Real Requirements: How to find out what the requirements really are

Tom Gilb
Tom@Gilb.com

Abstract. This paper gives advice to help you find the requirements that really matter to your stakeholders. It focuses on stakeholder value delivery as the appropriate level, rather than the technical solutions that so many specify as their requirements. It suggests several specific methods for determining what the real requirements are. This includes some technology unique to this author, such as the Impact Estimation method.

INTRODUCTION

'Real requirements' are the requirements that express what key stakeholders really want; no matter what they, or marketing people, *tell* you they want. '*False* requirements' are *so-called* requirements that, at best, might *support* (that is, be some kind of 'design' for) the delivery of the *real* requirements. At worst, they bear no relationship to what people really want. Typically, false requirements are one of the following:

- A total misunderstanding;
- So badly formulated that they are misunderstood;
- A design or strategy that might at best work to *some* degree, but that will not *fully* satisfy the real needs;
- A design or strategy that people believe in, but in fact it will not do *any* good, and may well be negative, or at least, a dangerous drain on resources.

In my consulting business, I consistently meet systems engineers, managers, and directors, who are stuck at the level of false requirements. Most do not realize that they are struggling because of their requirements. None of them are trained to seek out the real requirements systematically.

SETTING THE SCENE

I am often asked to look at either technical requirements or management objectives. Invariably, I have to find out what the key requirements are: they are rarely properly specified. Instead, they tend to be stated as follows:

- Listed as 'critical business requirements,' but not quantified;
- Listed as 'Expected Outcomes' of the plan, but still in vague terms;
- Totally missing as an explicit specification, but implied by other specifications, such as the vague benefits, or vague 'top business requirements';
- Implied by a mass of technical specification;
- Implied by tasks. If you directly ask the managers, "What do you expect to result if this task is done?" They will name some result, but that result is not actually in the plan, and certainly is not *quantified*.

The aim of this paper is to provide you with some ideas on how to tackle these problems and find your real requirements. The basic 'plot' is as follows:

- Method 1: **Identify the Core Requirements**: Make sure you are dealing with the *real* top-level objectives (for example, ask someone knowledgeable, "What is the major point of all this?");
- Method 2: **Use Quantification**: *Quantify* the nice sounding words about benefits and critical business objectives;
- Method 3: **Evolve Requirements**: start with a reasonable initial set of the critical few requirements. Evolve throughout the lifecycle of the project and product the clarity, correctness and relevance of the requirements, by getting feedback from stakeholders;
- Method 4: **Analyze Stakeholder Value**: Keep your focus at all stages of development (requirements, design, reviews, testing) on the generation of acknowledged stakeholder value. Move value towards planned value targets, as set in requirements. Make sure the value to cost (profitability) relationship is formally considered.

To give some further guidance, here is a set of six underlying principles that these methods encompass:

- Principle 1: Extend the scope of a requirement so that it more realistically deals with the real business situation, or technical environment, from beginning to end, and it avoids unnecessary constraints in doing so.
- Principle 2: Focus on your strategic requirements, and don't get involved in your boss's 'fundamental' requirements, or your subordinate's 'means' requirements.
- Principle 3: Confront your stakeholders with your clarified interpretation of their vaguely articulated requirements, and ask them if that is what they really want. Then ask them if that is the most important thing for them, or if there is anything more important to them.
- Principle 4: Focus on *quantifying* the 'top ten' most critical requirements (Note: a maximum of 10!). They are the main reasons for project funding.
- Principle 5: You can only realistically expect to learn the true requirements by frequent result delivery to stakeholders, and consequent analysis of reaction.
- Principle 6: The 'Real Requirements' have the higher value-to-cost ratio; they are the most profitable ones to do.

CASE STUDY: A WEB SYSTEM

Ask an Initial "Why?"

