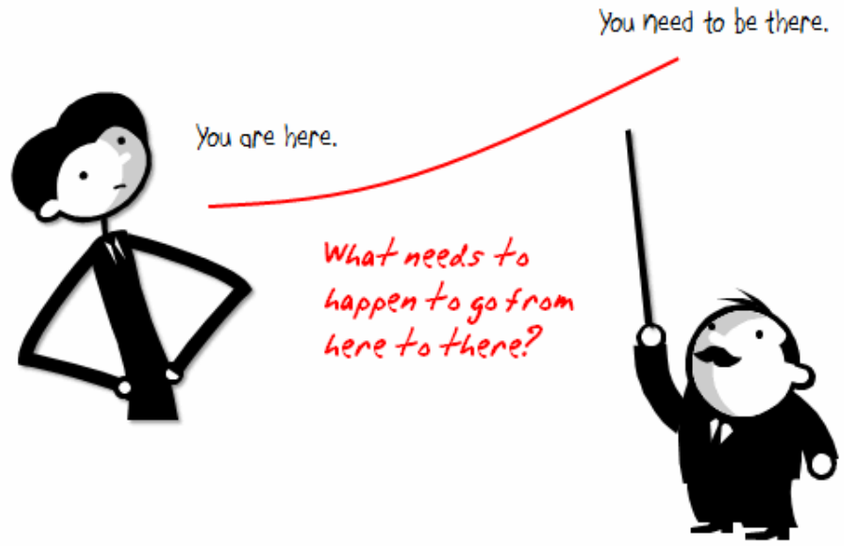
Operational Cost Efficiency Scale: <Increased efficiency (Straight through processing STP Rates)>

Cost-Per-Trade Scale: % reduction in Cost-Per-Trade
Goal (EOY 20xy, cost type = I 1 – REGION = ALL) **Reduce cost by 60%** (BW)
Goal (EOY 20xy, cost type = I 2 – REGION = ALL) **Reduce cost by x %**
Goal (EOY 20xy, cost type = E1 – REGION = ALL) **Reduce cost by x %**
Goal (EOY 20xy, cost type = E 2 – REGION = ALL) **Reduce cost by 100%**
Goal (EOY 20xy, cost type = E 3 – REGION = ALL) **Reduce cost by x %**

Quantifying Critical Values

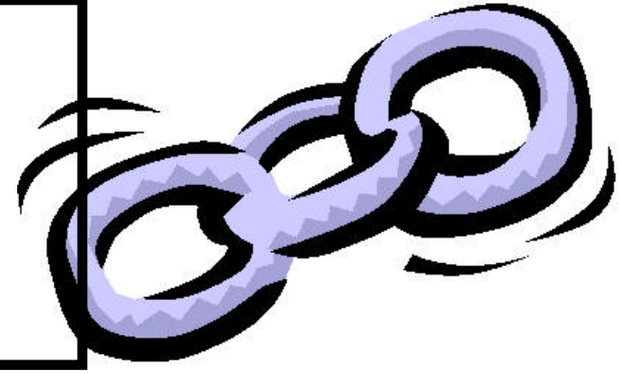


Real Example
"Platform Rationalisation Initiative"
"<u>Main Objectives.</u>"
<u>London Multinational Bank</u>



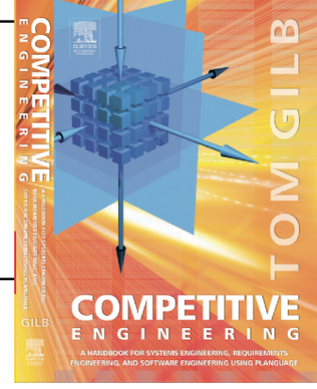
- **Rationalize into a smaller number of core processing platforms. This cuts technology spend on duplicate platforms, and creates the opportunity for operational saves. Expected 60%-80% reduction in processing cost to Fixed** Income **Business levies.**
- **International Securities on one platform, Fixed Income and Equities (Institutional and PB).**
- **Global Processing consistency with single Operations In-Tray and associated workflow.**
- **Consistent financial processing on one Accounting engine, feeding a single sub-ledger across products.**
- **First step towards evolution of "Big Ideas" for Securities.**
- ***<u>Improved development environment</u>*, leading to increased capacity to enhance functionality in future.**
- **Removes duplicative spend on two back office platforms in support of mandatory message changes, etc.**

LINK WORDS: OBJECTIVE:ARCHITECTURE
RULE 4. No Design/Architecture



- **Rationalize into a smaller number of core processing platforms. *This cuts technology spend* on duplicate platforms, and *creates the opportunity for* operational saves. Expected 60%-80% reduction in processing cost to Fixed Income Business lines.**
- **International Securities on one platform, Fixed Income and Equities (Institutional and PB).**
- **Global Processing consistency with single Operations In-Tray and associated workflow.**
- **Consistent financial processing on one Accounting engine, feeding a single sub-ledger across products.**
- **First step towards evolution of "Big Ideas" for Securities.**
- ***Improved development environment*, leading to increased capacity to enhance functionality in future.**
- **Removes duplicative spend on two back office platforms in support of mandatory message changes, etc.**

How can we improve such bad specification? (‘Planguage’)



Development Capacity:

Version: 3 Sept 2009 16:26

Type: Main <Complex/Elementary> Objective for a project.

Ambition Level: radically increase the capacity for developers to do defined tasks. <- Tsg

Scale: the Calendar Time for defined [Developers] to Successfully carry out defined [Tasks].

Owner: Tim Fxxx

C**alendar Time**: defined as: full working days within the start to delivery time frame.

Past [2009, {Bxx, Lxx, Gxx}, If QA Approved Processes used, Developer = Architect, Task = Draft Architecture] **15 days** ±4 ?? <- Rob

Goal[2011, { Bxx, Lxx, Gxx }, If QA Approved Processes used, Developer = Architect, Task = Draft Architecture] **1.5 days** ± 0.4 ?? <- Rob

Justification: Really good architects are very scarce so we need to optimize their use.

Risks: we use effort that should be directed to really high volume or even more critical areas (like Main Objective).

QUANTIFICATION OF ALL CRITICAL VALUES AND QUALITIES

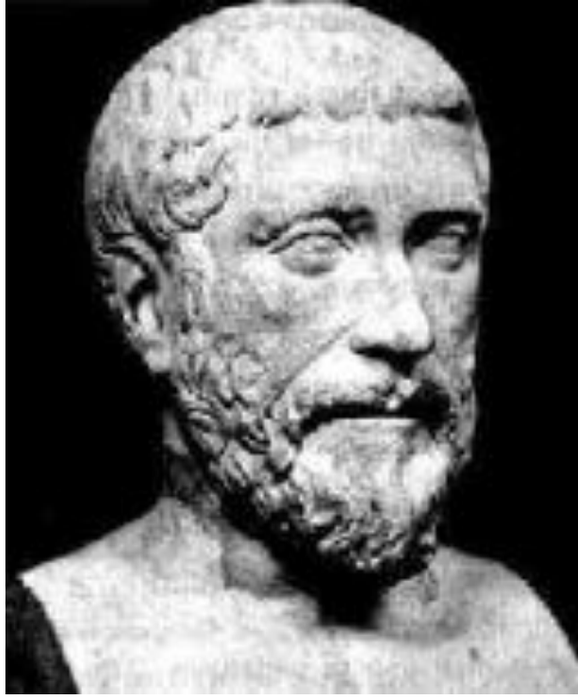
no management bullshit

no user stories

all improvements quantified/estimated/tracked

all qualities quantified/estimated/tracked

Philolaus on Numbers



- Over four hundred years BC,
- a Greek by the name of
- **Philolaus** of Tarentum said :

- *"Actually, everything that can be known has a Number;*
  - *for it is impossible to grasp anything with the mind or to recognize it without this (number)."*

Best regards (Aug 2005),
N.V.Krishnawww.microsensesoftware.com

How to Quantify any Qualitative Requirement

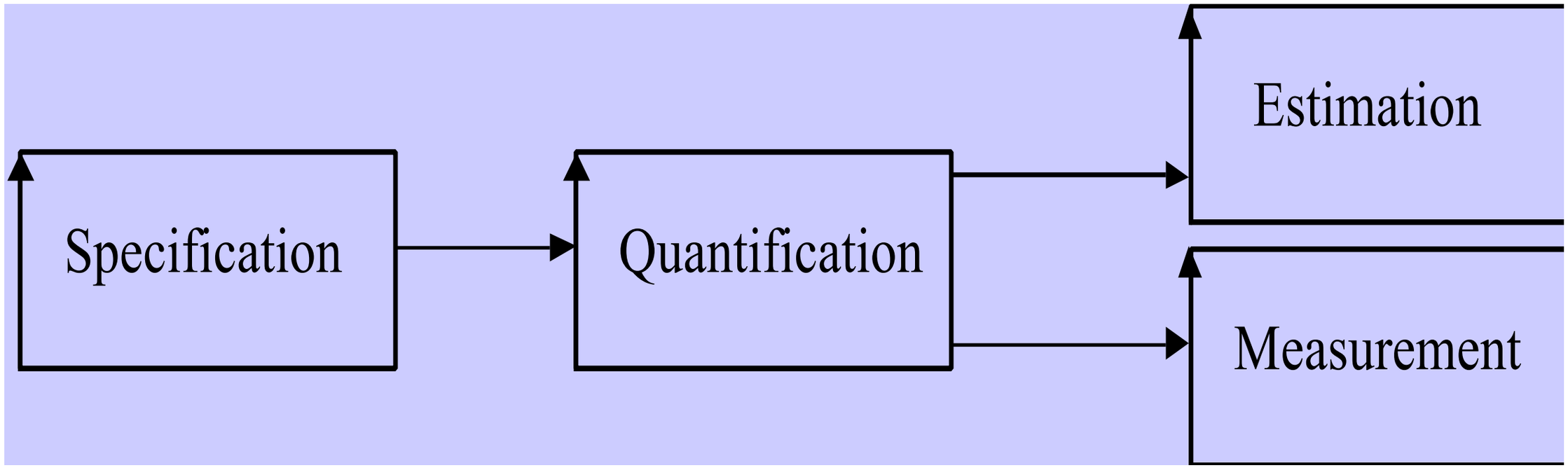


Diagram from 'Competitive Engineering.' book.

Quality Quantification Methods #1

- Common Sense, Domain Knowledge
  - Decompose “until quantification becomes obvious”.
  - Then use Planguage specification:
    - **Scale**: define a measurement scale
    - **Meter**: define a test or process for measuring on the scale
    - **Past**: define benchmarks, old system, competitors on the scale
    - **Goal**: define a committed level of future stakeholder quality, on your scale.

