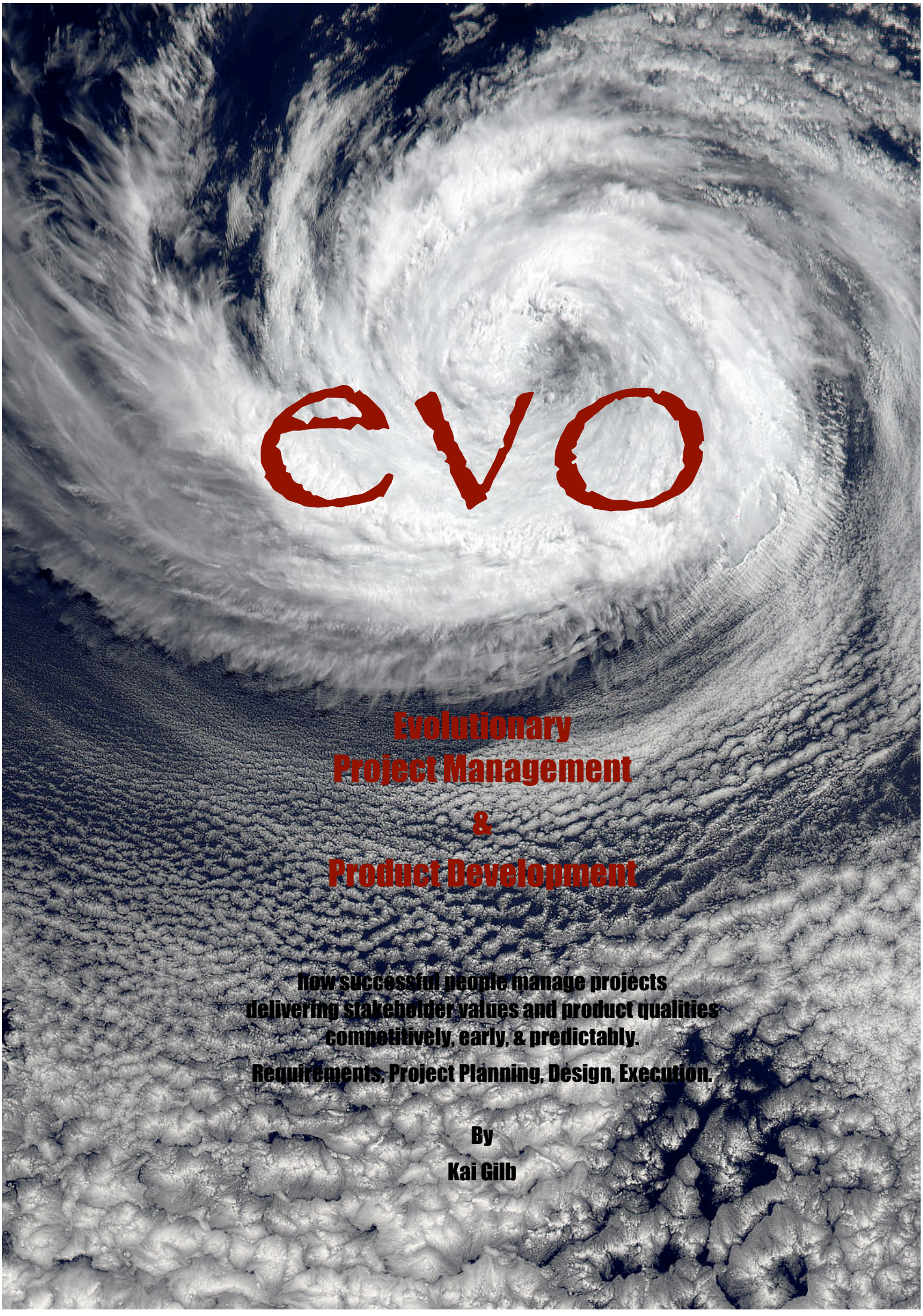

evo
Evolutionary
Project Management
&
Product Development
how successful people manage projects
delivering stakeholder values and product qualities
competitively, early, & predictably.
Requirements, Project Planning, Design, Execution.
By
Kai Gilb

Table of Contents

Introduction to Kai, Tom and the methods

 For more information on Kai Gilb & Tom Gilb see: http://www.Gilb.com

Many multinational companies practice Evolutionary Project Management. Several have adopted the methods, or parts of them, as company policy for how to develop and deliver products, as well as how to run the whole organization from the top. The methods are also practiced by small companies as well as by individuals on their own little projects. Evolutionary Project Management scales gracefully from the largest to the smallest projects.

Evolutionary Project Management is also practiced on a wide variety of challenges. Our own clients span from electronics, technology, banks, organizational management, military, telecommunications, internet companies, software developers, aircraft and aerospace, railroad, as well as spiritual, aid & environmental organizations. The methods are applicable to any type of planning, project, management, development, creative process and thinking process.

It seems everyone who ever learned Evolutionary Project Management appreciates how powerful it is compared to their old methods. History is also proving that when using other methods, project delays and cancellations are normal, while when using Evolutionary Project Management people learn to expect success. For the people and organizations that understand the methods they are obviously more powerful. The biggest challenge seems to be in changing organizational cultures and individual habits and paradigms.

Evolutionary Project Management was initially developed by my father and partner Tom Gilb in practical use together with a long list of clients. I joined Tom professionally in 1992. Much inspiration has come from the work of Dr. Deming and Dr. Jurans work on statistical manufacturing & management methods. In addition to the organizations using these methods, there is a long list of professional individuals contributing as well. Independently of Tom and me, other organizations and people have also developed methods that build on similar principles.

Evolutionary Project Management has many names among our clients. In this book I have chosen to use Evolutionary Project Management or Evo. Evolutionary as in quickly evolving towards Stakeholder Values & Product Qualities while learning through early feedback. Other names for Evo include; Evolutionary Delivery, Evolutionary Management, Requirements Driven Project Management and in Toms latest book he calls it Competitive Engineering.

When understood, Evo is very simple and can be used by anyone. It's less complicated than most other project management methods, yet it will enhance the most challenging and competitive of projects. It uses advanced methods of measurement and control, yet the application of it is so practical and simple and it supports creativity. All of our clients have in common that they want to improve, most of them want to be the worlds best in their field, or to keep that position.

Intended Readers of this book

Program managers, project managers, project planners, anyone planning anything, people developing products, contract writers, engineers and people managing outsourcing and purchasing are examples of intended readers. The book is written to be broad in its application, and to be accessible to the readers.

The management of Projects, a brief history

<- Quote summary chapters

Overview of Evolutionary Project Management

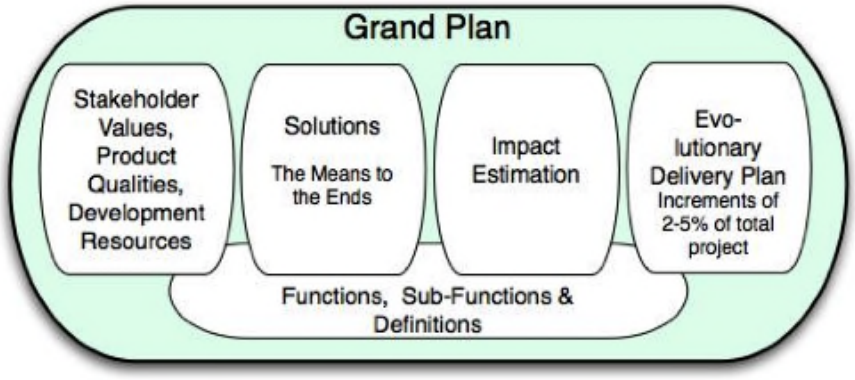


Illustration: The central elements of Evolutionary Project Management are:

Stakeholder Values & Product Quality Requirements & Development Resource Budgets
- how much good stuff & for how much Resources,

Solutions
– ideas on how to reach the Stakeholder Values & Product Quality Requirements within Development Resource Budgets,

Impact Estimation
– maps the Solutions to the Stakeholder Values, Product Qualities & Development Resources to see if we have adequate and powerful ideas to meet the Stakeholder Values & Product Quality Requirements within the Development Resource Budgets,

Evolutionary Plan
– initially a rough plan of the sequence, and how we will develop and evolve towards the Stakeholder Values & Product Quality Requirements. Necessary details of the Evolutionary Plan evolve together with the rest of the project as we develop the product/service,

Functions
– describes what the system does,

Definitions
– describes words and concepts.

Evolutionary Project Management consists of commonsense ideas and principles organized into a practical method. When understood, the ideas and principles are obvious to most everyone, except where they contradict what is currently practiced by an individual. Then additional time and open-mindedness can be necessary to drop the old habits.

In Evolutionary Project Management we start by understanding who the Stakeholders are, what level of improvements they want and what level of Product Qualities are necessary to give the Stakeholders the improvements they want. We call it Stakeholder Value & Product Quality Requirements. In a fairly traditional way we list Project Development Resources like money, time & people, but we do not commit to the whole budget at once. We only commit to using a small part of the allocated resources at a time to prove that we know how to deliver value early. The focus of any project will be to deliver the Stakeholder Values & Product Qualities within allocated Development Resources.

All Stakeholder Values & Product Qualities are variable; they vary from worse towards better. We identify the variables and specify where we are, and where we want to go, a Goal level, and when we want to be there, a date. We can add other points on the variables, such as where the competition is, and where we think the market will be in the future.

When we thoroughly understand where we want to go, the Stakeholder Value & Product Quality Requirements, and when we want to be there, we identify potential Solutions for getting there. Using an Impact Estimation Table, we engineer the Solutions as best we can to optimally meet our Stakeholder's Value & Product Quality Requirements.

We develop a step-by-step plan called 'Evolutionary Delivery Plan' for delivering, not Solutions, but improvements to Stakeholder Values & Product Qualities. Initially the Solutions and the Evolutionary Delivery Plan is at a high overview level.

We begin developing and delivering improvements to Stakeholder Values & Product Qualities. Picking ideas from the Solutions, creating Evolutionary delivery Cycles, we detail, develop, control the Product Qualities & deliver improvements to real Stakeholders, or as close as we can get to them.

From the beginning, we collect real-time feedback on the actual improvements the Evolutionary Deliveries have on moving us towards our Stakeholder Value & Product Quality Requirements, as well as on actual Development Resource consumption. From the feedback we learn what works and what does not, what we understand and what we don't, where there are challenges, what we did not know from the beginning, and about new technology and techniques that were not even available when the project started.

We adjust everything, as needed, based on what we learn during development. We do not even necessarily detail Solutions or Evolutionary Deliveries before they are the next ones to be developed and delivered.

This overview explains parts of the method in principle, but there are too many possibilities and variations that require more than an overview to understand. For instance, when projects are big and complicated, several layers of hierarchy and several concurrent Evolutionary Delivery cycles can be used.

Stakeholder Values & Product Qualities

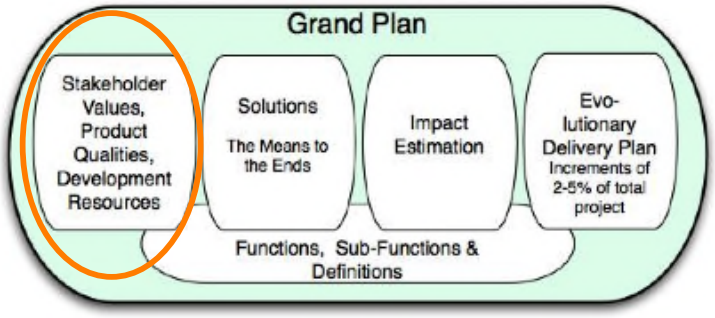


Illustration: First we will look at advanced ways of thinking about, then specifying Stakeholder Values, Product Qualities & Development Resources.

Stakeholder Values and Product Qualities describe the areas of improvements a project is intending to deliver. I divide the improvements into two categories, Stakeholder Values and Product Qualities.

Stakeholder Values describe the improvements a Stakeholder needs or wants with or without a specific product.

Product Qualities describe the quality attributes of a specific product or service. It is normally intended that the Product Qualities will deliver improvements on the Stakeholder Values.

Other names for Stakeholder Value & Product Quality Requirements include: Objectives, Strategic Goals, Requirements, Aims, Ends, Targets, Purposes, Ambitions, Qualities, Non-Functional Requirements, Intentions and Tom Gilb calls it Performance Attributes in Competitive Engineering.

It is not so critical what we call it (hint, stay away from the term non-functional requirement;-), but it is essential that we understand what it is, and how to express it in such a way that we finally can deliver the actual real need the project is intending to deliver.

This chapter will clarify the difference between ends (Stakeholder Values & Product Qualities) and means (Solutions or Designs), it will discuss the importance of understanding, specifying, communicating the Stakeholder Values & Product Qualities. We will discuss what Requirements are and how they are different from Stakeholder Values & Product Qualities. Finally, I will show a powerful way to specify Stakeholder Value & Product Quality Requirements in a practical way that is clear, measurable & testable.

Stakeholder Value & Product Quality Requirements. Where are you?

If you have managed or taken part of a project before, there is a chance you thought you succeeded, did OK, completely failed, or that you have no knowledge of how well you did.

Picture: Robin Hood suit, bow and arrow over shoulder, busy painting a target around a shot arrow.

> Guaranteed 'success' principle: If we have no interest in the actual success of a project, state the Stakeholder Value and the Product Quality Requirements; unclearly, without numbers, and in a way that can not be tested. Whatever happens, claim success!

If our project is complex & competitive and we want to create value, we must know:

- who the critical Stakeholders of the project are.

- what improvements the Stakeholders desire.

- how much the Stakeholders are willing to pay to get those improvements.

- how all this changes in time.

Then we must know how to deliver the desired improvements to the Stakeholders.

When the desired Stakeholder Values are delivered, within the Development Resources, then we can truly call our project successful.

Most project planners' do not have a clear idea of what Stakeholder Values & Product Qualities are. Under the headline of Requirements they simply state ideas that seem to be good for their project, they mix in Requirements, Solutions and Functionality without feeling any guilt.

By learning to identify the real end state Stakeholder Values & Product Qualities, and the art of specifying them clearly, measurably and testable, we will be years ahead of those who don't know how. It will open up for creativity, enabling us to meet deadlines to predetermined Stakeholder Value & Product Quality levels within Development Resources, enabling us to thrive on change, beat our competitors and to realize our dreams.

Here is a challenge for you. In these real life examples, taken out of professional project plans for hundred-thousand-dollar projects, can you separate the real Stakeholder Values & Product Qualities from the Solutions? Do so by underlining all words containing real Stakeholder Values & Product Qualities.

<Kirkens Nødhjelp Example>
<Ericsson Example>

Interactions are Qualities

To my limited knowledge, everything within the known universe interacts in some way with other things in the universe. If something did not interact with humans in some way, we could not even know it existed, except by theory or imagination, even that in itself is some kind of interaction.

Illustration: Interactions are everything. We would not even know of an apple, had it not been for the interactions between the apple and us. We see, feel, taste and smell it etc. These interactions between the apple and us, I call interacting qualities or product qualities. The qualities of an apple (or any product) are what make us appreciate it. It might look-, smell-, feel-, great or less so. It will be healthy or less so. It will store well or rot fast.
We choose to buy and eat one food (or any product), rather than another, based on the qualities that we want. To develop or buy an apple we must understand its qualities, the same holds true if we want to successfully develop any product or service.

<<<Picture: lots of things with radiating, communicating interaction qualities. Examples; boy, computer, car, mirror, ocean, starving boy, carrots, link them together with interaction arrows.>>>

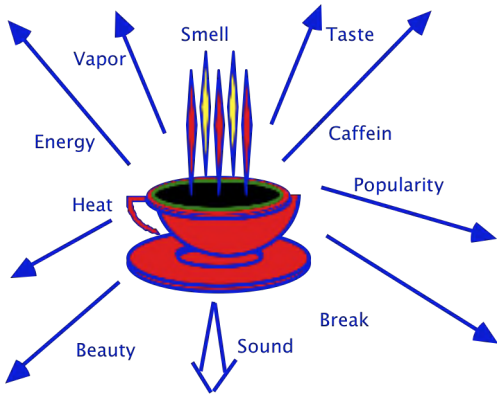


Illustration:

All the interactions, what I call Product Qualities, are what makes a product appealing or not, something a person would buy, or not, why we pick one product over the other. A person likes one thing and not another. Why? They like the way it looks, or the taste, or that it rarely breaks down, or because it is easier to use, or faster, or that it brings them health, wealth or a smile.

The Interactions

Relationships are everything, the thing in itself is not the point, the thing in itself is not a Stakeholder Value or Product Quality. Take a beautiful 200 year old vase that has been in the family for generations, and that you inherited from your grandmother.

What makes this vase valuable to you? It is its qualities, its relationship with you; memories, its beauty, its venerability.

To another person, not a family member, not knowledgeable about its age and history, the vase has different qualities. It might be useless, worn and without value?

Think of something you have or want that is special to you! What makes it special?

Let's say it is a friend, what makes this friend special? Is it that he has two arms? a head? lungs? toes? Clearly not! It is his qualities that makes him special to you. The joy, the sadness, the ability to listen, understand, relate, love, respect. Yet project management methods typically do not specify these critical qualities (humor, love, respect, or user-friendliness, security, reliability, portability). But list out the parts (arms, head, lungs, or GUI, password, inspection, object oriented, user manual, titanium). Completely missing what is important.

Let's buy a car!

Have you ever bought a car? or a bicycle? What makes you decide to buy a particular car? How is one car different than another car?

The 'What' or the Function

A car is a car. We do not decide to buy one product, 'Car', over another product. 'Car', because of what it is, they are both a 'Car'. 'Car' is what it is, and it is called 'Car' because it has a fundamental set of Functions. 'Car' refers to a set of Functions that all cars have, something like transferring people from a to b on roads. Functions describes what it does. It is binary, it is either a car or it is not a car. It does not vary. Ford makes cars, Volkswagen makes cars, Honda makes cars, and they are all the same as far as what they do, their Functions. If you want a Truck, Van or a Motorcycle, or something that is not a car, then you need it to do something else than what you would expect a car to do.

A Function is binary, it is either there, or it is not there.

We can also divide Functions into Sub-Functions. The Sub-Functions makes up the Functions of a car. These Sub-Functions are also binary, either they are there or they are not there. I find that Sub-Functions are part of the Solutions to Stakeholder Values & Product Qualities. Solutions contain Functions with Qualities as well as that which delivers the Functions with its Qualities.

Illustration: All cars have more or less the same Sub-Functions. We don't select cars based on the Sub-Functions. We expect all cars to have a given set of Sub-Functions, like steering, breaking, an engine etc.
Sources: www.history.rochester.edu/steam/brown/sailing-car.jpg

The 'How well' or the Product Qualities

The Product Qualities is what differentiates cars (or any product that has the same Function), the interactions between the car and people, or the car and other things, that make us desire one car over the other.

