Competitive Product Engineering:
10 Powerful Principles
for winning product leadership,
through advanced systems engineering

- Copyright © 2006 by Tom Gilb.

Iver Holtersvei 2, NO-1410 Kolbotn, Norway, Tom@Gilb.com,

- www.Gilb.com, +47 66801697

- **Abstract**:
    - Some product developers are still trapped thinking narrowly about their technology – they do not have enough customer focus, and they do not get good enough feedback from the customer and support team 'real world'. These principles will help refocus them.

Here is an overview of the principles I recommend:

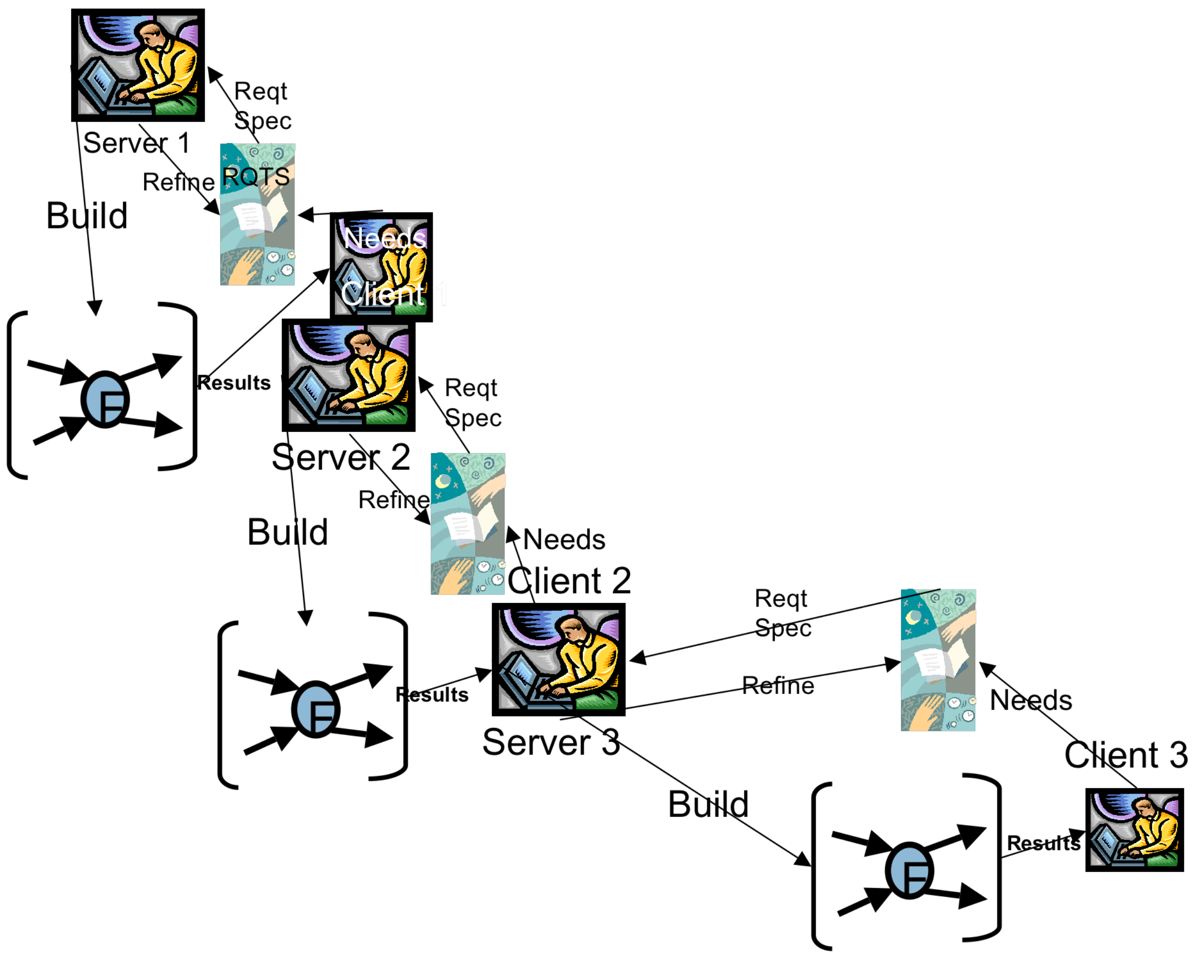
- 1. Stakeholder Focus
    - Formally identify all your product stakeholders: all '35' of them – any group who can influence success or failure. Don't over focus on users alone.
- 2. Stakeholder Value Focus
    - Identify the value sets of the stakeholders, particularly *qualities*. Quantify the valued quality levels, with a scale of measure and a goal level.
- 3. Value Delivery First
    - Plan to deliver highest possible value to some stakeholders, as early as possible.
- 4. Learn Rapidly
    - Get quantified measurements from early frequent increments of your product, regarding real quality levels and costs. Analyze this data in relation to expectations.