Ask "Why?" and the answer will often take you to a 'more-real' requirement. Here is an example based on a UK client case in 2004:

> **Requirement: To reduce time and effort putting information on our websites.**
> *Michael: We are looking at $500,000 investment in better website software.*
> *Tom: Why?*
> *Michael: 'Time To Market'.*
> *Tom: What do you mean by that?*
> *Michael: 'I mean we need to get corporate information to the websites as soon as possible, with little effort on the part of our staff. Currently it takes too much time and effort, and is often forgotten.'*
> *Insight: Define the following requirement:*
> **Info Time To Market:**

Scale: Time from company opportunity exists to update information on the web, until it is first actually accessed by the intended recipients.
Past: Hours to days.
Goal: Seconds.

By changing the requirement definition from the narrow idea of faster web updates, to the broader idea of getting information from the point that it originates in the company to the point it (via a website) is actually accessed by its intended readers, the project gained as follows:

- It avoided sub optimization of the 'quick web update' alone;
- It can focus on ways to shorten the time from 'company events' until 'we start a web update';
- It can focus on ways to make sure potential web readers know they should access that update;
- It can evolve the results in a stream of practical changes starting with particular areas of the company, and the current web technology.

Ask "Why?" Again

However, don't stop there just because you get a 'more-real' requirement by asking one "Why?" More 'Why? Cycles' might be even better – up to a point. For example, take the previous example of defining the requirement in terms of 'time from 'event to web' and 'web to customer'. Ask the question "*Why?"* yet again. *"Well, to make sure that the market knows about our great prices and products as soon as possible."*

So, let us raise the level to an *even more real* requirement. Let us avoid the whole 'Web' idea. It is really a 'design' and not a *requirement*. We reformulate accordingly as follows:

Information Time to Market:
Scale: Time elapsed *from* when events create new information, *until* that information is in the hands of the intended customers and users.
Goal: Seconds.

If using the website is *sometimes* a good solution for this requirement, fine. However, if there are faster or more cost-effective ways of achieving it, then this new requirement dictates *avoiding* the web and using the better methods (like direct email, personal contact, telephone, SMS, MMS and TV Announcements).

Now let's discuss in more detail some of the key points being made by this case study. To do this, let's return to the methods and principles outlined earlier.

METHOD 1: IDENTIFY THE CORE REQUIREMENTS

Extending the Scope

Principle 1: In order to avoid unnecessary constraints, extend the scope of a requirement so that it more realistically deals with the real business situation, or technical environment, from beginning to end.

Often a requirement is unnecessarily, and unrealistically narrow. This is frequently because the focus is on a technical tool, such as a computer, rather than the complete business process.

In the case study, the example is the requirement 'to reduce the time to put information on our website'. This can be fruitfully expanded to include the opportunity time before actually starting the website update itself. This means the time from when someone or something should be able to realize that information should be updated on a website. It also should include the time

after the update, until one or more potential website readers actually access the information, or even until they successfully act on it.

The other key point is to extend scope by cutting out the arbitrary design concept. In the case study, this involved cutting out from the requirements all the explicit statements about using a website to hold the information. The important business point, the 'real requirement, being that the information is somehow in the hands of the people who can act on it.

Sort Out the Fundamental, Strategic, and Means Requirements

Principle 2: Focus on your strategic requirements; don't get involved in your boss's 'fundamental' requirements, or your subordinate's 'means' requirements.

Ralph Keeney (1992) suggests a simple scheme of separation of requirements into three categories:

- *Fundamental* Requirements are at your bosses level or above. They are outside your scope and direct responsibility - although your job is to contribute to them;
- *Strategic Requirements* are the requirements you set, and you are responsible for at your level of work. By meeting these requirements you will contribute to your fundamental requirements;
- *Means Requirements* are requirements that are exclusively in support of meeting your strategic requirements. You have delegated them to organizational levels below your own. You should not get involved in setting them, or working to meet them.

The main point here is to carefully avoid taking direct responsibility for anything except your strategic requirements. They are your 'real' requirements. The other ones will drain your energy from the real ones.

Stakeholder Focus

Always be systematic in identifying all your critical stakeholders, and use their values to help you see interesting requirement ideas. Normal projects have typically 10-40 identifiable stakeholders. Stakeholders have values that can be satisfied by one or more potential requirements for your product. If you focus too narrowly on a few stakeholders (such as just the users and the customer), you are going to miss dozens of real requirements. The higher the priority of the stakeholder, the more we need to focus on their requirements as realistic priorities. For example:

Stakeholder: Salesperson.

