The public link to the live stream on LinkedIn is: https://www.linkedin.com/feed/update/urn:li:ugcPost:6830473178213572609/

Agile Engineering

12 Aug 2021 21:00 CET
'UN-Common Sense'. Live event

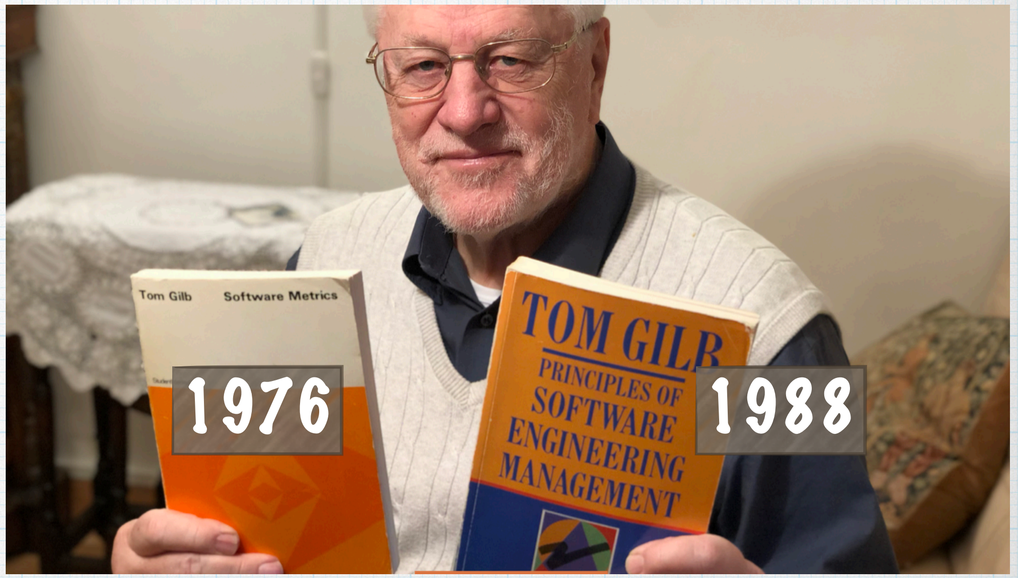
A 15-minute introduction to a discussion with

Joshua Barnes
Al Shalloway
Steve Tendon
& Tom Gilb

These Slides

https://tinyurl.com/AgileEngineering

Gilb Agile Library



Tom with Early Agile Engineering Books

Joshua Barnes <jbarnes@processmentors.com>
Al Shalloway <al.shalloway@pmi.org>,
Steve Tendon <steve.tendon@tameflow.com>
Tom@Gilb.com, Gilb.com, +47 92066705, @ImTomGilb,

Agenda: 1 minute each

Agile Engineering (AE): or 'Quantified Value(s) Agile'

1. Real Software *Engineering*, not just about coding, Not 'CodeFlow' like some agile methods.
2. Systems (level) Thinking: even for 'Programs' you need to integrate people, data, legal, hardware, cloudware and more.
3. Stakeholder Engineering: not merely 'customer and user': Stakeholder Stories, Stakeholder Xperience (SX not UX)
4. Simultaneous Multi-Values and Multi-Cost Requirements; as Agile *Efficiency* Measure, "Agility for Efficiency".
5. Multi-Value Flow Optimization, Multi-Constraint Consideration.
6. Dynamic Design to Efficiency (Value/Cost): The Architect in the Agile Loop (IBM Cleanroom, Evo)
7. 100X Defect-Prevention from Requirements; using Spec QC + Planguage; at Intel, in practice. (Terzakis)
8. Dynamic Stepwise Priority Computation, based on Efficiency and Constraints. Using Impact Estimation Tables.
9. AI, Web 3.0, Solid, Symantic Triples, Ontology& Digitization. = Very High-Tech Agile Future: Bye Bye Yellow Stickies.
10. Scale-Free Agile methods, as proven big time, at Intel, and other places. 11. Agile Engineering on a small scale (16 Norway Developers, 4 x 4 Teams, at Confirmit, capture international market with dramatic product quality increases)
12. "Principles of Agile Engineering" : Logical Common Sense.
13. *µ*Acts: + -> Tailored Practices -> Tailored Methods: BYOM Bring Your Own Method. 'Essence' and D.A.

https://www.dropbox.com/sh/wcl343wcopg2z7v/AAA2-Lk6mkaq1nWyT0TrLwjda?dl=0

Here are pdfs with free links to my Value Agile Stuff: "Gilb Agile Library"

Books, Papers, Slides, Video Interviews, Training Course Videos, Conference Presentations, and Historical Contributions and recognition

https://www.dropbox.com/sh/wcl343wcopg2z7v/AAA2-Lk6mkaq1nWyT0TrLwjda?dl=0

Gilb Agile Library

Agile Engineering (AE): or 'Quantified Value(s) Agile'

On step-results measurement, and retreat (='agile') possibility

'A complex system will be most successful
if it is implemented in small steps
and if each step has a clear measure of
successful achievement
as well as a "retreat" possibility to a previous
successful step upon failure.'

(p. 214), Software Metrics 1976-7.

See Slide Presenter Note for more detail, or ask me tom@Gilb.com

I was 35 years old when this was published. I had 16 years **'agile'** (or 'Evo', 'deliver value to stakeholder in small increments, measure value and retreat if necessary') experience by then (from 1960 Dobloug Case, and UiOslo Publisher/Admin (1968) etc).

Tough Questions for Agilistas

What methods can you use if: ....

