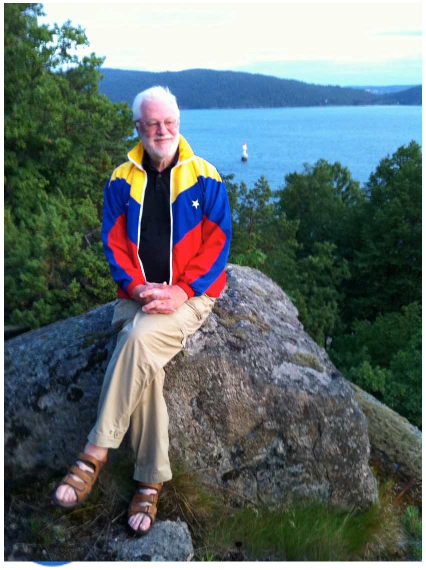
"What is drastically wrong with most software engineering modeling languages and approaches, and 10 necessary principles for a really good modeling language"

Tom Gilb, Norway

@ImTomGilb, Tom@Gilb.com. www.Gilb.com



45 minute lecture

Quality Days Conference, Vienna

10:45 to 11:30 , Wednesday January 15 2014

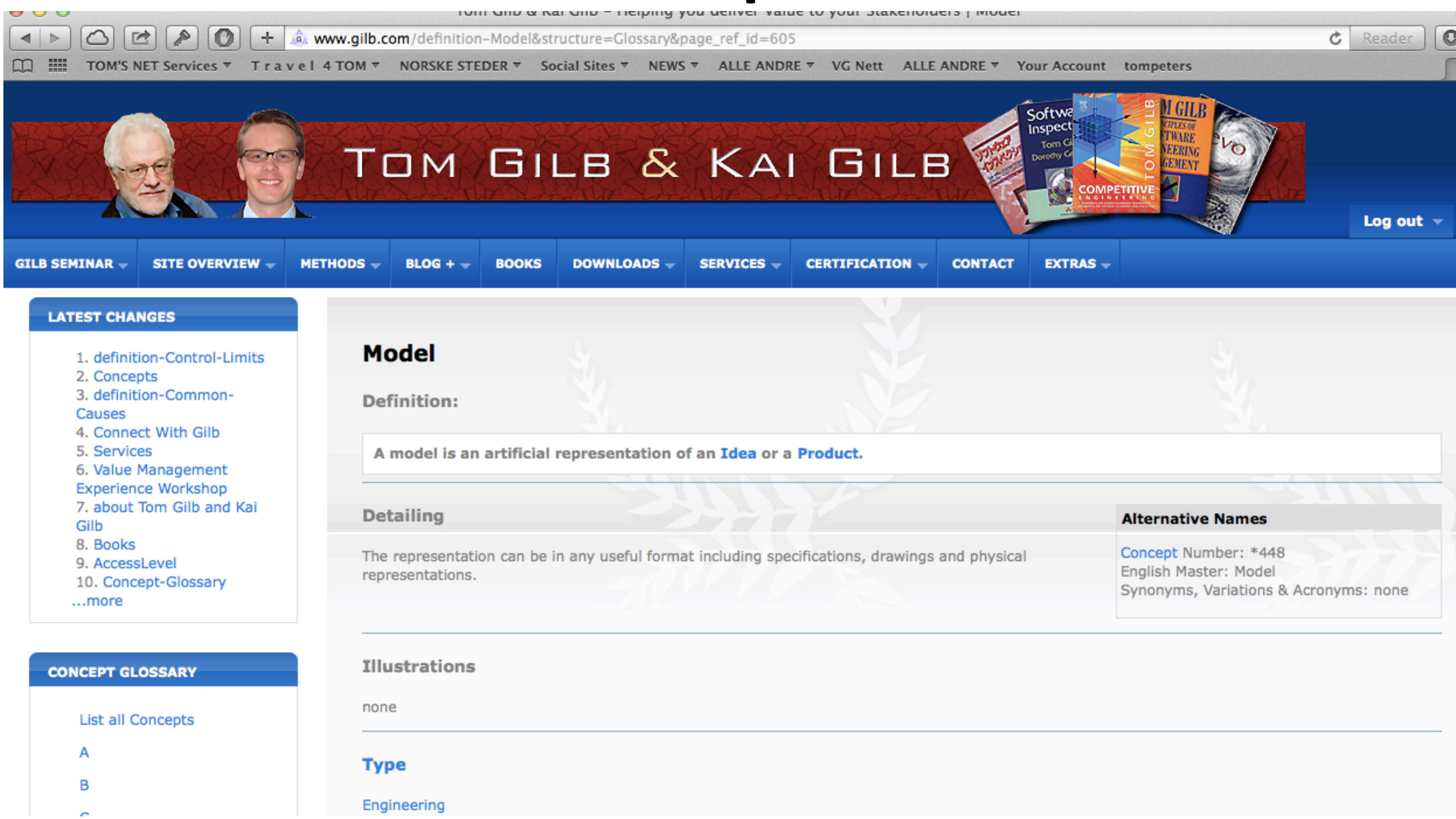
Link to Yesterdays workshop slides on Modelling

http://tinyurl.com/QWGILB

"How to Model Qualitative aspects of software *requirements*, software *architecture*, and *project progress* – quantitatively" (3.5 hours)

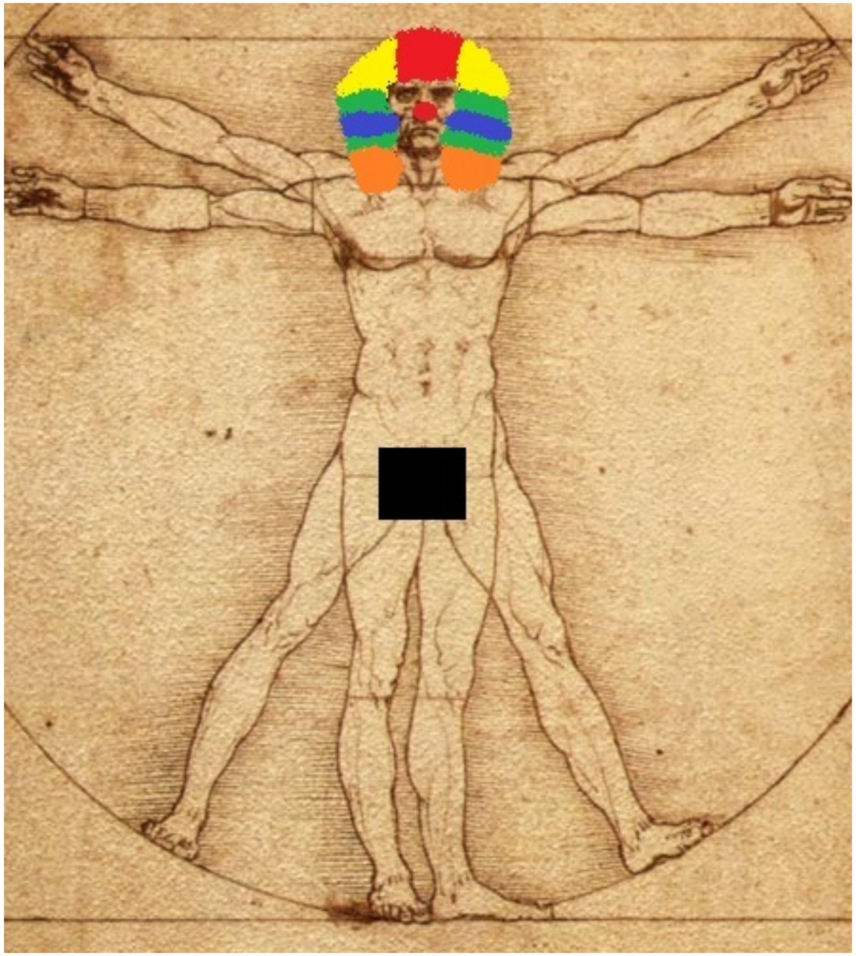
Concept Definitions

“A Model is an artificial representation of an idea or a product”

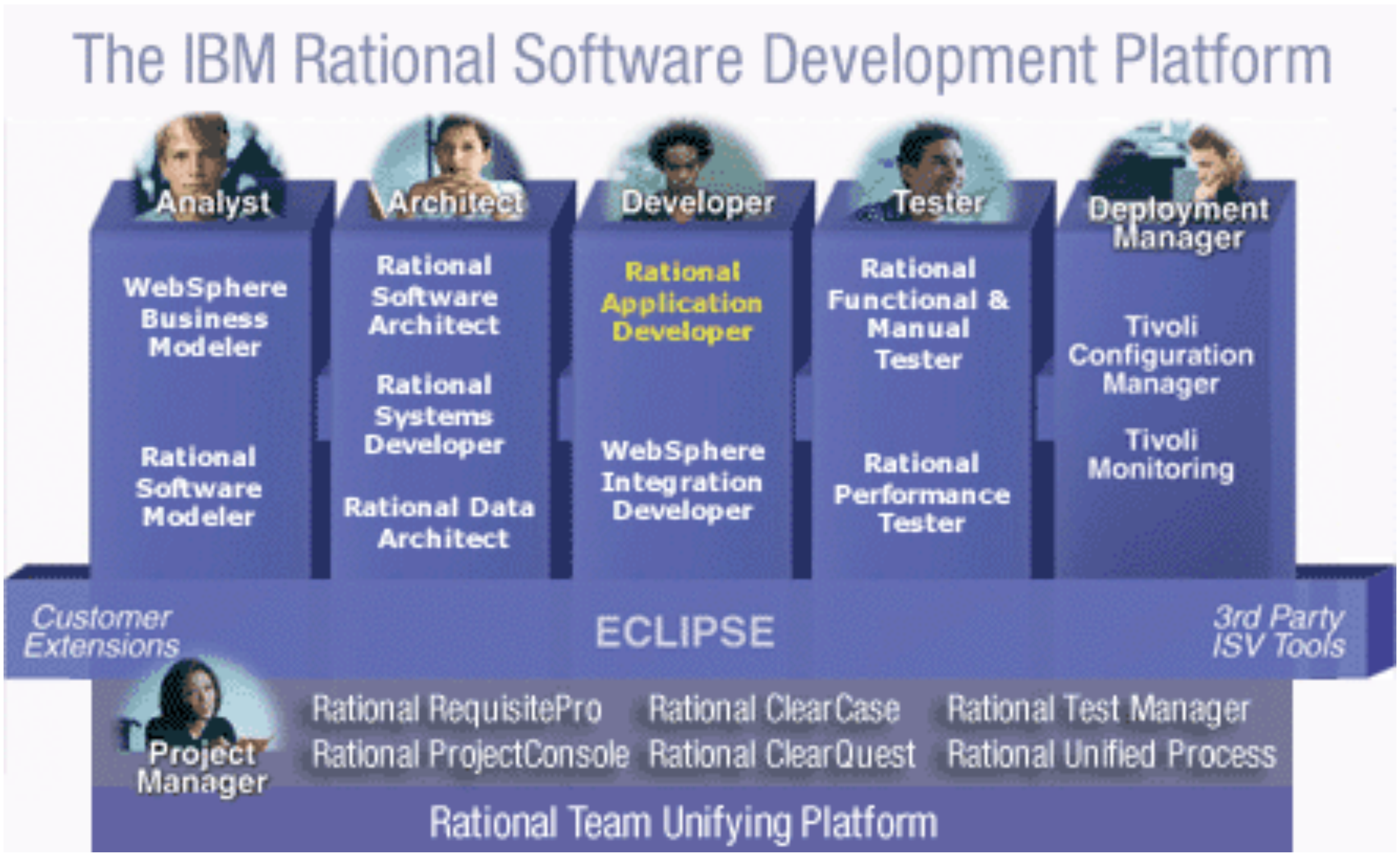


All technology ideas can be modelled in a limerick

- The limerick packs laughs anatomical
- Into space that is quite economical.
- But the good ones I've seen
- So seldom are ‘clean’
- And the ‘clean’ ones so seldom are comical



So, at the dinner last night with Alex and Ines from the IBM Rational Stand



I made up a 'Model' Limerick

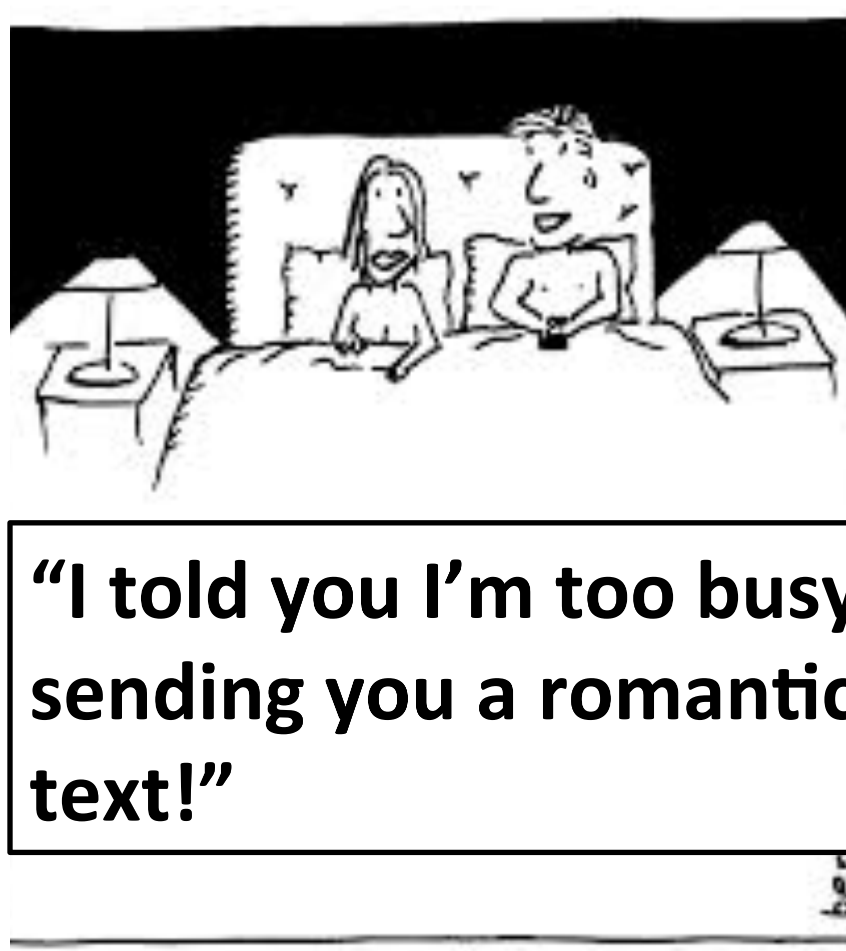
The Model

- There once was a lad with a model

Reality

- There once was a lad with a model

- Who could not get so much as a cuddle



“I told you I’m too busy sending you a romantic text!”