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

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

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

Quality Quantification Methods #**2**, **Look it up in a book**

Chapter
5

Scales of Measure

How to Quantify

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

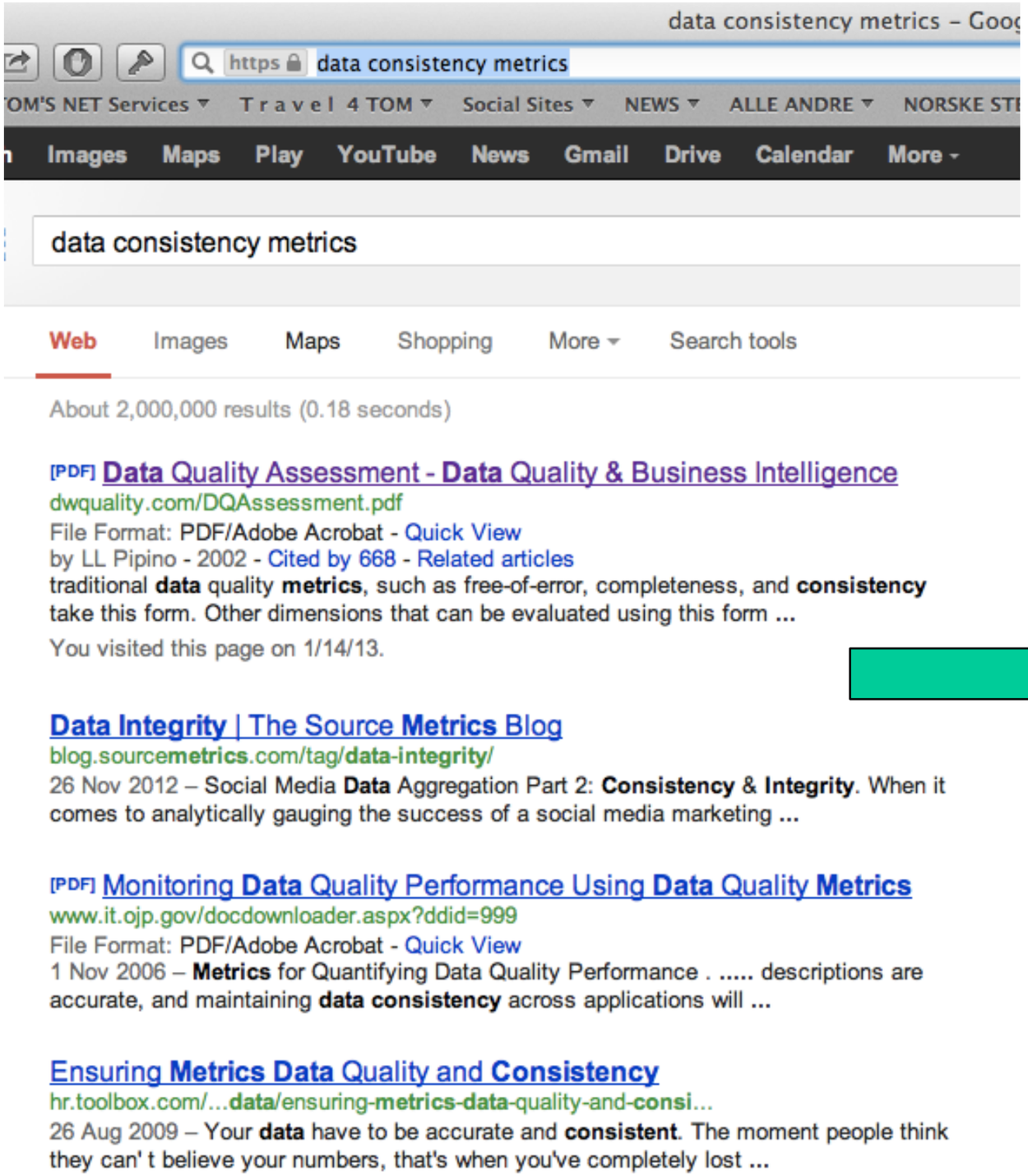
Problem Recognition:
Scale: C
system]
electroni
Adminis
Scale: C
assigned
Tool Co
Scale: C
acquire
and qua
Problem
Scale: C
toms an
Change
Scale: C
the nece
Note: Th
Quality
Scale: C
Modifica
Scale: C
correctio
Modifica
Unit T
Scale:
Integr
Scale:
Beta T
Scale:
releas
Syste
Scale:
Recovery:
Scale: Clock hours for defined [User Type] to return system to the state it was in prior to the fault and, to a state ready to continue with work.

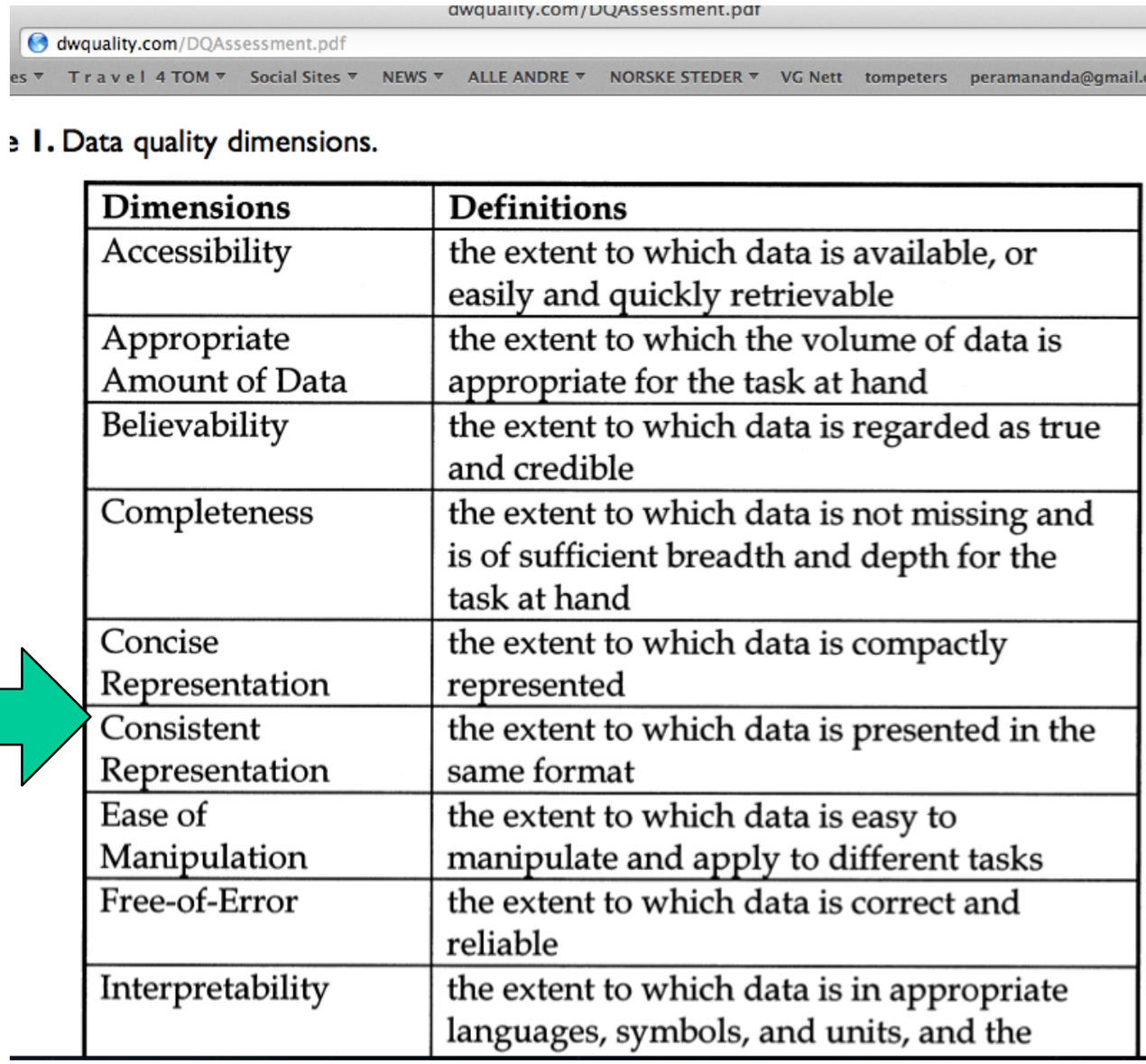
Source: The above is an extension of some basic ideas from Ireson, Editor, Reliability Handbook, *McGraw Hill, 1966* (Ireson 1966).

Quality Quantification Methods #**2**,
Look it up in a book

Tool Collection:
Scale: Clock hours for defined [Maintenance Instance: Default: Whoever is assigned] to acquire all defined [Tools: Default: all systems and information necessary to analyze, correct and quality control the correction].

Quality Quantification Methods #**3**, Google It



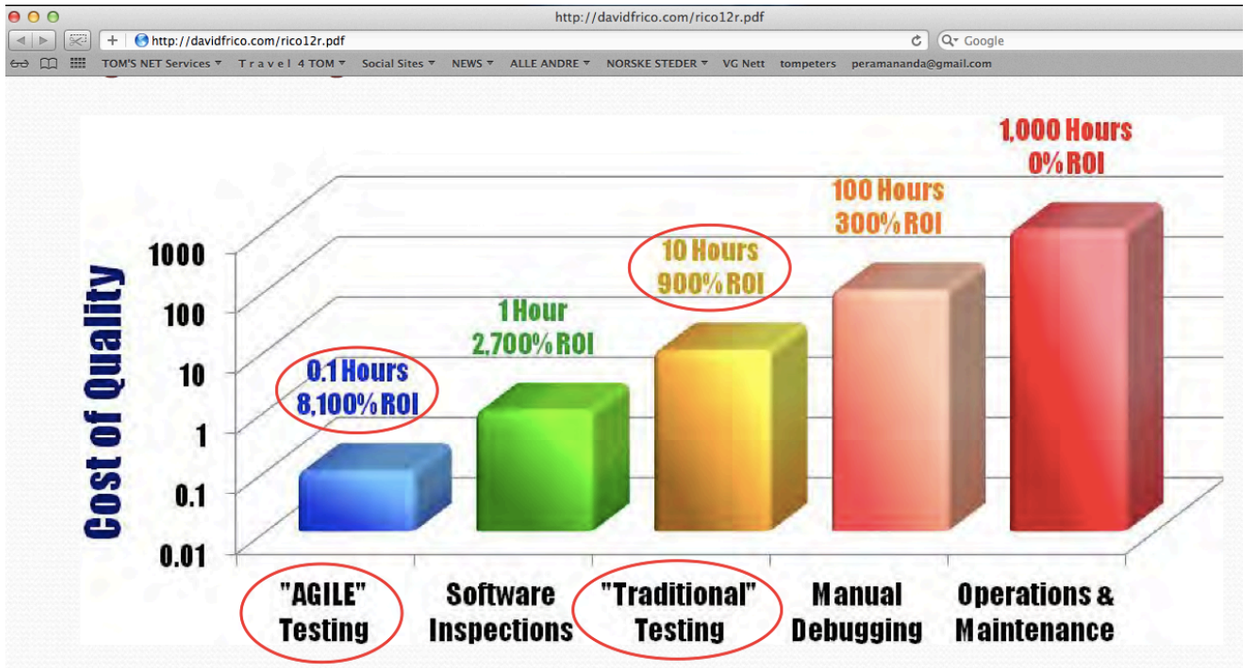


1. Data quality dimensions.

Dimensions	Definitions
Accessibility	the extent to which data is available, or easily and quickly retrievable
Appropriate Amount of Data	the extent to which the volume of data is appropriate for the task at hand
Believability	the extent to which data is regarded as true and credible
Completeness	the extent to which data is not missing and is of sufficient breadth and depth for the task at hand
Concise Representation	the extent to which data is compactly represented
Consistent Representation	the extent to which data is presented in the same format
Ease of Manipulation	the extent to which data is easy to manipulate and apply to different tasks
Free-of-Error	the extent to which data is correct and reliable
Interpretability	the extent to which data is in appropriate languages, symbols, and units, and the