Function Requirement: Needs credible product demonstrations.

Design Idea: Deliver limited function, but deliver reliable and good user interface releases of the final product, that are suitable for sales demonstrations much earlier than the final product.

Asking the Stakeholders "Why?"

Principle 3: Confront your stakeholders with your clarified interpretation of their vaguely articulated requirements, and ask them if that is what they really want. Then ask them if that is the most important thing for them, or if there is anything more important to them.

No matter what signals, orally, written, reviewed or approved, the stakeholders have given you

earlier, there are several reasons why you may not really understand their 'real' requirements. These reasons include:

- Wrong Source: Someone else may have spoken for them (like a marketing person, a previous departed/failed manager, someone so high up in their hierarchy that they did not understand the real priorities);
- Our Misinterpretation: We may well have misinterpreted their vague ambiguous signals, and our attempts to clarify (using Ambition, Scale, Past, Goal specification language) might help to bring out our *own* misunderstandings. Stakeholders will not necessarily initially be clear about their real requirements. But, if we bother to go through this feedback and correction cycle, we have a chance to find out what the 'real requirements' are;
- Changed Needs and Values: They may have changed their ideas about requirements since the initial data was captured. This can be due to management changes, experiences, competition, new technology, new economics, new laws and regulations. There are plenty of reasons for changes. We have to accept that, and capture and update their real requirements quickly. And, thank them for the update – and remind them that we are eagerly awaiting any other updates they may have at any time. We should not ever berate them for not giving us these requirements earlier!

Asking Five (5) 'Whys?'

As part of Root Cause Analysis, Taiichi Ohno recommends asking "Why?" five times whenever a problem is encountered. The aim being to identify the root cause of the problem, so that effective countermeasures can be developed and implemented. Ohno attributes his problem-solving approach (the so-called '5 Whys') to Sakichi Toyoda's sense of how to spend time well. "Stand on the production floor all day and watch," Ohno urges would-be problem-solvers, passing on the Toyoda legacy of careful, patient observation.

So I am recommending using this technique, initially intended for root cause analysis, for a related purpose of *finding the core requirement*. Effective designs can then subsequently be determined, and applied, to solve the defined root (core) problem.

Stop Asking 'Why?' When….

Continue asking 'Why' until:

- You have clearly reached your boss's level of concern;
- You are clearly outside of anything you can hope to influence;
- You have clearly made progress in an interesting direction, and can use it to discuss with others about whether to continue or not.

METHOD 2: USE QUANTIFICATION

Expressing requirement levels using *numbers* is a basic and powerful technique for finding out what the real requirements are. By quantifying, we move from highly ambiguous words (such as 'highly portable' or 'very secure') to testable clarity about the real requirements.