Learn from deviations as rapidly as possible, and change plans to reflect realities.

- 5. Delivery Frequently
    - Plan to deliver increments of performance, quality and function on a frequent basis, for example *weekly*. This pace will be a useful discipline, ensure management control, and send positive messages to stakeholders.
- 6. Deliver Early
    - Get something out to stakeholders at the very beginning of your development project. This might involve improving existing products, using some of the ideas for new products. But it sends signals to the market, and makes sure your team is well grounded in reality.
- 7. Hit the ground running
    - Make sure that each product increment is industrial standard quality. No second rate prototypes.
- 8. Quantify Valued Qualities
    - All critical product qualities must be specified quantitatively, otherwise you have lost control of them.
- 9. Control Value-to-cost ratios
    - Make sure you keep your eye on the value-to-cost ratio of each step. That will remind your staff that it is about profitability, not technology.
- 10. Rapid Reprioritization
    - When stakeholder feedback, or measures of delivered value and cost dictate it, change your plans to maximize your profit.

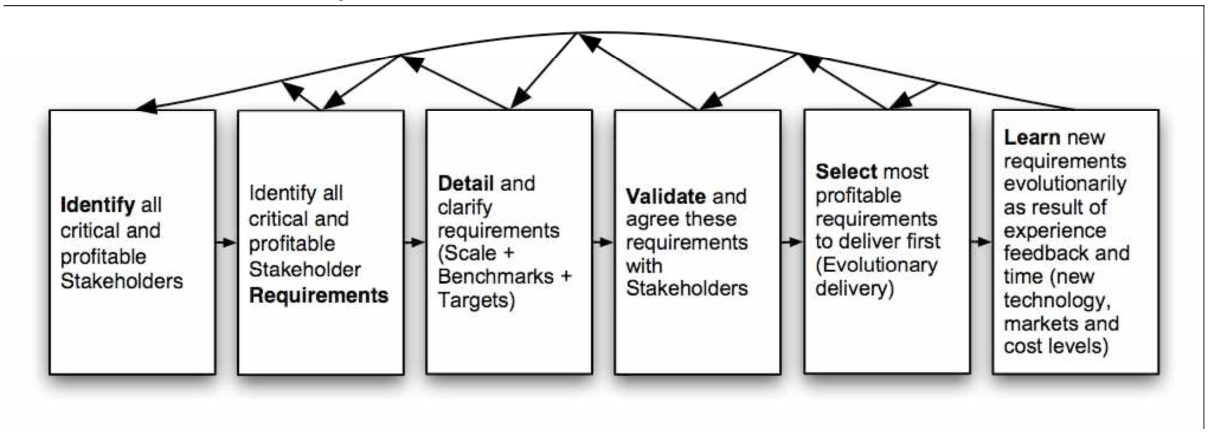
Detailed Discussion about the principles

- 1. Stakeholder Focus
    - *Formally identify all your product stakeholders: all '35' of them – any group who can influence success or failure. Don't over focus on users alone.*