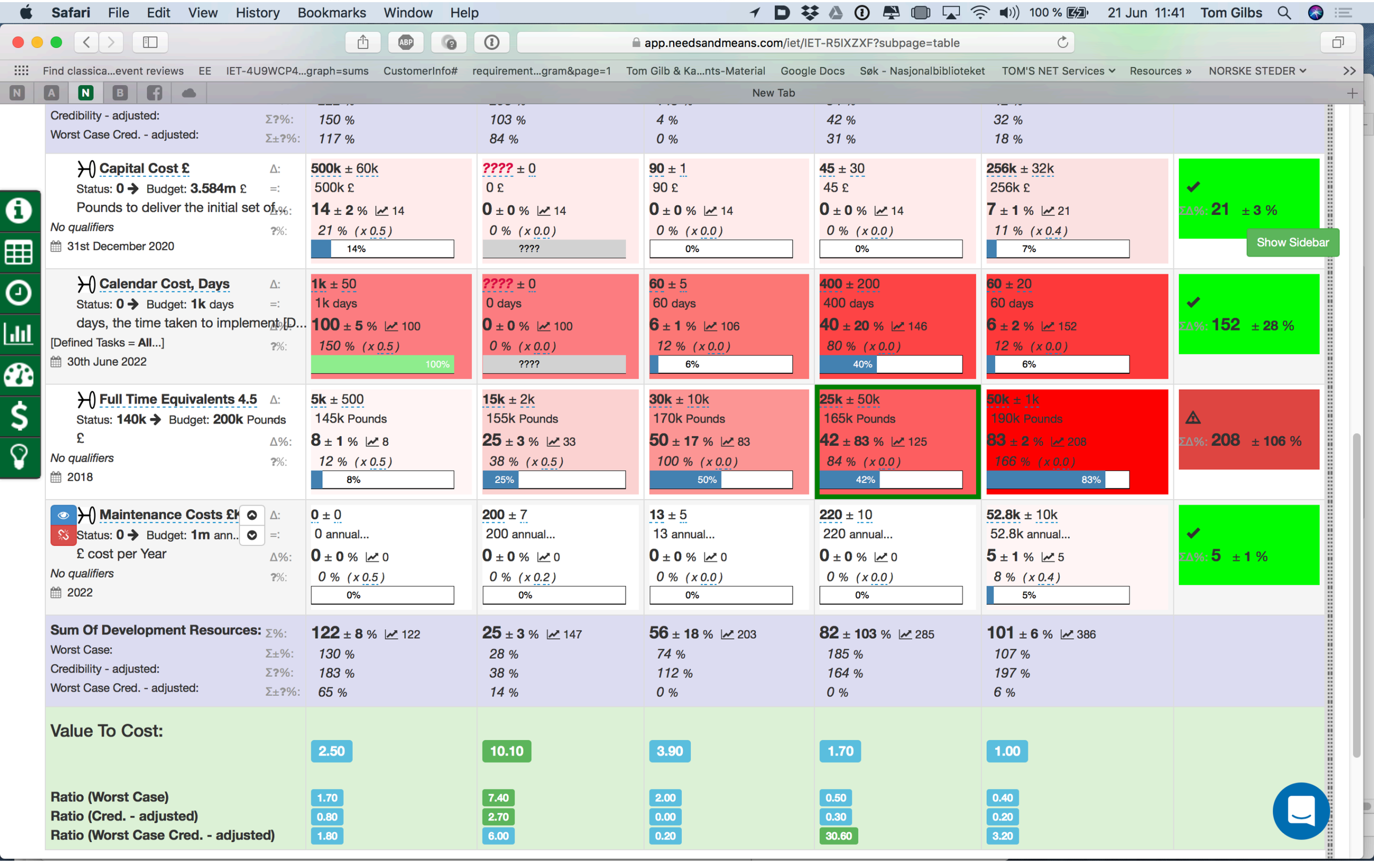
Resources

- **If you do not manage ‘resources’ explicitly and continuously**
- **Your resource use will kill your project and product.**
- **You have to ‘design’ low costs**
- **Costs are not inevitable, they are often avoidable: if you are smart and continually focussed on cost management**
- **Agile (step by step delivery) gives many special opportunities to manage costs**
  - **Early warning during development**
  - **Early opportunity to discover operational costs**
  - **Opportunity to change design in order to lower costs**
  - **Opportunity to see quantified Value/Cost ratio in practice, and improve ‘efficiency’**



Rico, D. F. (2012). *The Cost of Quality (CoQ) for Agile vs. Traditional Project Management*. Fairfax, VA: Gantthead.Com.

Keeping Track of 4 Types of Resources



Constraints

Planguage Conc

- **All requirements 'constrain' us but**
  - **some types of requirements INTEND to constrain us**
  - **Scalar Constraint levels of values and resources**
  - **Function Constraints**
  - **Condition Constraints**
  - **Resource Constraints**
  - **Value Constraints**
    - **Tolerable, Fail, OK, Survival 'levels'**

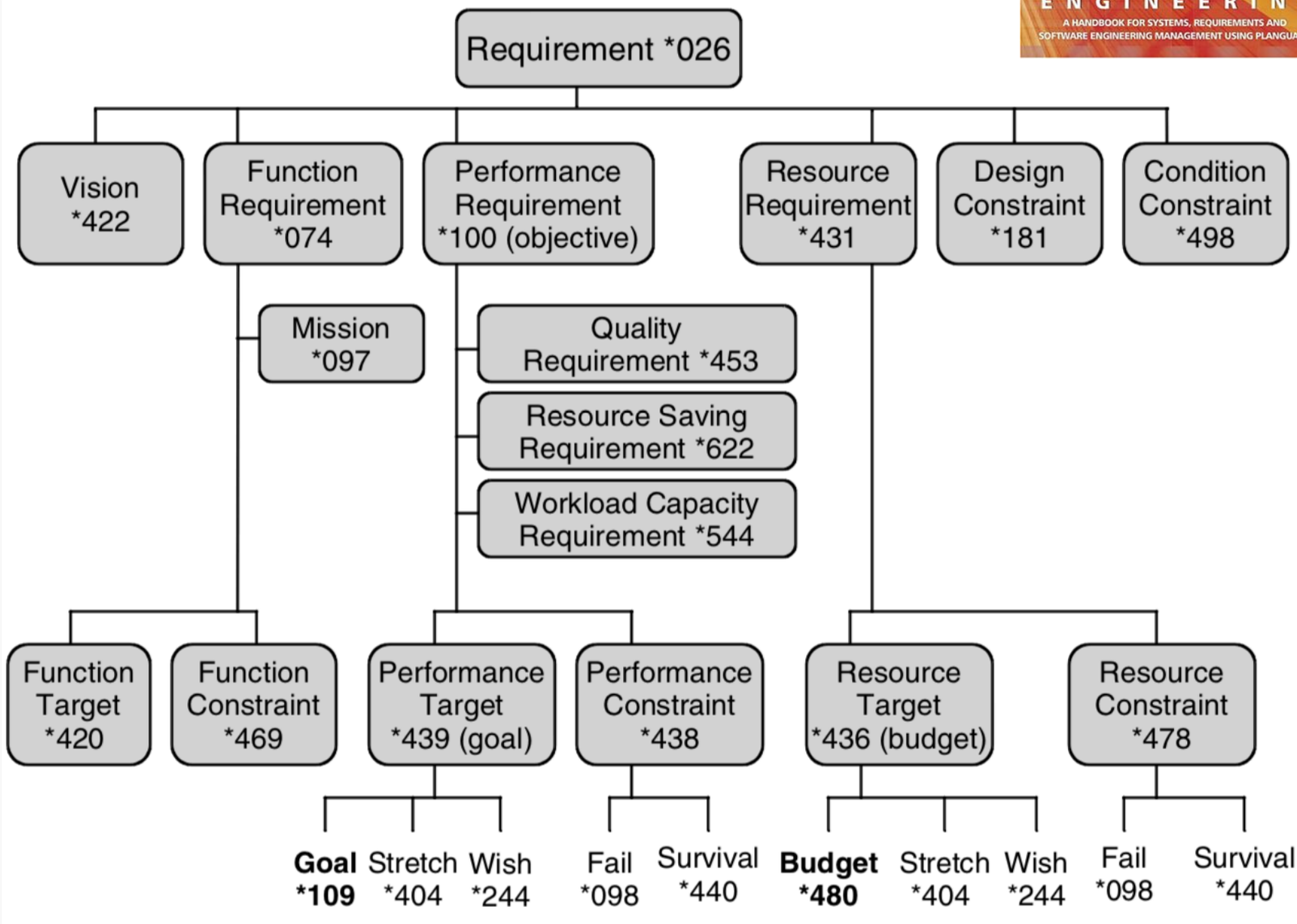
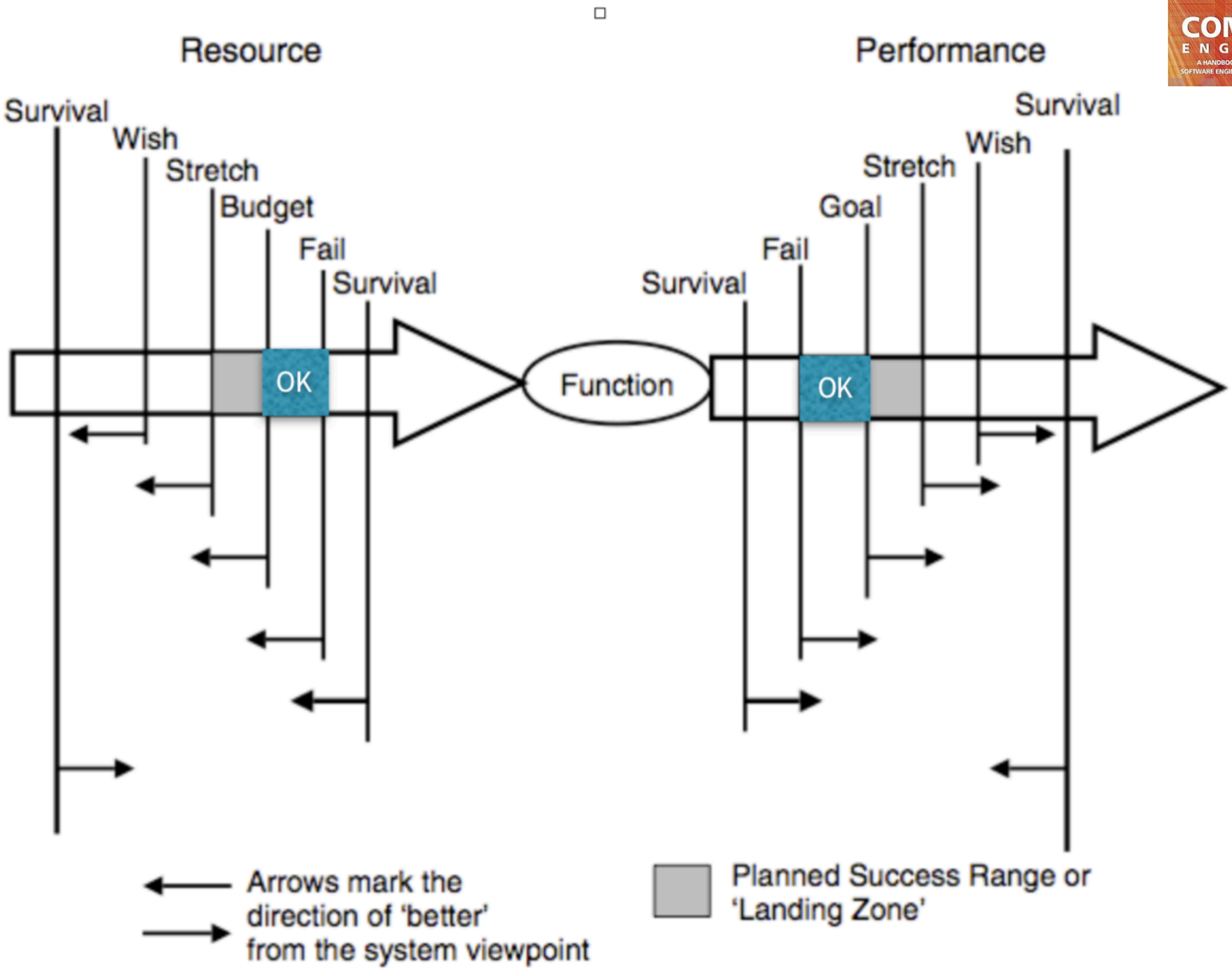