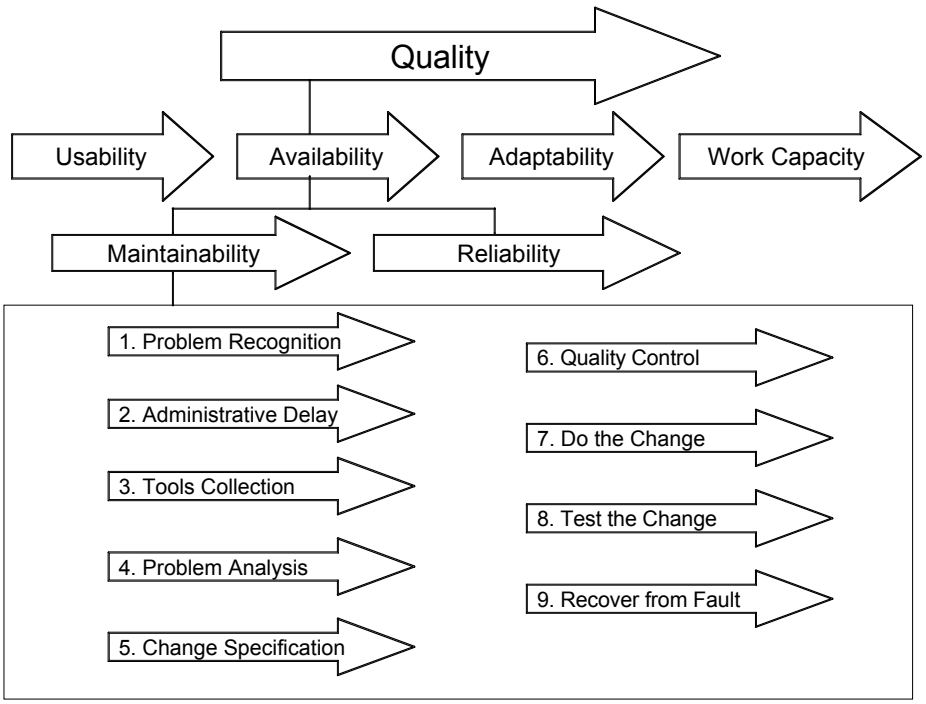

Figure 1 shows a quality hierarchy for the complex requirement, 'Maintainability'. 9 distinct sub-attributes are defined.

In my experience, in addition to conventional quantification of speed, volume and costs, *all* quality requirements can be expressed *quantitatively*. The process for doing this is described in (Gilb 2005, Chapter 5: Scales). The basics of quantification include:

- Copy known quantification ideas; an Internet search will expose much of this knowledge. (Gilb 2005) gives some basic patterns, such as for Adaptability;
- Be prepared to decompose complex qualitative ideas into a set of different scales of measure. There is often no single scale of measure that will adequately cover all the measurable dimensions of the concept we are trying to define. Real requirements are often complex. I have examples of Usability using 10 scales of measure (for example, Learning Time), Maintainability with about 10 (for example, Administrative Delay Time), and Adaptability with about 18 (for example, Data Portability).

One practical use of a requirements hierarchy is that if we are initially given a requirement lower down in the hierarchy, we can work on the hypothesis that the real requirement is further up the hierarchy. For example, when people speak about reliability, they often really care most about the availability of the system, but they don't articulate this because they are not trained to think this way.

Find the 'Top Ten'

Principle 4: Focus on <u>quantifying</u> the 'top ten' most critical requirements (Note: a maximum of 10!). They are the main reasons for project funding.

People often hand me dozens of pages of 'Functional Requirements' and 'Non-Functional Requirements'. Because of all this detail, the *really* critical and key requirements are either given only as a brief summary, or they are invisible. We quickly lose track of the main reasons for the project.

Here is an a real extract of some 'top level benefits' as handed to me by a client:

"Projected benefits of this include:

- *reduced time lost in planning;*
- *quicker identification of actual and potential operational problems;*
- *reduced time in vehicle tracking for customers and internal purposes;*
- *better matching of operational costs and effort to sales contracts;*
- *better information for future contract negotiations & renegotiations".*

Source: Transportation Business, Europe 2004.

The benefits are, *nowhere* in the 37-page requirement document, made more specific or quantified. However, the following level of detail is typical for the greater volume of specifications in the same document:

"The following header information should be available for any single unit:

- *VIN,*
- *tracking number,*
- *chassis number/ serial number,*
- *sales order number,*
- *customer's batch number*
- *make, model type (and model),*
- *colour and option information (if available)*
- *most recent location and date that this vehicle was tracked,*
- *status & condition,*
- *date status changed,*
- *final destination & dealer (if specified)." (op.cit.)*

Further, this is a typical example of the level of detail specified for functions:

> *"It must be possible to find the current disposition of all cargo against a single contract breakdown, showing numbers of units at current locations and predicted numbers at future locations for as far as the current schedules run." (op.cit.)*

This specification was automatically numbered (that is, no long term identity tags created). There was no source, no rationale, no time limit for the report. Also there was no reference to any assessment of the current technical feasibility, nor of the ability of the organization to carry out this project.

METHOD 3: EVOLVE REQUIREMENTS

Principle 5: You can only realistically expect to learn the true requirements by frequent result delivery to stakeholders, and consequent analysis of reaction.