Its cost, comfort, acceleration, braking ability, its precision in steering, fuel mileage, style and beauty, its maintenance costs, road handling, ease of opening the hood, intuitiveness of the air-conditioner, intuitiveness of adjusting the seats, space, environmental pollution, quietness, etc., as well as customer satisfaction, friendly sales staff, image etc. are all examples of what I call Product Qualities that we might evaluate when buying a car.

Illustration: The car industry serves the market by offering cars with a verity of Product Qualities. Considering safety, performance, comfort, handling, power, economy, cost etc. the car designers put together cars with product qualities that suit most needs. One car is not necessarily better than another, it has different Product Qualities suited for different needs, different Stakeholder Values. What differentiates one car from another? Its Product Qualities!

The 'How' or the Solutions

Different Solutions create diverse Product Qualities in cars. It is the specialty of the designers and engineers, using Solutions, to create a car that has the desired Product Qualities.

How they design and built the car, is not really of concern for us as users, as long as it delivers the desired Product Qualities.

Illustration: If you think the 'How" should be the users concern, think of it this way. If you require a specific tire, ask yourself why, and you will come up with Product Qualities that end up in the 'How well' section. Product Qualities like grip, endurance, maintenance, style, etc. If you require a certain engine, with certain amount of horsepower, ask yourself why? Again, you will end up with Product Qualities that end up in the 'How well' section; acceleration, reliability, maintenance, sound, pollution, fuel efficiency, etc.

The 'How', I have chosen to call Solutions in this book. It is up to the engineers or the people developing the product or service to find the best Solutions, it is their job to, understand the technology, have the knowledge, invent when necessary, the optimum set of Solutions to create the desired Product Qualities.

Why people communicate 'How' and not 'How well'

Mostly people communicate the 'How' Solutions and the 'What' Functions instead of the 'How well' Product Qualities. It is not our tradition to communicate 'How well Product Qualities', it is not thought in school. We are not skilled at communicating the 'How well' Product Qualities and interactions. People assume that if they get ABS brakes, they get great braking qualities, that if they get an air-bag, they get safety. Most people find it easier to express braking quality specifying the technology that gets us the desired quality rather than specifying the braking quality directly.

For people in sales it might be good enough list the technology that gives the desired Product Quality, but if we want to excel in product development and project management, it is essential and necessary to be able to think, compare and specify Product Qualities directly.

Why we should first communicate the 'How well' Product Qualities and not the 'How' Solutions

If we specify the How, the Solutions, in a Requirement document, then engineers, developers and project implementers usually feel they have to implement what is specified, even if they know about or could find better ways of achieving the desired results. If we do not have to find the best possible solution, but just implement what is specified, creativity dies, competitiveness dies, engineering dies, and eventually, as the competitors improve, business dies.

If we specify the How well, the Stakeholder Values & Product Qualities we want to achieve first, we can leave it to the creativity and engineering expertise of the developers to find the optimum How, the Solutions. They can systematically develop a product to optimally meet all the Stakeholder Values & Product Qualities simultaneously within the resource constraints. They can change the Solutions if the Solutions do not give the expected results. They can add Solutions late in a project that were not available at the start. Engineers can keep current on technology and development processes within their field. They can use their creativity, engineering skill & know-how to find ingenious ways of creating better products than the competitors.

Example:

Solution stated: In a Requirement specification it is stated that the car must have an engine of a specific type and model.

Product Quality stated: Acceleration: In a Requirement specification it is stated that the car must be able to accelerate from 0 to 100 Km/hour in 6 seconds.

Where the Solution (engine) is stated directly, the developers have little choice but to implement the specified engine, whether it gives the intended Product Qualities or not. Where the Product Quality (Acceleration) is stated the developers are free to find whatever Solutions best give them the desired Product Qualities. They also have the opportunity to use the specific engine specified in the above example.

Let's choose between two cars.

One car has ABS brakes, another doesn't. Which car would you choose?
Well, let me also tell you that the car with ABS brakes uses twice the distance to stop as the car without, even if turning is necessary during braking. Are you sure you want the car with the ABS brakes?

Again let's choose between two cars

'Car A' has air-bags, and 'Car B' doesn't. Which car do you choose? This time you might have stopped to reflect. What if 'Car A' with air-bags and all, gives substantially more damage to the driver and passengers, and the death rate is twice as high in 'Car A' than 'Car B' under the same crashing conditions.

Obviously the 'How well' is the determining factor when we buy anything, like a car, a computer system, a painting, a mobile phone or a house. It is also the determining factor for any project, whether developing a thing, a new business, a service project, any kind of plan or project. Despite this fact, most every project planning method and tool lacks the vital ability to control the 'How well', the Stakeholder Values and Product Qualities, they simply state the 'What' the 'How' and 'How much'.

We will learn how to envision, find, communicate, specify, develop, test, achieve and deliver the 'How well' Stakeholder Values & Product Qualities.

What is a Requirement?

Webster's New World™ College Dictionary (Third Edition)

re•quire•ment (-ment) n.

1 the act or an instance of requiring

2 something required; something obligatory or demanded, as a condition [the requirements for college entrance]

3 something needed; necessity; need

Requirement Definition for Project Management

Requirements are; anything Stakeholders require.

End-State and Solution Requirements

"It is important that an aim
never be defined in terms of activity or methods.
It must always relate to
how life is better for everyone."

DEMING93, page 52

I categorize Requirements into categories.

End-State Requirements

End-State Requirements are the required Functions, Stakeholder Values, Product Qualities & Development Resources.

Means Requirements, or Solution Constraints

Solution Constraints are the building blocks, processes, features, implementation, function, design, architecture that somebody, like a Stakeholder or somebody above ourselves in the development process, require us to use in our development process.

If somebody requires us to use a specific part, process or feature, requiring us to design our product using one design over another, it is a required Solution. I call it a Solution Constraint, as it is a Solution of the kind that constrains my right to find more effective Solutions to meet my End-State Requirements.

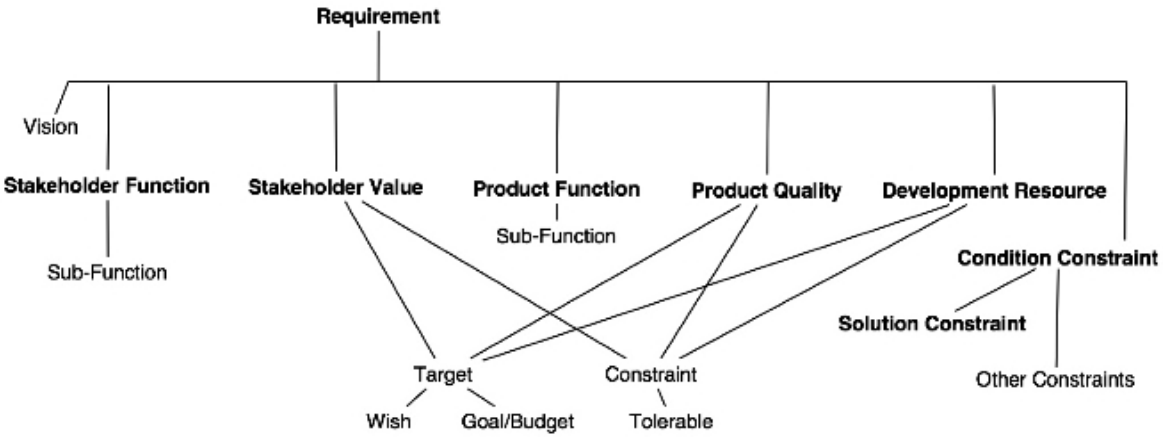


Illustration: requirement categories that I find important to keep separated when developing products.

Do we need to comply with all expressed requirements?

For an expressed Requirement to be a Requirement that our project is interested in fulfilling, I consider:

1. Who/what it came from.
   The people/products with an interest in our project I call Stakeholders. Does the Requirement come from a Stakeholder that we must or want to listen to?
2. Will it be profitable to fulfill the Requirement?
   Are the Stakeholder requiring the Requirement willing to pay for it?
3. Is it a market we and our company want to be in?
   Fulfilling a Requirement puts you in the market of people wanting that Requirement.

First in this book, we will discuss End-State Requirements, then we will tackle other kinds of Requirements.

Stakeholders

To find the Stakeholder Values and Product Qualities to consider in our project, first, I identify the critical parties and products with an interest in our project. Any person, group of people, or product that has or we want to have an interest in our project should be considered.

I call them Stakeholders.

Here is an example of a group of stakeholders:

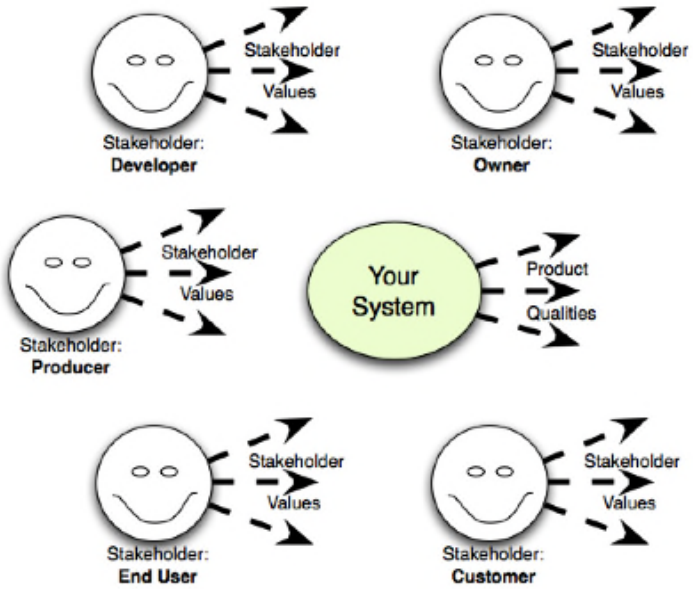


Illustration: For your system, the specific Stakeholders will be different, both in type and in number.

We can group Stakeholders or split one Stakeholder into two or more Stakeholders. For example, if we have two types of End Users, we could specify them together if they have similar Requirements or separately if they have different Requirements.

New critical Stakeholders with an interest in our project might appear at any stage of development. The Evolutionary Delivery method explained later is one effective way of sensing new Requirements and Stakeholders.

I examine each Stakeholder, and identify:

1) what each one wants to achieve, and

2) what we or our system can do to help them achieve it.

3) if we want to be in the business of meeting their needs.

Examples of Stakeholders and what they want out of a product.

Stakeholder: End-User.Bank-Customer:

Transaction-Speed: "improve the average time for a bank customer to do a transaction, like getting cash, moving money, getting a loan, etc."
Questions: "improve the time it takes for a bank customer from the time they have a banking question, to they have a understanding of the answer."
Comfortable: "improve the comfort level of a bank customer."
Availability: "improve the ability for the end user to do banking when they want to, where they are"

Stakeholder: End-User.Music-Lover: **Pleasure**, **Beauty**, **Choice**,

Stakeholder: End-User.Lake "or the people interested in the lake": **Oxygen**, **Algae**, **Bio-Diversity**, **Chemicals.**

Stakeholder: Customer.Bank: **Profit, Happy-Customers**

Stakeholder: Customer.Music-Stereo-Developer: **Market-Share, Reputation, Return-On-Investment**

Stakeholder: Customer.Lake-Environmentalists: **Learning, Happiness, Challenges, Public-Awareness**

Identifying Stakeholder Values

When I have found who our critical Stakeholders are, I identify what their needs and desires are, the Stakeholder Values, and then develop the product accordingly.

Often, I see products with a very narrow range of requirements, limited to system Functions and Solutions (Designs). To me, this is an indication of a project with an internal view; it has somehow lost the view of the outside world including its Stakeholders.

Successful projects deliver Stakeholder Values, and must therefore identify the current Stakeholders and their current values.

Stakeholders and their values will change

The Stakeholders identified in the beginning of our project might be different from the ones we deliver to at the end of our project. Some Stakeholders might withdraw, others might appear during a project, and we might initially not have correctly identified some Stakeholders.

In addition, the Stakeholders might not know in the beginning of a project what their values, their requirements, are. Even if they did know, their values often change in time.

Our project development method must be able to adjust according to changing Stakeholders and their changing Requirements.

Some projects 'freeze' their requirements. This is a sign that the project methodology is not able to cope with the reality of their Stakeholders changing needs. 'Freezing' the requirements might be handy for our project, but if our Stakeholder Values truly are changing, if the requirements set at the time when they where frozen have shifted at the time of delivery, and we do not change what we deliver accordingly, we will not satisfying our Stakeholders. We will not be customer focused, but internally focused.

Two types of Scalar End-States; Stakeholder Value & Product Quality

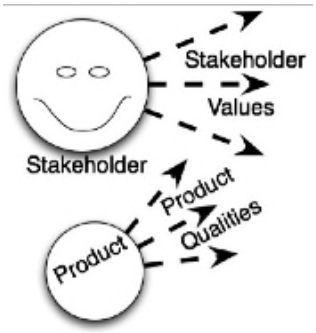


Illustration: I distinguish between the needs and desires of the Stakeholders (Stakeholder Values), and how well the product performs (Product Qualities).

Stakeholder Value; defined as: The achievement, benefit, experience, saving or profit that a Stakeholder value.

Some examples of Stakeholder Values:

achievement: achievement of task a Stakeholder wants to do.
benefit: stock value increase.
experience: excitement, fun and love.
feeling: feel safe, happy.
saving: training cost saving, efficiency improvement.
profit: money made.

Product Quality; defined as: Interaction between a product (or a system) and other entities, like other products, systems or people.

Some examples of Product Qualities: User-Friendliness, Portability, Availability, Reliability, Maintainability, Security, Performance.

Stakeholder Values are product independent. The Stakeholder Values can potentially be fulfilled with a variety of product combinations.

Product Qualities, like easy to use, or reliability, does not become Stakeholder Values if a Stakeholder wants, or “values” them. Solutions, like a password, do not become a Stakeholder Value if a Stakeholder wants, or “values” it. Stakeholder Values is a term used to define the values, achievements, benefits or profits that the Stakeholders need or desire.

To create a product (services, processes etc.) that Stakeholders would want to use or buy, we must ensure that the product can satisfy some of their Stakeholder Values. To do that, we must understand what the Stakeholder Values are. Only then we can design the appropriate Product Functions (what the system does) and a fitting level of Product Qualities (how well it does its Function).

Illustration: The skipper have needs and desires that can be satisfied using a sailboat with the appropriate qualities.

or generalized

Our Stakeholders have Stakeholder Values that can be satisfied using products with the appropriate Product Qualities.

Development Resources

Development Resources: The resources available to develop a product, I call Development Resources.

Development Resources can be anything that we can use to develop Solutions fit for achieving the Product Quality and Stakeholder Value Requirements. Development Resources are normally limited. Typical examples are: time, qualified people, money, space; and can also include anything else that we have a limited supply of, like water, air, a material (titanium), weapons, tools etc.

Relationships; Stakeholder Values, Product Qualities & Development Resources.

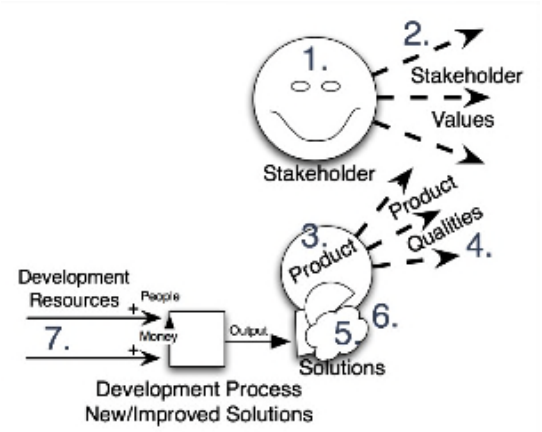


Illustration: (7) Development Resources are used to run the development process. The Development Process develops new improved (5) Solutions with enhanced (4) Product Qualities. When the Stakeholder uses the new product with the enhanced (4) Product Qualities it improves on their (2) Stakeholder Values.

The Product Qualities describes the abilities of a product without solving any Stakeholder Values. One example of a Product Quality is Reliability, how often a product fails. Other examples of Product Qualities are User-Friendliness, Comfort, Performance, Efficiency, Maintenance, Throughput as well as Costs.

Notice that the Stakeholder Values are not described by the Product Qualities like being easy to use, very fast, very reliable, or cheap, but the Product Qualities describe a product that might help meet the Stakeholder Values.

Imagine a product, any product, where the Product Qualities are great. Imagine this product not solving the challenges our customers or other Stakeholder needs solved. Obviously, we have to plan further than the Product Qualities.

The Product Qualities must be of such a nature that the product will meet or enhance the needs and desires of the Stakeholders (Stakeholder Values). The better our product is able to solve or meet the Stakeholder Values, the higher the chance is that our product will be successful.

Stakeholder Values and Product Qualities can be quantified and measured.

It is essential to observe that interacting qualities always vary. There is more or less user-friendliness, reliability, adaptability, style etc. It is not like a product either is user-friendly or not, it will be user-friendly to a varying degree. I often read in product requirement documents that a product must be very user-friendly. These words, very, increase, enhanced, improved, more, less, better, reduced etc. indicate that there is a variation, there is some state of less, and some state of more. If user-friendliness is an important Product Quality in the product I am part of developing, I always make sure that we quantify what we mean with user-friendliness. We state exactly how much more or less.

As the Product Qualities are interactions between two or more entities, Product Qualities can be observed. Since a Product Quality can be observed, it can in addition to being quantified, be measured. Since we can measure a level of Product Quality, we can know the level we have of a Product Quality we currently have, we can set targets fro improving the level, and we can track our progress towards achieving that level.

Stakeholder Values & Product Qualities. The Planguage starts forming

When we learn English in school, play on words is customarily encouraged as it is used in poems and novels. Using a language like English to communicate ideas precisely on the other hand is challenging. English is simply not very precise. It leaves a lot of room for interpretations.

We have over a period of several decades developed a method, a Planning Language or Planguage as we like to call it. My father Tom Gilb, started developing and using Planguage in the late 1960. It has proven remarkably helpful in communicating ideas in projects clearly so everyone involved understands the ideas as they where intended. It also guides and stimulates the thinking process and creativity of the people writing the specifications.

Planguage is very simple, yet very powerful. It is based on any language, like English, but it adds some structure and words with defined meanings. It is not trying to develop a fancy method, it is a reflection of reality, a way of describing products and projects.

First, I will demonstrate and explain how to specify one Product Quality Requirement using Planguage. Then I will demonstrate how all other Product Qualities as well as Stakeholder Values can be expressed using the same method.

This method of specifying the central Product Quality or Stakeholder Value Requirements works equally well on any kind of project. It works well if we are delivering a service, a product, planning a party, planning the objectives of a corporation, planning a social or environmental project, a software or hardware project or a project with any mixture of all of these, any kind of project. I am amazed at the different kinds of projects we have used these methods successfully on. If you feel the examples I use to describe the different elements in the specification do not relate to what you are doing, that is fine. Just know that the methods will work with your project! Later I will show examples of how to specify many other types of Product Qualities and Stakeholder Values. This will enable you to specify exactly what you need for your own projects.