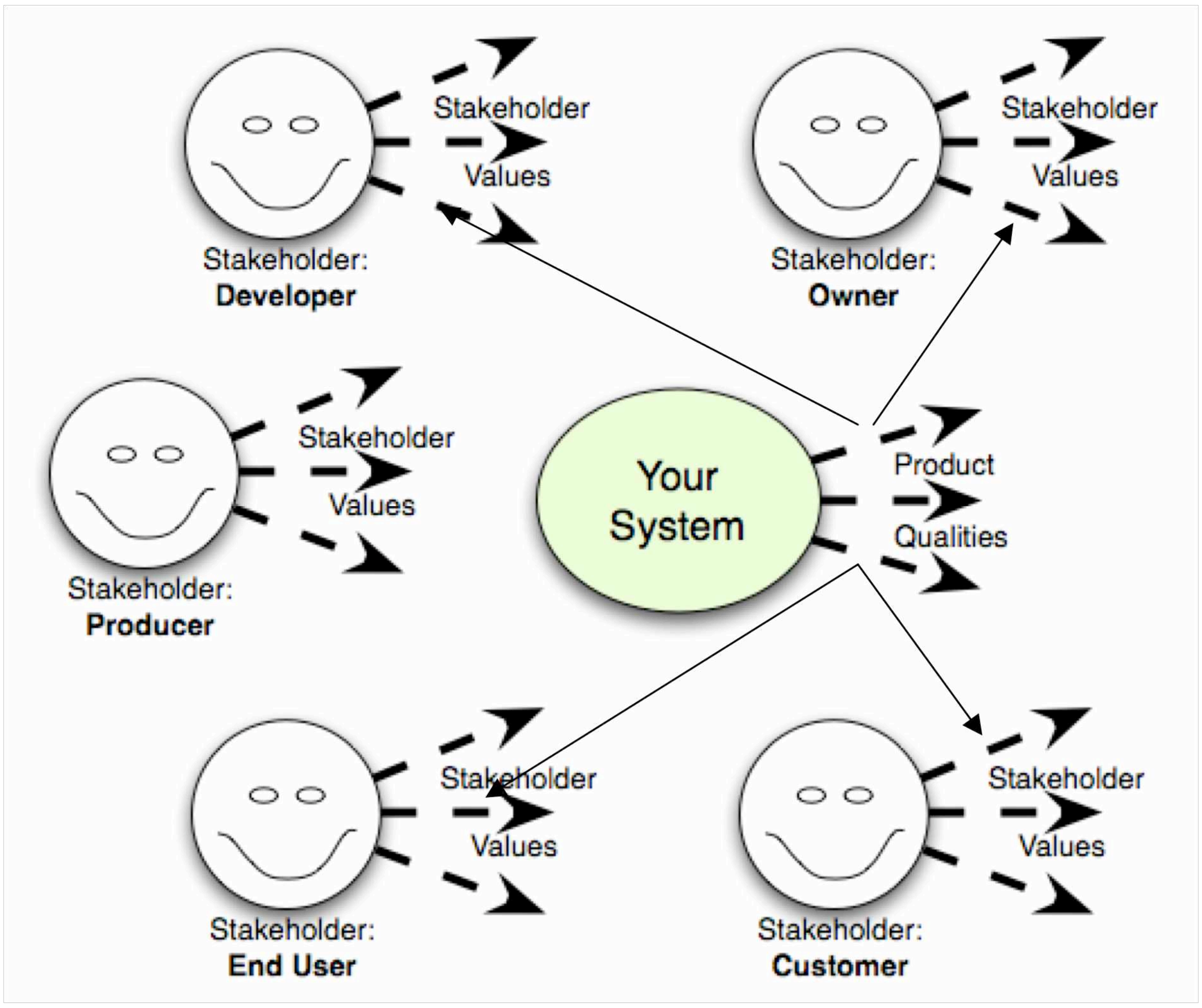
- Fig 1: Requirement stakeholder levels. A requirement is a client stakeholder need, that a 'server' stakeholder i s planning to satisfy.
- At the US operation of a multinational telecoms manufacturer, we were analyzing 50 pages of requirements for a more than $100 million product investment. After a few days we recognized that almost all the requirements were focussed on the end user (wandering from work to street to home with a single handset). At least 10 major stakeholders, such as their factories, and product installers, we not specifically identified. The needs of these critical stakeholders were not systematically identified and well specified. The corporation did not have stakeholder analysis processes in place. This may have been due to their transition from a monopolistic environment to a more competitive environment, as well as transition from small scale systems to very large scale systems.

- So, take a look at your formal practices, and the real practice of your stakeholder analysis process if you do have one formally. Are you really identifying to dozen or more interesting stakeholders, and their requirements? If not, then you can increase your real competitiveness by making this a formal process.
- Stakeholder analysis is not a new practice for real systems engineering cultures, but it may be poorly carried out by some cultures because of their market and technical history.


Identify all critical and profitable Stakeholders
Identify all critical and profitable Stakeholder **Requirements**
Detail and clarify requirements (Scale + Benchmarks + Targets)
Validate and agree these requirements with Stakeholders
Select most profitable requirements to deliver first (Evolutionary delivery)
Learn new requirements evolutionarily as result of experience feedback and time (new technology, markets and cost levels)

- Figure 2: A process for finding a more competitive set of requirements through more thorough stakeholder analysis. Notice that it is iterative, and probably eternal, as long as things are changing.

- 2. Stakeholder Value Focus
  - *Identify the value sets of the stakeholders, particularly qualities. Quantify the valued quality levels, with a scale of measure and a goal level.*



- Figure 3: Each stakeholder has multiple related values (example 'save staff costs') that need to be analyzed and prioritized, and finally a corresponding set of product quality levels must be determined for the product to best satisfy some of those values (like 'short product learning time').
- We need to analyze, specify and confirm what the various stakeholders *really* want. *Not* in terms of the product qualities, and not in terms of the *technology* itself. But. In terms of what stakeholders 'really want'. This is not always easy because you will experience that the stakeholders will tell you what they think you *can* deliver or will deliver. They may not realize what you might really deliver if only you knew what they 'really' wanted.
- If you make all the values that the stakeholders really want, then you will necessarily map some needs that you cannot or will not attempt to satisfy in a given product or