Learn evolutionarily what people really need; there are some requirements that we will never learn about in office meeting rooms. We need to deliver evolutionary system changes (according to perceived requirements) and keep in dialogue with the affected stakeholders. Analyze their reactions, study their activity, solicit their current opinions, and even solicit information from people affected by their use of the system (their 'customers').

	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

Table 1: The weekly evolutionary delivery cycle at Future Information Research Management (FIRM) company in Norway. Monday and Tuesday are regularly set aside for learning what the step result user stakeholders really think and want, compared to what we thought they wanted. (Johansen04). This was based on a model used at HP - see Table 2.

	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	

Table 2: An example of a typical one-week Evo cycle at the Manufacturing Test Division during a Project (May 1996).

METHOD 4: ANALYSE STAKEHOLDER VALUE

Principle 6: The 'Real Requirements' have the higher value-to-cost ratio. They are the most profitable ones to do.

You should be able to estimate, and later measure delivery of, the 'value set' expected from implementation of each requirement option (using Impact Estimation methods, Gilb 1998 and Gilb 2005).

It should be possible to estimate and measure the *set of costs* associated with a requirement. It should also be possible to measure actual costs after implementation, maybe by using pilot evolutionary step implementation before full scale-up, major stakeholder delivery, or product release.

As a consequence of this numeric value and cost feedback, it should be possible to get a clear picture of the requirements that have the highest values for the associated costs. On this basis it should be possible to get agreement with stakeholders as to their *real* requirements. These should be the ones with the highest value-to-cost ratios. This is close to a notion of 'profitability'.

The subtle point here is that the stakeholders are often unaware of several critical factors initially, when declaring their 'real' requirements. These factors are necessary to understand reality, in order to make realistic judgments and trade-offs, rather than just demanding certain levels of system performance:

- The *real* primary effect to be expected, based on past experience;
- The real primary effect *actually deliverable* in *our* environment;
- The *side effects* (other secondary impacts on critical requirement areas, such as portability, security and performance). These side effects can be positive and negative. They must be ascertained by estimation, and later by measurement, if we are to make a proper judgement of 'real' requirements.

The stakeholder/customer/user/marketing viewpoint is usually limited to a *narrow* (one performance/quality dimension, such as usability or portability) and *vague* (not numeric, and not based on any real measurement or contractual guarantees of delivered performance level) understanding/appreciation/belief/expectation. It is the job of the systems engineering professional to shed light on the facts for the stakeholder, so that they can make more informed decisions about their real needs.

In case the reader is not clear on my chain of thought here: I expect that any performance requirement will need some level of practical design engineering, to find real technology or architecture to implement it. That technology should have calculable impacts on one and more performance-quality aspects. If not calculable, then we cannot fairly select a design on the basis of it satisfying even one single one of our performance requirements to any degree. Once a satisfactory level of performance and quality impacts is ascertained, I assume the costs (time, effort, money, memory space) for initial development and operational use can be estimated, and

later measured in pilot evolutionary steps, to complete the picture of values/costs ratio. This picture gives a professional basis for determining the final acceptable real requirements.

The Priority of a Requirement

The highest priority requirements are probably the most 'Real'. But how do we determine the priority of a requirement? Priority can be based on a complex and changing set of data.

Here are some basic ideas. Priority is related to legal requirements, survival constraints, value, and potential long-term value. For example, you can evaluate a set of requirements for the time-savings they would give some stakeholder, such as corporate users. The biggest estimated savings are probably your highest priorities. Make sure those requirements are well formulated as quantified and testable items. See Table 3.

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000

Table 3: A set of results achieved by one client (Johansen 2004).

Here is how the client formulated the second requirement:

> ***Usability.Productivity****:*
>
> ***Scale:*** *Time in minutes to set up a typical specified MR (Market Research) report.*
>
> ***Past:*** *65 minutes.*
>
> ***Tolerable:*** *35 minutes.*
>
> ***Goal:*** *25 minutes.*
>
> ***Meter:*** *Candidates with Reportal experience and with knowledge of MR-specific reporting features perform a set of predefined steps to produce a standard MR Report.*

My client commented, "The focus is here on the day-to-day operations of our MR users, not a list of features that they might or might not like. We *know* that increased efficiency, which leads to more profit will please them (For example, 45 minutes multiplied by thousands of reports [will

save a considerable amount of money]").