- So he gave up the race

- For a real-quick embrace

- For a real-quick embrace

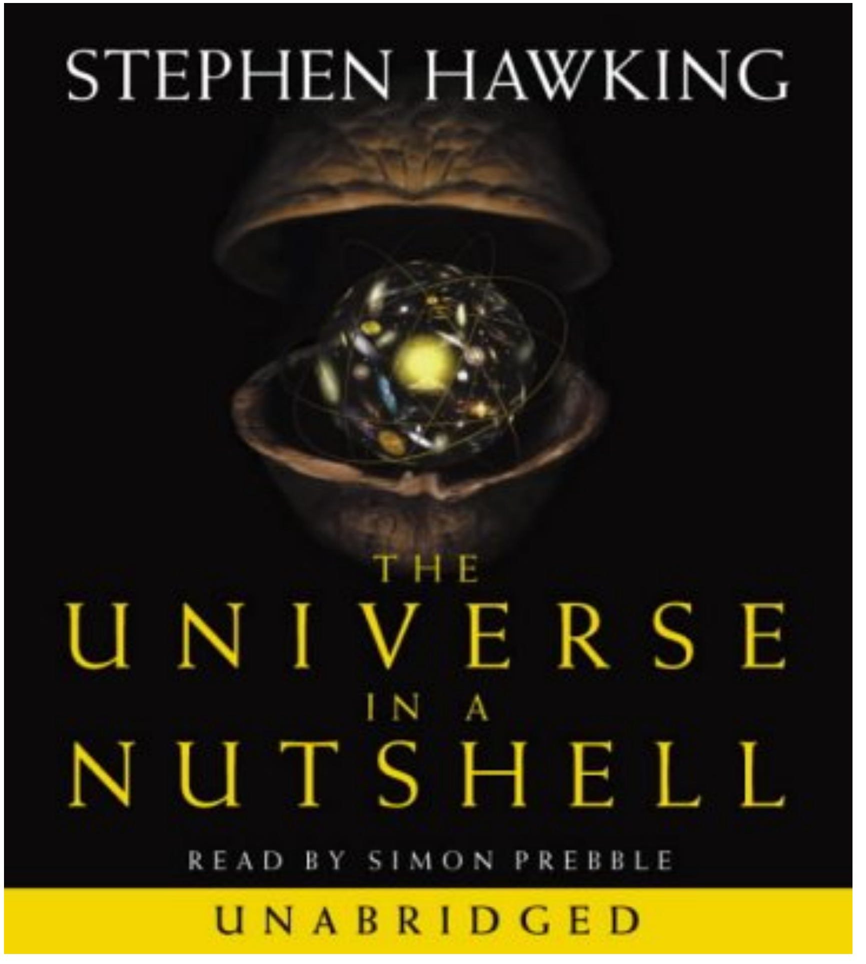
- And opted instead for an ODL

- **ODL** may refer to: Object Definition Language, specification language defining the interface to object types conforming to the ODMG Object Model;

- And opted instead for an ODL

My Talk, in a nutshell

- Software Modelling Languages (SMLs)
  - Must enable us to **describe *all* aspects** of a *software* system (including data, and qualities), and of its related system components (hardware, people, culture).
  - The Modelling Language must allow us to do things (present, analyze, estimate, manage risks, prioritize) that **a mere programming language** (or their high level reflection) is *not capable* of doing



Talk Focus

- I want to focus on things that most modelling languages *are failing* to do for us.
- This includes, failing to help us with:
  - Quality Management
  - Risk Management
  - Prioritization
  - Project Management
  - Economics Management
  - Systems Level Management

Some Fundamental Questions that software modelling languages should answer: *Requirements*

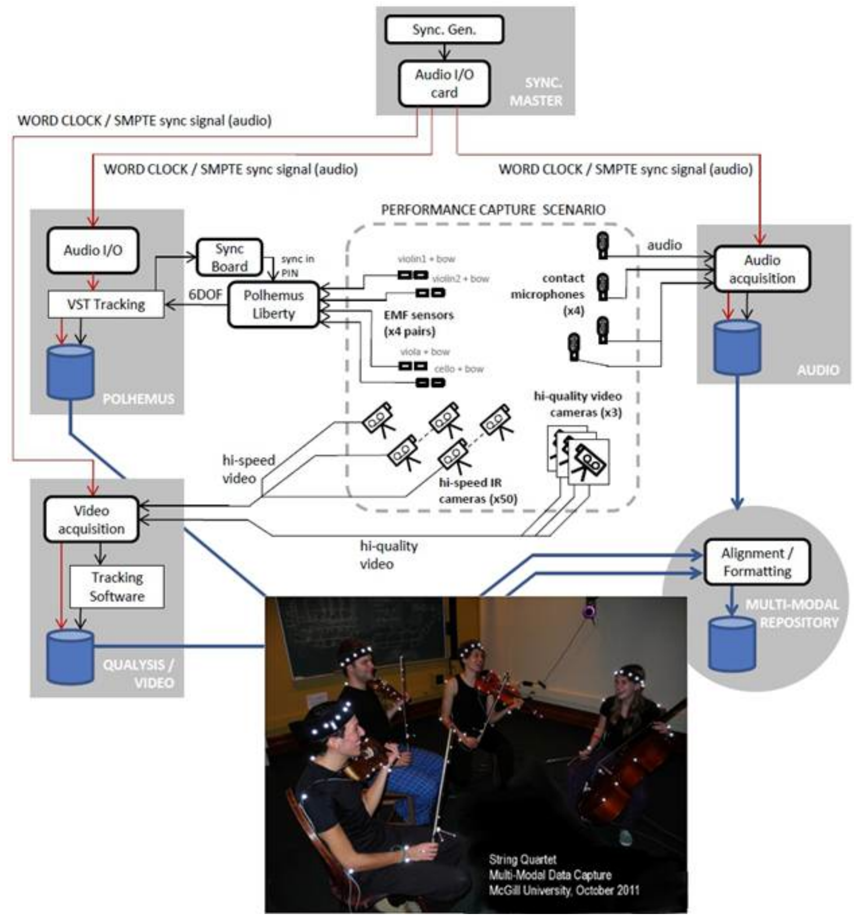
- What are the most **critical stakeholder expectations** for a 'new'/'Improved' system?
  - Are these improvements quantified?
  - Can the improvements be tested to prove presence?
-

Some Fundamental Questions that software modelling languages should answer: *Architecture*

- What are the main **architecture**/strategies which we expect will deliver the improvements?
  - How much **quantified impact** will the architecture have, on our **'improvement'** and 'cost' plans (requirements)
  - Is there **enough information** in the detailed component models, to allow us to add-up to the higher level properties
  - Or to allow us to hypothesize about the *incremental* effects of a given smaller component?

What is the *purpose* of a systems modelling language?

- To allow communication, analysis, and approval of planned requirements and design,
  - by stakeholders,
  - before commitment to
    - ‘Big bang’ development and expenditure
    - or at least before ‘scaling up’ after a pilot experiment
- To consider ideas
  - in a much *cheaper* way
    - than actually *building*, *changing*, or *buying*



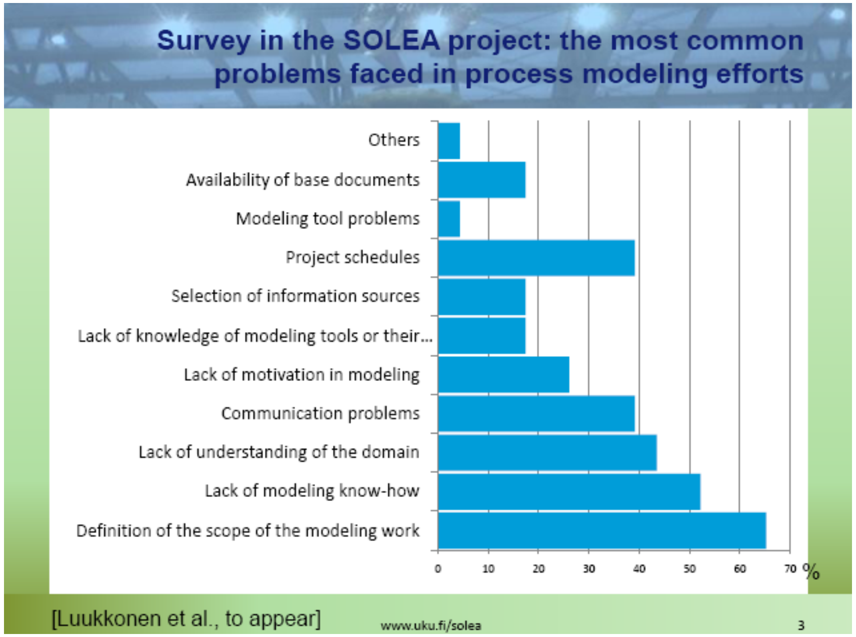
What are the important measurable* attributes of a systems modelling language?

- 1. Extendibility
- 2. Tailorability
- 3. Capability – Comprehensiveness - Richness
- 4. Ease of learning – Usability
- 5. Rigor – consistency – automatic interpretability
- 6. Ease of safe modifiability and growth
- 7. … etc. there are more of course.
  - * for scales of measure see (your free copy of) the Competitive Engineering book, Chapter 5
    - http://www.gilb.com/tiki-download_file.php?fileId=26
    - Or Google xxx Metrics, where xx is name of attribute.

What is wrong with most modelling languages?