product line. But at least you have a chance to consider it. And if you do this thoroughly then you are likely to set requirements for a more competitive product than you otherwise would have required.
    - There is a question of *who* should do this stakeholder value analysis. In theory it is your 'marketing' function. But my experience in computers, banking, and telecoms has taught me that the typical 'marketing' (or business analysis) function is poorly trained and managed to do such an analysis properly. They are more likely to be 'knee-jerk reacting' to direct customer input and competitive pressures. So if you want to be really competitive you will 'outsource' this analysis job to the systems engineering function, if that will make sure it gets done properly.
- 3. Value Delivery First
    - *Plan to deliver highest possible value to some stakeholders, as early as possible.*
    - This seems tougher than it is. We are so locked in conventional waterfall/grand design thinking about development, that we are not trained and cultured to ask the right questions (How can we deliver value early?). We are unlikely to see good answers even if they exist. One roadblock is that we fail to define the primary product objectives in terms of variable values (performance and quality levels). We move too quickly to product architecture and design, before we have established the real stakeholder values quantitatively.
    - But let us assume you have pinned down the critical qualities for stakeholder value satisfaction. Most engineers seem to make the mistake of going directly for meeting the final levels of performance by specifying an architecture and building the system/product.
    - There is a real possibility, always – in my experience, of delivering some of that performance increase, to some of the market, with some of the product partly developed. We just have to decide that 5% (of the final quality

improvement level) improvement, next month, is worthwhile, and figure out how to do it.

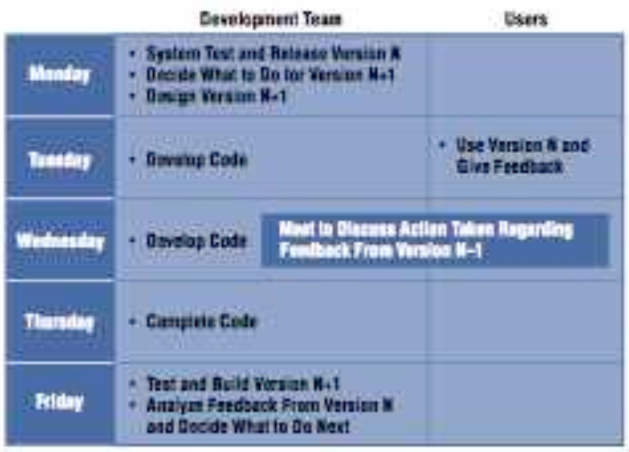
  - For example in the case of an advanced radar system, where the major ideas were delivery to a ship in 3 years, and having two radar antennas; we found that it was quite interesting to increase to the final level of 'accuracy of perception' (the major performance characteristic of the product) by building the target profile data up gradually, giving about 2% increase in perception capability at each cycle of improvement (one enemy plane at a time, most dangerous first) . We also found we could just as well deliver to existing ships, while we waited for launch of the new one.
  - Obviously the competitor who manages to deliver stakeholder value early, while their competitors are busy with late big bang projects, have a competitive advantage.
- 4. Learn Rapidly
  - *Get quantified measurements from early frequent increments of your product, regarding real quality levels and costs. Analyze this data in relation to expectations. Learn from deviations as rapidly as possible, and change plans to reflect realities.*
  - One major cause of large scale systems engineering failure [MORRIS] is that we do not imbed in our product development processes, good enough mechanisms for learning that our project is on a bad path. We learn, too late. Our systems are so complex that we have to take a very humble, traditional engineering, point of view and get early and frequent numeric feedback about 'everything'.
  - 'Everything' feedback includes feedback on costs, quality levels, development processes, staff quality, architecture, motivation, and management.
  - In reasonably small products we have succeeded in seeking, and getting, feedback, about emerging product qualities in stakeholder environments from the fist week and every week [FIRM]. Even in large DoD projects [CE, 9.8 Persinscom, US Army] we have been able to get 'next week' feedback (much to the DoD Amazement), by making

    small modifications to large existing systems, that had high effect. But in more likely cases regular feedback increments of a month [Example LAMPS system, Harlan Mills, IBM FSD] to a quarter [example Jet propulsion Labs, Control Rooms, Spuck] are possible.
  -
  - *Figure 4: each increment can be exploited to get data, learn and exploit the knowledge in immediate future steps.*
  - *By getting knowledge about new technologies, development methods and markets at the earliest possible moment, you obviously have a competitive edge over those who not.*
- 5. Delivery Frequently
  - *Plan to deliver increments of performance, quality and function on a frequent basis, for example weekly. This pace will be a useful discipline, ensure management control, and send positive messages to stakeholders.*
  - Most evolutionary development methods choose a constant delivery cycle duration [Larman] a week, two weeks and a month are regularly mentioned.