Here is the first example of one Product Quality related to a new Mobile Phone being developed:

Project (The Function): Mobile Telephone

Product Quality:

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past 35 min.

Goal [within 1 year] 5 min.

Let's go through each element step by step.

Name Tag

User-Friendliness.Learn.Contacts

I label all ideas, including Stakeholder Values & Product Qualities, with a unique name tag.

Usually, I write the name in **bold** letters, with the First Letters Capitalized. I connect words with a dash "-" and indicate a hierarchy with a dot ".". In this example, **User-Friendliness.Learn.Contacts,** User-Friendliness is one name, there is not a separate entity called User or Friendliness, .Learn is one level down in the hierarchy, .Contacts yet another level down. We can now have **User-Friendliness.Learn.Modes** and **User-Friendliness.Errors**, etc.

As we refer the name tags repeatedly throughout a project, and often in tables where we have limited space, I keep name tags short, about 2 to 35 letters.

After writing an idea, and giving it a unique name tag, I do not rewrite that idea again in another place for another purpose, I refer to the idea using its unique name tag.

Because we care, we give them names!

I have observed that requirements frequently either remains nameless or that a name is used on a section of text that contains several unique ideas or requirements, also, frequently the naming systems used does not ensure that the names used are stable with changes or additions to the requirements. When I read requirements that are not individually named, I know the company does not use those requirements in any meaningful systematic way, they are just written for the purpose of following a process, then more or less forgotten about.

Because you care about your requirements, you need to give them a unique name that is stable independent of changes in the specifications. If the requirement name is not fixed, stability and communication will be lost as other s will refer to the old name. With a named requirement you can trace the requirement and its consequences. With a named requirement, we can identify it, refer to it and talk about it, without having to describe it in detail first.

Name it or lose it, or, Give requirements a fighting chance.

If we state a requirement, without giving it a name, it is highly unlikely that it will be followed through the development process, to the developers, testers, quality control, manual writers, course writers, to the end users and other Stakeholders as well as to support and maintenance. If we give the requirement a name, at least we are giving everyone a fighting chance to implement the requirement at their level.

All things people know about have names, it is extremely useful for communication. We always name all ideas in the projects I am involved in.

Outlaw Copy and Paste of Requirements, or, how to avoid complete chaos.

With concern, I observe a norm of repeating a requirement several places in similar but different ways. One version of one requirement for the engineers, another version of the same requirement for the testing group, one for the marketing group and another version of the same requirement for the people writing the user manual. The justification for this practice seems to be that people have slightly different needs or something to do with readability. This practice is catastrophic and should be outlawed. It leads to variations in what is one and the same requirement. When a change needs to be done in one requirement, it is impractical to update them all. Usually it only gets updated in one instance of the requirement. The engineer builds the product based on one version of a requirement, the tester tests to another, the manual has its own twist, and the marketing people sells something different again. This leads to defects, problems, time delays, cost overruns, lawsuits and project managers losing their ability to manage.

Having given our requirements names gives us the ability to reuse a requirement just by referring to its name. This practice alone will reduce the amount of project documentation immensely. It will allow the tester to test from the same requirement that the engineer used when designing the product, and the marketing people will sell a product built from the same requirements they sell.

I often see requirement specifications that no one uses, except to write them. I often run into this situation when doing specification quality control (not the scope of this book), and we find Major defects with requirement specifications, but the customer argues that it does not matter, because no one uses these requirements anyway. What a sad waste of peoples time. By giving requirements names, and outlawing copy and paste, we can focus on one set of requirements, we can invest in the quality of that one set, and we can know that everybody on the team is working from the same requirements.

Not only requirements, but all elementary statement in a project plan needs its own name tag. This is true for Stakeholder Values, Product Qualities, Solutions, Evo Cycles and all other elements of a plan. Here are four simple guidelines for writing specifications.

Generic Guidelines

GEN.Elementary: Statements shall be broken up into their most elementary form.

GEN.Name: All statements must have a unique and unambiguous cross-reference capability, which is stable, independent of sequence and changes.

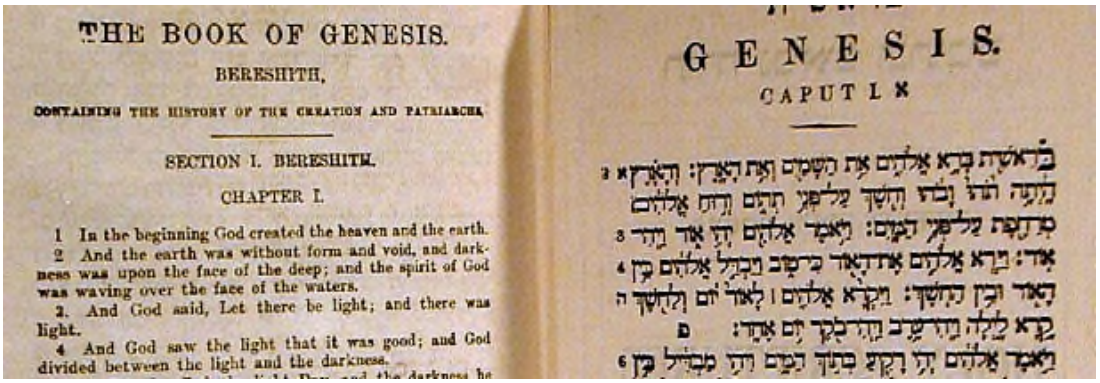
GEN.Unique: Ideas shall have one and only one single instance in the entire project documentation.

As the ideas are broken up into its most elementary form, named, then not repeated everywhere, readability is vastly improved. The authors can continuously enhance the ideas in both clarity and content. The reader gets one idea presented to them at a time, again enhancing clarity of ideas.

I do not recommended to use numbers like 1.1, 1.2, 1.2.1, 1.2.2 as name tags. It makes it difficult to adapt to the changes that will occur. If we add a new statement, 1.2.1, and every statement name after it changes (1.2.2 becomes 1.2.3 etc.), it breaks the guideline GEN.Name given above. All references to 1.2.2 now point to the wrong statement. I tag each idea with a name, not a number, this allow us to keep the same name on each statement even when we add or subtract other ideas.

Example: Imagine having a meeting in some town without a name, find an address there without a name, and find a meeting room there without a name, and have a meeting there with people without names. Somebody would have to describe to everyone going there where the town is etc. etc.

As crazy as this sounds, this is how many treat their requirements, with no names.



THE BOOK OF GENESIS.

BERESHITH,

CONTAINING THE HISTORY OF THE CREATION AND PATRIARCHS.

SECTION I. BERESHITH.

CHAPTER I.

1 In the beginning God created the heaven and the earth.
2 And the earth was without form and void, and darkness was upon the face of the deep; and the spirit of God was waving over the face of the waters.
3 And God said, Let there be light; and there was light.
4 And God saw the light that it was good; and God divided between the light and the darkness.

GENESIS.

CAPUT I.

Illustration: The authors of the Bible clearly understood the value of labeling statements. As the Bible is fairly stable, they can use numbers. Software programming languages depend on naming statements as well. It's time for all people writing project plans to start naming everything as elaborately as they do in the bible.

Love dies if there is no dedication.

Sri Sri Ravi Shankar, From Guru Purnima 1990

Control of a project dies the moment there is no dedication to track and follow-up each Requirement. Tracking and following-up requirements starts with giving each Requirement a unique stable name, and is only realistic if we have one and only one version of each Requirement.

<<<example of untagged req. to tagged req>>>

Scale

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Scale is short for Quantification Scale.

The Quantification Scale describes the Stakeholder Value or Product Quality, in a numeric way, outlining the variation from – infinity to + infinity, or the dimension of goodness from worst level to the best level.

The Quantification Scale describes a dimension of 'how well', which is critical to avoid failure and/or can ensure success in the project. The Stakeholder Values and Product Qualities are expressed through the Quantification Scale. Quantification Scale gives you the ability to asses a situation, compare it to a past or a competitors situation, set targets for success levels and set levels that would be intolerable.

It is crucial to find and specify the critical Stakeholder Values and Product Qualities, and to find out how to quantify them as directly as possible in the Quantification Scale. The Quantification Scale is the heart of Stakeholder Value and Product Quality specifications.

Stakeholders should normally approve the specified Quantification Scale, and that improvements along it are exactly what they want from the project, that improvements along the Scale will make them dance in the street and that they would love to pay for it, otherwise we probably will improve in the wrong direction.

The Scale contains a unit of measure, a rate to normalize and a description of its context. The unit of measure is left open (empty), as in "average time in minutes" without entering a specific number. This way we can reuse the Scale without rewriting it for each point along the Quantification Scale.

We never use more than one Scale per Stakeholder Value or Product Quality statement. When we need several Scales to express a Stakeholder Value or Product Quality, we split them up and create a new Stakeholder Value or Product Quality for each Scale with its own Name, and its own set of parameters (Past, Goal etc.)

Scale defines the 'how well' and should not be confused with how we measure. We will call the how we measure a Meter, as in speedoMeter. We will discuss the use of Meter after we have discussed Scale.

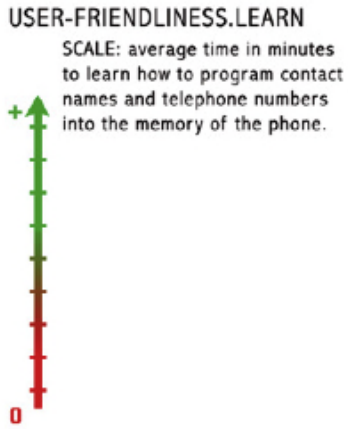


Illustration: We can describe all Scales graphically with an arrow. The root or the arrow representing the worst possible, and the arrow pointing towards perfection.

Without much trouble we use some units of measure like; US$, seconds, grams, lbs., Ohms, Watts, Joules, Calories, Celsius, Horsepower, Acre, Bars, knots, number of, meters, miles, gallons, liters etc.. There was a time when we did not know how to quantify these things. I am sure it was blissful days! Knowing how to quantify these variables has given us a certain level of control. Knowing how to quantify the value of money has enabled us to trade. Knowing how to quantify time has enabled us to synchronize. Knowing how to quantify weight has enabled us to make the same cake twice, with some level of control. Quantifying the

Stakeholder Values and Product Qualities gives my clients that same level of control over their project. And frankly, not knowing how to quantify the critical Stakeholder Values and Product Qualities leaves projects totally out of control.

The time has come for us to quantify and communicate to others, not only that which we know how to quantify, but that which is critical for us to control to ensure success, to get the Stakeholders what they want and need and to beat our competitors in delivering it.

> We keep on challenging: "Do you really love me? If you really love me then you won't do like this. You should be doing this way." When we keep some such measuring rods, then we are draining love.
>
> Sri Sri Ravi Shankar, Compassion And Trust

Not everything in life should be measured.

Choosing and composing our Quantification Scale

We have to choose our scales carefully. Quantify that which is important for success, and critical for survival.

My clients usually spend some time and go through a few iterations before they learn the Quantification Scales that works well for them. Some of my clients collect libraries of Scales that work well. In the beginning however, they often have to find and create their own customized and specialized Scales.

When faced with the challenge of writing Quantification Scales for the first time, a search on the Internet, using a key word, like ‘portability’, plus the word ‘metric’ or ‘quantification’, usually finds examples of other people who have gone before you. Normally you can adapt their work and customize it to suit your specific needs.

How to compose a Quantification Scale

A Scale consists of three elements;

1. unit of measure
2. a rate to normalize
3. description of its context.

1. The unit of measure is where we can specify the variation of the Scale. A higher or lower number before the unit of measure will be critical for the success or failure of the project. A Scale always contains a unit of measure, examples are: $, seconds, volt, transactions etc.

2. Usually a Scale contains a rate to compare and track changes, like: per year, per project, per 1000 transactions. A rate is a measure, quantity, or frequency, typically one measured against some other quantity or measure. The rate to normalize is usually fixed.

3. Then we put it in context. Explain what is being measured. Examples are; % of customers that are happy with the product, $ per year in service costs.

For all Product Qualities, like Performance, Availability, User-friendliness, Portability, Feel, Addictiveness, and for all Stakeholder Values like Cost reduction, Listening Enjoyment, Accuracy of estimates, etc. we can now create a useful Quantification Scale by combining a unit of measure, a rate to compare and track changes and a description of its contexts.

We can also quantify less technical Stakeholder Values & Product Qualities in the same way, like,
- number of refugees per year, we help returning to their home country and become self sufficient,
- % of native population that survives a war,
- % of families in our community that recycle paper, plastic, glass and compost.

Even when Scales are developed and reused within a domain, we always examine our specific project and our specific Stakeholders, decide what we want to accomplish, and modify the Scales to meet the needs.

> Make your desire bigger. Desire for the highest. Don't go for anything smaller than that. Hmm? Then also Divine Love dawns in you.
>
> Sri Sri Ravi Shankar

Stakeholder desires, the essence of Requirements are captured in the Quantification Scale.

Meter

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past 35 min.

Goal 5 min.

Meter describes the process we use to measure where we are, along the line defined by the Quantification Scale.

The Meter we chose must fit to the Scale. It must be able to give information about where along the Quantification Scale the Stakeholder Value or Product Quality is, just like a speedometer is able to tell us how fast our car is traveling.

The Scale gives us a variable, the Meter tells us where we are on that variable.

The Scale has no direct cost, the Meter often has, as it usually is a physical thing that requires some action or a process carried out by humans.

To verify that we have a useful Meter, we can apply the Meter, if it tells us where along the Quantification Scale we are, the Meter might be useful.

Examples of Meters:
Meter: Use a stopwatch and time it.
Meter: Manually count and record 10 randomly chosen people doing a task.
Meter: Take water samples at 1, 5, 10 and 30 meter depth and give it to the chemist to analyze.
Meter: Amnesty's International report.
Meter: Call 100 people and ask them....,
Meter: Make a one page question form and have 50 people answer.

Meters have variable:

1. accuracy
2. costs
3. time taken before they give us the results
4. credibility.

Meter for Development:

A Meter used during **development** should be **accurate** enough to gage progress, **inexpensive** enough so we can use it again and again and **fast** enough to give feedback to the current development cycle.

Inexpensive: Choose a Meter that is as **inexpensive** as possible. We will have to Meter many times during a Evolutionary project.

Accurate: Choose a Meter that is **accurate enough** to give us the feedback we need to gage progress towards our Goal levels.

Often people choose Meters that are very accurate, but too costly, or take too long time to get the results. It is better to choose a Meter that is barely accurate enough to indicate if improvements have been made from the last Evo Cycle, but so inexpensive that we feel encouraged to actually use it after a Evo Cycle, than a accurate one that we will not use because of time or money pressures.

Fast: Choose a Meter that gives us the results we need **fast**. We will be using the results from the Meter during our Evolutionary project cycles. Often we need the results of the Meter the same day, or quicker.

Meter for Final Delivery:

A Meter used for **final delivery** should be **credible** enough so Stakeholders believe the results and **accurate** enough so the variation in the results do not account for possible economic loss for the Stakeholders. I can also recommend listing agreeable Meters in the contract.

Accurate: Choose a Meter that gives accurate enough results so the measuring variation does not give room to cheat in the way of not delivering as promised. If we promise a Goal level of 91, and a result of 90 would not be acceptable, then the Meter must normally be accurate enough to show this difference.

Credible: A Stakeholder must feel comfortable with the credibility of the measurements taken with the Meter. During the development cycle we could use fast, inexpensive and accurate enough Meters to gage progress, for final delivery of a product it is usually more important that the Stakeholders believe in the measurement. If there are established ways of Metering or methods that cost more time or money, it might be necessary to use those, so as to get credible results that the Stakeholder can agree to.

Agreed: It can also be recommended that Meters used for final delivery be agreed on in the **contract**. No Meters are perfect, and trying to reach perfection can cost more than it will be worth both for the Developer and for other Stakeholders. Some practical compromise must be agreed upon.

We must be realistic when we choose our means of Metering. Do we really need this scientific study with a gigantic cost or can we do a little sampling ourselves? Do we want to wait for scientific results or can we find a faster way, which may not be scientifically valid, but will give us a quick low cost way to find out where we are on our Scale, and give us indications to how we are doing and where to go from there.

One neat trick to finding a cheap good Meter, is to use something that is already being Metered by ourselves or others, that way it ads no new costs.

Meter during operation

To gage changes during operation of a product it can also be useful to have Meters in place during operation. Here it can be useful to have built in automatic Meters that do not interfere with day to day operations.

Past

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

The Past together with Scale reads:

The average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone, used to be 35 min.

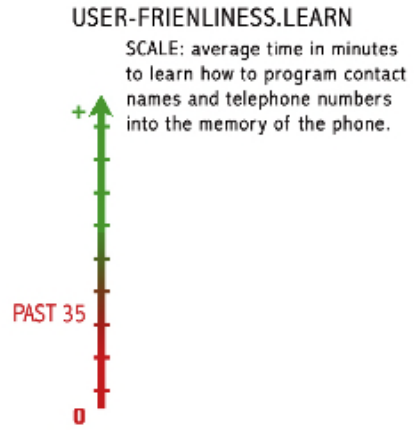


Illustration: the Past is any interesting reference point along the Quantification Scale.

Past is short for Past level. The most common use of the Past level is to reference how well a previous product did along the same Quantification Scale. The Past level can be any reference point of interest, and should be along the Quantification Scale specified.

It can be essential to know where along the Quantification Scale we and our Stakeholders are, before we decide where we want to be in the future. Specifying the Past level gives that benchmark and communicates it clearly. The challenge now becomes to move from where we are, the Past level, to where we want to be, the Goal level.

Say we are in Oslo Norway, the Past level, and we want to go to New York, the Goal level. Then we can probably tell something about the cost and the time (Development Resources) it will take us to get to New York.

If we don't know where we are, no Past information, our Development Resources are $50 and 8 hours, and the Goal is to go to Moscow, how can we know if we can get to the Goal?

If we say "we want cleaner drinking water!", but we don't know how clean the water is, the Past level, or how clean we want the water to be, the Goal level. We cannot make logical conclusion about the Solutions we need to apply to the drinking water, so we do not know how much Development Resources the Solutions will consume.

We need to know where we are, the Past level, and where we want to go, the Goal level, before we find Solutions to get there. The Solutions used are what determines the consumption of Development Resources, so when we understand what Solutions are needed we understand something about how much development Resources it will cost to get to the Goal level.

Goal

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Goal [within 1 year] 5 min.

Goal together with Scale reads: Our Goal is for the average time in minutes to learn how to program contact names and telephone numbers into the memory of the phone to be 5 min. within 1 year.