1. If you have over 100 types of Stakeholders, each with 3 conflicting requirements?

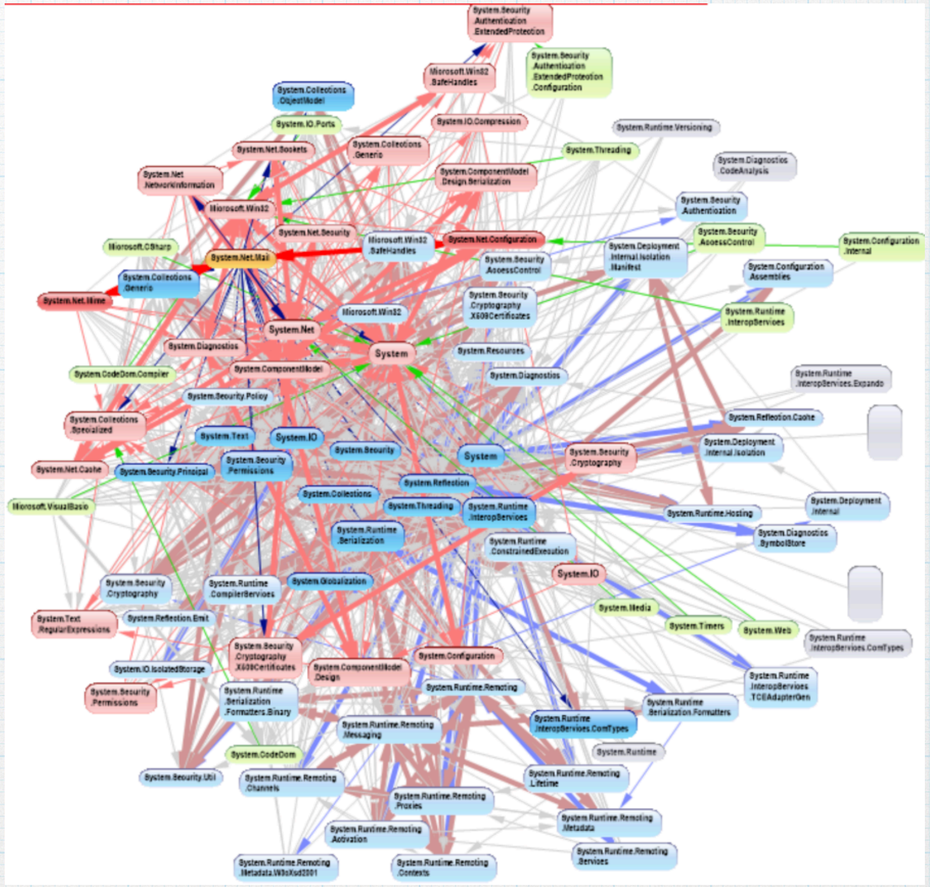
https://tinyurl.com/StakeholderBook

2. You build a large government system, and need measurable results for Ministers and Public, before next election in one year?

3. Your project uses 1,000 Software Engineers, and is 3 years late?

Hicom, Siemens, Case PoSEM, 1988, 13.7.4. 1984-5, Günther Rabb, Bernhard Falkenberg. Aslo CE book p. 314-6, "Using common sense"

Solved using Increments (Evo) and Quantified Quality (Software Metrics)

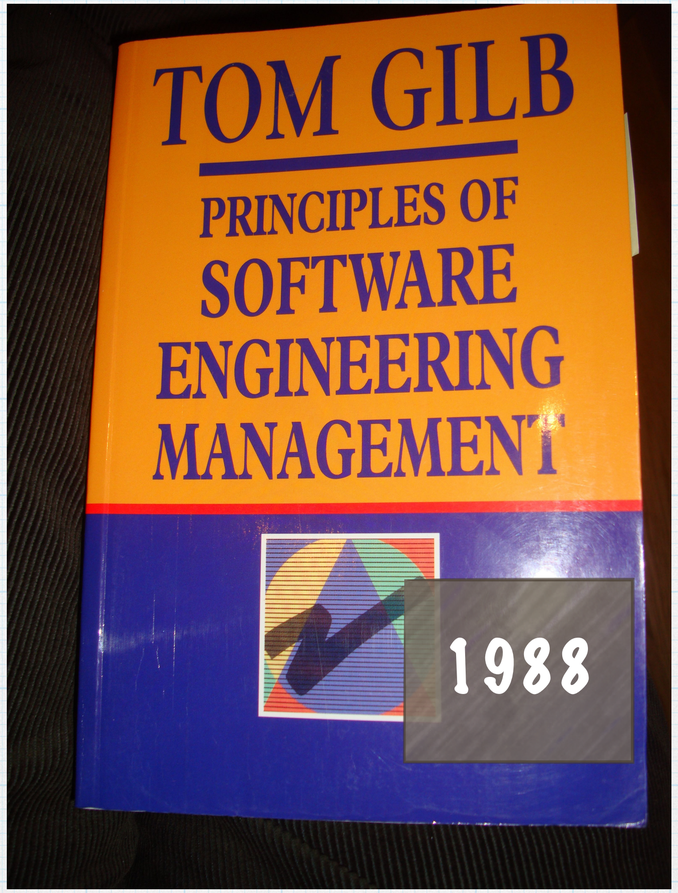


https://i.stack.imgur.com/GvrCA.png

http://concepts.gilb.com/dl876
20 Tough Technical Questions about requirements and designs for Keynote Intel April 2016

1. Real Software *Engineering*, not just about coding, Not 'CodeFlow' like some agile methods.

* Logicware
* Dataware
* Peopleware
* Hardware
* Softcrafters
* Other Stakeholders
  * Legalware (like GDPR EU)
  * Planware
  * Cultureware



Chapt 15 Deeper Perspectives on Evolutionary Delivery
www.gilb.com/dl561
144 Principles