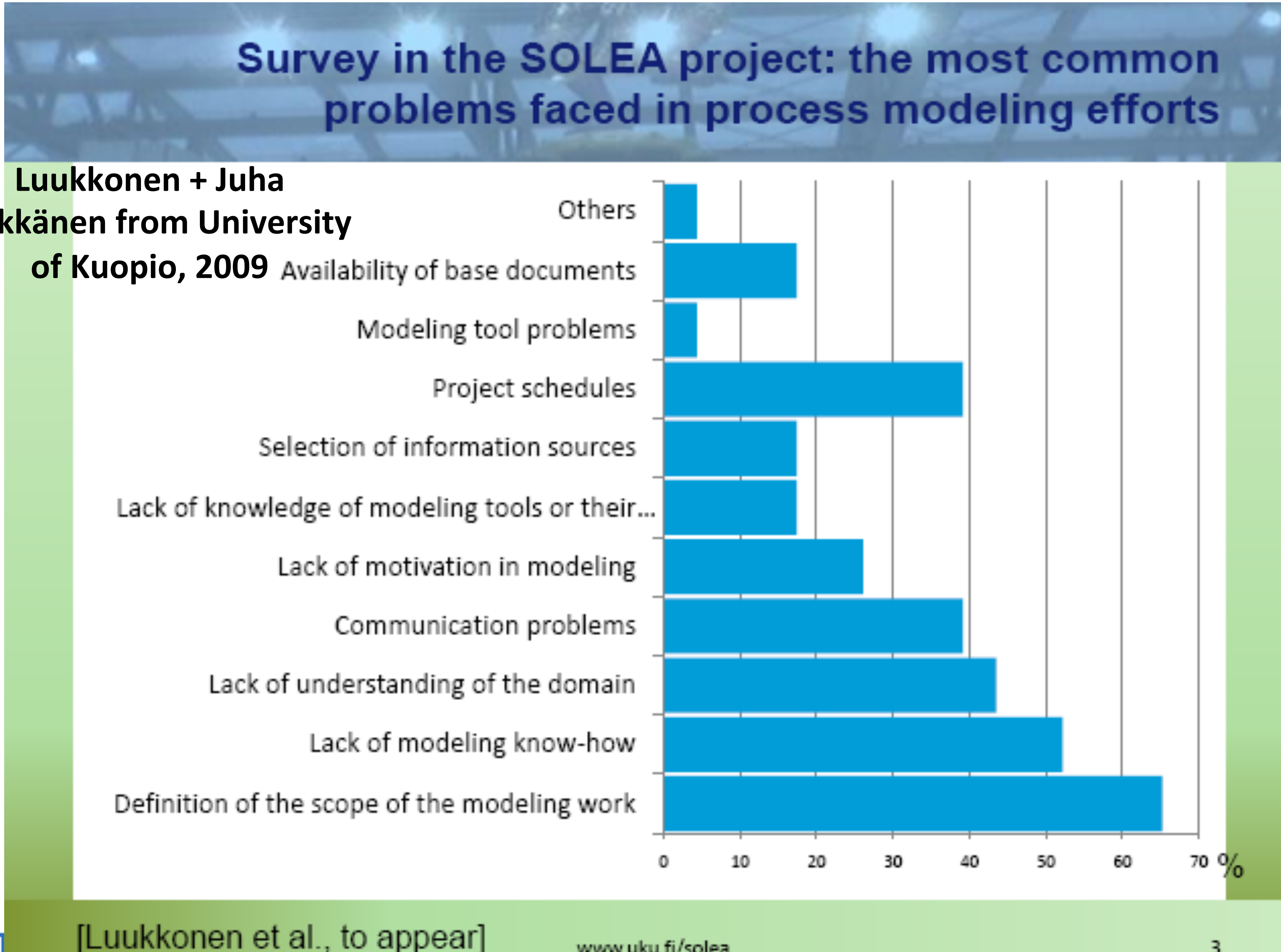
- **GILB'S LIST**
- Qualities Modelling
- Risk Modelling
- Multidimensionality
- Priority Analysis
- Systems Level Thinking
- Connection to Real World Implementations
- Economic Thinking
- Top Level Stakeholder Intelligibility
- Auditability: Review Capability

Juha Mykkänen List



See enlargement, next slide

http://www.metacase.com/blogs/jpt/blogView?searchCategory=DSM&index=31



Luukkonen + Juha Mykkänen from University of Kuopio, 2009

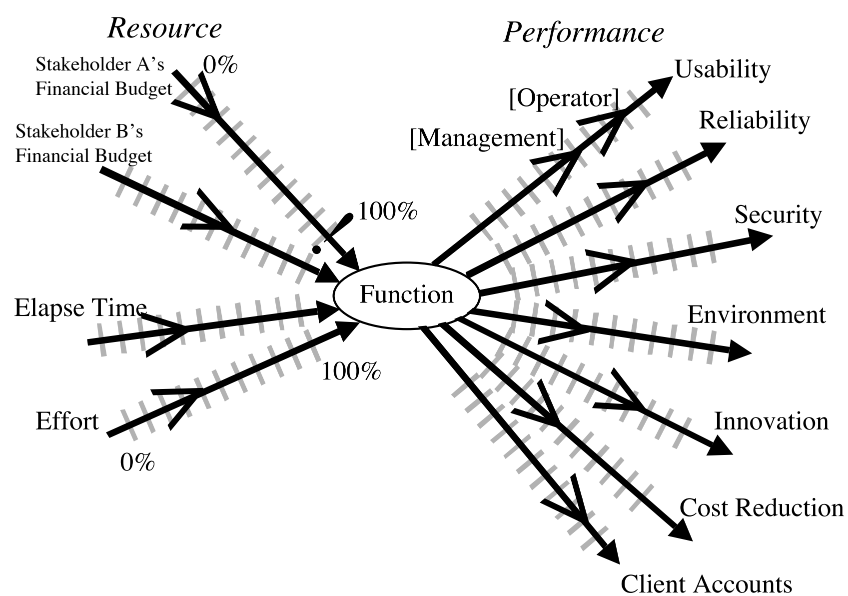
10 Principles of a 'Good' Systems Modelling Language & Process

- **The Clear Benefits Principle**

- 1. Using a Modelling Language
  - must give us some clear overall benefits (net time saved?),
  - compared to simply building the system
  - without modelling first

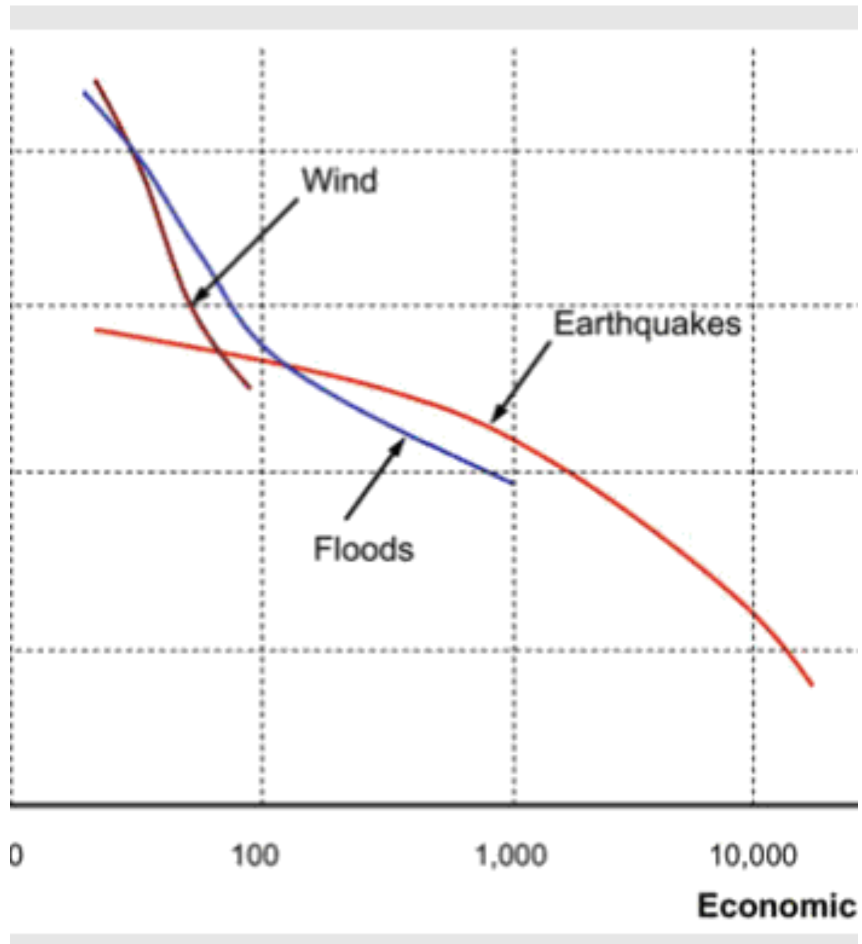
Quality Principle

- 2. You must be able to model the multiple critical system qualities
  - and other performance characteristics:
  - both in terms of requirements,
  - and in terms of design characteristics;
  - and the relationship *between* design and requirements.



The Risks Principle

- 3. Modelling must allow you to explicitly document
  - all manner of risks
    - and proposed mitigations,
  - and their relationships to any other parts of the model



Design Spec Enlarged 2 of 2

==== Priority & Risk Management ========

Assumptions: *<Any assumptions that have been made>.*

A1: **FCCP is assumed to be a part of Orbit.** FCxx does not currently exist and is Dec
Requirements Spec. <- P
discussions AH MA JH EC.

Consequence: FCxx
estimation and cos

A2: **Costs**, the developme
All will base on a budget
The ops costs may differ s
hardware. MA AH 3 dec

A3:Boss X will continue to

A4: the schedule, 3 years,
can in fact deliver, OR we
budget. If not "I would ha

A5: the cost of expanding Orbit will not be prohibitive. <- BB 2 dec

A6: we have made the assumption that we can integrate Oribit with PX+ in a sensible way, even in the short term <- BB

Dependencies: *<State any*

D1: FCxx replaces Px+ in time. ? tsg 2.12

ASSUMPTIONS:
- broadcasts critical factors for present and future re-examination
- helps risk analysis
- are an integral part of the design specification

DEPENDENCIES:

Risks: *<Name or refer to tags of any factors, which could threaten your estimated impacts>.*

R1. FCxx is delayed
2.12

R2: the technical **in**
& we must redevel

R3: the and or scala
allow us to meet th

R4: **scalability** of O
especially <- BB. Pe

R5: re Cross Desk r
technical design. **Solution not currently known.** Risk no
solution allowing us to report all P/L

Risks specification:
- shares group risk knowhow
- permits redesign to mitigate the risk
- allows relistic estimates of cost and impacts

Issues: *<Unresolved concerns or problems in the specification or the system>.*

I1: Do we need to put the
objectives (Ownership).
differentiator. Dec 2.

I2: what are the time scal

I3: what will the success
are actually being asked

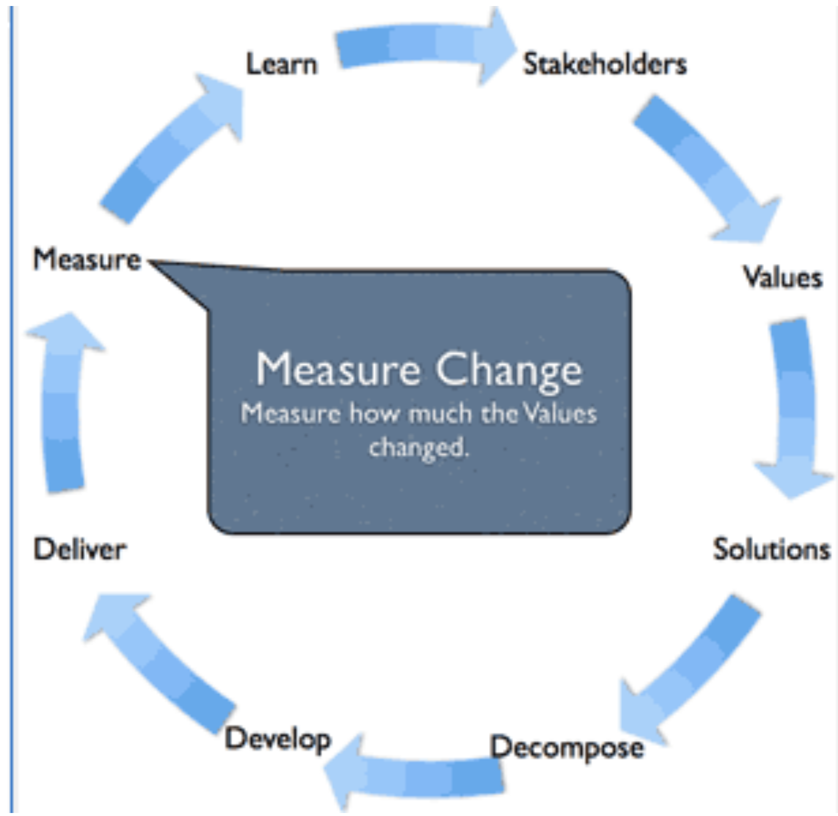
I4: for the business other
lack of clarity as to what
might differ from Extra a

I5: the degree to which this option will be seen to be useful without Intra Day. BB 2 dec

Issues:
- when answered can turn into a risk
- shares group knowledge
- makes sure we don't forget to analyze later

The Dynamic Multi-dimensionality Principle

- 4. Modelling must explicitly allow and promote specification and analysis of all critical attribute dimensions
  - This should include feedback from incremental deployment of system components to the model
    - Not just big upfront modelling
  - The initial model must be capable of being used and integrated, in actual gradual deployment, to the system components