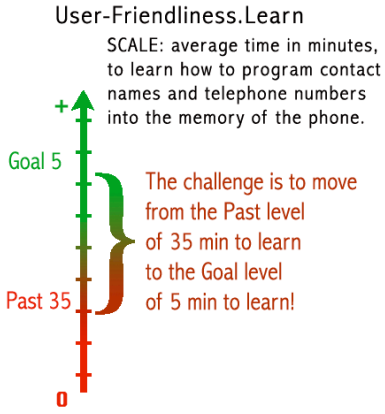
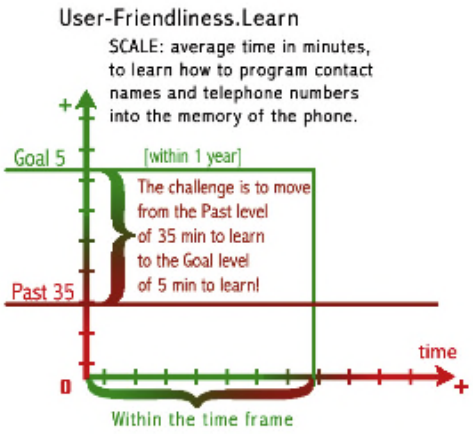


Illustration: For everybody on the development team, moving from the Past levels to the Goal levels becomes the reason for existing.

Goal levels are committed or promised targets along the defined Quantification Scale. By setting a Goal level, we commit to reaching that level, including committing Development Resources.

Moving from where we are now, the Past, to where we have committed to be in the future, the Goal level, becomes the project. Not reaching an agreed Goal level is seen as some degree of failure. Once a Goal level is reached, there are no further commitment of resources to continue improvement on that Scale for that Goal level. The Goal level becomes a stoplight, it dynamically communicates that in this area we are done. Development Resources can then be reallocated towards reaching other Goal levels.

In some cases we never actually measured with a Meter where we where along a Quantification Scale, but having quantified the Stakeholder Value or Product Quality and set the Goal levels guided the designers in choosing the appropriate solutions.



The Goal levels have timeframes for when they shall be delivered. This timeframe together with other conditions that apply I specify in what I call a qualifier. I use [square brackets] in front of each Goal level and specify the qualifying information within the [square brackets].

“Goal [within 1 year] 5” is an example of using the qualifier. It specifies that we plan to reach the Goal level of 5, within 1 year.

We will describe the [Qualifier] and numerous ways to use it in the advanced Planguage chapter. For now, for each Goal level, include a date within the [Qualifier] to communicate when the Goal level is expected to be delivered.

We can express many Goal levels for each Scale.

Example:

Goal [within 1 year] 5

Goal [within 2 years] 3

Because Goal levels are committed to or promised, they are set taking into consideration; all other Requirements (Especially Stakeholder Values and Product Qualities), existing technology, our competence, benchmarks (see Past & Record, the cost of beating the Record is normally unknown and exponentially expensive), competition and marketplace (see Trend), and Development Resources available including delivery schedules.

Exact Goal levels are best determined using a combination of actual Evo Cycle deliveries, Impact Estimation Tables IET, and field and technology experts. As everything around us changes and our understanding of our capabilities will improve, it is best if exact Goal levels can allow for adjustments during a project. A change of a Goal level can be initiated either from a Stakeholder or from the developers, and might require renegotiation about prize.

To have sales people promise specific Goal levels without them first asking technology experts is a sure and common way to destroy profitability. i.e. sales signing a ‘lucrative’ contract that promises 99.999% availability will do the damage.

See ‘Wish’ for a way of setting a level without promising anything.

Stakeholder Value & Product Quality Examples.

Stakeholder Values

Sale.Paperwork

Scale: average time spent, per sale, doing paperwork related activities.

Meter: look it up in the time management tool.

Past [2004] 2 hours.

Goal [2006] 15 minutes.

Sale.Quote

Scale: average time, from customer asks a salesperson for a price quote, until they have an official quote.

Meter: pretending to be a customer, ask 3 different salespeople for a quote, time it and compute the average.

Past [2004] 25 min.

Goal [2006] 5 min.

Transport.Time

Scale: average travel time, from our New York offices, to our London offices, door to door.

Meter: ask 2 of our people to time it the next time they make the journey, take the average.

Past [2004] 15 hours.

Goal [2006] 9 hours.

Training.Cost

Scale: average cost, to train, an employee.

Meter: ask the training department for this information.

Past [2004] $15.000.-

Goal [2006] $7.000.-

Sales.Total

Scale: total sales in $, per quarter.

Meter: look it up on the quarterly report.

Past [2004] $3M.

Goal [2006] $4M.

Teller.Transaction.Time

Scale: Average time, from a user is ready in front of a cash-machine with his credit card in his hand, until he leaves the teller, with cash in his pocket.

Meter: Stake out one of our tellers, time 20 people, and use average.

Past [2004] 4 min.

Goal [2006] 1 min.

Teller.Security.Rob

Scale: average number of robberies, per million transactions, related to our cash-machines.

Meter: get the number of robberies per month from the police, and the number of transactions in that period from our internal records. Calculate the where we are on the Scale.

Past [2004] 5

Goal [2006] 0.5

Product Qualities

User-Friendliness.Operate.Install

Scale: average time in minutes, to install, the system.

Meter: have 3 people install the system, time them, and average the time.

Past [2004] 120 min.

Goal [2006] 30 min.

User-Friendliness.Operate.Smooth

Scale: average number of mistakes done, per hour of use, by an operator with more than 100 hours experience using the system.

Meter: manually observe and count the mistakes done by 3 operators.

Past [2004] 4

Goal [2006] 0.4

User-Friendliness.Operate.Intuitive

Scale: average number of times, per hour of use, an operator with more than 5 hours experience using the system, either needs to look something up, or does not do the right thing on their first try.

Meter: manually observe and count the number of 3 operators.

Past [2004] 4

Goal [2006] 0.4

User-Friendliness.Operate.Learn

Scale: average time to learn how to do 10 defined tasks,
from they have the system in their hands with the intention to learn the tasks,
until they can repeat the tasks without referring to any instructions or notes.

Meter: manually time 7 users.

Past [2004] 180 min.

Goal [2006] 30 min.

Availability

Scale: average % of time, from 7am to 9 pm, that the system is up and running and available to the users.

Meter: create and install an application that automatically keeps track of the time the system is available.

Past [2004] 95 %

Goal [2006] 99 %

Reliability

Scale: mean time between failures.

Meter: create and install an application that automatically keeps track of the time between failures.

Past [2004] 9 hours

Goal [2006] 100 hours

Maintainability

Scale: mean time to repair.

Meter: create and install an application that automatically keeps track of the time the system is down.

Past [2004] 5 hours.

Goal [2006] 0.5 hour.

Portability

Scale: % saving, in total cost, to port the system to a new platform, compared to building a new system from scratch on a new platform.

Meter: port a tiny part of the system to a random platform, and make the estimation based on the resulting costs.

Past [2004] 0%

Goal [2006] 30%

Robustness.Drop.Break

Scale: average height of fall, onto concrete pavement, that the product can sustain without damages, other than outer surface scratches.

Meter: drop tests.

Past [2004] 0.7 meter

Goal [2006] 1.5 meter

Robustness.Drop.Stop

Scale: average height of fall, onto concrete pavement, that the product keeps functioning, if it is dropped.

Meter: drop test.

Past [2004] 1 meter.

Goal [2006] 2 meters.

Standby-Time

Scale: average standby time, from full charge, until automatic turn off, after using the product regularly everyday for 6 months.

Meter: charge, turn on, and let the phone sit connected to a carrier, and clock the result.

Past [2004] 24 hours.

Goal [2006] 72 hours.

Customer-Satisfaction

Scale: average customer satisfaction rating, from 1 to 6, where 1 is worst and 6 is best.

Meter: survey.

Past [2004] 2.5

Goal [2006] 4

Learning

Scale: average score on course completion exam, ranging from 0%=worst to 100%=best.

Meter: test results.

Past [2004] 55%

Goal [2006] 75%

Performance

Scale: time to complete 1.000.000 transactions.

Meter: records.

Past [2004] 7 sec.

Goal [2009] 2 sec.

Lake.Health

Scale: average number, of fish, per 100 cubic meter water.

Meter: sample nets.

Past [2004] 2

Goal [2009] 20

Stakeholder Values & Product Qualities. From your previous plan towards the Evo way

The 7 Whys ? and 1 step back.

Stakeholder Value & Product Quality Requirements are something desired in the future, an end state. Solutions are means of getting there. There are always many possible potential Solutions to reach a set of Stakeholder Value & Product Quality Requirements. All roads lead to Rome!? Many Solutions can satisfy the Requirements. A critical part of creating a competitive product or service is to find and use Solutions that are cheap and fast, yet satisfies the end state Requirements. I.e. To find the most direct route to Rome. If one specifies Solutions in the Requirement specification, instead of the real Stakeholder Value & Product Quality Requirements one can not pick the Solutions that best satisfies the Requirements to satisfy them. It is also highly likely that after developing and delivering Solution based requirements that the real requirements will not be satisfied, and that the Stakeholders will be unhappy about the outcome. I.e. one follow a route that does not lead to Rome, or does eventually get you to Rome, but at a high cost in time and money.

To find the real Requirements, I start a process of asking "why?" 7 times. Why? do we Require this Requirement? I take the answer from that question and again ask "why?" then "why?", "why?", "why?", "why?" and for the seventh time, "why?"

Depending on the specification I might formulate my "why?" question slightly differently each time. i.e. Why do we want this? Is this really what we want? What do we require this for? What will it give us? What will this do for my Stakeholders?"

For each "why?", I find a higher level Requirement, I am closer to the real End-State Requirement. Moving from sub-product levels to product levels to Stakeholder Values, to higher level Stakeholder Values, to levels so high that it is not the concern of this project or organization, then I stop asking "why?" and go one level back.

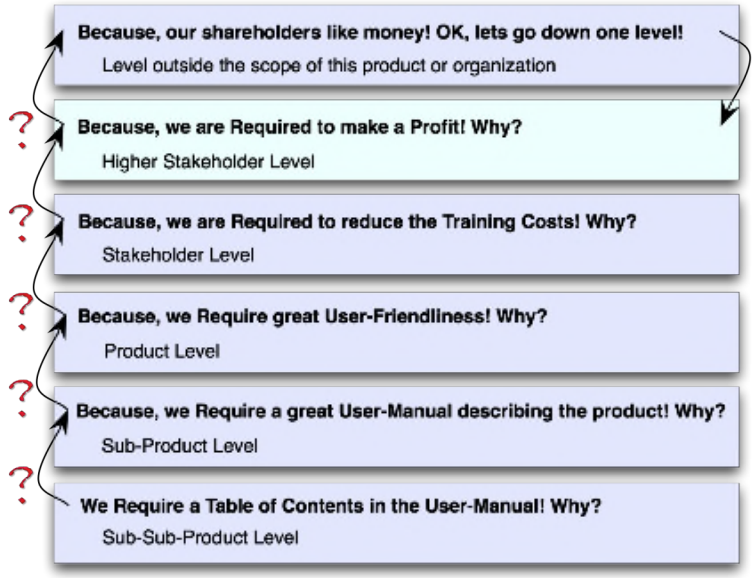


Illustration: A Table of Contents in a User-Manual is written down as a Requirement. Many people would just list that as a valid Requirement, and start building a User-Manual with a Table of Contents. If we ask why?, why do we want a Table of Contents in our User-Manual? We see that a Table of Contents is a Solution that is there to support the idea of a great User-Manual. We move one level closer to the real need,

the real Requirement. Then we continue this process of asking why?, and it turns out that a great User-Manual is also just a Solution for a Product Quality of User-Friendliness. Again we ask why do we want our product to be user-friendly?, and it turns out that User-Friendliness is there to help reduce the Training Cost which again is there to support the company in making a Profit. When we ask why do the company want to make a Profit?, we start getting answers that move outside the scope of the product and organization, and we can step down to Profit.

A level's existence is justified by the level before it.

Viewed from the level preceding it, every level is a Solution, or set of Solutions. A level's justification for existence is only the satisfaction of the level before it. Competitive advantage is achieved by designing each level so it best satisfies the level before it, with the minimum amount of Development Resources.

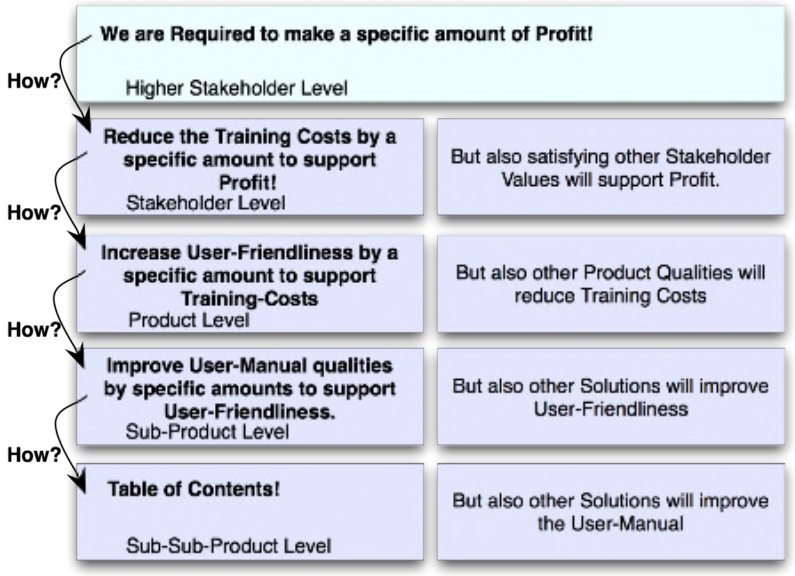


Illustration: Then we can continue the process in the other direction. How much Profit do we want? How much must we reduce the Training-Costs to get that Profit? What else must we do to get the Profit we want? To reduce the Training-Costs what can we do? Improve User-Friendliness, yes, but what else? What about; training the trainers better? Improving the Training Material? Home study training? Etc. To improve User-Friendliness, we can improve the User-Manual, but what else? What about; redesigning the User Interface so new users do not need to look things up in a User-Manual? What about designing a Mouse-Mat that has the answers to the 10 most common things looked up in the User-Manual. Etc. If we do want to have an improved User-Manual, how should the Table of Contents be for it to improve the User-Friendliness, and what other than a Table of Contents can we do improve the User-Friendliness?

Through the process of asking "why?", the actual reasons for what we are doing is revealed. We find the true Stakeholder Value & Product Quality Requirements. Only when we understand them can we make intelligent decisions about what Solutions are best suited to satisfy those Requirements.

Just because something is listed as a Requirement, does not make it so! Challenge it by asking why?

> Beauty has no utility, because beauty is not a "means," it is an end in itself.
> All that is useful in life is only a means.
> All that is useless, is an end in itself.
> Do you see that? What is the use of being in love?
>
> -- Sri Sri Ravi Shankar, Beauty And Innocence

Some examples of asking why?

Example: Mobile Phone:

In a project plan, under the heading of Objectives or Requirements. We have specified a specific type of keypad, made out of rubber. Is this the final state, an end state Product Quality? Well, let's ask why? Why do I want to use a rubber keypad?

Let's say we come up with the answer that, it is comfortable. Great! We have now moved one step closer to our real end state, the Product Quality. We want a comfortable keypad. What we have done is opened up for an endless number of possible ways to reach our end state. We are no longer restricted to just building a rubber keypad. We can look at other possibilities for creating a comfortable keypad. We can try something completely different, or we can build on the idea of a rubber keypad. Let's build on the idea for a moment. What about shaping the keys in a round way, to make them comfortable, or what about making sure the keys are firm using a stable base. The possibilities are now endless. Engineers can now get to work, and come up with Solutions that will make a very comfortable keypad. Can you think of anything that will make a keypad comfortable?

Is this idea of a comfortable keypad our highest level of Requirements? Well, let's ask Why? again: Why do our Stakeholders want a comfortable keypad? Hmm, let's say we come up with the idea that it will help drive the sales, it will improve sales. Great! We have now moved one level higher to the Stakeholder level. So the Product Quality of comfortable keypad is there partly to support a Sales with their Stakeholder Value of improving sales.

We might find answers to our question of "Why?", that takes us far beyond the scope of our project or organization. In that case, we can go a step or two back, until the answer is related to our project. Let's say we ask; Why do I want to sell more units of mobile phones? Answer: For the company to make a profit! Why do our company want to make a profit? To feed the employees and their families! Answer: Why feed the families? To keep everyone healthy and happy. Why keep people healthy and happy? Well that is an interesting question, but it is so far upstream of our project, that we will not tackle that directly. Where do we end our quest of Whys? Maybe “At selling more units or making a profit.” Even the company making a profit, might not be our responsibility, and normally it would not be the job of a project manager to deal with that directly, but it might be valuable for a project manager to understand the company objectives set on Profit and somehow tie their product into that.

Example Sara:

Saras goal statement: "My goal is to get a university degree in Spanish!"

Does this statement look like an end state requirement to you? Let ask: "Why do I want this?"

Why?: "Why do I want to get a university degree in Spanish?"

Saras answer: I can expand my possibilities of getting interesting jobs, and I would like to be able to speak with more then the English speaking world.

Great!, let us ask Why? one more time!

Why?: "Why do I want to expand my possibilities of getting interesting jobs, and why would I like to be able to speak with more then the English speaking world?

Saras answer: I want to have fun, and learn.

Great!, again.

Why?: “Why do I want to have fun, and why do I want to learn.”

Saras answer: “I don't know! Got to do something!”

Great! I think we are looking at Saras Stakeholder Value Requirements.

Fun and Learn are the ultimate Stakeholder Values for Sara.

What looked like final Requirements where in reality Solutions, Solutions to reach the hidden Requirements. The degree, the interesting jobs, the ability to speak to Spanish speaking people, are all possible Solutions to achieve the Stakeholder Values of fun and learn. We have not ruled out the degree in Spanish, she can still get that, we have just widened the gate considerably in the magnitude of possible Solutions we can use in meeting Saras real Stakeholder Values.

Separating the ends from the means and finding the real End-State Stakeholder Values & Product Qualities will give us countless options of potential Solutions to create a great product that we otherwise would not have. Knowing Saras End-State Stakeholder Value, she no longer has to get a university degree in Spanish. There might be several other ways to have fun and learn.

To expand our flexibility thereby the possibility and our ability to succeed in our projects, it is essential to understand the difference between ends and means, and to separate the Stakeholder Value Requirements from the Product Quality Requirements from the Solutions to getting there.

Let's say a customer of yours is running a business and you are developing a product to sell to them. Will your product be able to solve some of the challenges your customer is facing? Let's say they need to decrease the time it takes to do a specific transaction, can they get that reduction using your product? That is the Stakeholder Value! Let's say your customer needs to increase their production efficiency, can your product help them with the increase? That is the Stakeholder Value! Will your product, project or service help your Stakeholders meet their Stakeholder Values? Will it help people do whatever they want to do? That is the Stakeholder Values!

Coming back for more

Successful project management starts with understanding who our Stakeholders are, getting a clear idea of what they want to accomplish, then helping them accomplish it.

If we are assisting our customers achieve their objectives, they will come back for more.