	Development Team	Users (PMT, Pros, Doc writer, other)	CTO (Sys Arch, Process Mgr)	QA (Configuration Manager & Test Manager)
Friday	✓ PM: Send Version N detail plan to CTO + prior to Project Mgmt meeting ✓ PM: Attend Project Mgmt meeting: 12.00-15.00 ✓ Developers: Focus on genereal maintenance work, documentation.		✓ Approve/reject design & Step N ✓ Attend Project Mgmt meeting: 12-15	✓ Run final build and create setup for Version N-1. ✓ Install setup on test servers (external and internal) ✓ Perform initial crash test and then release Version N-1
Monday	✓ Develop test code & code for Version N	✓ Use Version N-1		✓ Follow up CI ✓ Review test plans, tests
Tuesday	✓ Develop Test Code & Code for Version N ✓ Meet with users to Discuss Action Taken Regarding Feedback From Version N- 1	✓ Meet with developers to give Feedback and Discuss Action Taken from previous actions	✓ System Architect to review code and test code	✓ Follow up CI ✓ Review test plans, tests
Wednesday	✓ Develop test code & code for Version N			✓ Review test plans, tests ✓ Follow up CI
Thursday	✓ Complete Test Code & Code for Version N ✓ Complete GUI tests for Version N-2			✓ Review test plans, tests ✓ Follow up CI

- Figure 5: The regular cycle at [FIRM] which is essential identical to Hewlett Packard’s cycle (below [May]).

Fig. 3. An example of a typical one-week EVO cycle at the Manufacturing Test Division during project



	Development Team	Users
Monday	• System Test and Release Version N • Decide What to Do for Version N+1 • Design Version N+1	
Tuesday	• Develop Code	• Use Version N and Give Feedback
Wednesday	• Develop Code	Meet to Discuss Action Taken Regarding Feedback From Version N-1
Thursday	• Complete Code	
Friday	• Test and Build Version N+1 • Analyze Feedback From Version N and Decide What to Do Next	

-

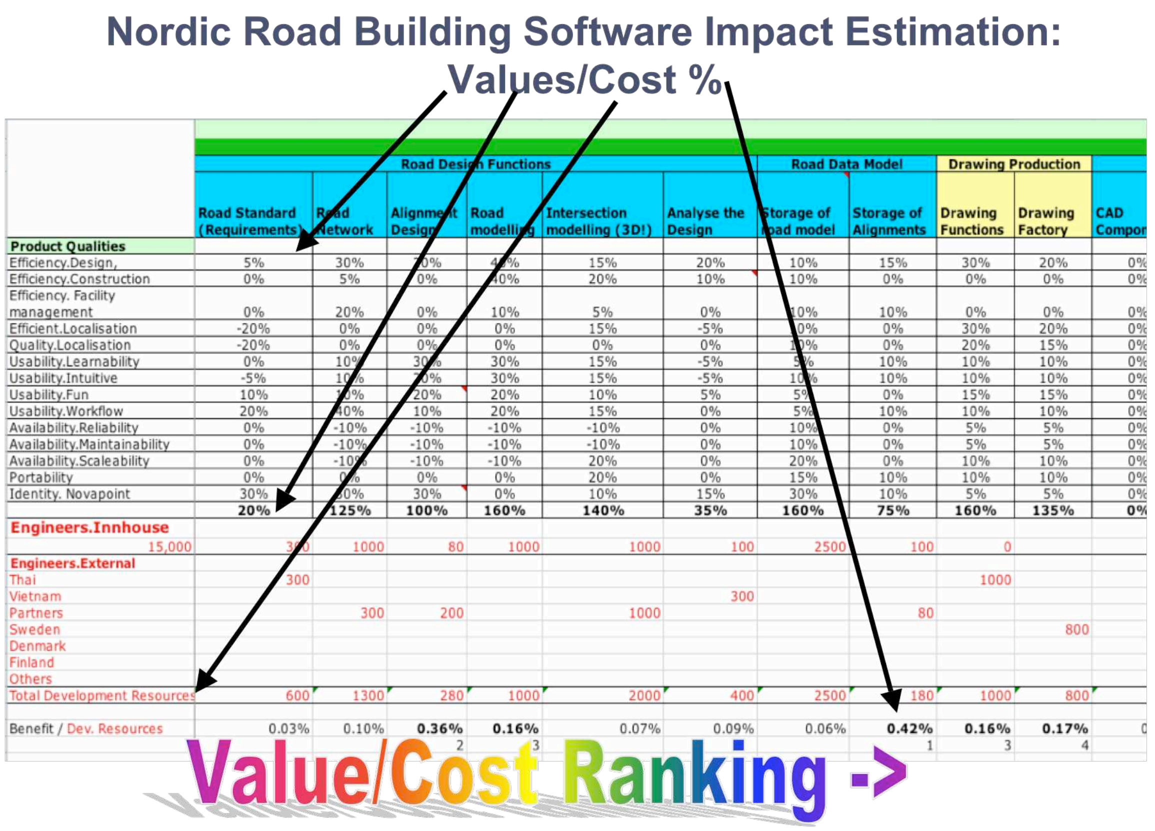
    - There are several competitive advantages of a frequent cycle. Here are a few.
        - The entire development process is tested and tuned at each cycle – for example integration testing. Badly organized and managed developments are exposed early before they can do competitive harm.
        - The opportunity to deliver high value early is more likely. This attracts customers and defeats competitors.
        - It is almost impossible to have severely delayed projects [Larman]. Market or customer confidence is built on results early and steadily
- 6. Deliver Early
    - *Get something out to stakeholders at the very beginning of your development project. This might involve improving existing products, or using some of the new ideas for new products. But it sends signals to the market, and makes sure your team is well grounded in reality.*
    - Getting something early to *stakeholders* (I did not say *final* customers) has the following advantages amongst others:
        - *Some* stakeholders, like sales and training, can better prepare for their roles when the product is released to the main customers.
        - *Some* markets can be used to field trial, and early market test the product
        - Current customers who might be tempted by competitors will remain interested and patient about the new product development if they can see real improvements to their old product, or at least credible demonstrations of the new product.
- 7. Hit the ground running
    - *Make sure that each product increment is industrial standard quality. No second rate prototypes.*
    - It would be easy to misunderstand that early increments were ‘quick and dirty’ prototypes. This need not be the case, and it should not be.
    - Harlan Mills [Mills], IBM Federal Systems Division, was always very clear about achieving final quality levels at