Figure G20
Requirement Concepts.

Planguage Scalar Constraints and Targets



Priorities (Dynamic)

Multi Play – Suited for diversified line of business

Ready to assemble – Mix and match components to optimize CAPEX/OPEX based on business needs

Rapid Implementation – Open architecture for quick time to market

Dynamic Scaling – horizontal and vertical

Adaptability - Modular plug-ins that leverage existing services & legacy assets

Realtime Visibility - Monitor resources, connectivity and control access

Flexibility - Customizable & configurable modules

Performance - Optimized network utilization

Feature rich COTS- ready to deploy

http://www.triwaytechnologies.com/telecom-suite/

Priority Determination: ‘as you go’, just like in a restaurant

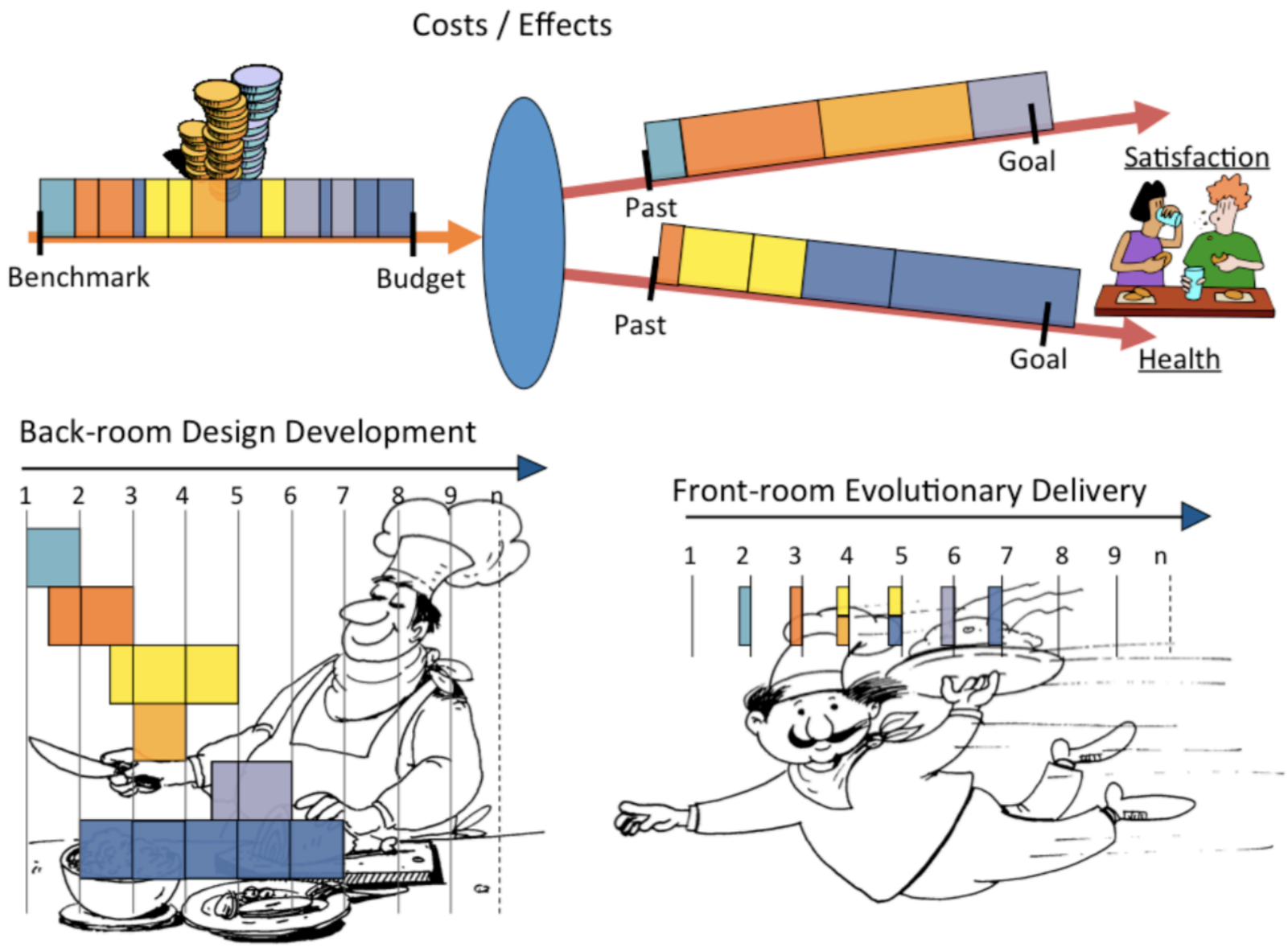


Figure 6.3 One visualization of the prioritization problem.
On the one hand, we are investing up front in the back room, consuming limited budget, and not immediately getting any value back. Is this a wise investment? A necessary evil? But we can track incremental value delivery from Past to Goal, and see the value build up. We need to figure out the lowest-cost set of sub-strategies to reach our Goal levels. Reality is of course at least ten times more complicated than this simple model.

Confirmit, Oslo
Computing residual priorities

Priority Signals

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	1150	500	1000
			Development resources		
64,0			0		84

Figure 6.7 . Incremental Value Tracking at Confirmit.

Decomposition

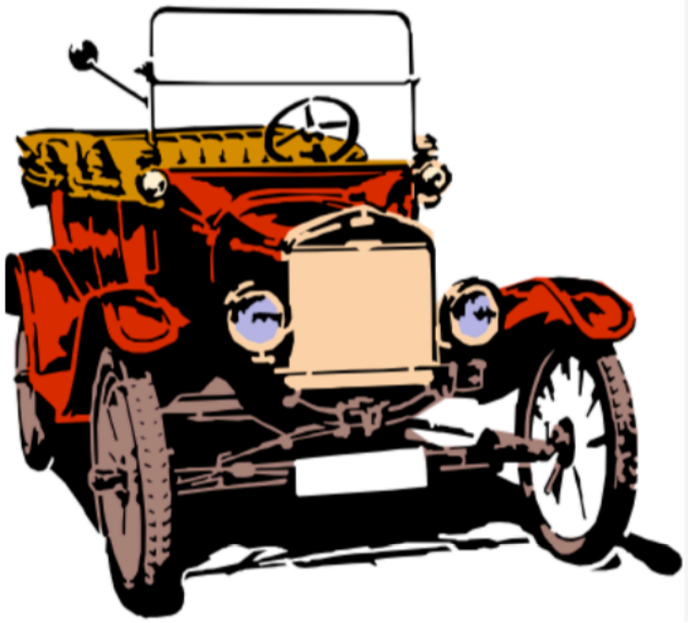
"Nothing is particularly hard if you divide it into small jobs."

"There are no big problems, just a lot of little problems"

"Obstacles are those frightful things you see when you take your eyes off your goal"

--Henry Ford, 1863-1947

•



Ford in 1919

Musk the De-composer: Agile Car Manufacturing

***Elon Reeve Musk* - Chairman & Chief Executive Officer Tesla.**

"Okay, I think that's a pretty open-ended questions, but – we have a philosophy of just continuous improvements, so every week there are approximately 20 engineering changes made to the car.