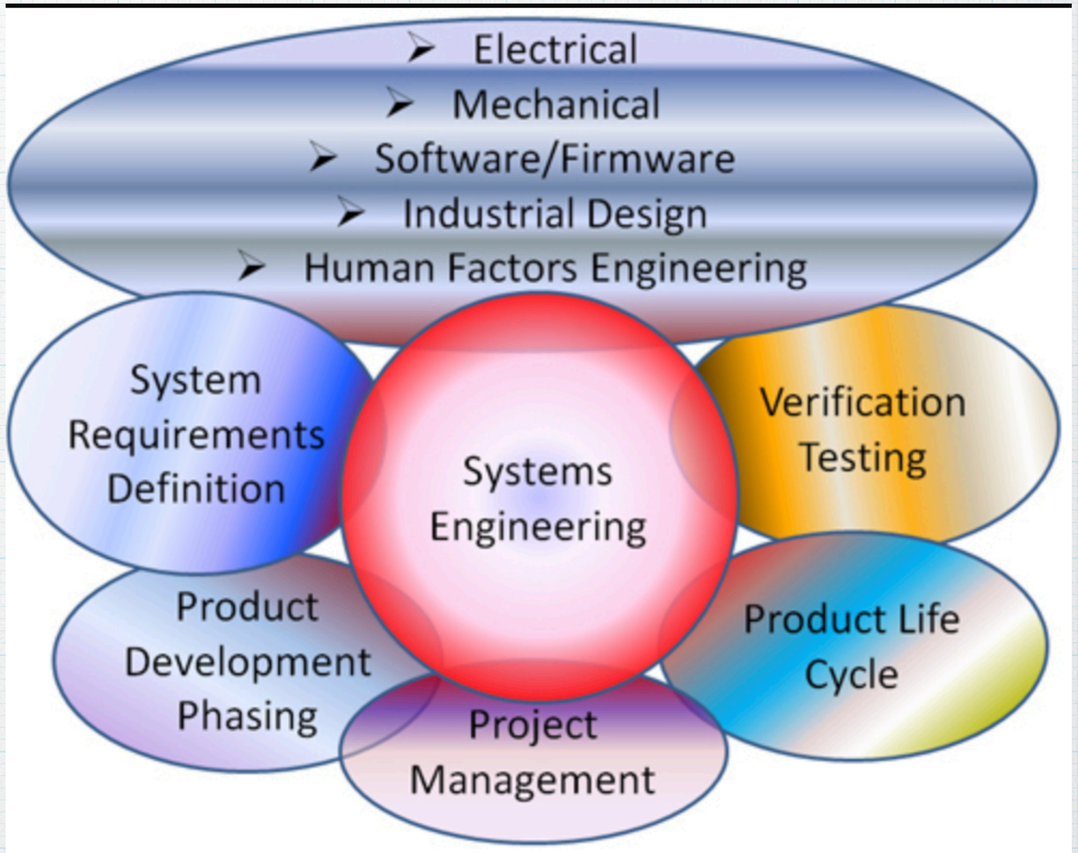
https://www.gilb.com/p/competitive-engineering
(free pdf)

10 Chapters in 10 Parts = 100 Tools

2. Systems (level) Thinking:
even for ‘Programs’ you need to integrate
p*eople, data, legal, hardware, cloudware* and more.

These Slides

https://tinyurl.com/AgileEngineering



http://www.aesaustin.com/images/systems-engineering-services_large.jpg

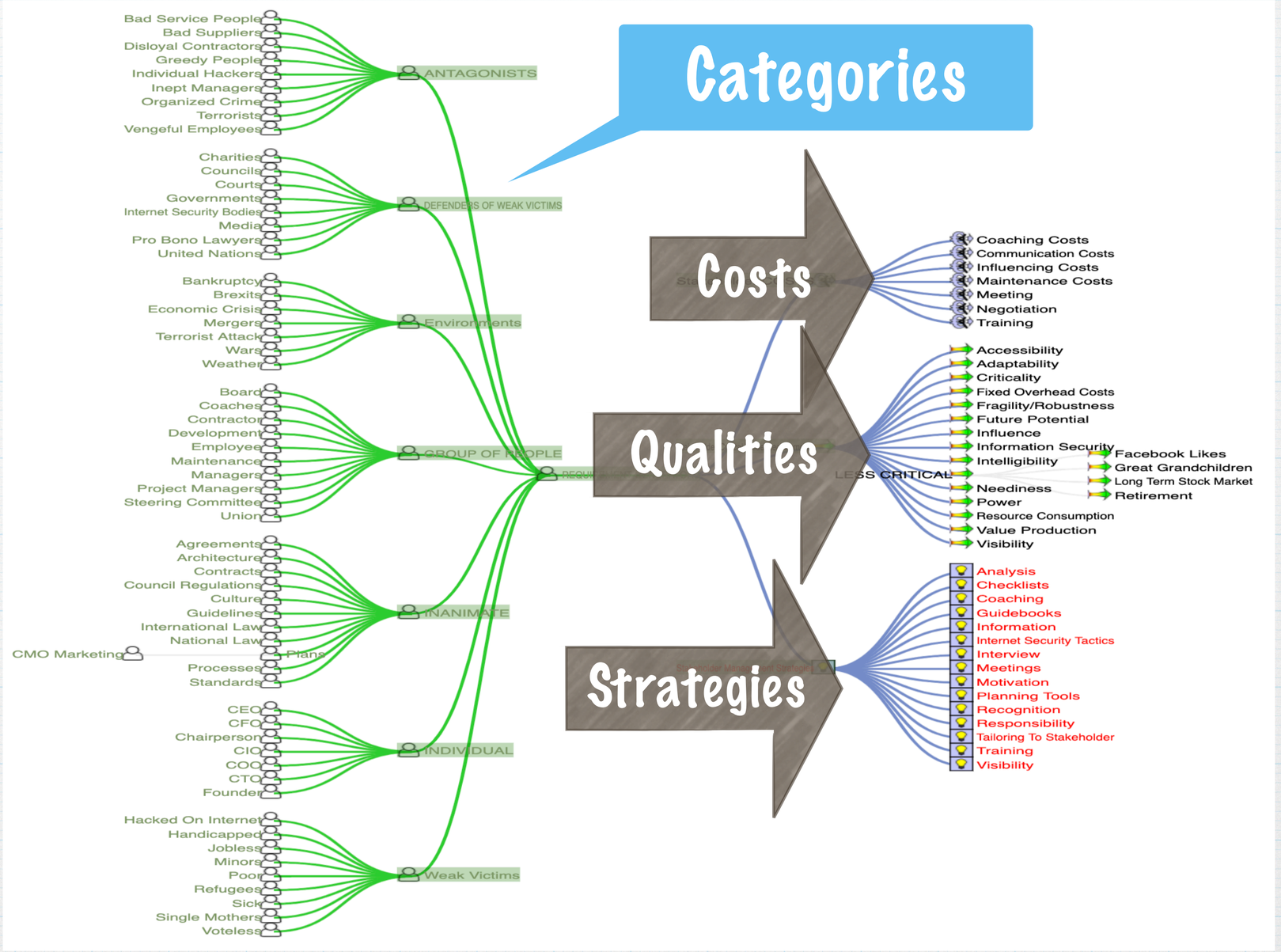
3. Stakeholder Engineering: not merely ‘customer and user’:

Stakeholder Stories, Stakeholder Xperience (SX not UX)

* When you scale up from small simple systems (OKR, User Stories, Few Teams)
* You need ‘engineering’ to keep track of the Stakeholder complexity and costs.