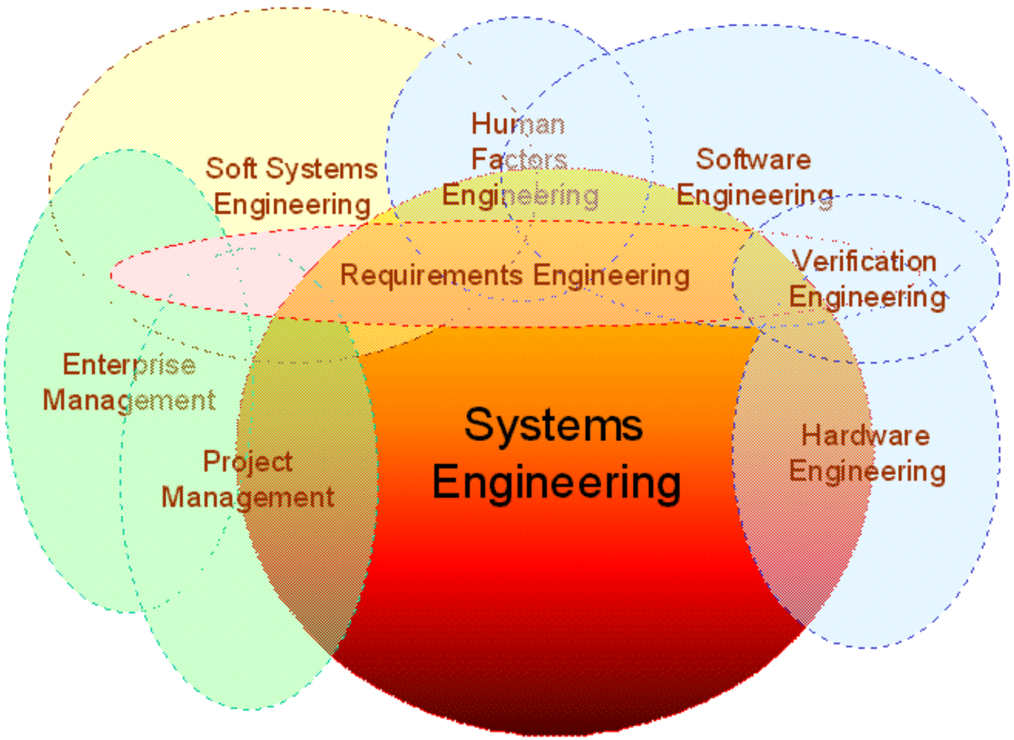
Gilb's 'Evo' Cycle, see Gilb.com

The Dynamic Priority Principle

- 5. Modelling, should give management, information, to decide on the *priority* of work sequencing, and on exclusive choices
  - In relation to one or more prioritization policies
    - Effectiveness, risk, efficiency (cost effectiveness), politics
  - And do so dynamically as the implementation and planning progress

The Systems Engineering Principle

- 6. Modelling should permit and encourage
  - complete thinking
  - about the larger system
    - or technology and humanity involved
  - It should not be limited to a single domain
    - such as function or algorithm alone



Healthcare Impact Estimation

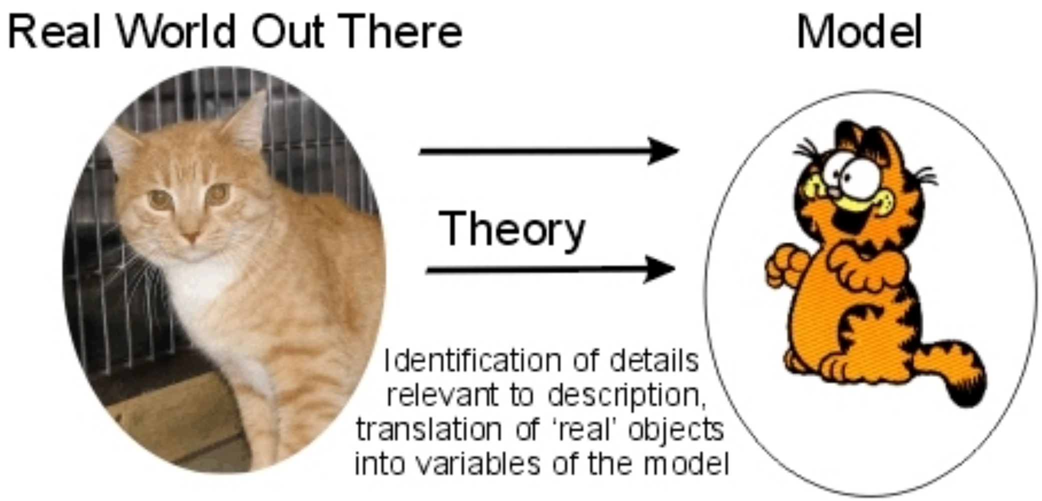
Man-Chie Tse[1,2] & Ravinder Singh Kahlon [1,2]
{Man-Chie, Ravi}@dkode.co

HEALTHCARE SYSTEM IMPACT ESTIMATION

	Automate Rules	Web Self Service	Decision Support	Total Impacts
Increase Transmission of Requests *(30 minutes → 10 minutes)*	10 minutes 100%	3 minutes 100%	-	200%
Decrease Number of Errors Occurring *(353 per week → 30 per week)*	100 errors 80%	< 50 90%	-	170%
Decrease Time for Processing of Requests *(70 minutes → 10 minutes)*	35 minutes 70%	-	< 10 minutes 90%	160%
Decrease Time to Learn process *(1 day → 1 hour)*	-	1 hour 100%	10 minutes 103%	203%
TOTAL DESIGN REQUIREMENT IMPACT	250%	290%	193%	

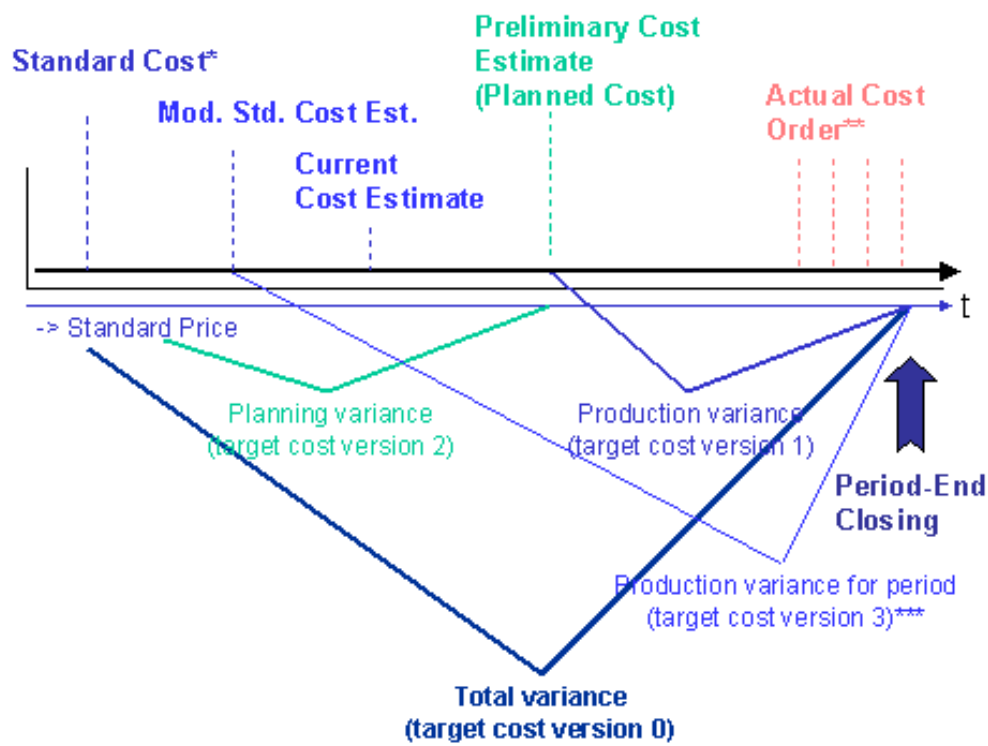
The Real-World Connections Principle

- 7. Modelling should allow and encourage
  - clear specified connections
  - to real development implementations,
  - and to previous, existing and planned future systems.



The Economics Principle

- 8. Modelling must permit, encourage, and easily integrate - considerations of **several** types of resources - related to the model: such as
  - Building and Planning Costs, time, skills
  - Initial investments
  - Recurring lifetime maintenance costs
  - Decommissioning costs
  - Reuse and Porting costs

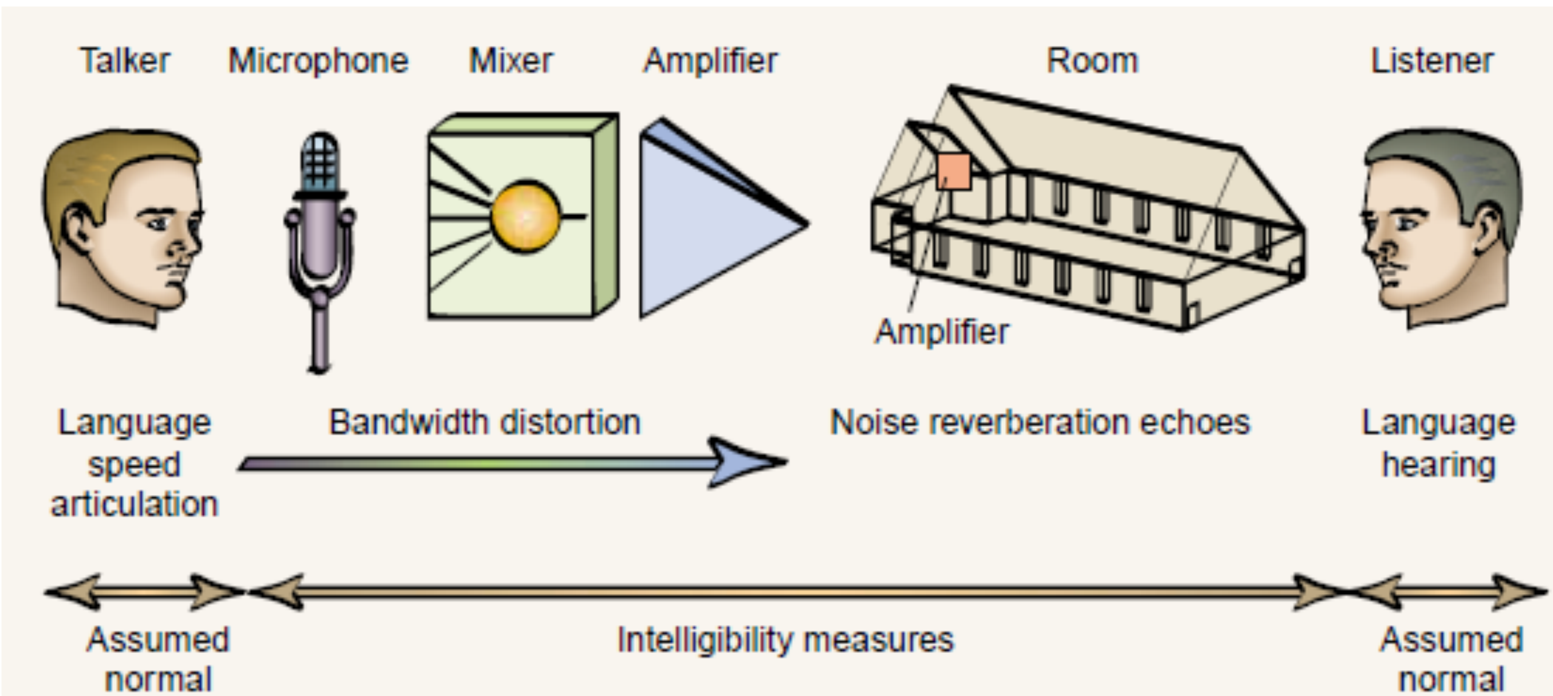


Development Resources swipe and

Product - Solution - VKoT			swipe payments		economic overview		Netbank ajax		Netbank server		payment.tonon		search.contexta			
Value Knockout Table																
© 2013 Kai@Gilb.com Value Management																
Certificate holders are granted permission to use.																
Development-Resources			*units*	*% of Budget*	*units*	*% of Budget*	*units*	*% of Budget*	*units*	*% of Budget*	*units*	*% of Budget*	*units*	*% of Budget*	*units*	*% of Budget*
Work-Hours			**400**	**2%**	**700**	**4%**	**5000**	**25%**	**1000**	**5%**						
0	28000	20000	50	0%	100	1%	1000	5%	300	2%	10000	50%	7000	35%		
			1	2%	0.2	6%	0.5	38%	0.5	8%						
				2%		**4%**		**25%**		**5%**		**0%**		**0%**		**0%**
				0%		1%		5%		2%		50%		35%		
				2%		6%		38%		8%						
				106.27		**59.32**		**6.86**		**34.92**		**91686507**		**48611111**		**7.75**
Value Name Tag				71.43		45.76		4.45		22.03		8.08		2.05		
Status	*Tolerable*	*Goal/Budget*		22.49		9.05		1.94		13.28		25686507		79629629		2.58
when	*when*	*when*		31.44		13.88		0.28		-2.54		1.91		-0.68		
				213%		**208%**		**171%**		**175%**		**367%**		**194%**		**0%**
				52%		25%		38%		31%		-37%		123%		