early incremental deliveries. Microsoft was clear about having ‘shippable quality’ for early milestones [Cusumano] which are evolutionary steps (of 6-8 weeks) before a release.
    - The competitive advantage of high quality early is, among other things:
        - You prove the basic architecture will allow you to achieve necessary high levels
        - Your early increments can be demonstrated in the field and will have tolerable quality for real users, and early adopters.
        - The true costs of development and of hardware components, to achieve the necessary industrial quality levels, is likely to be correctly understood.
        - You are less likely to get project delays while trying to get necessary quality levels.
        - If market or contractual deadlines must be met, you are more likely to be ready to do so without cheating on quality.
- 8. Quantify Valued Qualities
    - *All critical product qualities must be specified quantitatively, otherwise you have lost control of them.*
    - I regularly find on both systems engineering projects (aircraft for example) and software engineering projects that many of the acknowledged most-critical product qualities are not specified quantitatively. Example ‘intuitiveness” and ‘adaptability’ [CE, ch 5 for examples].
    - *"In physical science the first <u>essential step</u> in the direction of learning any subject is to <u>find principles of numerical reckoning</u> and <u>practicable methods for measuring</u> some <u>quality</u> connected with it.*
    - *I often say that when you can <u>measure</u> what you are speaking about, and <u>express it</u> in numbers, you know something about it;*
    - *but when you cannot <u>measure</u> it, when you cannot <u>express it in numbers</u>, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge,*

but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be."

- *Lord Kelvin, 1893, Lecture to the Institution of Civil Engineers, 3 May 1883* From http://zapatopi.net/kelvin/quotes.html
- In my practice we make it a point to define useful quantifications for all critical product qualities. We have found that they all can be quantified. This means we can define competitive levels of the critical qualities, and make sure they are in fact provably delivered – or not.
- People are not trained to do this. They are even convinced that certain concepts are 'qualitative' (and cannot be quantified).
- Those who take the trouble to quantify critical product qualities, where others do not bother will get a competitive advantage.
- One computer equipment manufacturer client of mine, with 23,000 employees made it serious corporate policy to quantify all qualities, on my advice, from the top down. They went into profit for the next 14 years, unlike any competitor. I like to think there was a connection.

- 9. Control Value-to-cost ratios
  - *Make sure you keep your eye on the value-to-cost ratio of each step. That will remind your staff that it is about profitability, not technology.*