Stakeholder Engineering.
By Tom Gilb
Leanpub.com/StakeholderEngineering
Released 27 July 2021, Leanpub

1.7 A generic hierarchical stakeholder pattern, with detailed examples of categories.



Figured 1.7 B This is a top-level example of an overview of a useful set of interesting stakeholders, together with our objectives in managing them (arrows), our potential strategies, for better stakeholder management (lightbulbs), and the associated cost aspects of managing stakeholders.

4. Simultaneous Multi-Values and Multi-Cost Requirements; as Agile *Efficiency* Measure, "Agility for Efficiency".

* All critical stakeholder values and costs
* Must be considered
* Not just code production flow

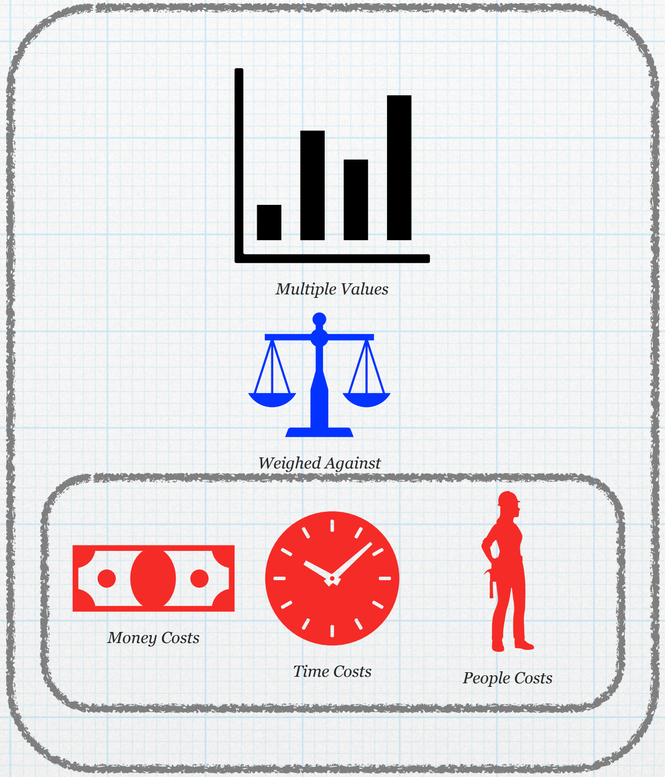


Figure 2.1 Multiple Value attributes of an architecture / Multiple cost attributes x Risks = Architecture Efficiency

X  = Arch Eff

Risks and Uncertainties

Video 3 Feb 2 2021, Chapter 2 Architecture Efficiency
https://www.youtube.com/watch?v=tL7-RTqNuNM&feature=youtu.be

5. Multi-Value Flow Optimization, Multi-Constraint Consideration.

Can 'your agile' really handle 10 objectives and 5 costs At the same time?

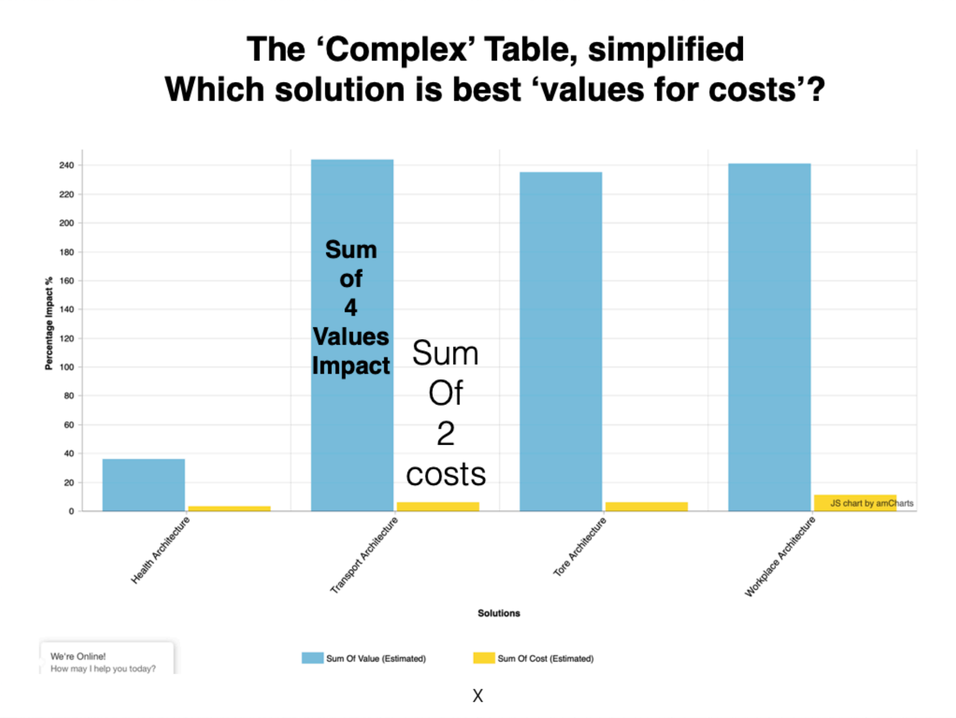


Figure 2.3 The table data on previous page (Fig 2.2) Summarized as a Bar Chart. Without any adjustment for Risks. It is a tight competition for 3 of the architecture options. The risk factors, and using numbers, will finally decide for the 4th on

Figure 2.2 A ValPlan Impact Estimation Table (IET) adding up the values for each 4 sub-architectures, and adding up the costs, and then modifying the V/C by 3 different types of risk and uncertainty (see Part 12 for details).