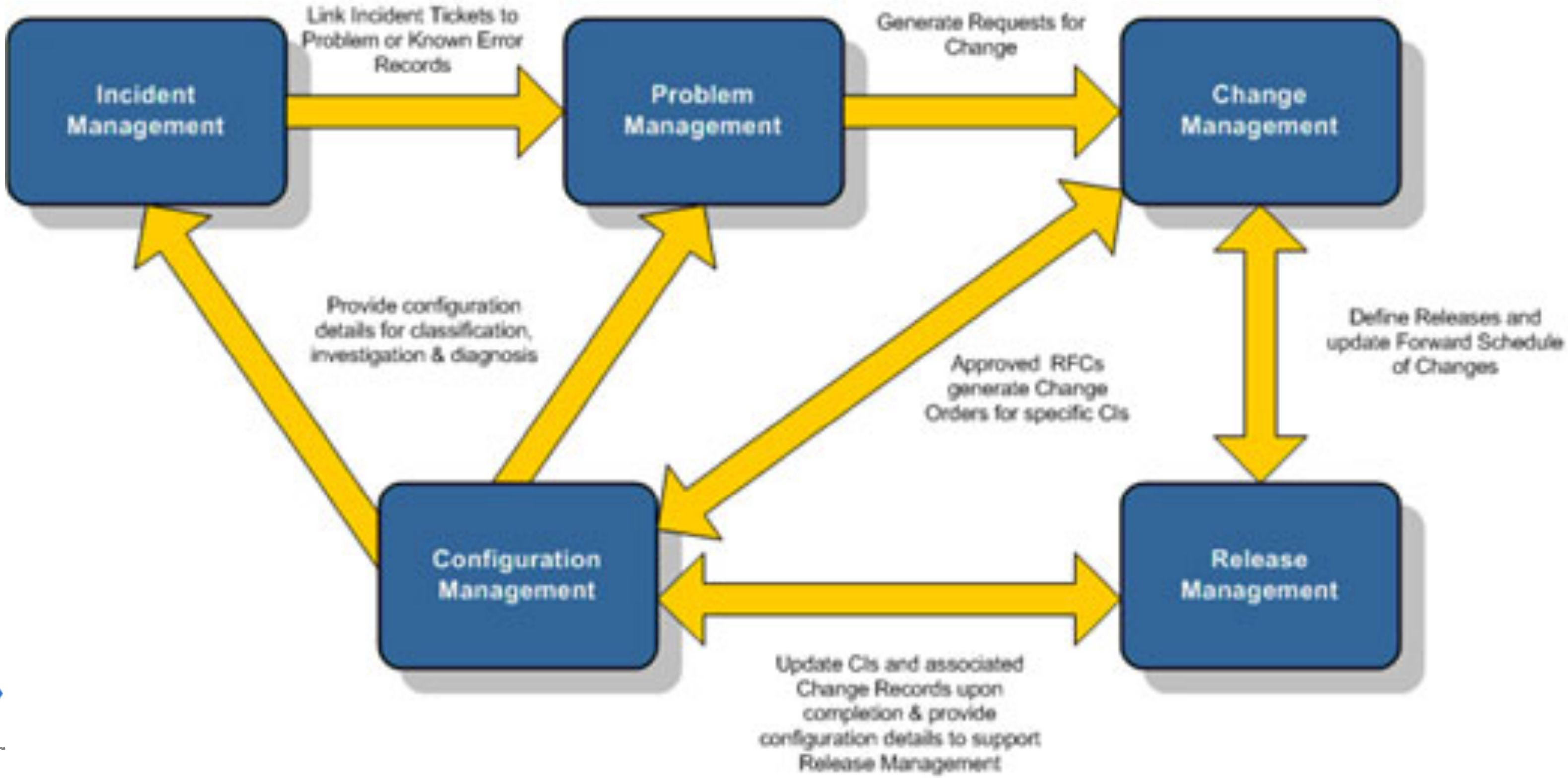
The Stakeholder Intelligibility Principle

- 9. The Modelling Language must enable
  - highly intelligible presentation
  - for relevant stakeholders,
  - of the technical, political and economic consequences
    - of the model information
  - On a continuous updated basis
  - Interpreting the underlying technology
  - Making underlying specifications and assumptions available for review and analysis



The Auditability Principle

- 10. The Modelling language must be capable of quality analysis or ‘auditability’ by review processes,
  - According to defined rules, principles and standards
  - In an economic and selective manner



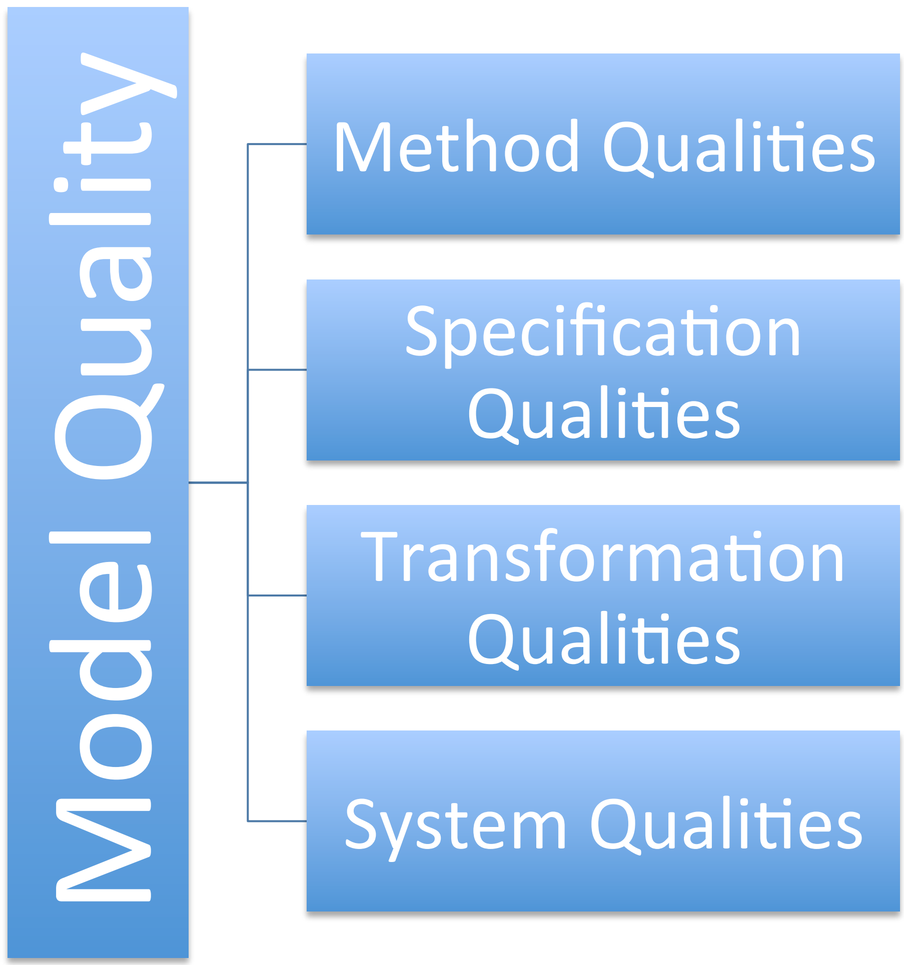
Qualities of Modelling Languages

Stakeholder Intelligibility

- “Modeling languages are intended to be used to precisely specify systems so that stakeholders (e.g., customers, operators, analysts, designers) can better understand the system being modeled.
- The more mature modeling languages are **precise, consistent and executable**.”

- http://en.wikipedia.org/wiki/Modeling_language

Levels of Quality



http://www.bth.se/fou/forskinfo.nsf/alfs/c4cc7a58a7b8bbc2c125739f004abd76/$file/Rapp10.pdf

Some Modelling Languages

- See **more detailed** analysis and comment in my workshop, yesterday 14 Jan 2014
- http://tinyurl.com/QWGILB

Quality Function Deployment (QFD)

Much less well-defined, and much less objective quantification than Impact Estimation Tables

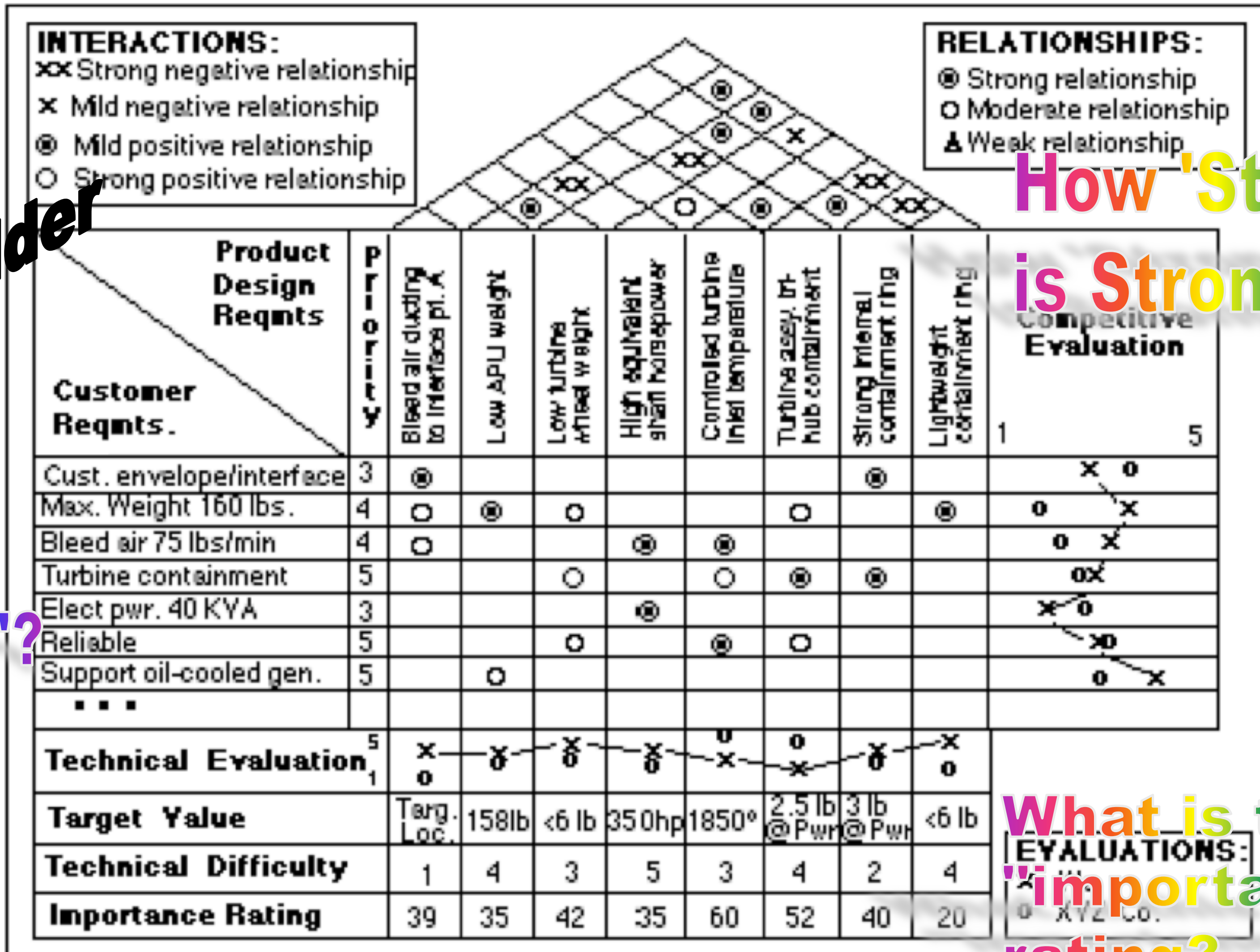


Figure 4 QFD House of Quality

Planguage stages

Man-Chie Tse1,2 & Ravinder Singh Kahlon 1,2
{Man-Chie, Ravi}@dkode.co

PLANGUAGE STAGES

SYSTEM FLOW ABILITY	TARGET GROWTH *(Window Perspective)*			PERFORMANCE *(Parameters Configuration)*			DIRECTION *(Optimisation)*
SYSTEM OBJECTIVES	Vision + Rationale	Stakeholders	Function + Performance requirements	Scale	Past + Goal + Fail	Design Feature + Rational + Dependency	Decision Making
SYSTEM FACTORS	The key stakeholders should provide a vision view of the existing issues to be addressed & expectations required	Some stakeholders are unidentified from the system; therefore identification of all is required	Identify the requirements from the stakeholders which provide insights towards the scope of the system	Pinpoint existing measures present within the organisation. Planguage commends using all values to facilitate constraint	These are scale performance specification, setting parameter target area(s) for achievement	Identification of key designs to the requirements and establish possible credential impacts	The specification of performance should be reflected to the requirements and likely impact.
STAGES	Establish the Vision	Identify all the stakeholders	Obtain & analyse the requirements	Determine the scale measures for performance requirements	Establish the levels on the scales of measures	Identify some potential design solutions	Obtain agreement from the relevant stakeholders
	STEP 1	STEP 2	STEP 3	STEP 4	STEP 5	STEP 6	STEP 7