So it's not nearly as discrete as you're alluding to. With other manufacturers, they tend to sort of bundle everything together in a model year.

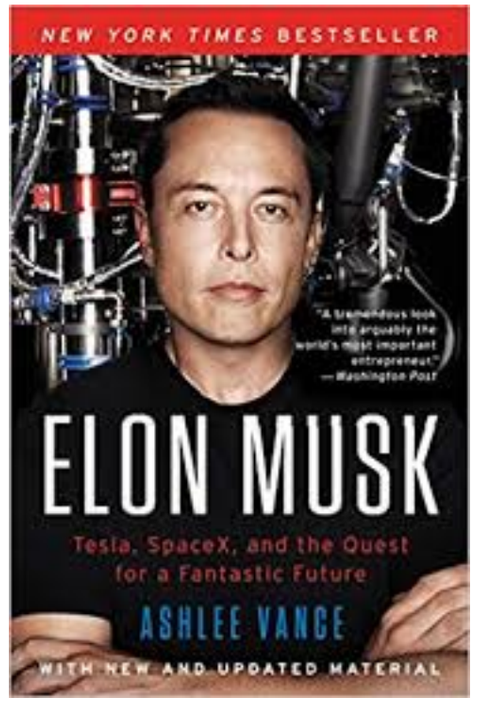
In our case, it's a series of rolling changes. So model year doesn't mean as much. There are cases where that step change may be a little higher than normal as, for example, with having the Autopilot camera, radar, and ultrasonics.

But we try to actually keep those step changes as small as possible.

And so that – I mean, essentially like the common questions that I get is from friends, they say, "when should I buy a Model S?" and my answer's always "right now," because – and they say, "well, aren't you going to make a better one in six months?" I'm like, yeah, of course.

But if their goal is to only buy a Model S when there aren't significant improvements happening, then they will never buy one."

Quote 5.3 A. Musk on eternal continuous Tesla improvement.20/week x 50 weeks = 1,000 improvements per year. Source: http://seekingalpha.com/article/3642146-tesla-motors-tsla-elon-reeve-musk-on-q3-2015-results-earnings-call-transcript?page=2&p=qanda&l=last November 3 2015.



Risks

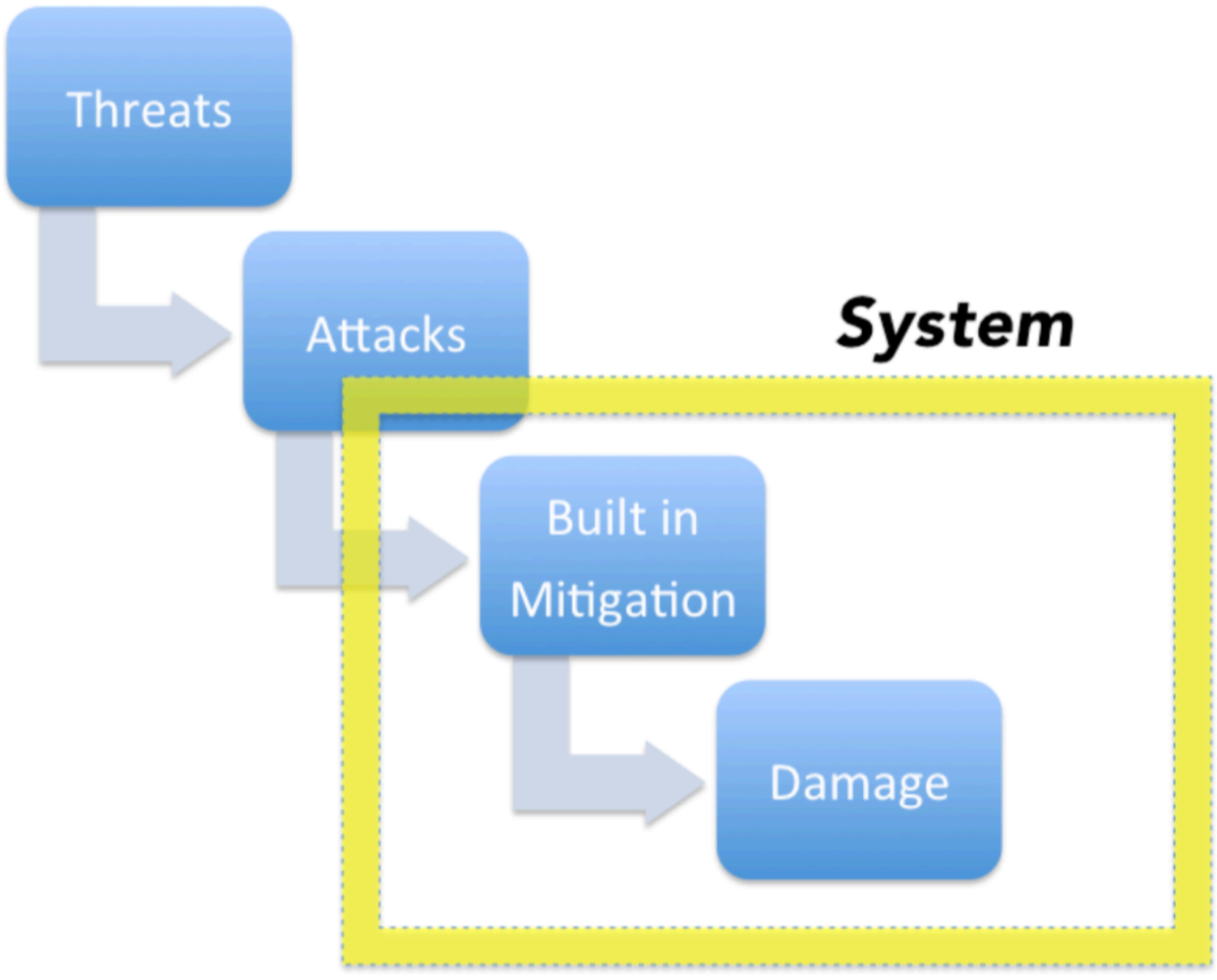
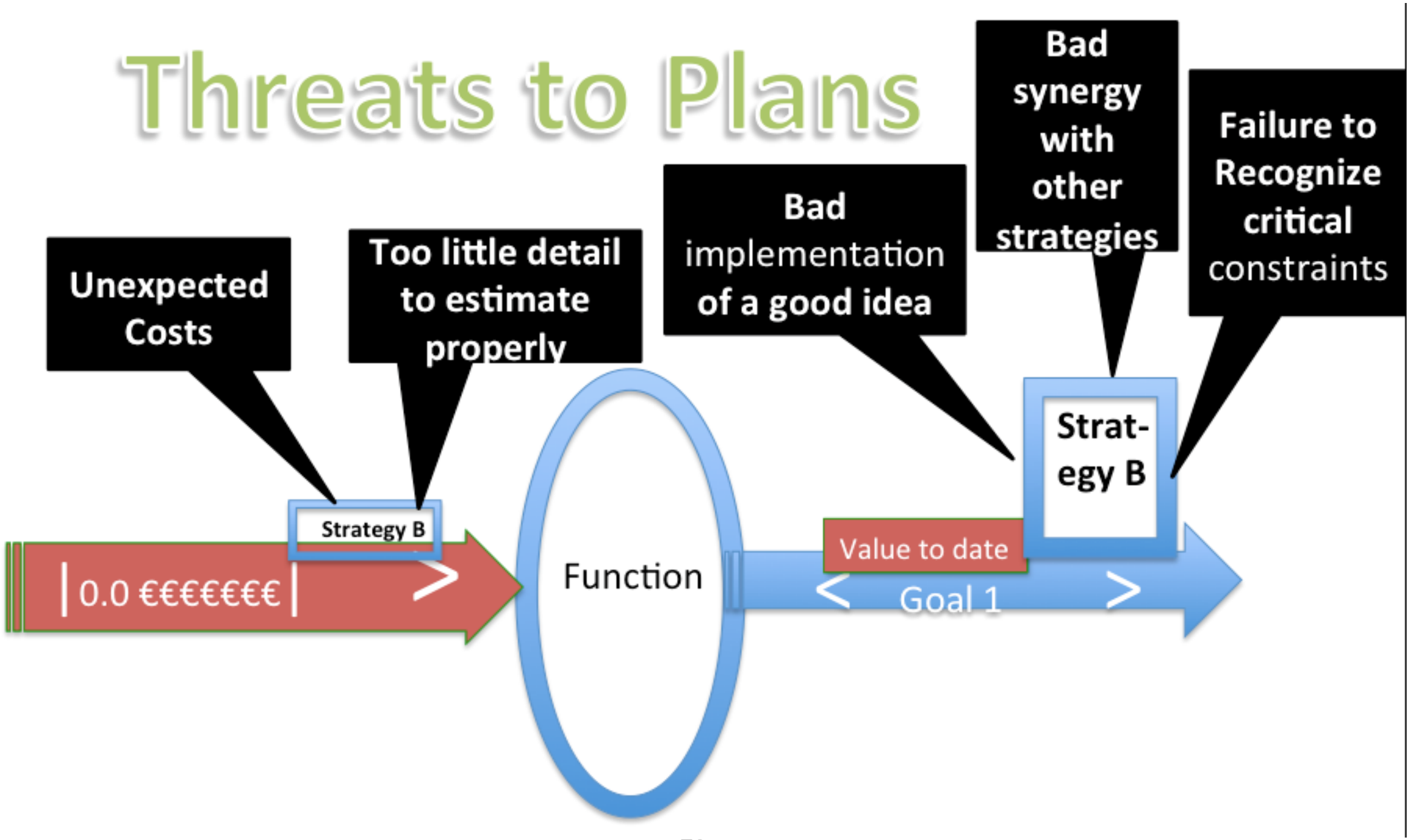