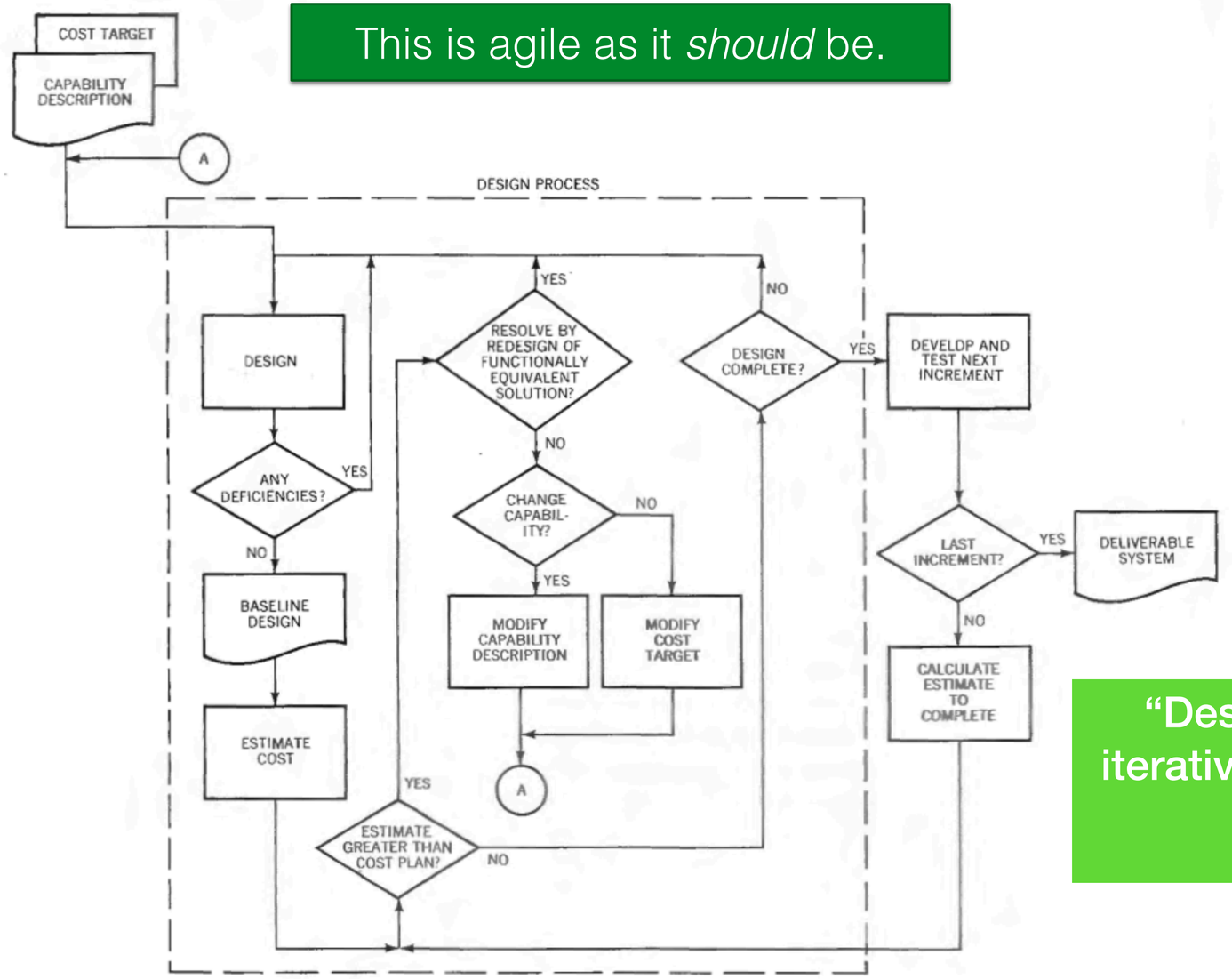
The actual risk numbers are not visible here (quite noisy) , but if we choose to display them, or drip down into cell, they are there to analyze or audit.

Systems Enterprise Architecture (SEA) BOOK
https://leanpub.com/SysEntArchBook
2021, $4.99 to $9.99 +VAT EU

6. Dynamic Design to Efficiency (Value/Cost): The Architect in the Agile Loop (IBM Cleanroom, Evo)

* Does your 'Enterprise Architect'. Haha :)
* Redesign things
* If necessary
* For better cost or quality
* At every 'sprint' ?
* And achieve on-time, under budget, high quality in defence and space software?

Figure 3 Design-to-cost



This is agile as it *should* be.

"Design is an iterative process"

Source: Quinnan, IBM SJ, page 473

http://trace.tennessee.edu/cgi/viewcontent.cgi?article=1004&context=utk_harlan

And PoSEM page 105, Figure 7.10

Value Agile book free

https://www.dropbox.com/sh/c2g7ib3z2g2uzfw/AAAypXlN0yA2WS4obwjDzZR3a?dl=0

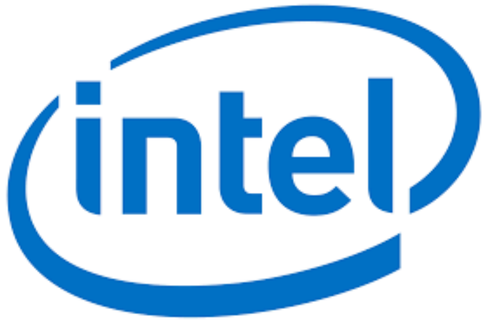
7.
>100X Defect-Prevention from Requirements input; using Spec QC + Planguage; at Intel, in practice.