Impact Estimation Elements

Man-Chie Tse1,2 & Ravinder Singh Kahlon 1,2
{Man-Chie, Ravi}@dkode.co

VALUE Decision Tables: 1 of 3 Levels

Product - Solution - VKoT			swipe payments		economic overview		Netbank ajax		Netbank server		payment.tonon		search.contexta			
				213%		208%		171%		175%		367%		194%		0%
				52%		25%		38%		31%		-37%		123%		
				123%		119%		50%		9%		59%		99%		
Value Requirements			*units*	*% of Goal*	*units*	*% of Goal*	*units*	*% of Goal*	*units*	*% of Goal*	*units*	*% of Goal*	*units*	*% of Goal*	*units*	*% of Goal*
Snappiness			**10**	**71%**	**-5**	**-36%**	**10**	**71%**	**12**	**86%**	**-1**	**-7%**	**14**	**100%**		
85	90	99	5	36%	2	14%	5	36%	3	21%	5	36%	10	71%		
5-Dec-13	5-Jun-14	5-Jun-14	0.1	7%	0.3	-11%	0.7	50%	0.1	9%	0.1	-1%	0.5	50%		
Reliability			**10**	**11%**	**30**	**33%**	**90**	**100%**	**80**	**89%**	**-1**	**-1%**	**-5**	**-6%**		
30	60	120	1	1%	7	8%	2	2%	9	10%	2	2%	1	1%		
5-Dec-13	5-Jun-14	5-Jun-14	0.4	4%	0.8	27%					0.7	-1%	0.2	-1%		
Usability.Intuitiveness			**40**	**100%**	**80**	**200%**					**30**	**75%**	**40**	**100%**		
30	40	70	10	25%	5	13%					10	25%	20	50%		
5-Dec-13	5-Jun-14	5-Jun-14	0.9	90%	0.5	100%					0.8	60%	0.5	50%		
Productivity-Task			**-3**	**30%**	**-1**	**10%**					**-30**	**300%**				
30	25	20	1	-10%	1	-10%					10	-100%				
5-Dec-13	5-Jun-14	5-Jun-14	0.7	21%	0.3	3%										
PV5																
1	2	3														

Strategy & Architecture 'Tags'

Sum of strategy impacts on all Product Values

Product Values

Product - Solution - VKoT			swipe payments		economic overview		Netbank ajax		Netbank server		payment.tonon		search.contexta			
				213%		208%		171%		175%		367%		194%		0%
				123%		119%		50%		9%		59%		99%		
Value Requirements			units	% of Goal	units			% of Goal	units	% of Goal	units	% of Goal	units	% of Goal	units	% of Goal
Snappiness			**10**	**71%**	**-**	**-36%**	**10**	**71%**	**12**	**86%**	**-1**	**-7%**	**14**	**100%**		
85	90	99	5	36%		14%	5	36%	3	21%	5	36%	10	71%		
5-Dec-13	5-Jun-14	5-Jun-14	0.1	7%	0	-11%	0.7	50%	0.1	9%	0.1	-1%	0.5	50%		
Reliability			**10**	**11%**	**3**	**33%**	**90**	**100%**	**80**	**89%**	**-1**	**-1%**	**-5**	**-6%**		
30	60	120	1	1%		8%	2	2%	9	10%	2	2%	1	1%		
5-Dec-13	5-Jun-14	5-Jun-14	0.4	4%	0	27%					0.7	-1%	0.2	-1%		
Usability.Intuitiveness			**40**	**100%**	**8**	**200%**					**30**	**75%**	**40**	**100%**		
30	40	70	10	25%		13%					10	25%	20	50%		
5-Dec-13	5-Jun-14	5-Jun-14	0.9	90%	0	100%					0.8	60%	0.5	50%		
Productivity-Task			**-3**	**30%**	**-**	**10%**					**-30**	**300%**				
30	25	20	1	-10%		-10%					10	-100%				
5-Dec-13	5-Jun-14	5-Jun-14	0.7	21%	0	3%										
PV5																
1	2	3														

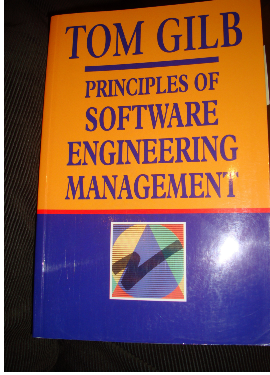
% of 'way to Goal level Impact' of " Economic Overview" Strategy on The 4 Requirements

Modelling using the Real System
Robert Quinnan, IBM FSD Cleanroom

- It raises the question of
  - when modelling is *less useful than*
    - actual building

- For source and detail, see IBM Sys J 4/80

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 manageme ... actice carries cost management far ... nd managerial practices are applied in an ... nsistent with cost management. Th ... ing a design, estimating its cost, and ens

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

-

He goes o ... y either redesign or by sacrificing 'plann ... for a single increment, the 'developmen ... gn of the others.'

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

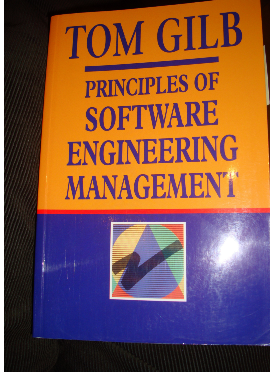
It is clear from this that they avoid the big bang cost estimation approach. Not only do they iterate in seeking the appropriate balance between cost and design for a single increment, but they iterate through a series of increments, thus reducing the complexity of the task, and increasing the probability 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. . . 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 o… y either redesign or by sacrificing 'plan… for a single increment, the 'developmen… gn of the others.'

'Design is an ite… ous level.' (p. 474)

It is clear … only do they iterate in seeking the app… they iterate through a series of increm… obability of learning from experience, won… ecomes a fact.

'When the devel… te the remaining increments is co…

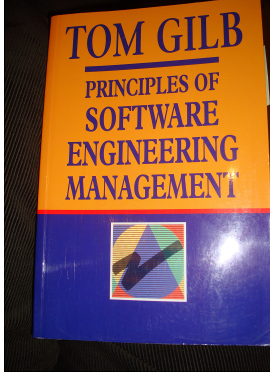
Source: Robert E. Q… . 19, No. 4, 1980, pp. 466~77

This text is cut from…

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

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.

“Design is an iterative process, ..”

'<u>Design is an iterative process </u>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 <u>they iterate through a series of increments, thus reducing the complexity of the task</u>, <u>and increasing the probability of learning from experience</u>, won as each increment develops, and <u>as the true cost of the increment becomes a fact</u>.

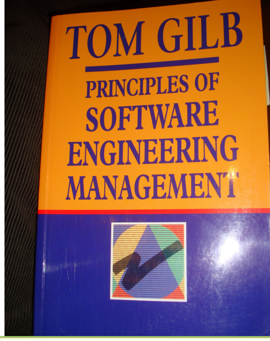
'When the development and test of an increment are complete, <u>an estimate to complete the remaining increments is computed</u>.' (p. 474)

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



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

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 <u>they iterate through a series of increments, thus reducing the complexity of the task</u>, <u>and increasing the probability of learning from experience</u>, won as each increment develops, and <u>as the true cost of the increment becomes a fac</u>t.

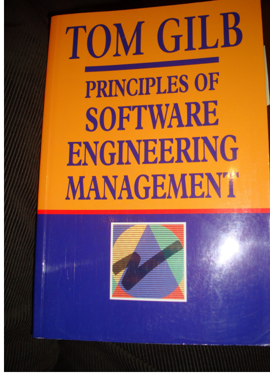
'When the development and test of an increment are complete, <u>an estimate to complete the remaining increments is computed</u>.' (p. 474)

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



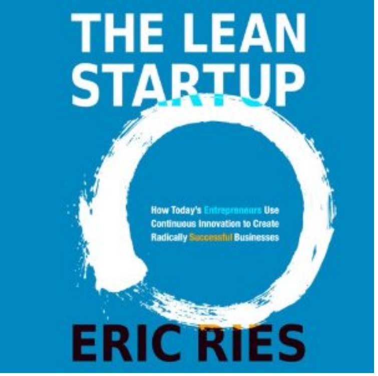
Quinnan describes the process control loop used by IBM FSD to ensure that cost targets are met.

"an estimate to complete the remaining increments is computed."

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

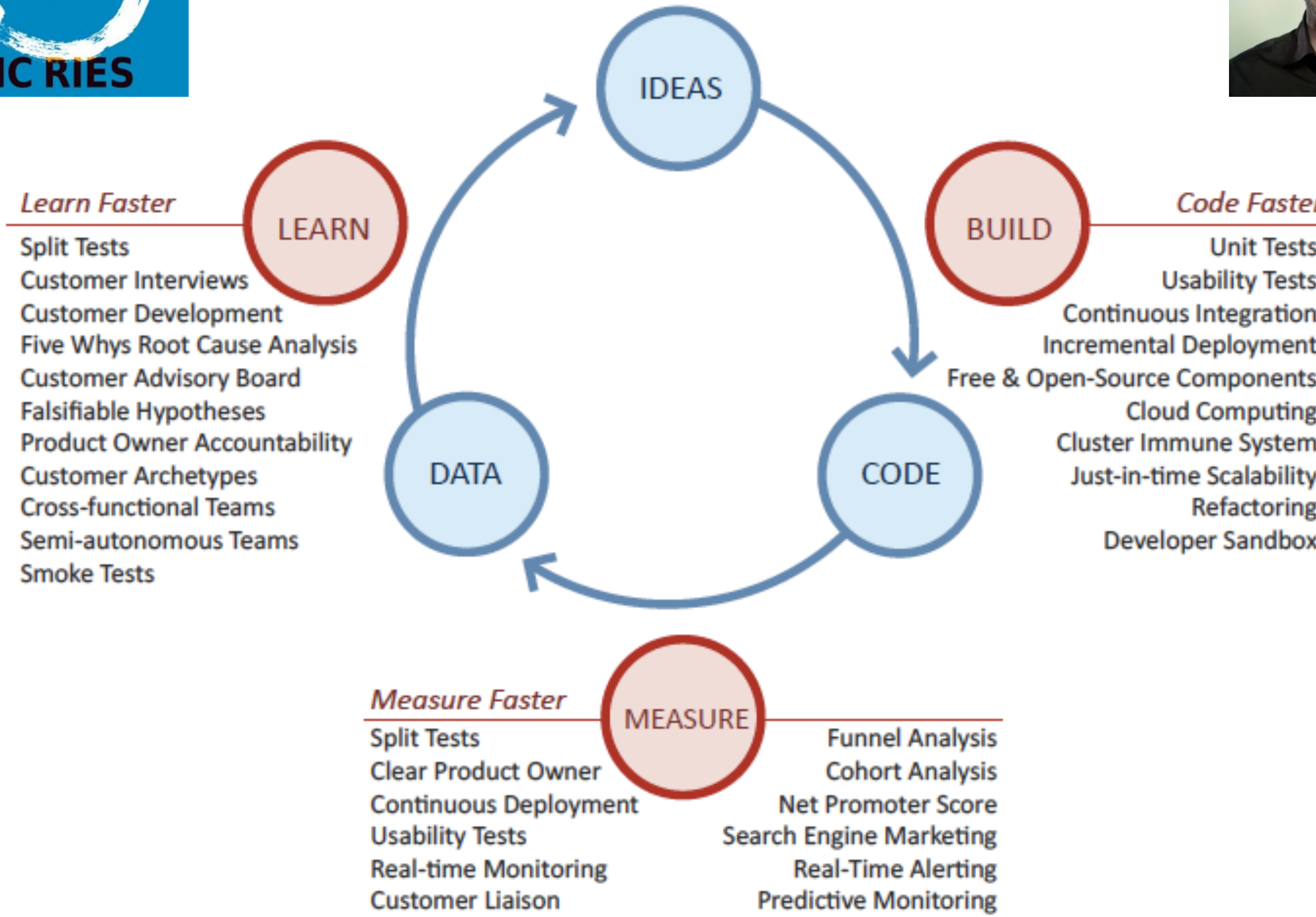
Source: Robert E. Quinnan, 'Software Engineering Management Practices', IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 466~77

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



Lean Startup

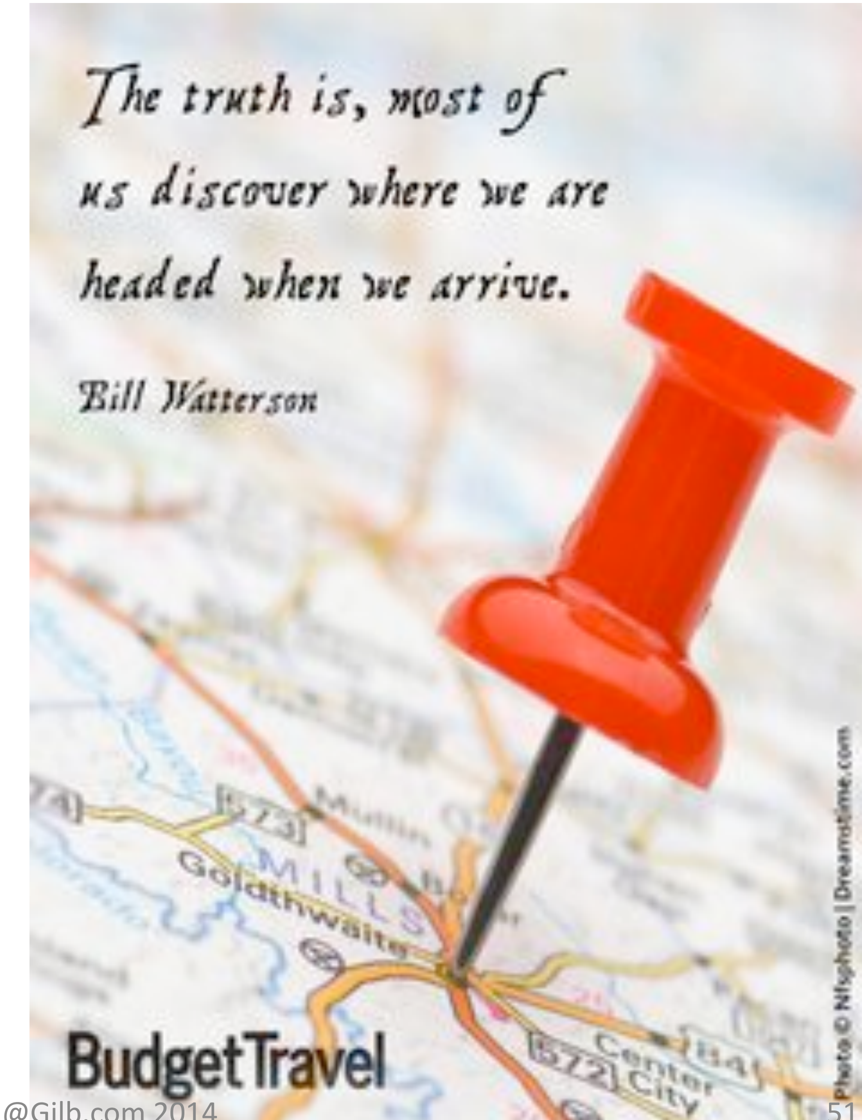
Realistic modelling by testing design hypotheses live
60 times a day



ROC: Resource Oriented Computing Modelling without Artificial Models

- What if, the real system was easier to build than the model?
- *What if the real system gave more reliable information about expected performance and quality than any modelling language?*
- What if the real system solved many of the problems we want to solve by modelling?
  - (Scalability, Legacy, Maintainability, Availability, Connectivity)?
- *What is your system is highly self-optimizing based on experience?*
- Would this change our current view of modelling?