My 7 steps for Success in Project Management

1. Stakeholders: first I identify all our Stakeholders,

2. Challenges: then I find out what they want to accomplish, their Stakeholder Values, and in what areas they have difficulties in meeting them.

3. Stakeholder Values: after that I target helping our Stakeholders meet some of their Stakeholder Values, especially those areas they find difficult. I specify the Stakeholder Values and set Goals levels to achieve, this becomes the Stakeholder Value Requirements.

4. Product Qualities: then I find out what Product Qualities & Development Resources our product needs, to help them achieve their Stakeholder Values. This becomes the Product Quality Requirements and the Development Resource Budgets

5. Solutions: next I find and engineer the Solutions to make a product with those Product Qualities within the available Development Resources. Using an Impact Estimation Table, I rate what Solutions are most effective compared to the Development Resources they consume.

6. Evo: then I divide the winning Solutions into Evo Cycles that I can deliver within a short timeframe (1 or 2 weeks). Each Evo Cycle has to has to give improvements to the Product Qualities in the direction of the Goal levels. Included in each Evo Cycle is measuring what improvements are actually achieved.

7. Repeat: then I start at the top again, but mostly I go quickly to step 5 or 6. I repeat until I can claim success.

Project Management Success example for consultants:

When meeting clients as a consultant, sometimes just given a few minutes with key individuals, my father has taught me to first ask them (our key Stakeholder) what and where they have difficulties getting the improvements they desire.
We spend a few minutes to help them articulate that desire in a clear, measurable, meaningful way, as taught in this book.
If they where not very interested in what we had to say before, they are now. We are now talking about

their desires, and the ones they have difficulties meeting themselves.
With their desires, their Stakeholder Values, clearly identified, we can effectively look for powerful Solutions to satisfy them. If we find effective ones, we become part of the team to meet their challenges, not some consultants with our own agenda.

Stakeholder Values - Product Qualities – Solutions, one follow the other

1. When all is said and done, we will be judged based on our ability to satisfy the Stakeholder Values.
2. Deliver a product with the appropriate Product Qualities to satisfy the Stakeholder Values.
3. Design/Engineer the product using Solutions that gives the desired Product Qualities.

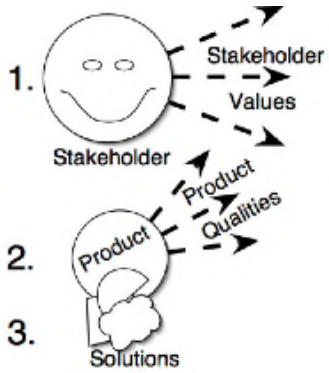


Illustration: Satisfaction of Stakeholder Values is based on delivering a product with fitting Product Qualities. To develop and deliver the desired Product Qualities, we develop the product using the appropriate Solutions.

Summary of chapter; Stakeholder Values & Product Qualities.

We now know how to describe a Stakeholder Value or a Product Quality along a Quantification Scale and specify two important points along the Scale, the Past level, and the Goal level, and to specify when the Goal level is to be achieved using the [Qualifier]. We have learned to distinguish between the quantification "Quantification Scale", and the measurement process "Meter".

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past: 35 min.

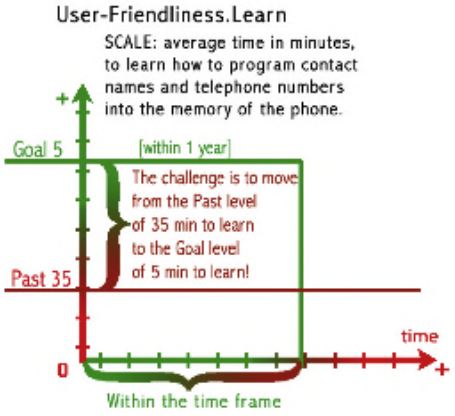
Goal [within 1 year] 5 min.

For the purpose of easy reading we can write the Requirement statement in plain English:

The average time in minutes to learn how to program contact names and telephone numbers into the memory of the phone will be reduced from 35 min. to 5 min. within 1 year.

The plain English version above might look attractive at first, but I have found it to quickly become long, complex and unusable when there are more details, as in many Past and many Goal levels in the Product Quality.

Or we can show it graphically.



We now know how to specify and communicate the improvements, the Stakeholder Value & Product Quality Requirements, in a way that is clear, easy to understand, easy to develop, precise, quantified, measurable, track-able and testable.

I describe all critical Stakeholder Values & Product Qualities in this way.

When all Goal levels are met in a project, the project loses its priority. It is like a stop sign for project managers. We are done!, but normally if we want to keep competing in the market we need to set new Goal levels for the next version of the product.

Many developers continuously develop one type of product, where the main Function is rather stable. That is, they develop mobile phones, or cars, or a web browser, or a tool to trade stocks. In these cases, the focus for the next release of the product will be improved Stakeholder Values and Product Qualities.

Principle: The Stakeholder Value & Product Quality Requirements of your projects must be equally clear, measurable and testable as the objectives in your favorite competitive sport.

Games like chess and sports like football have a few very clear objectives. Everybody playing knows what they are. When discussing practice or strategies, people can argue about the goodness of the strategies (Solutions) related to winning. Over time people will get better and better because they will not stray of course, clear objectives keeps athletes on their paths.

Our projects requirements can be much more complex. It is exceptionally important that we understand and define the Stakeholder Value & Product Quality Requirements, the end states of our projects, in such a way that everybody involved knows what they are and understand what they mean. Only then, can everybody pull, push and kick the project in the same directions. If we don't have this clarity of project Stakeholder Value & Product Quality Requirements, people working with the projects will steer off course, they will work on things that will not be in the interest of the project. Everybody will be pulling in different directions.

Illustration: Thor Heyerdahl transported the papyrus ship Ra II from the building site to the ocean using means available during the Predynastic period. It was made possible by hundreds of people pulling (in the same direction) to move the ship the <<<7 km???>>>.

Functionality, Function & Sub-Function

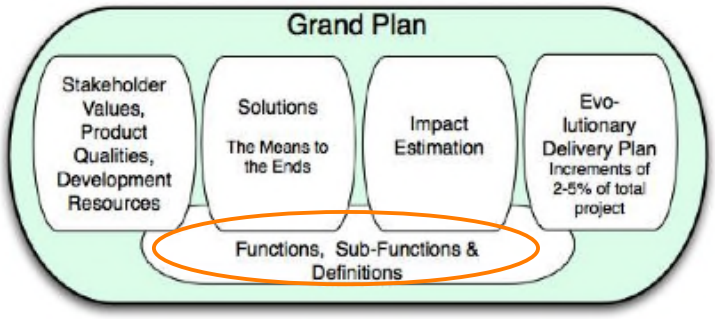


Illustration: In this chapter we will cover Function, Sub-Functions and Definitions. What it is and how we can specify it.

Functions and Sub-Functions

Defining Terms; Function, Sub-Function, or, Functionality what?

Most people use the word function & functionality very loosely. Let us define them in a way that is meaningful for describing, developing and delivering products.

Function Defined

Function Defined: A Function is what a system does.

Often I express the Function as what it is (not what it does). The Function of a television is to receive and display movies, images and sound. I usually just define the Function as Television. A car is a car, a truck is a truck and a laptop computer is a laptop computer. At other times there is a need to express the actual pure form of 'what it does'.

A Function is binary, i.e. it either does it or it does not do it.

Functionality or Sub-Function Defined

I will use the word Sub-Function in this book, it is the same as Functionality.

Sub-Function Defined: Sub-Function is a breakdown of the Function, what the system does.

Sub-Functions are binary as well. Examples of Sub-Functions in a car are brakes, steering wheel and seats. If we want to be pure in our expression of Sub-Functions and express it in what the Sub-Function does, replace brakes with stop, steering wheel with steering, and seats with a place for people to sit.

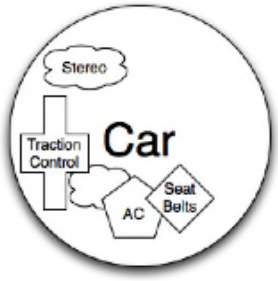


Illustration: The Function (Car) is made up by many Sub-Functions (Stereo etc.).

Function & Sub-Function – What are they?

Functions exists with Stakeholder Values and Product Qualities, or, I can not know about the existence of a Function without it interacting with me

All Functions and Sub-Functions have Qualities "attached" to them. Functions do not exist without their Qualities. The Product Qualities of a Function is the relationship between that Function and other products or people. And it is the Qualities of the Functions that delivers the value, and that cost to develop. Therefore, when we specify a Function, it is essential that we specify the Qualities of the function as well.

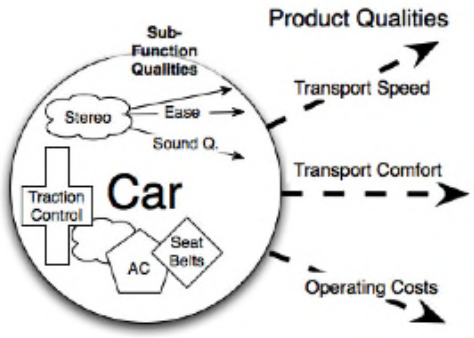
Illustration: The Functions does not determine the value nor the cost of a product. These two chairs deliver the exact same Functions. The chairs have very different Product Qualities. The Product Qualities are delivered thru the use of different Solutions. The Solutions used determine the Development Resources spent.

Solutions, a "package" of Function and Quality to deliver to the Requirements level above.

Product Functions together with Product Qualities can be viewed as a Solution to deliver Stakeholder Values.

Sub-Functions together with the Sub-Function' Qualities can be viewed as Solutions to deliver Product Qualities (that again can deliver Stakeholder Values).

The Function 'car' has Product Qualities, and together they can be used by a Stakeholder to help satisfy their Stakeholder Values. In a car, a sub-function (stereo) together with its Qualities, can be used in a car to help satisfy the overall Product Qualities of the car (eg. Transport Comfort).



How to Write Functions & Sub-Functions

Pure Function & Sub-Function specifications

If we have a pure understanding of a Function, what the system does, we can keep Stakeholder Values, Product Qualities, Solutions specifications separated from the Function and Sub-Function specifications. Functions, Stakeholder Values, Product Qualities & Solutions have many-to-many relationships, one Solution effects many Product Qualities and one Product Quality usually needs many Solutions to be created. To be able to optimize the Solutions needed to best deliver the many Product Qualities we must be able to create a strong meaningful link between all of them. This is hindered by writing a Solution together with a Product Quality in the same sentences.

Example:

We will use soft rubber buttons to create a comfortable keypad.

In the above statement, it is assumed that the main solution to create a comfortable keypad will be the use of soft rubber buttons. This it might do, but there are other solutions we can use as well. How do we link in additional Solution that will help create a comfortable keypad? How much will soft rubber buttons go towards meeting the comfortable keypad Product Quality Goal? What about negative side effects on other Product Qualities like reliability? or production costs? Are there other Solutions that might do the job better? There are no good answers to these questions when they are written together! By separating the Solutions form the Product Qualities etc. we can build up relationship information that can help us answering these essential questions.

Often, we don't need to specify the Function & Sub-Functions

The better we get at using Stakeholder Values, and Product Qualities to describe our projects, the less we will need to use Function & Sub-Function specifications at all. They can be contained within the Stakeholder Value & Product Quality specification.

Example of how Function specifications becomes obsolete.

Old way of specifying a product by using Function specifications, and omitting Stakeholder Values: *Produce a report, and deliver it to the desk the next trading day.*

Rewritten into its elementary ideas, where each sub-idea gets its own name tag.

DeskReport

ProduceRep: Produce a report

Deliver: Deliver the report to the desk.

Timing: the next trading day.

Now that the Functions are spilt up into separate elements, we can talk about each one separately, improve & develop them separately, and trace them thru to completion separately. I could for instance refer to **DeskReport.ProduceRep** and it would be clear which part of the specification I was talking about. This practice also makes it clear to the developers what they must develop.

DeskReport.Deliver.Timing is a poorly specified Stakeholder Value, hiding as a Sub-Function. There is clearly a time variation hiding here.

Let me rewrite the Function Specification of **DeskReport** into a Stakeholder Value:

Report.Timing.Ana

Scale: average time, from trading opening bell, until Report.Ana, with data from the previous trading day, is available to traders at the Desk.

Past [2004] 1.4 hours

Goal [2006] 0.2 hours

In the above example, when the Function specification was rewritten to a Stakeholder Value, it includes the information contained in the Function specification (ProduceRep, Deliver and Timing), so we do not necessarily need the Function specification any more, it has become redundant. Delete it! The Stakeholder Value points us directly to the real reason for the Function specification.

No new Functions are delivered to the Stakeholders, or, This is what we have always done!

Many developers believe that the main thing they deliver to Stakeholders are new Functions. Seen from the Stakeholders point of view, this is rarely if ever true. The Stakeholders have been doing what they have been doing before they where given our product to do it better. Remember that Function is defined as what it does. Better would be the improved Stakeholder Values and Product Qualities. Our product might expand in the Function it can perform, take over Functions done manually or by other systems, and I would hope, do the Function previously performed by another system better. Most system development is about improvement in how well the function is performing. Delivery of the Function is a given, success is determined by how well the Function works.

Illustration: What this Stakeholder' does, the Function, is done manually as well as with some current products. Our new product can do what was done by our old product, plus what some other products where doing, and it can do some of the work that was previously done manually. The main purpose of introducing our new system is to do some of the Functions that the Stakeholder already are doing better, i.e. improve the Stakeholder Values and Product Qualities.

The Stakeholder can choose to expand their main Function, what they do, if they like or not. They can do it, using the new Functions that our new product can do, or by some other means.

What we do is improve the Stakeholder Values by improving the Product Qualities. The improvement needs to be identified, specified and delivered. When we deliver a new system to a Stakeholder, we will find that in one way or another, the Stakeholder can already do the task we might think we are adding. We need to understand what they are doing, -the Stakeholder Functions and Sub-Functions, and more importantly, understand how we can improve on the qualities of those.

Example of No new Functions

I was holding a short presentation on the ideas in this book with a team that develops an office application suit. In the discussions, they seemed to be focused on Functions and Sub-Functions, so I challenged them with the following question:

What Function or Sub-Function does their text editor have, that the old tools of paper and pen does not have?

They started answering with obvious Sub-Functions like editing styles, fonts, cut & paste, but they quickly realized that all those Sub-Functions where performed with paper and pen as well. Then they went on to more sophisticated Sub-Functions like spell checking. I pointed out that people have spell checked long before the computer, that I could spell check right in front of their eyes on a peace of paper, and they quickly agreed. They become quiet for a moment, finally a person suggested video. I then reminded them of the movies we all made as kids, where we took a small notepad and drew a little cartoon on each page, one drawing slightly different than the next, to produce a video when flipping thru the pages quickly.

I believe that everyone in the room understood that by being focused on Functions they where chasing the wrong requirements. For many tasks, like writing this book, a computerized text editor can be easier, faster, prettier, more accurate at editing, cut & paste, spell checking, etc., and it certainly can contain stunning looking videos that make my flip pad look dull. For other tasks the Product Qualities of paper and pen are still unbeatable.

But computerized text editors contains NO new functions compared to the old system of using paper and pen etc..

When consulting on projects, I examine the project in the view of what value is added. It is rare that we are adding any new Functions as seen from the Stakeholders.

<<<<<<<<write functions initially, and delete them as we write more meaningful Product Quality specifications. Or just use skip writing functions all together. Or, in lack of good Product Quality specifications, keep the Function specifications around, as a bare minimum.>>>>>>>>>

<<<<< Explain how there are no new Function as seen from the stakeholder, 1. Stakeholder Function needs – what they do, 2. Stakeholders old system, and 3. the new System>>>>>>

Write Functions & Sub-Functions

Name

As with all other statements in a plan, individual Functions & Sub-Functions need their own name tag that is unique, stable and broken down into its elementary statements.

Description

Then the Function or Sub-Function needs to be described. Make sure Functions and Sub-Functions describe what the system does or must do, and not how or how well it will do it. Functions and Sub-Functions usually require only a few words to describe them. The Function of a digital camera is to take pictures and transfer them to a computer, and not much more.

Example

Functions

DigiCam:

.Camera: A camera that take pictures

.Digital: store the pictures digitally.

Transfer:

.From: from the camera

.To: to a computer

Sub-Functions

Shutter-Button: A shutter release button.

The Software Industry Scandal, or, Total reliance on Functionality (Functions)

It is widely known that software projects have a scandalously track record of delivering projects. Different studies form all around the world shows that a high (50%-80%) percentage of software projects are failures. Many speculate and report on different reasons for this high failure rate, and many use it as an argument to sell their product or idea. Reasons typically mentioned range from inaccurate estimates to lack of risk management, from high complexity to sloppy development practices, from poor communication to use of immature technology, but one reason mentioned first in almost all studies is related to bad requirements, and I concur, they are normally in bad shape, but how bad are they?

In the software industry, I have found that most development teams are almost entirely focused on what I call Functions, and they call functional requirements. In additions to the Functions they usually have a section called non-functional requirements, that are either left empty or contains vague buzzwords like, highly user-friendly or state of the art reliability. The so-called non-functional requirements are mostly ignored in further development because they are stated in vague terms that could mean practically anything. They are not quantified, not measurable, unclear and no one can be held responsible for failing to deliver such vague ideas.

This is a perfect setup for disaster for any type of project, because, what will determine a projects success or failure, and what costs time and other recourses to develop, are what they call non-functional requirements, the ones that are either missing or is expressed in useless ways.

We need basic Functions in our product to be in a marked, but our customers select our product or our competitors products (that all have more or less the same Functions) based on adequate Product Qualities, how well the product satisfies their Stakeholder Values and on Price. Similarly the failures on a grand scale in software projects, are not due to lack of core Functions, but due to lack of adequate Product Qualities, a miss alignment to Stakeholder Values and a lack of control over Development Resources.

Yet, believing Functions are everything, and forgetting about what really matters, the Stakeholder Values & Product Qualities, is an established practice in the whole software industry, it is thought by our professors, it is taught in almost all software books and reflected in our actual software development processes. Little else is taught to software ‘engineers’.

But it gets worse…

When analyzing the so-called functional requirements, they almost always consist mainly of, not Function(al) Requirements, but Solutions (Designs). The Solutions meant to satisfy the Requirements!

Where are the actual Function Requirements? Usually one can find an incomplete set of them hidden somewhere in-between the Solutions!

Where are the critical Stakeholder Value or Product Quality Requirements? Except for useless statements about having state of the art very high quality. The sad answer is, they are not!

Solutions (disguised as Functions) are there, and they cost to implement. But there is usually no estimate for how much the Solutions will cost. So the people responsible for finance and deadlines have stipulated their

budgets, but what they do not realize is that software people are not trained in, and generally do not care about costs.

How did the Solutions that are there (disguised as a Functional Requirement) get there? Did anyone do any engineering, any logical selection process where they weighted benefits of a Solution against its drain on Development Resources?