* An engineering requirements language (Planguage) gives 10X better (clearer) requirements
  * (10.06 defects vs 100+)
* Then Spec QC (see CE book)
* Reduces defects by 50X more
  * 10 defects -> 0.2 DPP
* > 20,000 Intel Engineers trained in Planguage,
* In use at intel for over 20 Years

AGILE INSPECTIONS: Reviews by sampling and measuring defects. Extreme inspection and reviews based on objective and quantitative review methods.

http://www.gilb.com/dl239

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-

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

ort paper [1] that

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%***				

Figure 4.2.2.1 [12] TERZAKIS INTEL 2011 AND 2013. Practical industrial cases. SQC and Planguage
https://selab.fbk.eu/re11_download/industry/Terzakis.pdf

8. Dynamic Stepwise Priority Computation, based on Efficiency and Constraints. Using Impact Estimation Tables.

* Can you compute priority at every increment?
* Based on values, costs and risks?

Values

Possible increments

Priority Calculation

5.4.1 Priority evaluation of 4 sub-architectures, in relation to 5 Performance Requirements,and 2 Budgeted Resources.

Corona Virus Management Norway Total Architecture

From Level: Stakeholder **To Level:** Stakeholder

Settings... | Add | Help me!

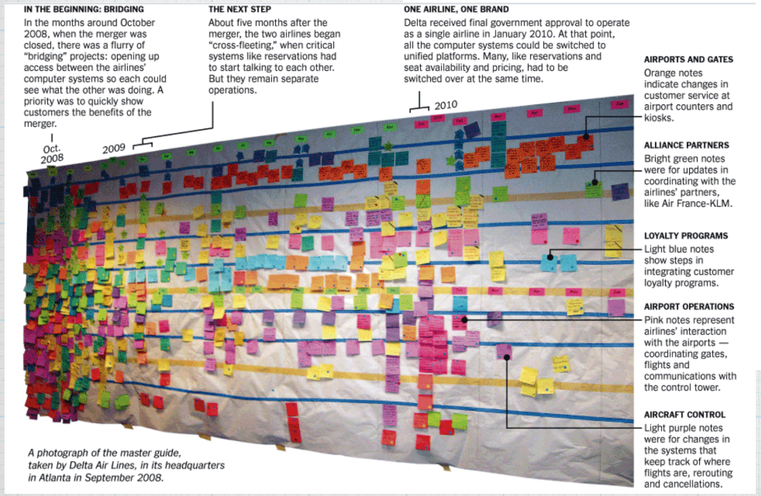
Requirements		Health Architecture	Transport Archite...	Tore Architecture	Workplace Archite...
Collect Information Status: 50 → Wish: 90 % [Relevan...	Δ:	5 13 %	20 50 %	25 63 %	30 75 %
Education Status: 42 → Wish: 95 % [Student...	Δ:	3 6 %	30 57 %	30 57 %	20 38 %
Get People Where They Need ... Status: 42 → Wish: 99 [Important...	Δ:	???? ????	30 53 %	2 4 %	???? ????
Healthy Employees Status: 70 → Wish: 99 [Work Acti...	Δ:	???? ????	10 34 %	20 69 %	25 86 %
Stay Healthy Status: 30 → Wish: 90 % [Capacit...	Δ:	10 17 %	30 50 %	25 42 %	25 42 %
Sum Of Values:	Σ%:	36 %	244 %	235 %	241 %
Days To Implement Status: 0 → Budget: 1k Days Neede...	Δ:	30 3 %	10 1 %	15 2 %	10 1 %
Capital Cost In Million NOK Status: 0 → Budget: 1k Million No...	Δ:	0 0 %	50 5 %	40 4 %	100 10 %
Sum Of Development Resources:	Σ%:	3 %	6 %	6 %	11 %
Value To Cost:		12.00	40.70	39.20	21.90
Ratio (Worst Case)		7.30	21.60	25.80	18.80
Ratio (Cred. - adjusted)		0.70	13.50	14.50	5.00
Ratio (Worst Case Cred. - adjusted)		2.80	22.50	113.40	292.00