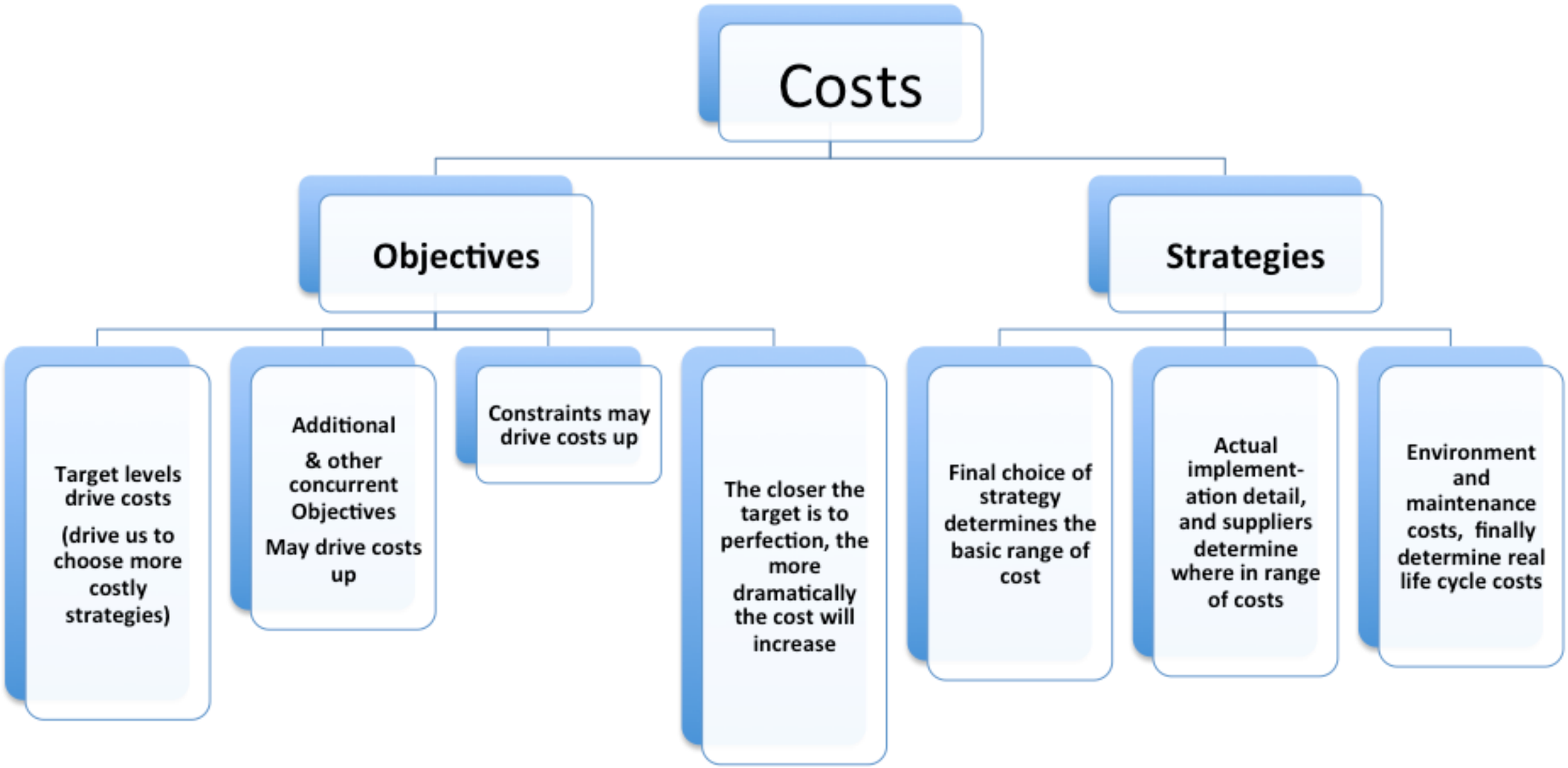
Figure 7.2 The risk environment. Threats may turn into real attacks, which get confronted with our planned mitigation. If the mitigation plan is unsuccessful, damage results, of various kinds. Notice that 'built-in to the system mitigation' is one type of mitigation. We can also mitigate risks at earlier stages, such as in planning, contracting, building trials, tests etc.

Various Risks to Plans

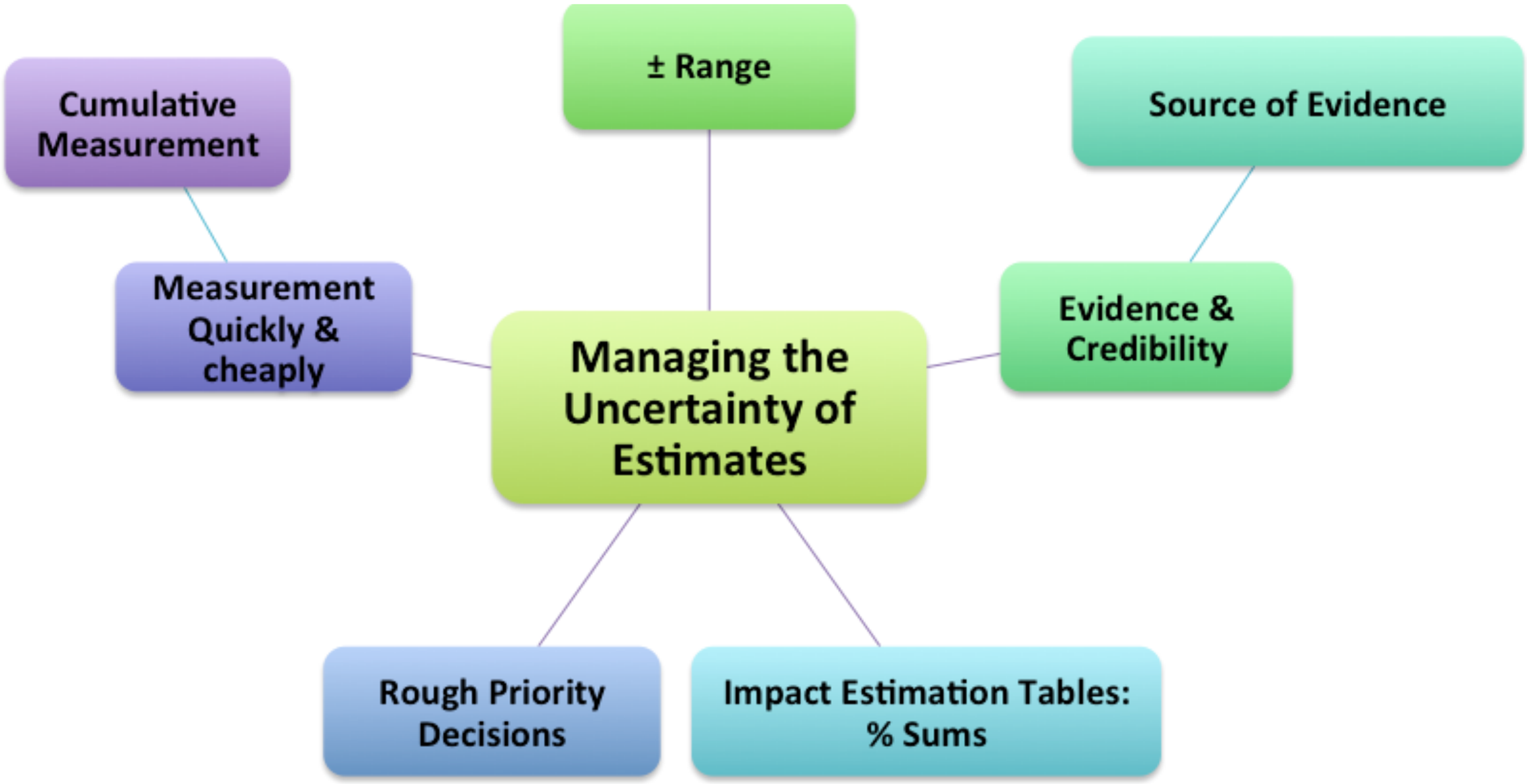


Design Strategy Risks

Cost Risks


Costs
Objectives
Strategies
Target levels drive costs
(drive us to choose more costly strategies)
Additional
& other concurrent Objectives
May drive costs up
Constraints may drive costs up
The closer the target is to perfection, the more dramatically the cost will increase
Final choice of strategy determines the basic range of cost
Actual implement-ation detail, and suppliers determine where in range of costs
Environment and maintenance costs, finally determine real life cycle costs

Risk Tools in Impact Estimation



12:00 TO 13:00
Measuring the Effects of your Designs on the Product Values and Costs: How to understand your design impacts, and design options.

- VALUE DECISION TABLES (VDT) = Impact Estimation

- Estimation, ± uncertainty, evidence, source,

- Value Budget: Value Result

- Differential Analysis Each Step

- Architecture Change if Necessary

- Dynamic Prioritization

VALUE DECISION TABLES (VDT) = Impact Estimation

- **if you believe in a design,**
- **then you should put your *mind* where your *heart* is,**
- **and estimate how good it is**
  - **for a well-defined purpose**

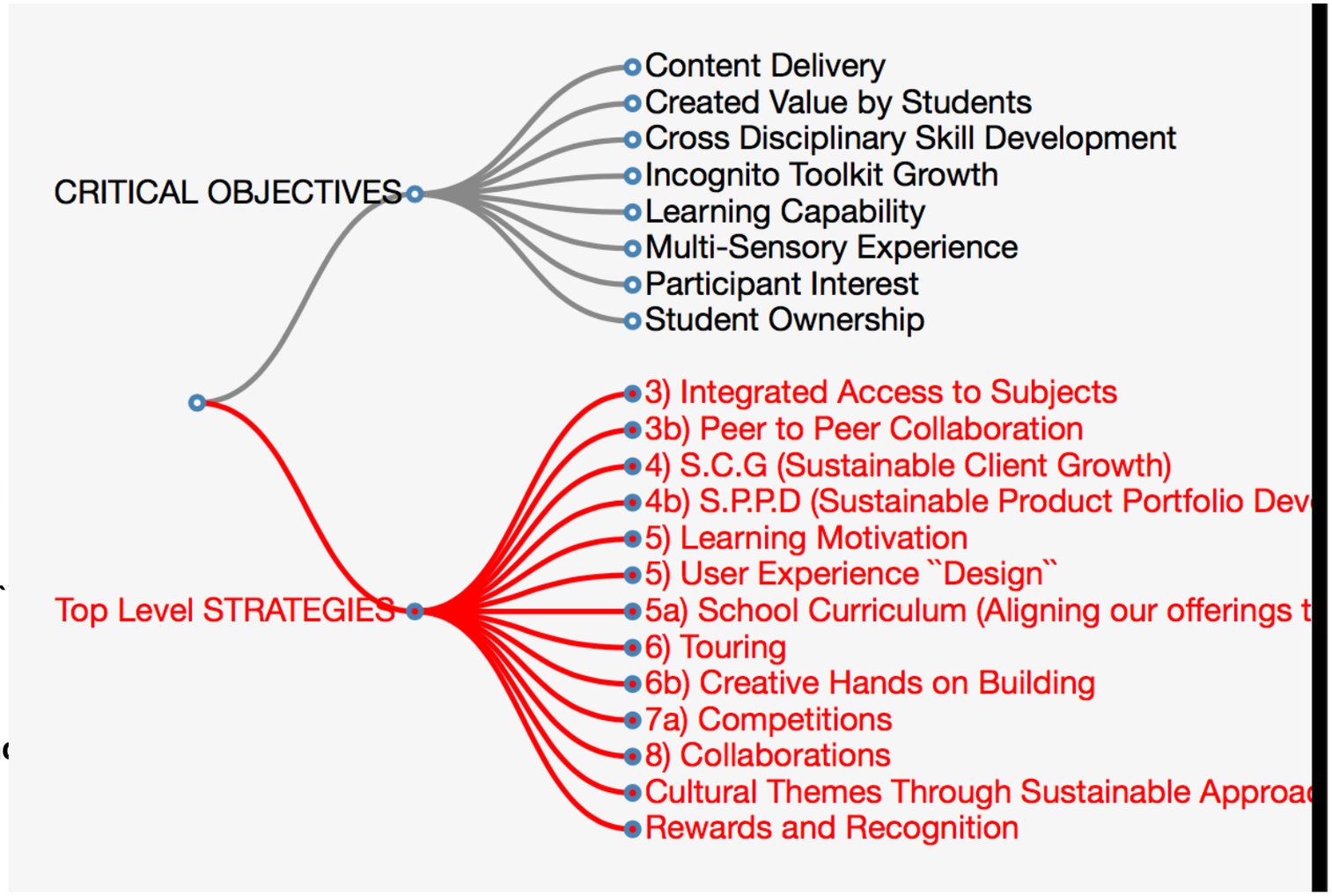
	Facebook	Profiler	Umantis BM	Total Impacts on Objectives
Attract Talents 271 -> 700	70% ± 10%	0% ±10%	50% ±5%	120% ±25%
Win Talents 53 -> 100	30% ± 20%	50% ± 10%	30% ± 10%	110% ±40%
Perfect Match 25% -> 75%	10% ± 10%	30% ± 10%	30% ± 10%	70% ±30%
Total Impact of Solutions	110% ± 40%	80% ± 30%	110% ± 25%	