Nordic Road Building Software Impact Estimation: Values/Cost %

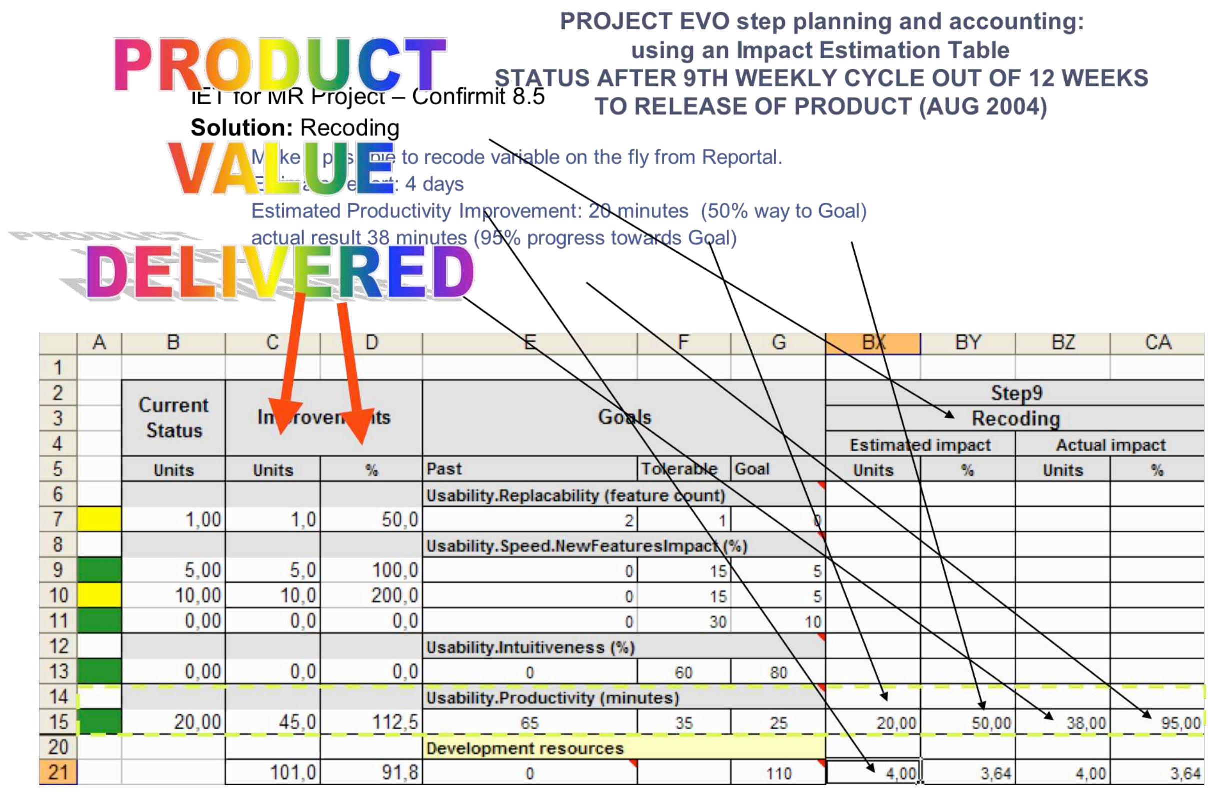
	Road Design Functions						Road Data Model		Drawing Production		
	Road Standard (Requirements)	Road Network	Alignment Design	Road modelling	Intersection modelling (3D!)	Analyse the Design	Storage of road model	Storage of Alignments	Drawing Functions	Drawing Factory	CAD Compor
Product Qualities											
Efficiency.Design,	5%	30%	20%	40%	15%	20%	10%	15%	30%	20%	0%
Efficiency.Construction	0%	5%	0%	40%	20%	10%	10%	0%	0%	0%	0%
Efficiency. Facility management	0%	20%	0%	10%	5%	0%	10%	10%	0%	0%	0%
Efficient.Localisation	-20%	0%	0%	0%	15%	-5%	10%	0%	30%	20%	0%
Quality.Localisation	-20%	0%	0%	0%	0%	0%	10%	0%	20%	15%	0%
Usability.Learnability	0%	10%	30%	30%	15%	-5%	5%	10%	10%	10%	0%
Usability.Intuitive	-5%	10%	20%	30%	15%	-5%	10%	10%	10%	10%	0%
Usability.Fun	10%	10%	20%	20%	10%	5%	5%	0%	15%	15%	0%
Usability.Workflow	20%	40%	10%	20%	15%	0%	5%	10%	10%	10%	0%
Availability.Reliability	0%	-10%	-10%	-10%	-10%	0%	10%	0%	5%	5%	0%
Availability.Maintainability	0%	-10%	-10%	-10%	-10%	0%	10%	0%	5%	5%	0%
Availability.Scaleability	0%	-10%	-10%	-10%	20%	0%	20%	0%	10%	10%	0%
Portability	0%	0%	0%	0%	20%	0%	15%	10%	10%	10%	0%
Identity. Novapoint	30%	30%	30%	0%	10%	15%	30%	10%	5%	5%	0%
	20%	**125%**	**100%**	**160%**	**140%**	**35%**	**160%**	**75%**	**160%**	**135%**	**0%**
Engineers.Innhouse											
15,000	300	1000	80	1000	1000	100	2500	100	0		
Engineers.External											
Thai	300								1000		
Vietnam						300					
Partners		300	200		1000			80			
Sweden										800	
Denmark											
Finland											
Others											
Total Development Resources	600	1300	280	1000	2000	400	2500	180	1000	800	
Benefit / Dev. Resources	0.03%	0.10%	**0.36%**	**0.16%**	0.07%	0.09%	0.06%	**0.42%**	**0.16%**	**0.17%**	0
			2	3				1	3	4	