9. AI, Web 3.0, Solid, Symantic Triples, Ontology& Digitization. = Very High-Tech Agile Future: Bye Bye Yellow Stickies.

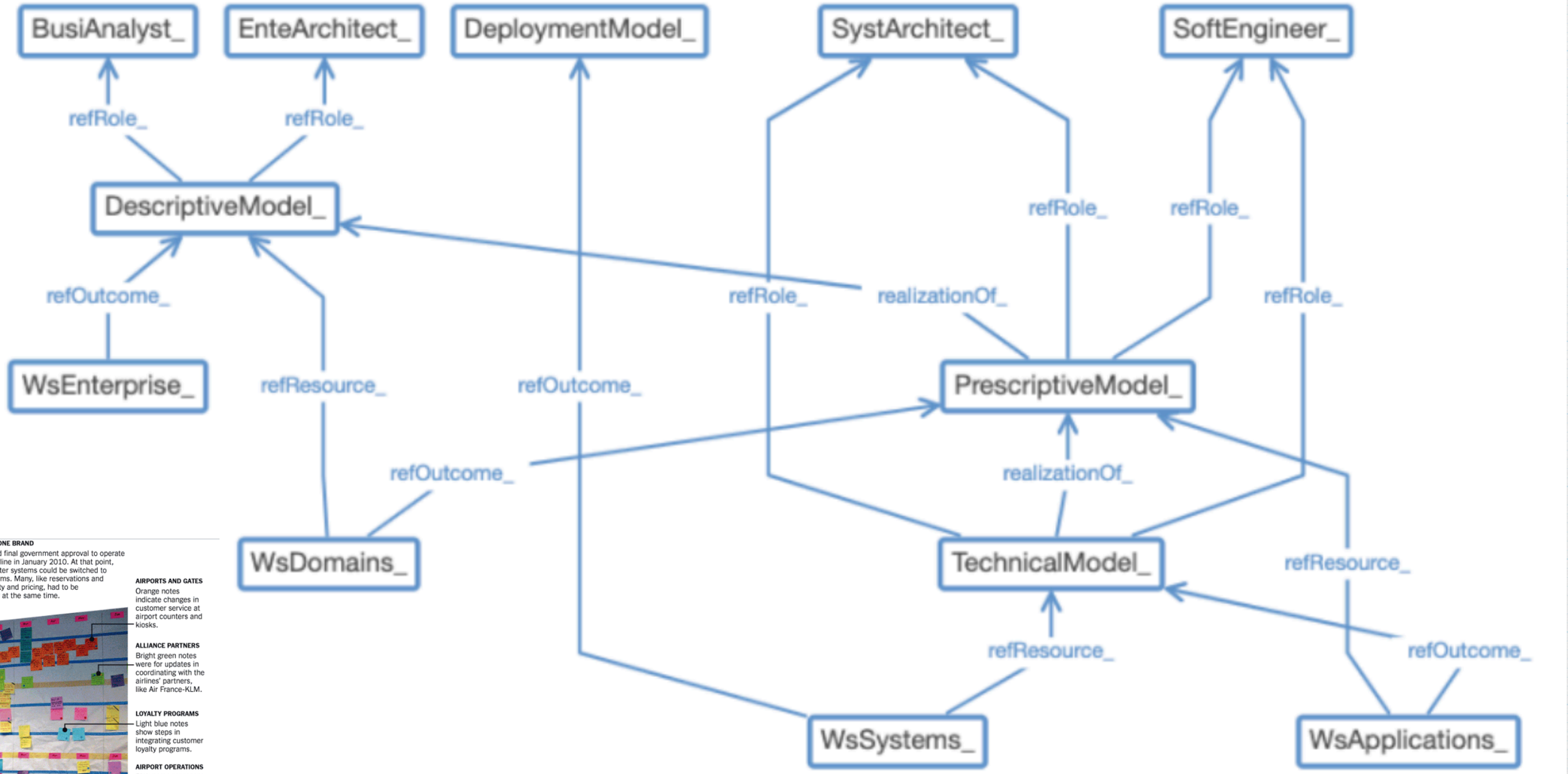
Digitize system relations

Yellow Stickies are for small, short term, non-critical systems

'Kid Stuff'



A photograph of the master guide, taken by Delta Air Lines, in its headquarters in Atlanta in September 2008.



https://caminao.blog/ea-in-bowls/ courtesy Rémy Fannader

10. Scale-Free Agile methods, as proven big time, at Intel, and other places.

IEEE Paper: https://www.computer.org/csdl/magazine/so/2021/04/09461035/1uCdHyJJMAw

*Are your Agile methods Scale-Free?

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@construx.com

Gilb. "SCALE-FREE:
Practical Scaling Methods for Industrial Systems Engineering"
lecture slides, http://concepts.gilb.com/dl892
2016, Considerable citation of Intel experience with Planguage method, by Erik Simmons.

11. Agile Engineering on a small scale

(13 Developers + 3 Testers, Norway, 4 x 4 Teams, at Confirmit, capture international market with dramatic product quality increases)

- Parallel Multiple -Quality Engineering using Evo and Planguage
- Real quick, (weekly increments) quarterly results released to international market
- 1 Day training for entire org. (abt. 68)
- All 16 devs had engineering degrees. NTNU

EVO Plan Confirmit 8.5 in **Evo Step Impact *Measurement***

4 product areas were attacked in all: **25 USER Qualities** concurrently one quarter of a year.

Total development staff = 13

Impact Estimation Table: Reportal codename "Hyggen"

9

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportRep (min		es
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewRepo (seco		s)
1,0	14,0	100,0	15		1
			Development resources		
203,0			0		91

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		7	7
			Runtime.ResourceUsage.CPU		
3,0	35	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0	800	100,0	800	0	0
			Runtime.Concurrency (number of users)		
350	110	146,7	150	500	1000
64			Development resources		
			0		84

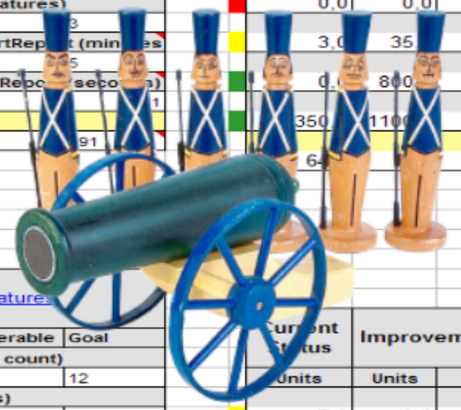
8

3

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

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

3



August 25, 2014

Case slides: http://concepts.gilb.com/dl33

12. "Principles of Agile Engineering" : <u>Logical Common Sense.</u>

1. quantify critical improvement objectives
2. estimate multiple impacts of strategies
3. reject polluted specifications
4. plan for 1 week, before starting value delivery
5. deliver some value every week or 2% of time
6. measure real value and costs and learn fast
7. contract for value delivery, not for work done
8. operate at the systems level, not the 'code' level
9. let critical stakeholders decide your critical requirements
10. Keep it simple: top 10 Objectives quantified is master, everything else is a servant

Value Agile 2020
BOOK: leanpub.com/ValueAgile
SLIDES: http://concepts.gilb.com/dl974
VIDEO https://www.gilb.com/blog/Agile-Tools-for-Value-Delivery-by-Tom-Gilb

13. *µ*Acts: + -> Tailored ***Practices*** -> Tailored ***Methods***:

BYOM Bring Your Own Method. ‘Essence' and D.A.

SimPlan

Simple Planning Language

Micro-Practices (µActs)

Small acts that can make a big difference to your plans

Figure: Cover.
The Large-sized book on Planguage [1]

SimPlan book 2021 in progress (In Pages)

https://www.dropbox.com/sh/3h6iwlz29vi3tvm/AACuH_ufpP9IZF9NrnmA0s31a?dl=0

14

‘Stakeholder Engineering: *The ‘Stakeholder Systems Engineering’ Discipline.*

https://tinyurl.com/StakeholderBook

July 2021
Free Download for YOU. Above URL.

Do Share with a friend, but not on Internet (Use then *Leanpub.com/StakeholderEngineering*).

Note it is for sale now too *(some people appreciate things they have to pay for)*, so this is the **public** reference.

Stakeholder Engineering.
By Tom Gilb
Leanpub.com/StakeholderEngineering
Released 27 July 2021, Leanpub
$5-$10, Big Free sample available here.