No!

It is impossible, as they do not even know what the expected benefits are!

<<<No!

Did anyone consider the risks?

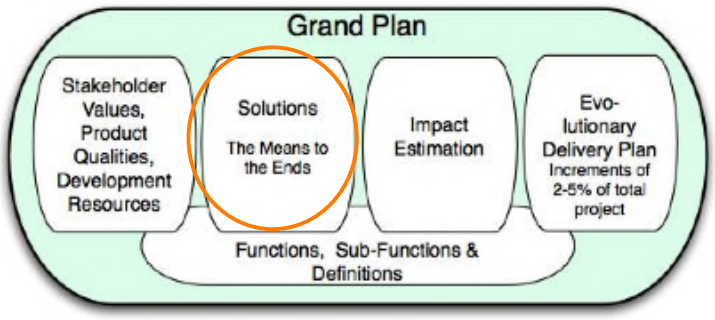
Did anyone consider the costs?>>>

I hold that before we have repaired this disastrous Requirement situation, the software industry will continue to develop substandard products and have no control over time or money budgets.

My hope is that competition will slowly force us to see how flawed our thinking has become, and bring us back to a sensible way to think about project management and product development, and I hope this book can be part of such a change.

A hint of the software industry' immature understanding of product development, is the word used for Stakeholder Values &/or Product Qualities, Non-Functional Requirements. The words Non-Functional says something about what the Requirement is not, but nothing about what it is. It's like describing a friend as, a non fish. While that is probably a true statement, it is not informative. I believe the use of the word non-functional came from software developers being overly focused on Functions, and some Stakeholders (like a user) made some requirements that where not Functions (like Usability), so they called it a non-function. As a consultant, whenever I see the words non-functional requirements in my clients documentation, I immediately know that they do not have control over Stakeholder Values or Product Qualities. Lacking that control, they don't have control over costs either.

Solutions or Means, or, What we do, the nuts & bolts, the actual code, the way we work, it!s how it works.



Solutions & Development Processes. Where are you?

Working with many different types of projects from all around the world I have observed a universal culture where the main Stakeholder Value & Product Quality Requirements are poorly specified, unclear and polluted with a mixture of Solutions (design), Functions (functionality), processes, even development process items like rules for how to write documents. With a poor understanding of the Stakeholder Values and Product Qualities, people write the Solutions (design) and then more detailed Solutions. With the detailed Solutions, people typically do a good job, except for the fact that the actual Solutions are fundamentally flawed as the Stakeholder Values & the Product Qualities are not well understood nor well expressed. Solutions other than those that contribute towards meeting the Stakeholder Values & Product Qualities within the budgeted Development Resources are a waste of time and effort.

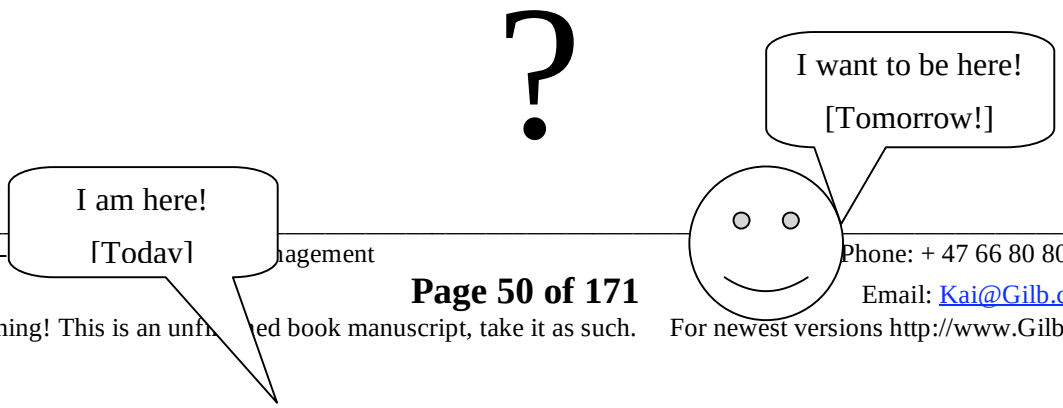
In this chapter I will introduce a better approach. I will argue that what people call Solutions or design, and Functionality might not even be that. Without a good understanding of the Stakeholder Value & Product Quality Requirements, the probability of writing intelligent Solutions are remote. It usually results in wasted time, money and effort. In this chapter, we will learn what Solutions are and how to specify them.

In the chapters on Impact Estimation, we will learn how to match and quantify well-constructed Solutions to well-constructed Stakeholder Values & Product Qualities within the allocated Development Resources.

Solutions. Basic ideas and principles

Definition of Solution:

General definitions of Solutions:
Solutions are the means to the ends,
or,



Solution

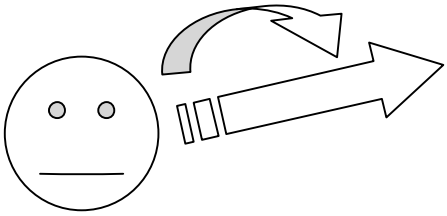


Illustration: Anything that will get me from where I am towards where I want to get, when I want to get there, within my constraints, I consider a Solution. This anything might be a combination of a tool, process, thing, idea, person, design, function etc. It does not matter what it is, as long as it gets me towards my Stakeholder Value & Product Quality Goals within my timeframe considering my conditions. Later, we will look at whether a given Solution fits us well.

Solution defined for project management: Solutions are anything, which moves us along Stakeholder Value & Product Quality Scales, from where we are, the Past (or Status), towards where we want to get, the Goal level, within defined conditions and Budgeted Development Resources.

Commonly used synonyms for Solution: Sub-Project, design, strategy, process, means, functionality, tactic, to-do tasks, architecture and method.

Solutions relate to the level preceding it.

Stakeholder Value and Product Quality Requirements define what we want to achieve using a Scale and a Goal level, while Solutions are intended to move us along the Scale from where we are towards the Goal levels.

To have a Solution, we must first understand the Requirement it is intended fulfill. Solutions only exist in the light of the Requirements it is meant to achieve. Solutions can only be as great as our understanding of the Stakeholder Value and Product Quality requirements are.

Discussions about for how good Solutions are, or which Solution is better, without having a clear idea of what kind of challenge it might solve, are fruitless. I avoid discussing which Solution we should use without having Stakeholder Value and Product Quality Requirements specified first.

Illustration text: Solutions must fit our specific challenges. If they don't take us closer to meeting our specific Stakeholder Value and Product Quality Goals, they are not really Solutions, at least not for us.

Solutions are anything that helps us meet our Requirements within our Constraints and our Development Resources.

To understand what are potential Solutions for us, we must first understand what our specific Requirements are. Nice things that do not help us meet our specific requirements are not Solutions for us.

Illustration text: Many people try to sell us Solutions that might be great for some things, but we must first find our challenges, our Stakeholder Value & Product Quality Requirements, then find Solutions that solves our challenges.

Your Solutions my Project

Your Solutions might become someone else's projects, and your project might be someone else's Solution.

The chain of someone's Solutions becoming someone else's projects might have many layers. The number of layers depends on, among other things; the size and complexity of a project, as well on how you choose to organize your organizational structure.

At every distinct level, we can describe the end states with its; 'what it does' (Functions), and its 'how well it does it' (using a Scale).

In smaller projects, we might operate with two levels, Product level & Solutions level. At the Product level we express the end states with Product Functions and Product Qualities. At the Solutions level, we might just specify solutions, like; use titanium, use a standard GUI layout etc.

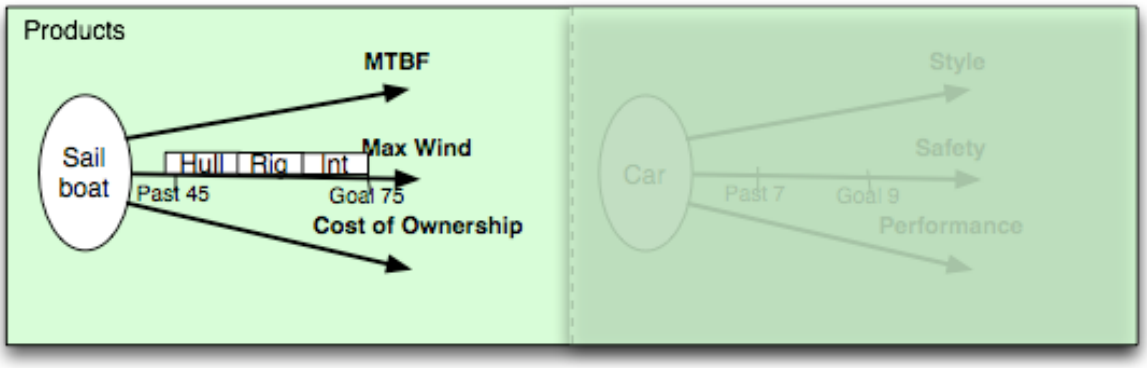


Illustration: In this example the Product Function = Sailboat, the Product Qualities are called = MTBF, Max Wind and Cost of Ownership, and the Solutions are called= Hull, Rig and Int.

In more complex projects, we might have 3 levels, a Stakeholder level, a Product level, and a Solutions level. Each level is viewed as Solutions for the level before it. The Products level are Solutions for the Stakeholder level, the Solutions level are Solutions for the Product level.

We can expand in both directions as needed. We can have several Stakeholders, some before others, we can have several products, and the products can be broken down several levels. Each level down are Solutions for the level above.

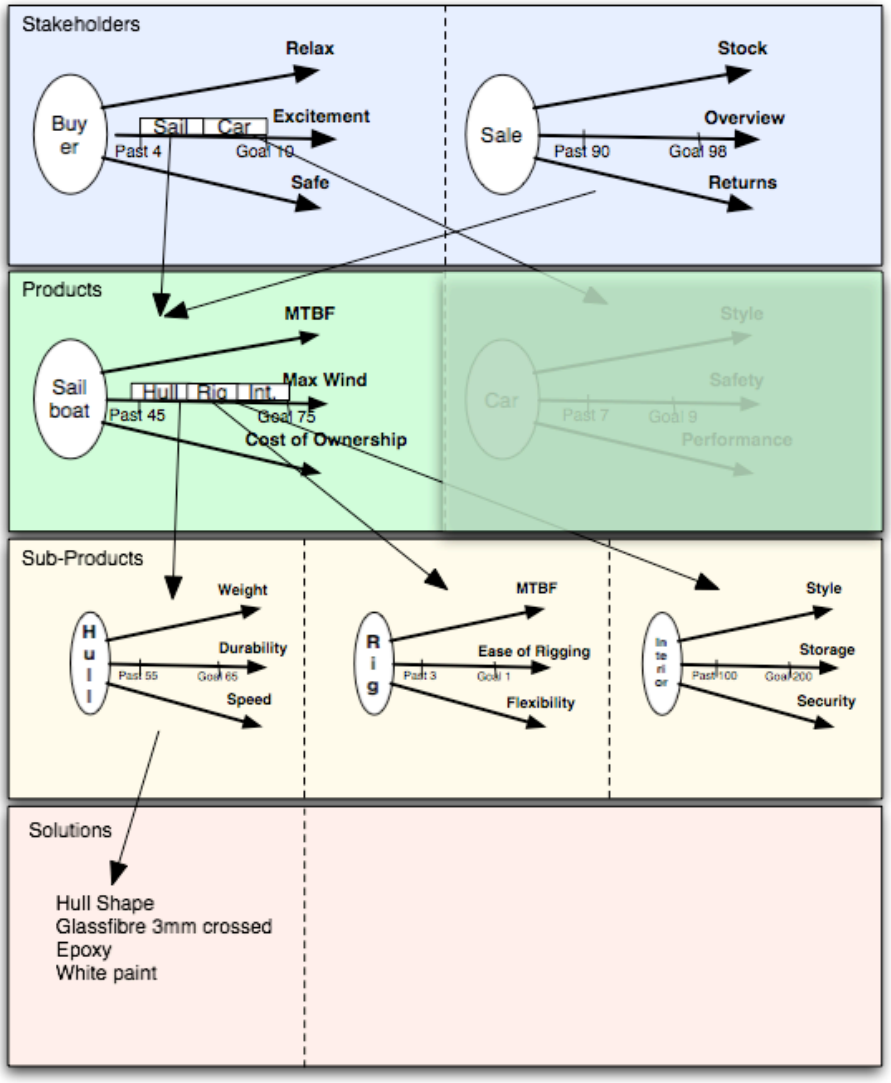


Illustration: In this example we are developing a Sailboat. We are showing four levels, each with several siblings.

At the Stakeholder level, we have two Stakeholders, a 'Buyer' and a 'Sales' Stakeholder. They both have a stake, an interest, in the product (sailboat) we intend to develop. The Stakeholder, 'Buyer', gets some of their Stakeholder Values satisfied with other products, like their Car, as well as the product we intend to develop. To make a successful product, we should clearly understand the Stakeholder Values of both these Stakeholders.

The Sailboat, is viewed as a Solution from the view of the Stakeholders, but it is the product for the people developing the Sailboat. The Product Qualities of the Sailboat will be critical for the ability of the Sailboat to satisfy the Stakeholder Values.

We can divide the Product into Sub-Products. The Sub-Products are viewed by the people responsible for the Sailboat project as Solutions to develop a Sailboat with the appropriate Product Qualities. From the view of the people responsible for a Sub-Product, it is viewed as a Product of its own right. The Sub-Products, like the Hull, has its own Qualities that are important to make the Sailboat with the appropriate Product Qualities.

To create the Sub-Products, we can decide on a set of Solutions to satisfy the Sub-Product Quality Requirements. For the Hull, they can include material, shape, building process etc.

For a shipping company we worked for, we helped them develop a system and we used about 5 levels as follows.

1. Shipping Company Owners

    Stakeholder Values: Profit, Market-share, Reputation..

2. Stakeholders

    Stakeholders: Customers, Crew, Ports… each with their own Stakeholder Values.

3. Shipping Company

    Shipping Company Functions: Ship Cars, Plan Schedules…

    Shipping Company Product Qualities: Effectiveness, Flexibility of Plan,..

4. Products

    Products Functions: Ships, Scanners, Scheduler…

    Product Qualities: User-Friendliness, Availability…

5. Sub-Products

    Sub-Products Functions: Capture-Data, Display-Data…

    Sub-Product Qualities: Data-Correctness, Data-Completeness

Illustration: Each level is a Solution to the level above, it has to be designed so it satisfies it.

All Solutions at any level can be described with their own Functions, Qualities as well as its physical implementation, as in what material, shape, design etc. is used.

At some point, we might choose to not describe the Solution in full detail with Function and Qualities, but simplify and describe how we will do it or what we will do. Examples are: use Titanium, or, program in C++, or, use Inspection, or, place the logo in the upper left corner.

When do we need to specify the Qualities.

Qualities are the base for deciding on the Solutions for the next level. If we want to make decisions about what to do, Solutions, for the next level down, we need to specify the Quality Requirements. If we don't, we do not have to. If you have decided to use a specific material, or a specific component, you do not necessarily need to specify the qualities, you can just use it.

Solutions. From the previous plan towards the Evo way

Solutions. The Planguage starts forming

When Solutions are not described fully, with Function and Qualities, but at the lower level, then Solutions can describe what we do.

The basic Planguage elements used for Solutions at the lower level are simple and familiar. We re-use many of the ideas used for Stakeholder Values and Product Quality specifications.

Each unique Solution is given a unique name, and we only allow one instance. Copy and paste of the Solution is outlawed. If we need to bring in the Solution to other places than the one specification of it, we refer to the name of it.

Solution Example

Short-Cut

Description:

.Names

.Button: A unique button for names,

.To: that takes the user directly to the names section of the phone.

Development Resources:

.Work-Hours: 100±30 work hours.

Naming Solutions

Notice how I give unique names to the Solution's sub-parts. I have an idea named "**Short-Cut**", then I have a sub-part I named "**Names**", which combined makes the name "**Short-Cut.Names**". If I like, I can have another but distinctly different idea related to **Short-Cut** and call it "**Messages**", then the full name will be "**Short-Cut.Messages**"

I have broken each element in the description of "**Short-Cut.Names**" into its components. One element I named "**Button**" another "**To**". I can refer to, develop and change each component separately. As an example, I can keep the "**Button**" but change what happens when a user press the button. Somebody can be responsible for implementing the "**Button**" Solution, and somebody else can be responsible fore implementing the "**To**" Solution.

Describing the Solution

Describe what the Solution is. Split it up into its individual ideas.

The content of a Solution should be comprehensive enough so the reader understands what the Solution is, have a good idea about how much Development Resources it will consume to implement, and give an idea of how it will effect the Product Quality & Stakeholder Value Requirements.

Scope of Solutions, a balancing act

Sometimes too little information is written down to describe a Solution, and the reader is left to guess what is intended. Even at early stages of development, when details of a Solution is not known, or when it is not yet considered desirable to spend time and effort to detail it, twenty or more words seems to be the minimum necessary to make oneself understood.

Early on in an Evolutionary project plan, we don't normally want to spend much time detailing Solutions. Because of Evolutionary planning' nature, it is likely that we, as we learn from hands-on experience and find better alternatives, scrap or alter the initial Solutions we came up with.

Therefore, initially, we describe Solutions at a high level without much detail. Only immediately before and while a Solution is actually built and implemented do we design and specify the necessary details. This practice prevents us from doing any work that could become wasted when we decide to do things differently. It also forces the people and their work done on paper to be real and practical.

In the **Short-Cut.Names.Button** example, no details are given about, the shape of the button, the material or the placement of the button. Technical details about how the button will work is not decided yet, it can be desided later in the development process when we get closer to actually developing the sub-Solution. The same can be said of **Short-Cut.Names.To**, there are no details of how this button will take the user to the names section. The details will probably be designed and specified at one point in time, but only if and when it is decided to actually develop **Short.Cut**. For now it is just one of many ideas, and I would not use precious Development Recourses to design and detail it yet.

Development Resource Consumption

I also make an estimate about the consumption of Development Resources for each Solution. Functions do not have a direct development cost, Stakeholder Values and Product Qualities does not have a direct development cost, the main drain on Development Resources comes from developing the Solutions. I estimate how much Development Resources each Solution consumes. If I want to know how much it costs to get from a Past level to a Goal level of a Product-Quality, I must decide what set of Solutions will get me to the Goal level, and ad the cost of the Solutions.

Side Kick - well-meaning project 'terrorists'

Sri Sri Ravi Shankar expresses Goals, Qualities and Solutions clearly when discussing the cause and the remedy of terrorism. Let us keep human values and life above any ideas and concepts. Let us reach noble aims by peaceful and non-violent means.

Terrorism: The Cause And The Remedy
by Sri Sri Ravi Shankar (www.artofliving.org)

The act which is only destructive and inflicts suffering both on oneself and others is terrorism. In such an act, human values are lost in the process of achieving a Goal.