- **If the map and the terrain disagree, trust the terrain.**
  - Swiss Army Aphorism



Reaping the Economic Dividend of ROC

• Architecture is 100% decoupled (not simply loose coupling) ◦ Hot-swappable

◦ Legacy coexistence

◦ Genuine reuse

◦ Unlimited evolvability

• Cheaper to develop

◦ 80% of a problem is solved by composition of existing tools ◦ Cheap to change - recomposition.

• Focus entirely on the domain problem

◦ Engineering levers available (eg throttle)

◦ Audit is built in

◦ Configuration Management: "Everything is a resource"

◦ Logging "in case it goes wrong" is redundant "execution state is a resource" Visualizer

• Constraints are spacial boundary conditions ◦ Trust and non-repudiation

◦ Validation, Semantic integrity

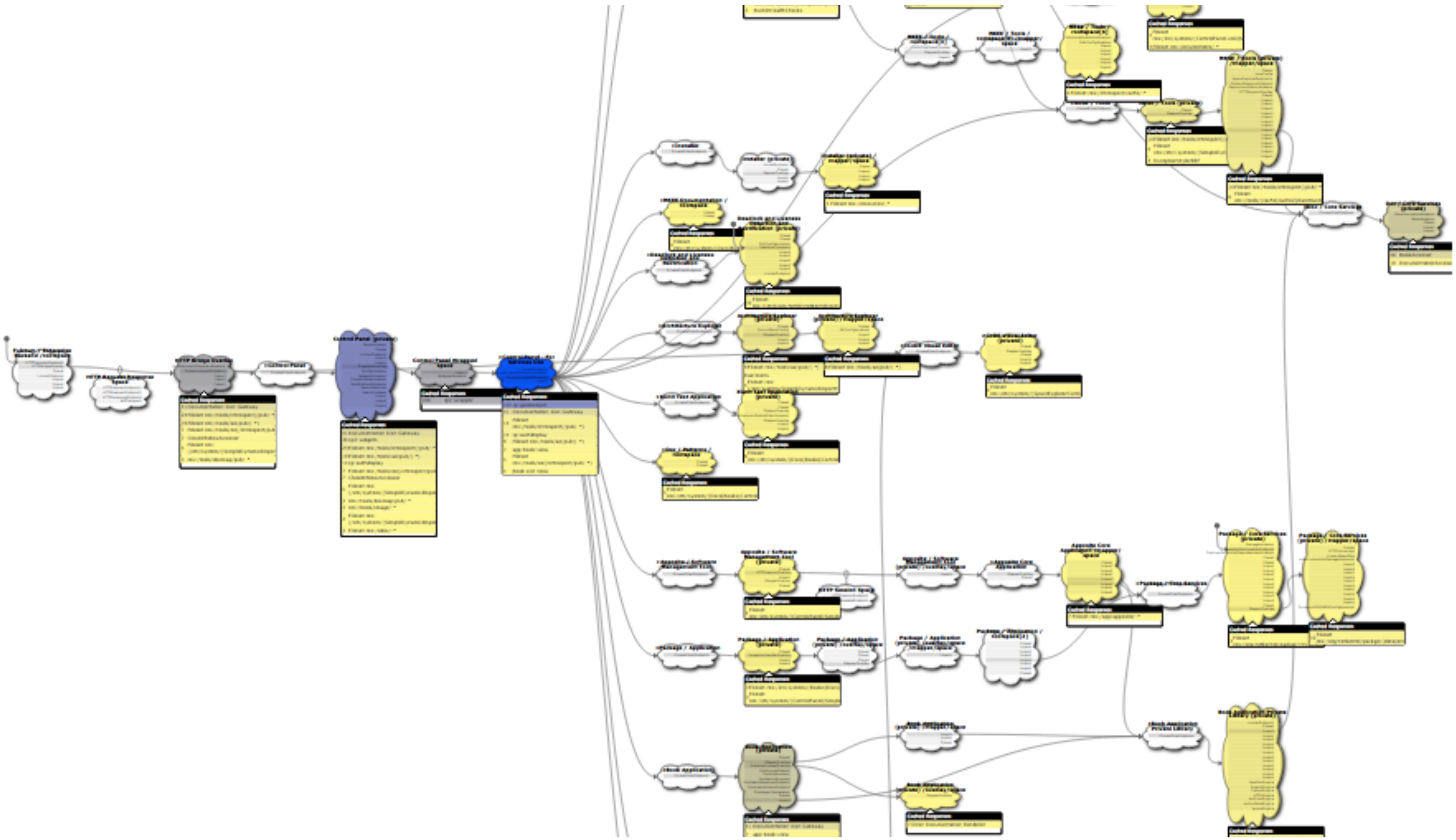
...and higher performance too...

Peter Rodgers
Founder and CEO

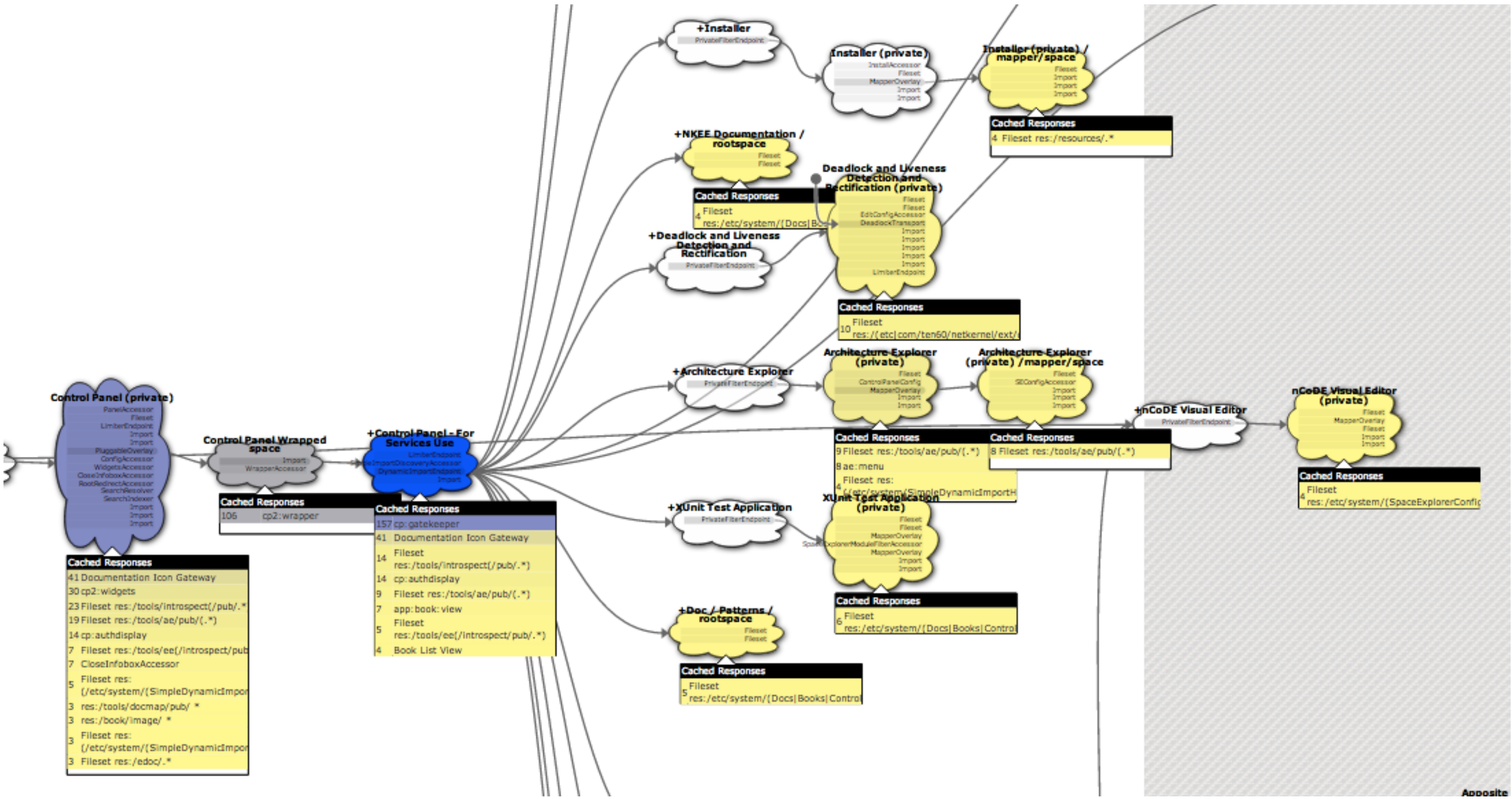
http://1060research.com/about/

ROC Architecture Model

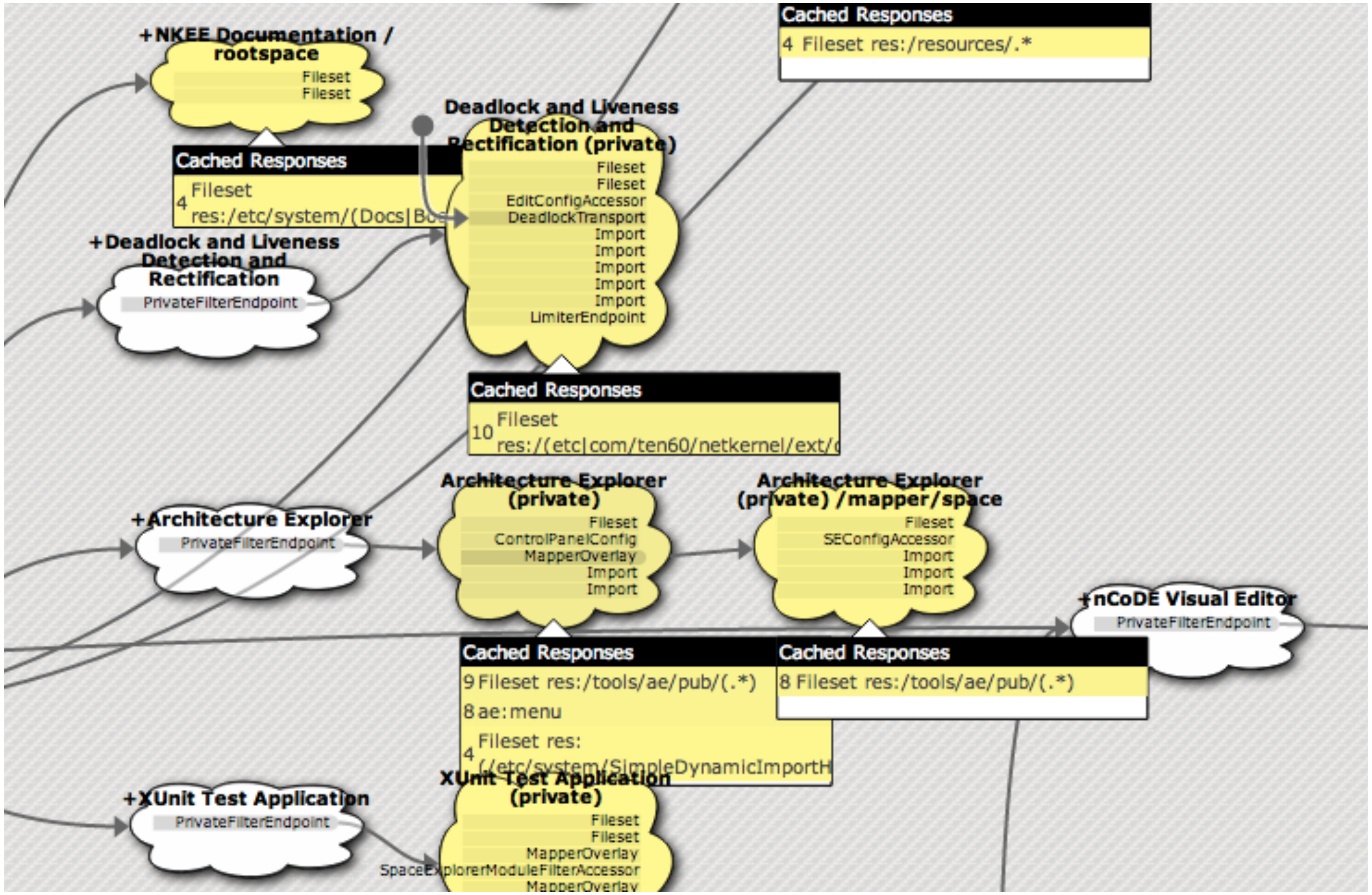
ROC Enlargement
Live Representations of real system