15
End of
Presentation

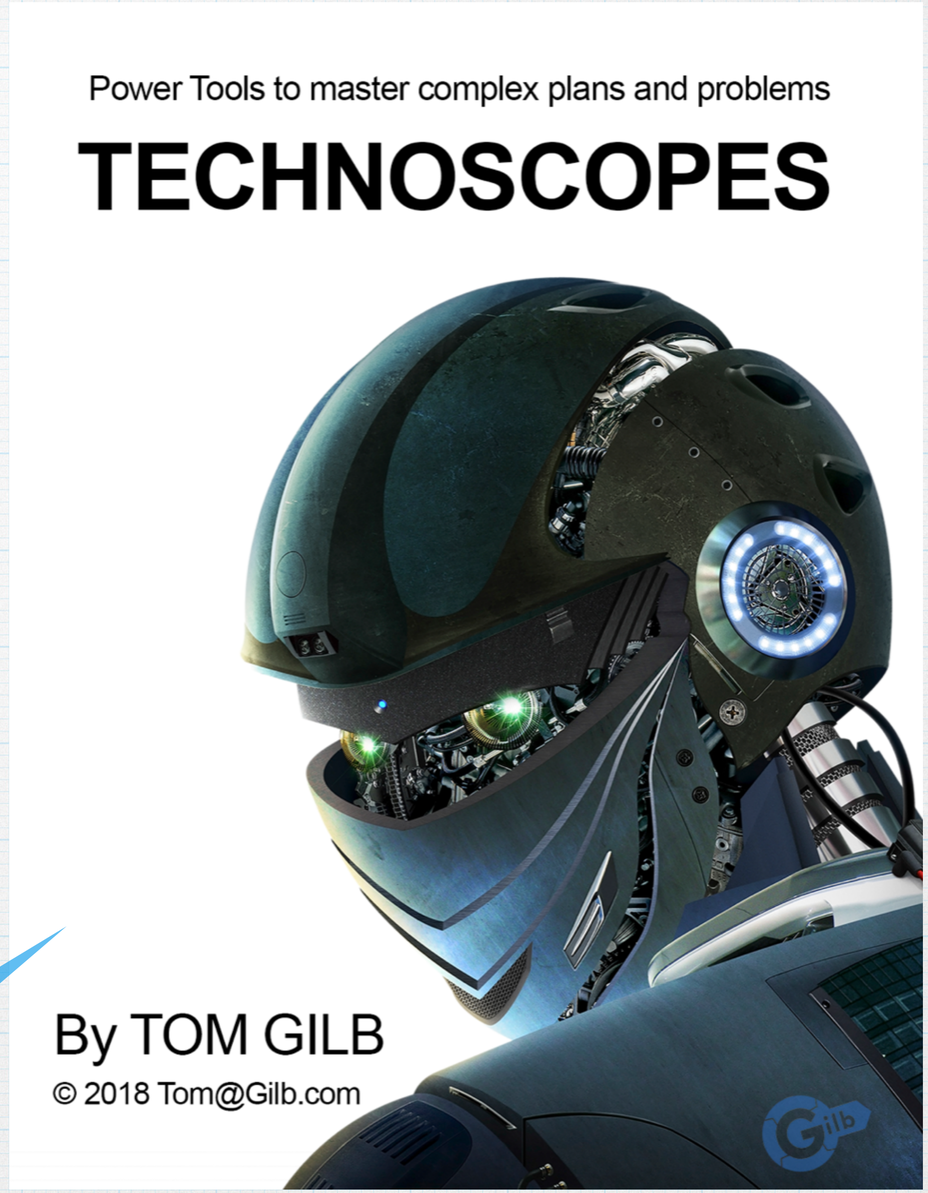
OK
Questions and discussion

Why isn't everybody doing 'Agile Engineering' ?

Why does 'conventional Agile'

fail to deliver, so often ?

100 Tools
2 Pages each



Technoscopes:
Tools for understanding complex projects
https://www.gilb.com/store/Pd4tqL8s
Price €14

Summary of Talk

Agile Engineering (AE): or 'Quantified Value(s) Agile'

1. Real Software *Engineering*, not just about coding, Not 'CodeFlow' like some agile methods.
2. Systems (level) Thinking: even for 'Programs' you need to integrate people, data, legal, hardware, cloudware and more.
3. Stakeholder Engineering: not merely 'customer and user': Stakeholder Stories, Stakeholder Xperience (SX not UX)
4. Simultaneous Multi-Values and Multi-Cost Requirements; as Agile *Efficiency* Measure, "Agility for Efficiency".
5. Multi-Value Flow Optimization, Multi-Constraint Consideration.
6. Dynamic Design to Efficiency (Value/Cost): The Architect in the Agile Loop (IBM Cleanroom, Evo)
7. 100X Defect-Prevention from Requirements; using Spec QC + Planguage; at Intel, in practice. (Terzakis)
8. Dynamic Stepwise Priority Computation, based on Efficiency and Constraints. Using Impact Estimation Tables.
9. AI, Web 3.0, Solid, Symantic Triples, Ontology& Digitization. = Very High-Tech Agile Future: Bye Bye Yellow Stickies.
10. Scale-Free Agile methods, as proven big time, at Intel, and other places. 11. Agile Engineering on a small scale (16 Norway Developers, 4 x 4 Teams, at Confirmit, capture international market with dramatic product quality increases)
12. "Principles of Agile Engineering" : Logical Common Sense.
13. *μ*Acts: + -> Tailored Practices -> Tailored Methods: BYOM Bring Your Own Method. 'Essence' and D.A.

Gilb Agile Library

OOPS !
Go back a slide

Here are pdfs with free links to my Value Agile Stuff: “Gilb Agile Library”

Books, Papers, Slides, Video Interviews, Training Course Videos, Conference Presentations, and Historical Contributions and historic recognition (I’m the Grandpa!).

https://www.dropbox.com/sh/wcl343wcopg2z7v/AAA2-Lk6mkaq1nWyT0TrLwjda?dl=0

Gilb Agile Library

These Slides

https://tinyurl.com/AgileEngineering