Some of the factors that lead to terrorism are:
* Frustration and desperation to achieve a Goal
* Confused emotion
* Shortsightedness and Impulsive action
* Belief in a non-verifiable concept of heaven and merit; a childish concept of God favoring some and angry at others, thereby undermining the omniscience and omnipotence of the Divine.

Terrorism induces fear psychosis in all, increases poverty, suffering and loss of life with no apparent gains. Instead of Solutions, the terrorist looks for destruction as an answer. If you simply criticize without giving a Solution, know that this criticism comes from the same seed as terrorism.

Although there are certain qualities you can appreciate in a terrorist such as:
* Fearlessness
* Commitment to a Goal
* Sacrifice

You will have to learn from them things that you should never do:
* Valuing some ideas and concepts more than life.
* Having a narrow perspective of life and dishonoring its diversity.

The Remedy for terrorism is:
* Inculcate a broader perspective of life -- value life more than race, religion and nationality.
* Educate people in human values – friendliness, compassion, cooperation and upliftment.
* Teach methods to release stress and tension.
* Cultivate confidence in achieving noble aims by peaceful and non-violent means.
* Create spiritual upliftment which can weed out destructive tendencies.

Question: Can it be that terrorism need not be only physical violence, but also cultural or economical?
Sri Sri: Yes.

Solution for economic violence – "Think globally, buy locally."
Solution for cultural violence – "Broaden your vision, deepen your roots."

Question: How does one cope with the aftermath of terrorism?
Sri Sri: Faith and prayer. When disaster happens, anger is inevitable. To take precautions that one does not react, wisdom is needed, not emotional outbursts. One mistake cannot be corrected

> by another mistake. Strive for multicultural and multi-religious education and spiritual upliftment to reach every part of the globe. For the world will not be safe even if a small pocket of people are left ignorant.

Sri Sri Ravi Shankar says, “Instead of Solutions, the terrorist looks for destruction as an answer.” He defines the opposite of Solutions as terrorism. In project management, those ideas that do not contribute towards meeting common Goals can be destructive to the project. It is the responsibility of a project manager to ensure that the project Stakeholder Value & Product Quality Goals, the ends, are well specified and communicated to everyone in the project, so everyone can focus on finding and developing Solutions to meet those ends. Otherwise, we will have well-meaning project ‘terrorists’ working with us.

Categories of Solutions with Examples

I find it useful to separate Solutions into categories. Here are four main categories.

Build Solutions

Build Solutions describe the actual parts, material, design and components of the product.

Examples are material used to make a product; computer code, the physical design of the product, parts to be used in the product.

Example:

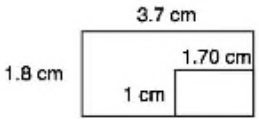
Casing

.Material

.Titanium: use titanium.

.Thickness: 10mm ± 0.2 mm.

.Design:



Sub-Function Solutions

Sub-Function Solutions are Sub-Functions that support a higher Requirement. They differ from ordinary Sub-Functions in that they are not a necessary part of the product, no one is requiring the product to perform a Sub-Function Solution, it is an option, a means to an end. A Sub-Function Solution does not tell us how, just what it will do.

Examples are any Sub-Functions that are not a requirement; air-cooling in a room, an address book on a telephone, internet in a hotel room.

Example:

Spellchecker

.Languages

.Euro: include all the European languages.

.US: include US English.

Process Solutions

Process Solutions describes processes that can be used to develop the product. Processes that alone will not make any product, but when applied to the developing process can increase the effectiveness of the development.

Examples are anything that can help the developers do their job better; computer applications, a system to sharpen the knifes regularly in a kitchen, a checklist to follow, a document change management tool, Evolutionary Delivery, Waterfall, Inspection.

Quality

Speckqc: use specification quality control

.Area:

.Req: on all new Requirements

.Qual: No more than 1 Major Defects per page remaining

.Sol: On all new Solutions

.Qual: No more than 3 Major Defects per page remaining

Testing: …

Product Qualities and Sub-Product Qualities

Sub-Product Qualities are Solutions to satisfy Product Qualities. Product Qualities are Solutions to satisfy Stakeholder Values. As an example a Product Quality of Usability can possibly be there to help us meet a Stakeholder Value about reducing the cost of ownership of a product. Usability is not an end in itself; it is a Solution to meet a higher end state. As seen from the perspective of Stakeholders, all Product Qualities are Solutions.

Example:

Usability

.Learn

Scale: average time to learn how to operate the system

Past [2004] 120 min.

Goal [2005] 30 min.

Solution Constraint

As far as possible, I refuse to accept Stakeholders requiring a specific Solutions. Solutions that are required, I call Solution Constraints. Solution Constrains prevents me from finding the Solutions that best reach the defined Stakeholder Value and Product Quality Requirements with minimum consumption of Development Resources. Because of the lack of culture in specifying Stakeholder Values and Product Qualities directly, many Stakeholders specify Solutions that they hope will give them the desired Product Qualities. The Stakeholders are rarely professionals at finding the best Solutions, they rarely have an overview of all the other Stakeholder Values and Product Qualities, and they rarely understand the technology or keep up with new technological possibilities, therefore they are doomed to find suboptimal Solutions. Often Stakeholders suggest Solutions such as a new button, tab, or menu in a software application. Many of my clients, against my recommendation, still take such Solutions suggested by Stakeholders and treat them as Solution Constraints. After some time of letting the Stakeholders drive the development with Solution suggestions, the product looks and behaves as if it was hacked together by amateurs (it was), not designed by engineers or professional developers. This leads directly to uncompetitive products.

A 7 Step Process to Challenge Solution Constraints.

I recommend challenging incoming Solutions that are initially listed as requirements using this simple process.

1. Stakeholder: Identify who requires the Solution.

2. Why: Ask the Stakeholder why they request or require that specific Solution.

3. Value: Specify the reason in the form of Stakeholder Values or Product Qualities using Scale Past and Goal.

4. Verify: verify with the stakeholder that we understood what they actually wanted with the proposed Solution and that it is covered in the Stakeholder Value or Product Quality we specified.

5. Improved: Then we ask the Stakeholder;

<<<<>>>"if we can find a alternative Solution that can meet or improve on those Stakeholder Values or Product Qualities for less Development Resources than the Solution initially proposed by the Stakeholder, if he would be interested in hearing about that Solution?"

6. Solution Constraint: We will normally attempt to document the adverse effects and show it to them. Examples can be longer development time, reduced speed, security, user-friendliness etc. If they despite of negative effects insist on the specific Solution, we have two options.

a. Accept: we accept the Solution as a Solution Constraint and hold them responsible in writing for any and all effects it has on Stakeholder Values or Product Qualities.

b. Abort: we might find that the proposed Solution has such negative effects on Product Qualities that are important to us that we choose not to deliver to that Stakeholder. This would mean loose their business. *An example of this would be a Solution that would make Security or Upgradeability or Maintainability so bad that our product would not be competitive for other Stakeholders.*

7. Potential: If the Stakeholder is open for other Solutions as long as we satisfies their real need now expressed as a Stakeholder Value or Product Quality, we put the proposed Solution together with the other potential Solutions, and if the initially proposed Solution is the best Solution we can come up with, great we can use it, and if we can find an even better Solution, we are not stuck with an un-optimized Solution.

So, as an example of this process, a Stakeholder requires the use of a specific battery in a mobile phone. We ask them why they want that battery type, and specify the answer in the form of Stakeholder Values or Product Qualities. This could be a combination of standby-time, service-costs and battery costs. We can then present them with the idea that if we can find a cheaper battery that will give longer standby-time & lower service-cost, we would like the option to use such a battery.

If they still insist on using an inferior more expensive battery, we can either accept that battery as a Solution Constraint. But then we must require a written signed statement that specify that they took this decision, despite of us suggesting to them a better option, and specify that they are responsible for the inferior standby time, the higher service cost and the expensive price. Alternatively we can reject their battery and risk losing the Stakeholder as a customer.

Solution Constraint Specification

A Solution Constraint is specified just like a regular Solution, except we add the authority requiring that Solution, and we need to identify it as a Solution Constraint. This can be done by keeping a separate section with all the Solution Constraint, or by adding Type: Solution Constraint.

Example:

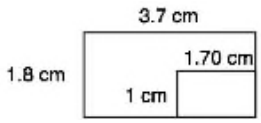
Casing

.Material

.Titanium: use titanium.

.Thickness: 10mm ± 0.2 mm.

.Design:



Type: Solution Constraint

Authority: Stakeholder: John Smith from Sales

The 'Authority:' parameter can also be used together with any other specification type. As in together with a Stakeholder Value, a Product Quality, a Function or a Evo Cycle.

Solutions Summary

Solutions are the means to the ends.

Solutions defined: Solutions are anything, which moves us along Stakeholder Value & Product Quality Scales, from where we are, the Past (or Status), towards where we want to get, the Goal level, within defined conditions and Budgeted Development Resources.

What are Solutions for some can be Projects for others.

All Solutions have Qualities, Functions, and Build Solutions to deliver the Qualities and Functions.

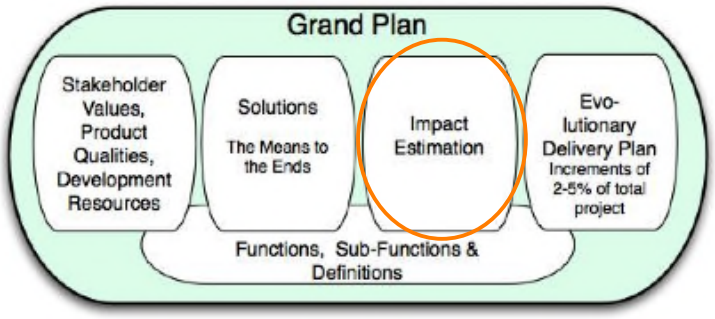
If we want to make decisions about what to do, Solutions, for the next level down, we need to specify the Quality Requirements. If we don't, we do not have to.

We write each individual Solution in one place only and give it a unique name. When we need to include that Solution elsewhere, we refer to the name of that Solution without repeating or re-writing it.

It can be useful to identify different kinds of Solutions, I use four basic categories of Solutions; Build Solutions, Sub-Function Solutions, Process Solutions and Product Quality Solutions.

When a Stakeholder requires a specific Solution, we call it a Solution Constraint, as it constrains us in finding a better suited Solution. I encourage my clients to challenge proposed Solution Constraints, as it constrains them from making better more competitive products.

Impact Estimation



Impact Estimation. Where are you?

Some fundamental questions project managers, architects and engineers must evaluate are:

a. Is a chosen Solution well suited to meet the Stakeholder Values & Product Qualities?

b. How does one Solution, intended to solve one Product Quality, impacts the other critical Product Qualities?

c. With a given set of Solutions, where do we have weaknesses, what Product Quality or Stakeholder value will not be met?

d. Is this set of Solutions sufficient to meet our set of Stakeholder Values & Product Qualities?

e.

When the answers to these questions are not known or unclear, it becomes obvious to a project manager that the project is out of control. Most project management methods will not be able to answer these critical for success questions. The problems originate from not clearly separating out Stakeholder Values & Product Qualities from Solutions, and from not stating Stakeholder Values & Product Qualities in a clear measurable and testable manner. Another practice, which is doomed, but is the norm, is to have one to one arguments on the value of Solutions. That is, a Solution is justified by its intended value on one Product Quality alone, not by its overall effect on the whole project with all of its Stakeholder Values & Product Qualities.

Example:

Let's assume we are developing a new car, and the Stakeholder Values & Product Qualities are unclear. We design a new engine part (means) that we argue gives the car more horsepower (end).

If we look back on the questions above, we will find that it is hard to answer any of them. If we do not know how much, if any, more power is needed, then maybe it is not solving any real Product Quality Requirement. Moreover, where is the argument about how it influences all the other Stakeholder Values & Product Qualities? What about fuel efficiency, time to market, cost, reliability, noise etc.?

Where the Product Qualities (ends) and the Solutions (means) are mixed together in the same documents, even the same sentences, the practice of one to one justification is as inevitable as it is detrimental. When the documents states that the new engine part is there because/for/to give more engine power, I know that we have lost control over our project.

The good news is that there is an easier, better, more logical way, of matching means with ends. It builds on an understanding that one must expressly separate the means from the ends. With this separation we can build a table (IET) evaluating the many effects the means have on the ends. And since we specify the Stakeholder Values and Product Qualities quantitatively using a Scale and a Goal level, we are also set up to quantitatively evaluate the effects the means have on the ends.

This evaluation method (IET) will also help you answer the fundamental questions I outlined above.

Impact Estimation. Basic ideas and principles

The understanding of how well our set of Solutions satisfies our set of Stakeholder Value & Product Quality Requirements we call Impact Estimation.

First let's use our basic examples of a mobile phone project, consisting of one Product Quality Requirement and one Solution.

Product Quality

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Goal [within 1 year] 5 min.

In this Product Quality Requirement, notice that it currently takes 35 min. to learn something (Past 35 min.), and that the Goal for our new system is for it to take only 5 min. to learn, all of this within one year (Goal [within 1 year] 5 min.). This difference between 35 min. and 5 min. clearly describe one of the Product Quality Requirements of this project.

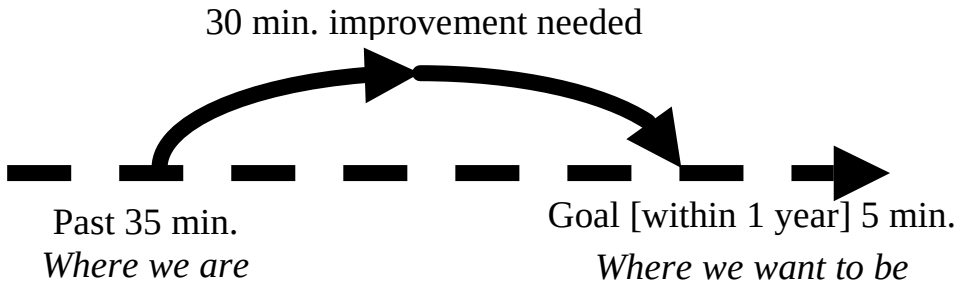


Illustration: The challenge of moving from 35 min. to 5. min. is clearly specified. Anything that gives an improvement from 35 min. towards 5 min. within the year I will consider a Solution. What is the best Solution, and do I have enough Solutions for the challenge?

One of the contenders for solving this challenge can be our **Short-Cut.Names** Solution.

Short-Cut.

Names.

Button. A unique button for names

To. that takes the user directly to the names section of the phone.

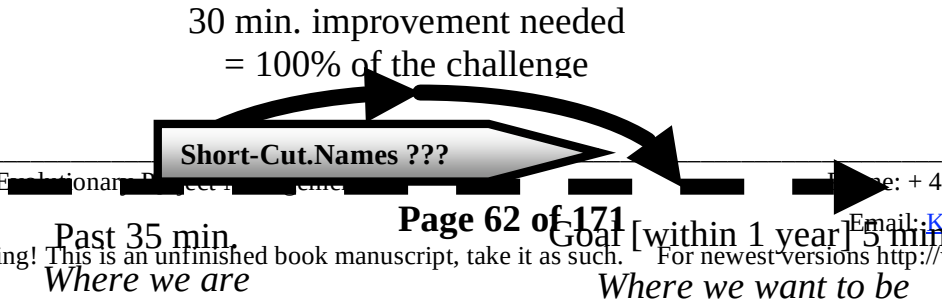


Illustration: Will the Solution **Short-Cut.Names** help solve the challenge of **User-Friendliness.Learn.Contacts** within 1 year? If yes, by how much?

Notice that a Solution like **Short-Cut.Names** does not necessarily solve the whole Product Quality Requirement. A 30 min. reduction is needed, 30 min. is 100% of the challenge to be solved. If we reduce the time it takes to learn with 15 min. we have solved 50% of our initial challenge.

Impact Estimation. Taking you from your previous plan towards the Evo way

<<<Kai Note: This is covered in the earlier chapter. Do I need more here or not?>>>

In our project plans,

Impact Estimation. The Planguage starts forming

Planguage systematically collects the differences between where we are - Past, and where we want to be - Goal. The difference between Past and Goal is treated as 100% of what needs to be improved on each Stakeholder Value and Product Quality. So if we have 10 critical Product Qualities, then we have to move from where we are, Past, to where we want to be, Goal, on all 10 critical Product Quality Goals.

We collect the Solutions we best think will solve these critical Product Quality Requirements.

Then we combine these elements in what we call an Impact Estimation Table. As the name applies, the Impact Estimation Table is a table that estimates what Impact our Solutions have on our Stakeholder Values or Product Qualities. We use numbers to show the estimated impacts, as numbers are much clearer than words, but the numbers are not intended as exact calculations of Impact, as that would be much to complicated, time consuming and practically impossible to accomplish. We have found that the best way of finding the real impacts of many Solutions on many Product Quality Requirements is to "just do it!" and measure the reality. The ‘just do it!’ method is covered in the chapters on Evolutionary Delivery. Even though the Impact Estimation Table is not a mathematical model by any accounts, our clients have found that incredible insights can be found by using the Impact Estimation Table as laid out in these chapters. The structure of the Impact Estimation table is reused in several variations, so learn it well and you will be rewarded by much improved control over your projects.

	Solutions		2	
	Short-Cut.Names		Buttons.Rubber	
Product Quality Requirements	Units	% Impact	Units	% Impact
User-Friendliness.Learn.Contac 35 5 by one year	-10	33%	-4	13%
Reliability 100 200 by one year	-5	-5%	10	10%
Development Resources	Units	% Impact	Units	% Impact
Project-Budget 0 100000 by one year	10000	10%	10000	10%

Table: Simple Impact Estimation Table

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Goal [within 1 year] 5 min.