- TSG 140519

Early Experience of the ‘Needs and Means’ Tool

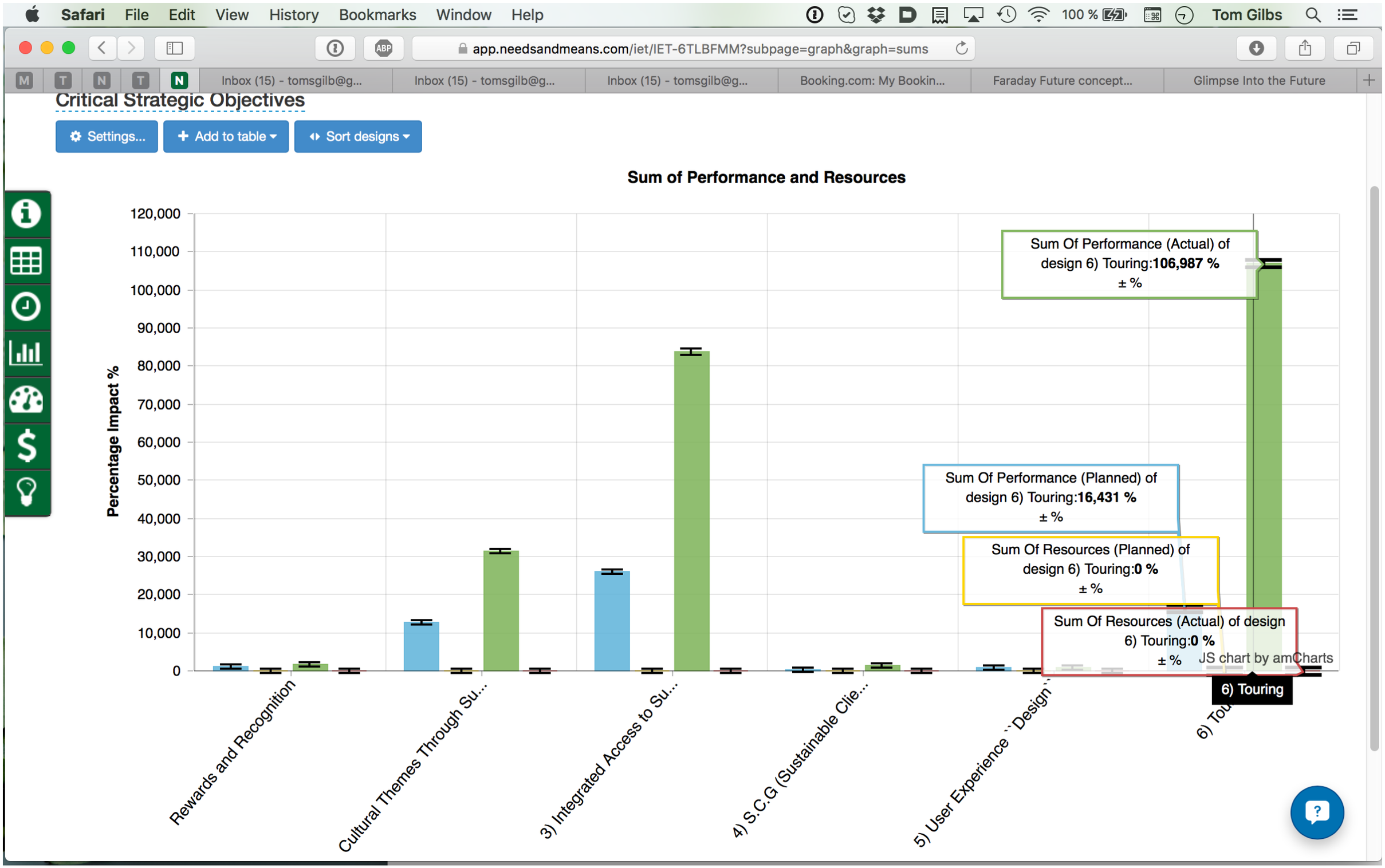
- Email January 11 2016

- **Double thumbs up for ‘needs and means’ (tool).**
- **I think every business, project, planning activity should use it!**
- **Time saver,**
- **and for me its amazing how you get the bigger picture instantly because it offers a real practical measure, unlike the usual hypothetical based tools that offer no measuring tool in addition.**
- **I mentioned to you the other day that it has ``unexpectedly` automatically shaped job descriptions for incoming staff with realistic deadlines.**
- **So I’m working hard to finish the finance projections bit and we can see what the effects are.**
- **I am honoured to have my project be the first real case study on N&M**



- gottfriedosei.ofei@gmail.com,
- **STARTUP ENTREPRENEUR, OSLO**
- **Incognito**

Visual Comparison of Strategies



IMPACT ESTIMATION TABLES FOR OVERVIEW OF ALL STRATEGIES, ARCHITECTURE IN RELATION TO OBJECTIVES, CONSTRAINTS AND RISKS

quantify the relationship between technology and business

(radically improve communication with your clients and managers)

Niels Malotaux

Impact Estimation principle

How much % of what we want to achieve do we achieve by this solution

Possible solutions to achieve it

Could we get all, within the budgets of time and cost ?

At what cost ?

	Design Idea #1	Design Idea #2	Design Idea #3	Total Impact
What to achieve: Objectives	Impact on Objective	Impact on Objective	Impact on Objective	Sum of Impacts on Objectives
Cost to achieve it: Resources Time Money	Impact on Resources	Impact on Resources	Impact on Resources	Sum of Impact on Resources
Return on Investment: Benefits to Cost Ratio	Benefits / Cost	Benefits / Cost	Benefits / Cost	

From Scales to Solutions

	Solution 1	Solution 2	Solution n	Total Impacts
Objective	Impact on Objective	Impact on Objective	Impact on Objective	Total Impact on Objective
Resources	Impact on Budget	Impact on Budget	Impact on Budget	Total Impact on Budget
Benefits-to-Cost Ratio	Ratio	Ratio	Ratio	

Courtesy Rolf Goetz

Impacts on Objectives

	Facebook	Profiler	Umantis BM	Total Impacts on Objectives
Attract Talents 271 -> 700	**70%** *± 10%*	**0%** *±10%*	**50%** *±5%*	**120%** *±25%*
Win Talents 53 -> 100	**30%** *± 20%*	**50%** *± 10%*	**30%** *± 10%*	**110%** *±40%*
Perfect Match 25% -> 75%	**10%** *± 10%*	**30%** *± 10%*	**30%** *± 10%*	**70%** *±30%*
Total Impact of Solutions	**110%** *± 40%*	**80%** *± 30%*	**110%** *± 25%*	

Courtesy Rolf Goetz

UNDERSTANDING DATA ENGINEERING
Design by estimating value effects and costs

ValPlan | Dashboard | Canvas | Tables | More... | Create | Data Enginee...

Data Engineering Demo For Talk 6 May 2019 Oslo / Value Decision Tables / SIMPLE IMPACT TABLE

SIMPLE IMPACT TABLE

From Level: *Level?* **To Level:** *Level?*

Settings... | Add | Sort | Duplicate... | Undo... | Δ: INC | Click to change tag of Use Safety & Security Standards

Requirements		Use Safety & ...	Use World Safety ...
Data Accessibility Status: **90** → Wish: **95** % success ... % success in accessing [Data Element ... [Data Element Types = **{Safety ...]** 03 May 2021	Δ: Δ%:	**3** **60** % 60%	**2** **40** % 40%
Sum Of Values:	Σ%:	**60** %	**40** %
Data Acquisition Costs Status: **0** → Budget: **100** € initial ... € initial cost of [Acquiring Data] o... [Acquiring Data = **Automatic Ac...]** 04 May 2021	Δ: Δ%:	**10** **10** % 10%	**42** **42** % 42%
Sum Of Development Resources:	Σ%:	**10** %	**42** %
Value To Cost:		**6.00**	**1.00**

Adding 1 Value and 1 Design

Untitled

SIMPLE IMPACT TABLE

From Level: *Level?* **To Level:** *Level?*

Settings... | + Add | Sort | Duplicate... | Undo... | Δ: INCREMENTAL | Help me!