Value/Cost Ranking ->

- *Figure 6: the use of an Impact Estimation Table [CE] to analyze the value to cost ratio of a number of alternative product development strategies for road building software.*
- *One lack of practice that is almost as unhealthy as not quantifying critical qualities, is to fail to look at the costs of individual technologies.*
- *Of course someone somehow came up with a cost estimate for the whole project, and got themselves a budget. That’s not it. We need to estimate costs at a more-detailed level. We need to use these cost estimates to make decisions about the value-to-cost ratio of our options. An impact estimation table (Fig.6) is a basic tool for helping us see these relationships.*
- *If you use this method of evaluating technological alternatives, you are more likely to be able to prioritize the high value to cost alternatives.*
- *This will aid your competitiveness by leading you towards more value for cost.*

- 10. Rapid Reprioritization

- *When stakeholder feedback, or measures of delivered value and cost dictate it, change your plans to maximize your profit.*
-


PRODUCT
VALUE
DELIVERED

PROJECT EVO step planning and accounting: using an Impact Estimation Table
STATUS AFTER 9TH WEEKLY CYCLE OUT OF 12 WEEKS TO RELEASE OF PRODUCT (AUG 2004)

IET for MR Project – Confirmit 8.5
Solution: Recoding
Make possible to recode variable on the fly from Reportal.
Estimated effort: 4 days
Estimated Productivity Improvement: 20 minutes (50% way to Goal)
actual result 38 minutes (95% progress towards Goal)

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7	🟨	1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9	🟩	5,00	5,0	100,0	0	15	5				
10	🟨	10,00	10,0	200,0	0	15	5				
11	🟩	0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13	🟩	0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15	🟩	20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

-
- Figure 7: An Impact Estimation table used to estimate expected benefit to cost ratios as a basis for selecting a particular strategy on an incremental step.
- We can evaluate real values and real costs against initial estimates, on a step by step basis. The feedback about reality can help us tune our estimations to be more realistic and to choose smarter options. This can improve the competitiveness of our development process.

Summing Up:

- Make critical product factors measurable
- Measure progress towards your goals often
- Learn rapidly from deviation
- Change fast, towards what works.

References

CE: Gilb, Tom, COMPETITIVE ENGINEERING: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering, Using Planguage. Elsevier 2005.

FIRM: **From Waterfall to Evolutionary Development (Evo)**
Or, How we rapidly created faster, more user-friendly, and more productive software products for a competitive multi-national market
Trond Johansen, Head of Project Management, FIRM A/S,
Trond.Johansen@firmglobal.com
Published in INCOSE 05 Proceedings. INCOSE.org. and EuroSPI, Trondheim 2004.

Also published in Torgeir Dingsøyr (Ed.): Software Process Improvement, 11th European Conference, EuroSPI 2004, Trondheim, Norway, November 10-12, 2004, Proceedings. Lecture Notes in Computer Science 3281 Springer 2004, ISBN 3-540-23725-9. http://www.confirmit.com/news/release_20041129_confirmit_9.0_mr.asp

Cusumano: Cusumano, Michael A. and Richard W. Selby. 1995. Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. The Free Press (A Division of Simon and Schuster). ISBN 0-02-874048-3. 512 pages.

May: May, Elaine L. and Barbara A. Zimmer. August 1996. The Evolutionary Development Model for Software. Hewlett-Packard Journal. Volume 47, Number 4, Pages 39–45. Available as pdf Adobe Acrobat file at http://www.hpl.hp.com/hpjournal/96aug/aug96a4.pdf

Larman: Larman, Craig and Victor Basili. June 2003. Iterative and Incremental Development: A Brief History. IEEE Computer. Pages 2–11. See www.craiglarman.com for a copy of this paper.

Mills: Mills, H. D. 1980. The Management of Software Engineering. Part 1: Principles of Software Engineering. IBM Systems Journal. Volume 19, Number 4. Reprinted 1999 in IBM Systems Journal. Volume 38, Numbers 2 and 3. A copy is downloadable from http://www.research.ibm.com/ journal/sj/194/ibmsj1904C.pdf/.

Morris: Morris, Peter W. G. 1994. The Management of Projects. London: Thomas Telford. ISBN 0 7277 1693 X. 358 pages. The American Society of Civil Engineers. The website http://www.indeco.co.uk/ has additional recent papers by Professor Morris.

Author Bio

Tom has been an independent consultant, teacher and author, since 1960. He mainly works with multinational clients; helping improve their organizations, and their systems engineering methods.
Tom's latest book is 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (Summer 2005).
Other books are 'Software Inspection' (with Dorothy Graham, 1993), and 'Principles of Software Engineering Management' (1988). His 'Software Metrics' book (1976, OoP) has been cited as the initial foundation of what is now CMMI Level 4.

Tom’s key interests include business metrics, evolutionary delivery, and further development of his planning language, ‘Planguage’. He is a member of INCOSE and is an active member of the Norwegian chapter NORSEC. He participates in the INCOSE Requirements Working Group, and the Risk Management Group.

Email: Tom@Gilb.com
URL: http://www.Gilb.com

Version Nov 9 2005