Requirements	Design Idea: Step 9 - Recoding			
	Estimated Scale Impact	**Estimated Percentage Impact**	**Actual Scale Impact**	**Actual Percentage Impact**
Objectives				
Usability.Productivity 65 <-> 25 minutes Past: 65 minutes. Tolerable: 35 minutes. Goal: 25 minutes.	**65 – 20 = 45 minutes**	**50%**	**65 - 38 = 27 minutes**	**95%**
Resources				
Development Cost 0 <-> 110 days	**4 days**	**3.64%**	**4 days**	**3.64%**

A detail of the real table below.

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9 Recoding			
3								Estimated impact		Actual impact	
4											
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

Table 4: An example of estimating the impact of a set of design ideas on a quality: Usability.Productivity. The goal was to reduce a task from 65 minutes to 25 minutes duration. The estimate was that the set of design ideas (Recoding) in Evo Step 9 would reduce the task time by 20 minutes (50% of the way to the goal), and would cost 4 days of effort to implement. At the end of the weekly increment cycle (when implemented), it was measured to actually reduce the time by 38 minutes (95% of the way to the long term requirement goal). It still only cost 4 days of effort, or 3.64% of the total budgeted effort (Johansen 2004).

SUMMARY

You can usually assume that the initial requirements statements, and even sometimes the *approved* versions of requirements, are not really the most important requirements. The real requirements are probably hidden. They are 'upstairs' - at a higher level of abstraction - ask the stakeholders "Why?" to get there (METHOD 1: IDENTIFY CORE REQUIREMENTS). Always clarify to remove any initial misinterpretations of vague and ambiguous requirements.

One excellent means of achieving clarity is to use quantification (METHOD 2: USE QUANTIFICATION).

Some real requirements are not available *initially*; they must be learned evolutionarily from practical feedback and interaction (METHOD 3: EVOLVE REQUIREMENTS).

The real stakeholder value levels and cost of delivering that value need to be ascertained (METHOD 4: ANALYZE STAKEHOLDER VALUE)

The energy needed to determine the real requirements is trivial compared with the energy lost striving to deliver false requirements.

REFERENCES

Gilb, Tom, "Impact Estimation Tables: Understanding Complex Technology Quantitatively", Crosstalk, December 1998. This article can be found in its entirety in PDF format on the Software Technology Support Center Web site at http://www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/12/gilb.asp [Last Accessed March 2005].

Gilb, Tom, Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, Elsevier Butterworth-Heinemann, 2005. ISBN 0750665076.

Johansen, Trond, (FIRM AS), *From Waterfall to Evolutionary Development (Evo) Or How to create faster, more user-friendly, and more productive, software*, EuroSPI Talk, Trondheim, Norway, November 2004.

Keeney, Ralph L., *Value-focused Thinking: A Path to Creative Decisionmaking*, 1992, Harvard University Press, Cambridge Mass/London. ISBN 0-674-93197-1. http://www.fuqua.duke.edu/faculty/alpha/keeney.htm [Last Accessed March 2005].

May, Elaine L., and Barbara A. Zimmer, The Evolutionary Development Model for Software, *Hewlett-Packard Journal*, August 1996, Vol. 47, No. 4, pages 39-45. http://www.hpl.hp.com/hpjournal/96aug/aug96a4.htm [Last Accessed March 2005].

Young, Ralph R., *Effective Requirements Practices*, Boston, MA: Addison-Wesley, 2001.

Young, Ralph R., *The Requirements Engineering Handbook*, Norwood, MA: Artech House, 2004.

Acknowledgement: This paper was edited by Lindsey Brodie (lindseybrodie@btopenworld.com), whose deep understanding of the methods has helped greatly to organize the ideas and present them intelligibly.