Requirements		Use Safety & ...	Use World Safety ...	Deep Database Qua...
Data Accessibility Status: **90** → Wish: **95** % success ... % success in accessing [Data Element ... [Data Element Types = **{Safety ...]** 03 May 2021	Δ: Δ%:	3 60 % 60%	2 40 % 40%	???? 0 % ????
Data Correctness Status: **50** → Stretch: **95** % of [Data... % of [Data Element Types] which [Data... [Data Element Types = **All** 04 May 2026	Δ: Δ%:	0 0 % 0%	5 11 % 11%	25 56 % 56%
Sum Of Values:	Σ%:	60 %	51 %	56 %
Data Acquisition Costs Status: **0** → Budget: **100** € initial ... € initial cost of [Acquiring Data] o... [Acquiring Data = **Automatic Ac...]** 04 May 2021	Δ: Δ%:	10 10 % 10%	42 42 % 42%	30 30 % 30%
Sum Of Development Resources:	Σ%:	10 %	42 %	30 %
Value To Cost:		6.00	1.20	1.90

Estimation, ± uncertainty, evidence, source,

- **Some estimates are better than others**
  - **we need to systematically capture data about why some estimates are better**
  - **to motivate people to give good estimates**
  - **to make quality control and auditing of decisions possible**
  - **so that management can know the level of risk they are taking if they say ‘GO FOR IT’**

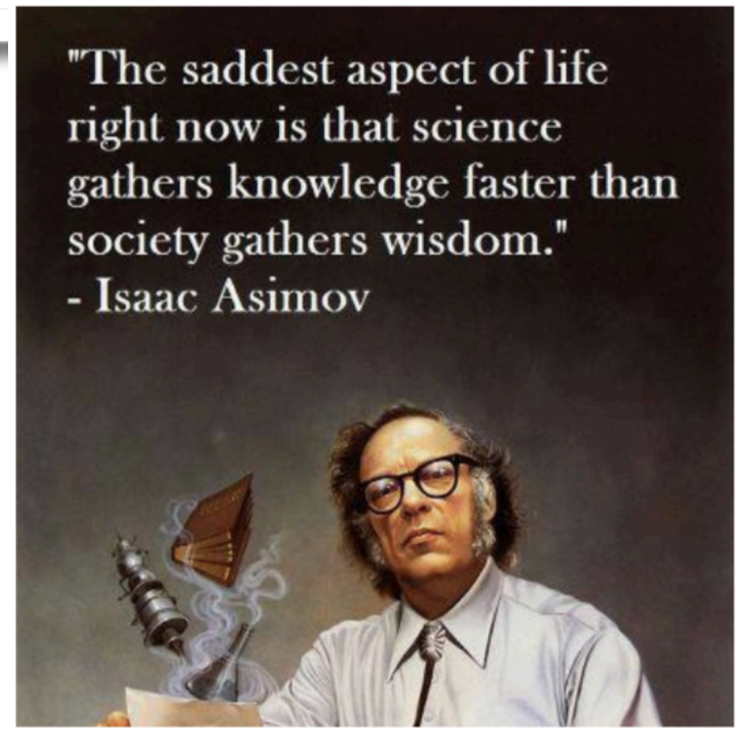
“I believe in evidence.

I believe in observation, measurement, and reasoning, confirmed by independent observers.

I'll believe anything, no matter how wild and ridiculous, if there is evidence for it.

The wilder and more ridiculous something is, however, the firmer and more solid the evidence will have to be.”

Isaac Asimov (2 Jan 1920 - 6 Apr 1992).



Explaining why you estimated an impact

Untitled

Settings... | + Add | Sort | Duplicate... | Undo... | Δ: INCREMENTAL | Help me!

Requirements		Use Safety & ...	Use World Safety ...	Deep Data
Data Accessibility Status: **90** → Wish: **95** % success ... % success in accessing [Data Element ... [Data Element Types = **{Safety ...]** 03 May 2021	Δ: Δ%:	3 **60** % 60%	2 **40** % 40%	???? **0** % ????
Data Correctness Status: **50** → Stretch: **95** % of [Data... % of [Data Element Types] which [Data... [Data Element Types = **All** 04 May 2026	Δ: Δ%:	0 **0** % 0%	5 **11** % 11%	25 **56** % 56%
Sum Of Values:	Σ%:	**60** %	**51** %	**56** %
Data Acquisition Costs Status: **0** → Budget: **100** € initial ... € initial cost of [Acquiring Data] o... [Acquiring Data = **Automatic Ac...]** 04 May 2021	Δ: Δ%:	10 **10** % 10%	42 **42** % 42%	30 **30** % 30%
Sum Of Development Resources:	Σ%:	**10** %	**42** %	**30** %
Value To Cost:		6.00	1.20	1.90

Comments:

Add Comment...

Hide Sideba

−Select Impact Target

Tag. Cost Impact:

Estimate:

Δ | 30 | ± | 20

Actual:

Δ | scale value | ± | 0

Credibility:

0.1

We know it has been done somewhere

Evidence:

Cost estimates are based on a very rough idea of what is involved in deep database diagnosis. The real costs vary from trivial to very expensive. Depending on what you program, and the degree of everyday application, and such things as accessing other databases, or using AI. However the good news is that this can be done in small steps where we can do high value things first, and measure effects and costs, and base decisions about scaling up on experience.

Source: by **tomgilb** - May 4th 2019, 20:08

Tom Gilb

Explaining why you estimated an impact

Untitled

Settings... | + Add | Sort | Duplicate... | Undo... | Δ: INCREMENTAL | Help me!

Requirements		Use Safety & ...	Use World Safety ...	Deep Data
Data Accessibility Status: **90** → Wish: **95** % success ... % success in accessing [Data Element ... [Data Element Types = **{Safety ...]** 03 May 2021	Δ: Δ%:	3 **60** % 60%	2 **40** % 40%	???? **0** % ????
Data Correctness Status: **50** → Stretch: **95** % of [Data... % of [Data Element Types] which [Data... [Data Element Types = **All** 04 May 2026	Δ: Δ%:	0 **0** % 0%	5 **11** % 11%	25 **56** % 56%
Sum Of Values:	Σ%:	**60** %	**51** %	**56** %
Data Acquisition Costs Status: **0** → Budget: **100** € initial ... € initial cost of [Acquiring Data] o... [Acquiring Data = **Automatic Ac...]** 04 May 2021	Δ: Δ%:	10 **10** % 10%	42 **42** % 42%	30 **30** % 30%
Sum Of Development Resources:	Σ%:	**10** %	**42** %	**30** %
Value To Cost:		6.00	1.20	1.90

Comments:

Add Comment...

Hide Sideba

−Select Impact Target

***Tag*. Cost Impact:**

Estimate:

Δ | 30 | ± | 20

Actual:

Δ | scale value | ± | 0

Credibility:

0 . 1

We know it has been done somewhere

Evidence:

Cost estimates are based on a very rough idea of what is involved in deep database diagnosis. The real costs vary from trivial to very expensive. Depending on what you program, and the degree of everyday application, and such things as accessing other databases, or using AI. However the good news is that this can be done in small steps where we can do high value things first, and measure effects and costs, and base decisions about scaling up on experience.

Source: by **tomgilb** - May 4th 2019, 20:08

Tom Gilb

Value Budget
versus
Value Result

Differential Analysis Each Step

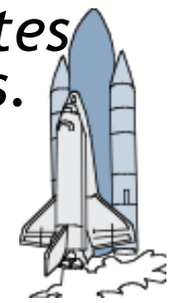
Architecture Change if Necessary

Dynamic Prioritization

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 bytes of program and data for ground and space processors in over a dozen projects.*



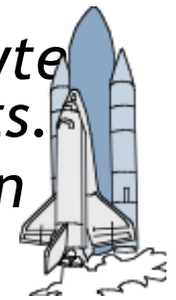
- ***- There were few late or overrun deliveries in that decade, and none at all in the past four years."***

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

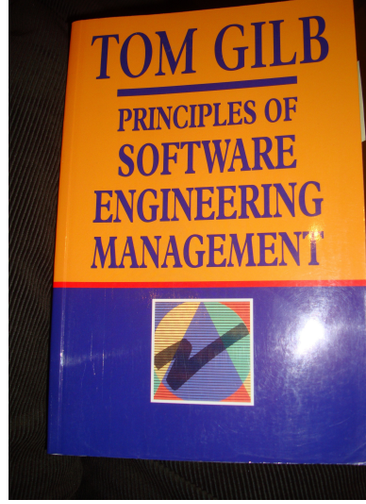
- *"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, f[illegible] about*
- *T[illegible]cts – cost overruns, late deliveries, unreliable and incomplete software*
- *Today [Ed. 1980!], management has learned to expect on-time, within budget deliveries of high-quality software. 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 softwa[illegible]million byte of pro[illegible]en projects.*
- *- Ther[illegible]e at all in the pa[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)

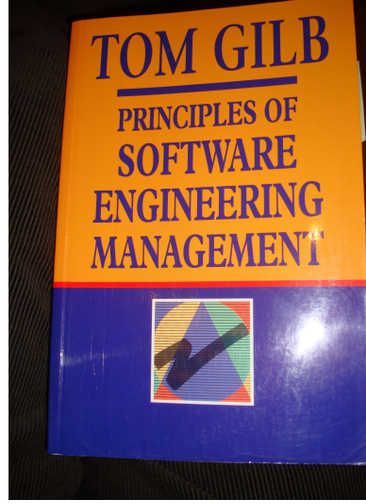
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability 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 by sacrificing 'planned capability.' When a s t of each increment can proceed concurrently with the

'Design is an iterativ

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

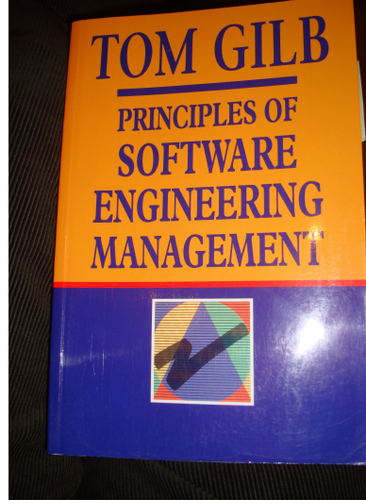
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability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

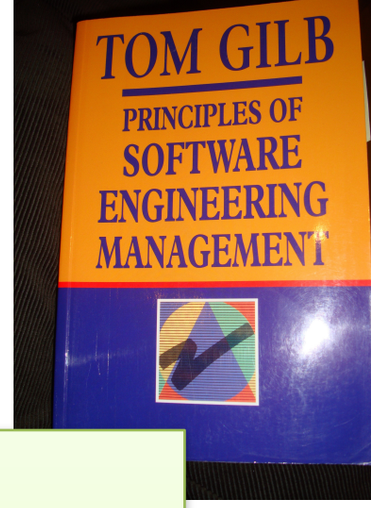
'Design is an iterative

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

END SLIDE FOR “Can we measure agility? Tools for practitioners”.

For the workshop session

23 May 2019, 10:00 to 13:00
Location: Polna 11, Warsaw

Tom Gilb
at Katowice
Masterclass, 2018
https://nowy.me/gilb/

Free Copy for
Workshop Participants only.
see details
in Presenter Notes of this slide
or Email Tom

https://www.gilb.com/store?tag=books
Link to Paid Books