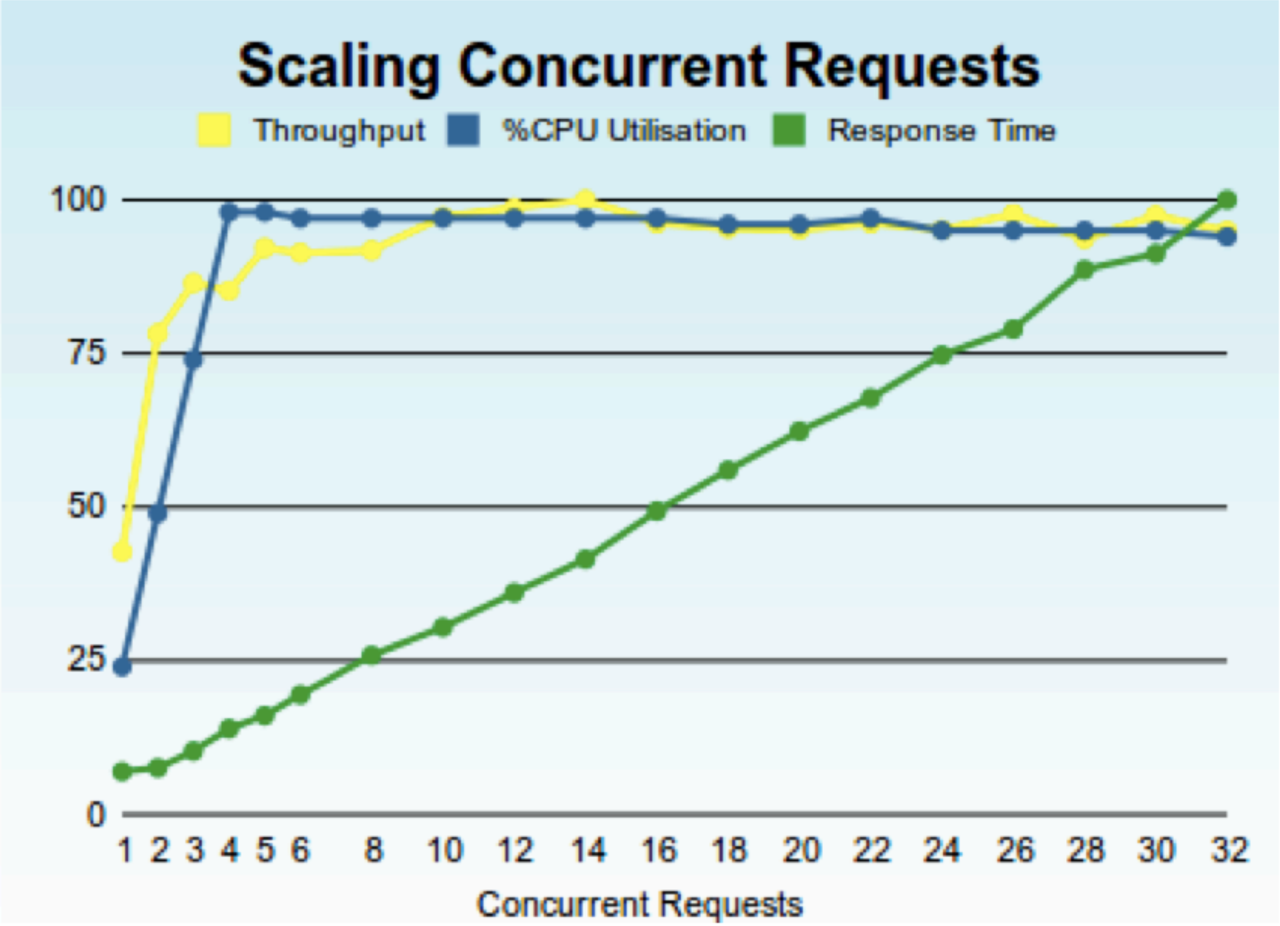
ROC Enlargement 2



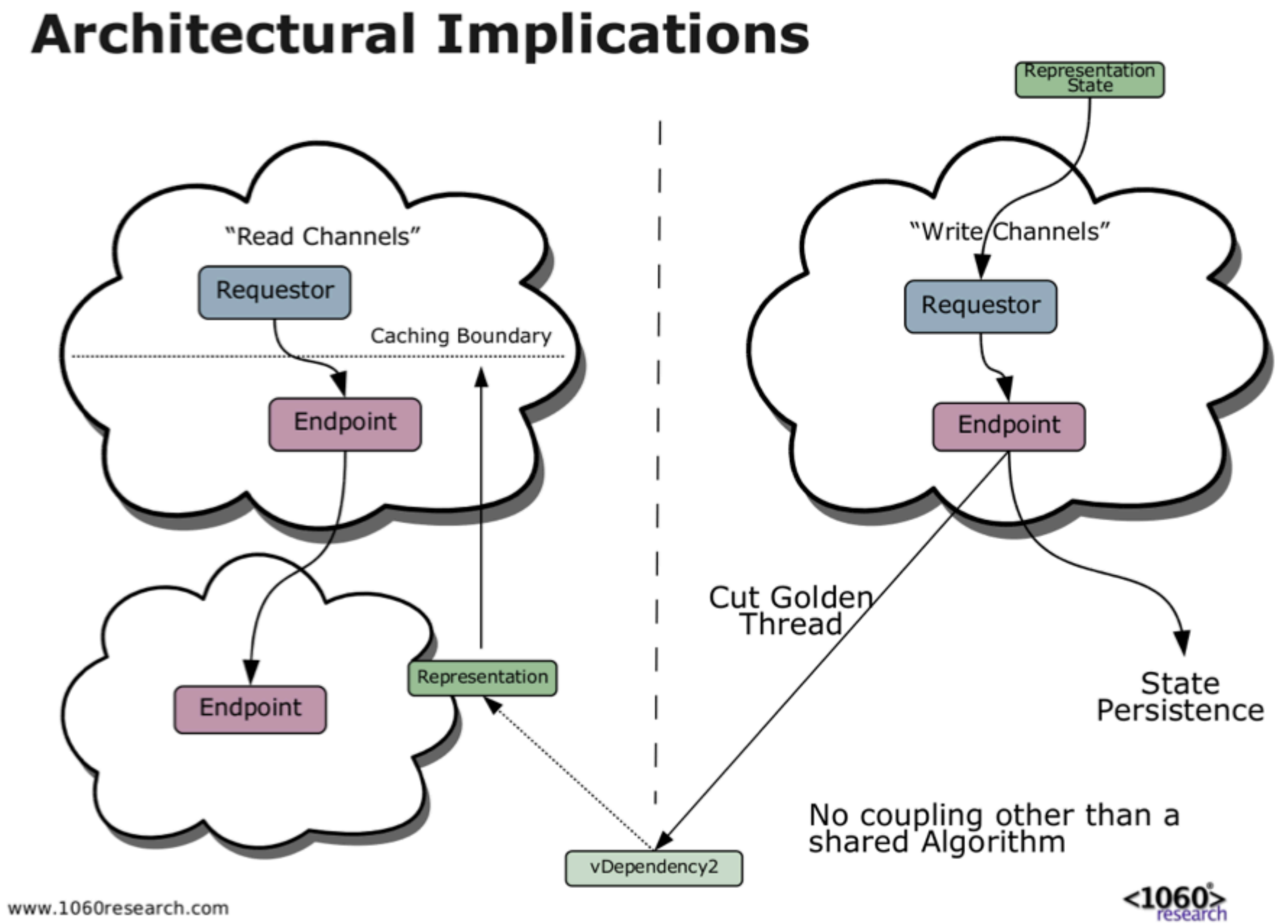
ROC Enlargement 3
Dynamic 'Modeling' of Live System



ROC Scalability



No Direct Coupling in ROC
Coupling is Real Time, as in WWW



NetKernel Capability Experience

(Source Peter Rodgers lecture, Oslo 2014)

The Resource Oriented Computing platform

- **General Standalone Application Server**
- **Embeddable as "ROC Engine"**

Proven with hard-core, carrier-class deployments

- **Telecoms**
- **Black Friday Retail**
- **Huge dot-com platforms**
- **Core Web Instracture - PURLs, Dublin Core**
- **Government Open Linked Data**

Separates Architecture from Code - brings engineering control to systems.

Systemic Memoisation (Caching) and Async Linear Scaling = Huge Performance Gains.

Changes Attainable Scale of Software

Changes Economics of Software - Eliminates Saw-Tooth build new and replace syndrome

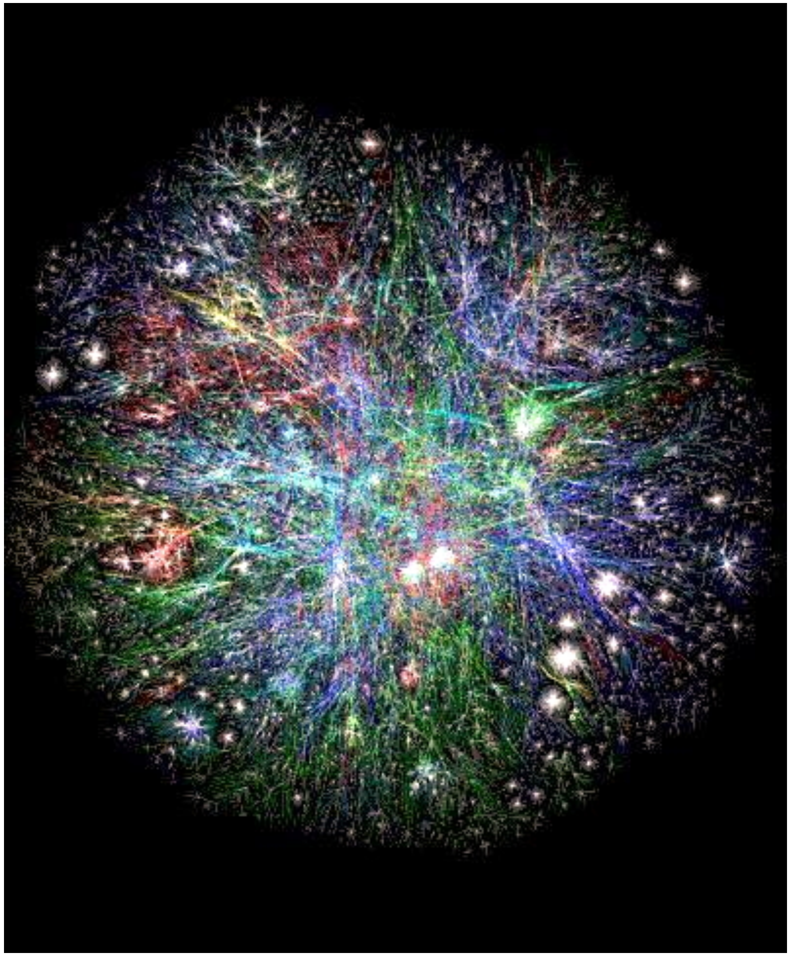
Brings the Web Inside, and makes it general purpose.

NKP takes ROC back *out* to enable amazing cloud architectures.

Netkernal ROC: The System is the Model

- ROC has made me wonder if we can go further than Cleanroom, Lean Startup, and Evo
  - which learn about the system, by incrementing one small real step at a time, and measuring the integrated effects of the increment

- It makes me ask the question, can we simply build a specification system that, *like the WWW it is modelled on,* is self measuring, self documenting, self optimizing, self maintaining, and automatically adaptable?



http://resources.1060research.com/docs/Collection-ROC-NetKernel-v1.1.1.pdf

Closing Limerick
(Especially for my Viennese Friends)

- If a **model** was reliably true
- Then that would be *really* new
  - Since models are *false*
  - As a 'Norwegian Waltz'
- So reality will just have to **do**

Free digital copy of
CE Book (Planguage Handbook)
on request to
Tom @ Gilb . com

The Oslofjord in background

My Summer Cabin

(Happy to give Austria access to the sea from my beach)