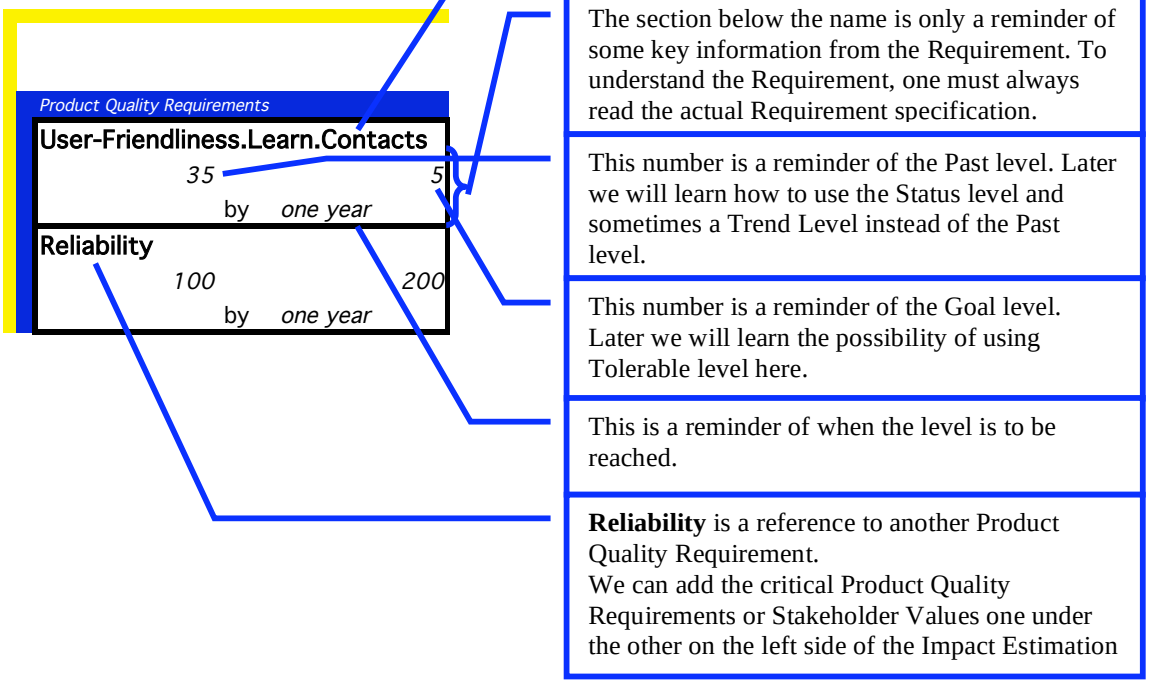


Illustration: The names of the Project Requirements are stated in the upper left corner of the Impact Estimation Table as a pointer to the master Requirement specification. User-Friendliness.Learn.Contacts

Project-Budget

Scale: Project development cost in US$.

Past 0

Goal [within 1 year] 100.000.-

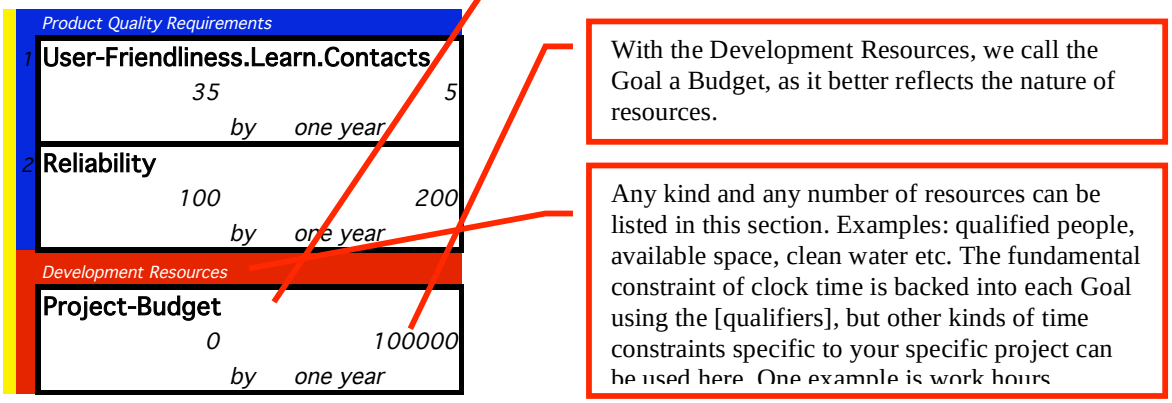


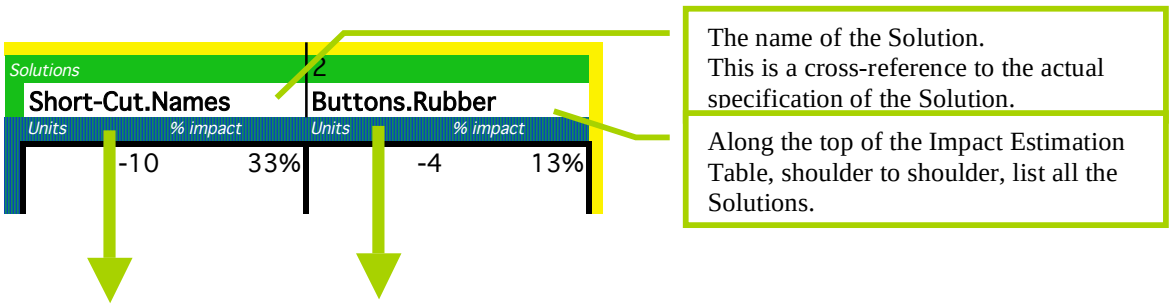
Illustration: Underneath the Stakeholder Value or Product Quality Requirements we list the Development Resources.

Short-Cut.

Names.

Button. A unique button for names

To. that takes the user directly to the names section of the phone.



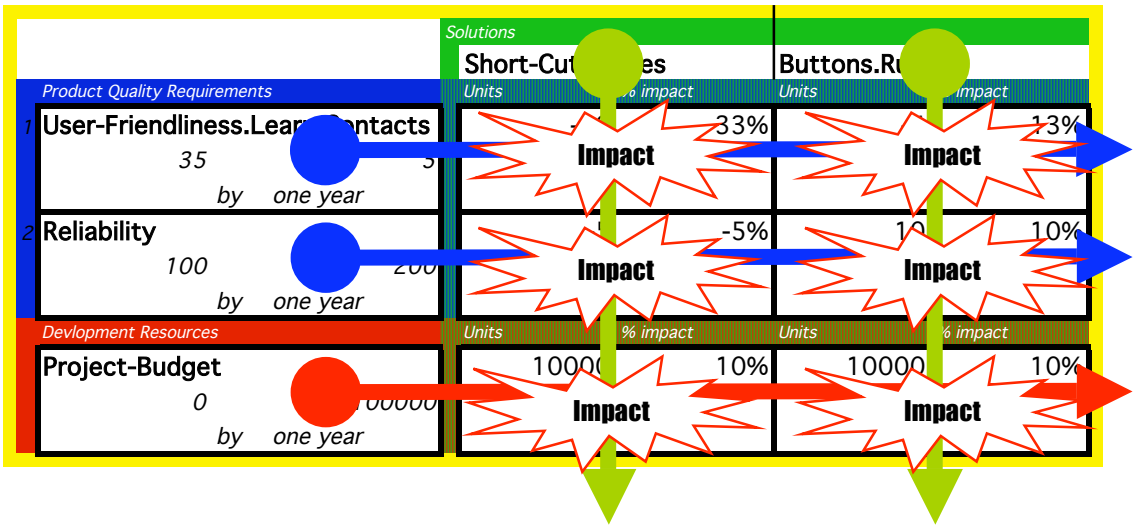


Illustration: We now have the foundation for estimating the impacts the Solutions have on the Product Quality Requirements. Every Solution will be evaluated against every Product Quality Requirement. Even when a Solution is not intended to help achieve a Product Quality Requirement, the impact has to be evaluated. We also evaluate how much Development Resources each Solution consumes.

> Breath is the link between your body and your spirit and your mind.
>
> Sri Sri Ravi Shankar, Breath is the Link

The Impact Estimation Table provides a link between our ends our means and our Development Resources.

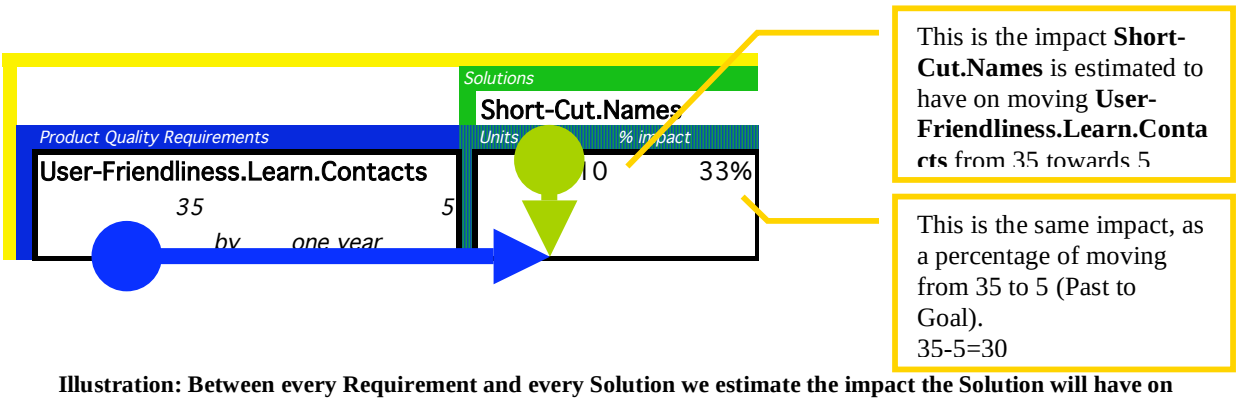


Illustration: Between every Requirement and every Solution we estimate the impact the Solution will have on achieving the Requirement. The impact is initially estimated using a number correlating to the Quantification Scale used in each Requirement. Then it is normalized using a percentage of Goal reached (this calculation can easily be automated by a spreadsheet program on a computer).

To understand this impact, we must first understand our Product Quality Requirement, then our Solution, then we estimate, based on our experience, how the Solution will impact the Requirement.

The Requirement is:

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Goal [within 1 year] 5 min.

~~30 min. improvement needed~~
~~= 100% of the challenge~~

Past 35 min.
Where we are

The Solution is:

Short-Cut.

Names. <- Company Research paper issue 108 page 23.

Button. A unique button for names

To. that takes the user directly to the names section of the phone.

Then we estimate, based on our experience, how the Solution will impact the Requirement.

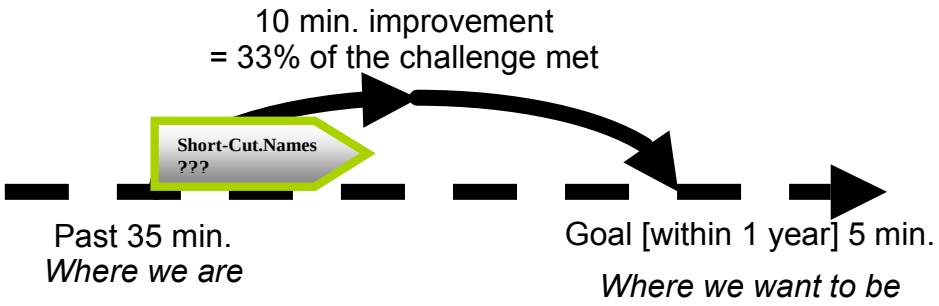


Illustration: The Solution is estimated to improve on the Requirement by 33%. Solutions for another 20 min. reduction is needed.

Ask the people with expertise in the Solution and the Requirement at hand to estimate the effect. Knowing or finding the Solutions that improve User-Friendliness on a mobile phone is the job of the engineers, that improve fuel consumption in an airplane is the job of the engineers, that improve Reliability of a car is the job of the engineers, that improve the living conditions of street kids in Brazil is the job of the field workers in aid organizations.

The job of a medical doctor is to understand health and diseases (Requirements), the cures (Solutions) and how to apply them, and to be able to estimate the effect (Impacts) of the cures (Solutions). Imagine a doctor who is unsure about the disease (Requirements), does not know what medicine (Solutions) to give or how well the medicine will cure you (Impacts). This doctor will soon lose her license. In the same way, in a project, we must collectively understand all our Requirements, possible Solutions and how the Solutions effect the Requirements and finally what combination of Solutions will satisfy our Requirements. The Impact Estimation Table brings all this together in a systematic orderly way that gives us great overview of all the important interactions.

When estimating the impact, our project teams knowledge, or lack there of, will become clear. We have recommended the Solution we call **Short-Cut.Names**, do we know what kind of effect it has on our Requirements. Have somebody used this Solution before? Did somebody read about its effect in a study? Or does somebody just think it will be a good idea? Maybe, for some impacts, nobody can even begin to imagine what the impact will be.

It is not necessary that we know all the impacts, or that all impacts are favorable. The act of filling out an Impact Estimation Table gives us immense insights into our project, its weaknesses, its strengths, where more knowledge is required, where us need an expert, which Solutions are better, which Goal levels are unrealistic or which ones will be the biggest challenge to achieve.

The cost of Solutions

Every project has some Development Resource limitations. Most all projects have some kind of deadline, this deadline is built into each Goal level using the [qualifiers]. Then most projects have some kind of money restrictions, and others have knowledge people, material, space, weight, etc. Anything we can run out of during the development of the project should be treated as a Development Resource. Each Solution must be accountable for how much of these scarce Development Resources they will consume. In the Impact Estimation Table, all the Development Resources are listed underneath the Stakeholder Values & Project Qualities. On the top horizontal row we list all the Solutions. There is an intersection between all the Solutions and all the Development Resources, where the Impact they will have is estimated, or how much Development Resources they will consume is estimated.

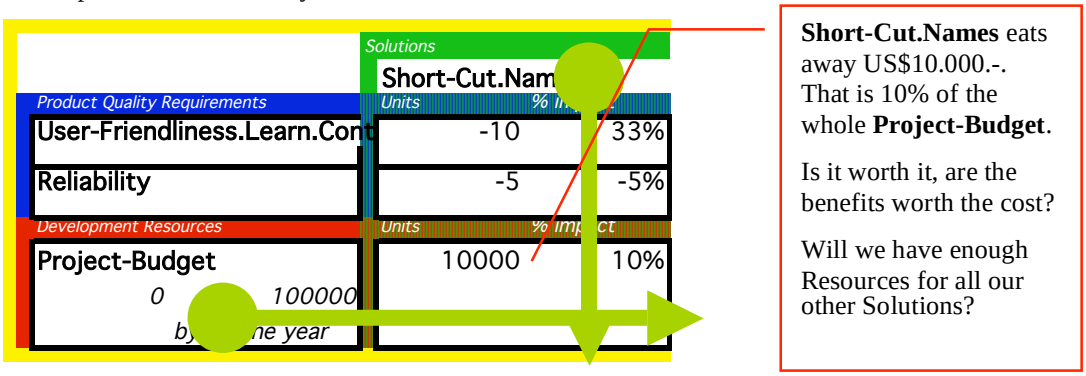


Illustration: In the Impact Estimation Table, estimate how much of our limited resources the Solution uses. This is done on all the Development Resources for all the Solutions.

Also notice that when estimating the effect a Solution has on a particular Goal level, the timeframe (and other conditions) given in the [qualifier] must be consideration in the estimation. We do not only estimate what impact the Solution will have, but what impact the Solution will have within the timeframe give in the [qualifier].

First real improvement, then % of Goal level met.

To keep the Impact Estimation table as realistic as possible, I have found it logical to start with the real number that corresponds to the Quantification Scale in the Requirement, then calculate the percentage from there. It is also possible, and often done, to estimate the percentage first, and not even include the real number.

Simplified Example:

Let us assume we have a Requirement, **Reliability**
mean time between system failure.
In the Past it failed every 50 days
and the Goal is for it to fail no more than every 100 days

Solution, **Backup-Battery**: We plan on adding a backup battery system that will kick in, in the event of power failure.

The next step is to estimate the effect the **Backup-Battery** system will have on: **Reliability** mean time between system failure, and how much it improves it from 50 days toward 100 days.

We need to ask ourselves if we have or anybody else have any experience on doing something like this? If we have or know about such an experience, what did they experience? If we do not know, we can guess on what the effect will be, or we can do tests. We should look at how often the system fails because of power failure.

We could just estimate that it will get better by 15%, but us will find it more credible if us figure out how many more days it will run before system failure. Let's say us conclude that it will run on average 10 more days, with **Backup-Battery** installed, how many % improvement from Past to Goal is that?

When the Goal level is a smaller number than the Past level

As in our example **User-Friendliness.Learn.Contacts**, the Past 35 is higher than the Goal 5, so a smaller number is the better result

.

Illustration: The Solution gets us from 35 minus 10 = 25. The real impact is -10 and should be expressed as such even though it is an improvement.

In the Impact Estimation Table, when expressing the Impact of a Solution where a lower number is better, the real number should than be expressed as a negative when the Impact is an improvement and a positive number if it is worsening. The percentage on the other hand, is a percentage of the Goal level met, and should therefore be expressed as positive if we are getting closer to our Goal level, and a negative if it is taking us away.

	Solutions	
	Short-Cut.Names	
Product Quality Requirements	units	% impact
User-Friendliness.Learn.Con	-10	33%
Reliability	-5	-5%
Development Resources	Units	% impact
Project-Budget 0 100000 by one year	10000	10%

Impacts between Solutions and Development Resources, shows the resources used. A positive number shows what is estimated for it to cost (this is negative), and a negative number would show an estimated saving, that is we make more resources if this Solution is used.

Using numbers or The fear of giving incorrect numbers

During school and quizzes, there typically is one correct number. If a child named Florence is asked, how much is 7+5, and the child answer 13, a big red mark is given. Years of this treatment has given many of us a fear of using numbers, in other than theoretical mathematically correct ways. Yet, in real life, numbers can be very useful, without always being theoretical mathematically correct. Let us assume that Florence grows up and becomes a bridge engineer. A bridge being designed, and she has to calculate the strength of it after 15 years of use. The realities of this is so complex that, if in her calculations, she comes close to reality, that will be good enough for practical meaningful use. In the same way, when running a project, we will find that using

numbers to guide us will be very useful and meaningful, even if the numbers are not correct. Therefore, I encourage the people in our projects to express improvements and impacts in numbers. It gives us a much better understanding of the weaknesses and strengths of our project, even if it is not correct.

Let us say your mother meets somebody they are talking warmly about. You ask your mother about her age, and your mother does not know, so she answers, she is young. Wouldn't it give you more useful information if she guesstimated the age to be between 50 and 65?

Percentage on Percentage, I am confused

If the Quantification Scale is expressed in a percentage, then it can be confusing to calculate the Impact of the real improvement and then the percentage improved. This is done in exactly the same way as with any other impact, we just have to stay more focused.

Simplified example of percentage confusion

Let us assume we have a Requirement, **Look**,
% of people that like the looks of our product.
In the Past 30% liked the product,
and our Goal level is that 80% will like it.

Solution; **Brushed-Metal**: We used to have a plastic case on our product, and we are planning on having a brushed metal case instead.

First let us estimate the real impact to be 25%. That means, we used to be at Past 30%, we will now be at Past 30% + Real Impact of 25% = 55% after the Solution is implemented.

We have moved from 30% to 55% on our way to 80%. The move from 30% to 55% is 50% of where we intend to go, so the Impact in % of the Goal level is 50%.

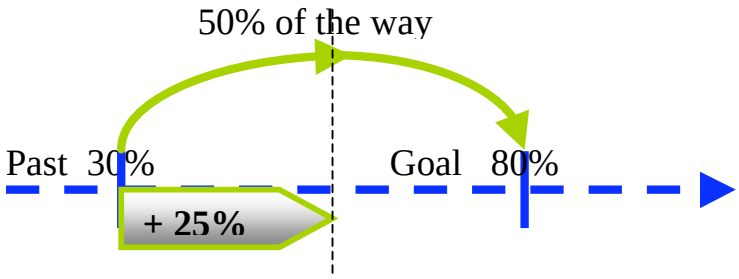


Illustration: Goal - Past = Real Improvement needed to meet Goal. 80% - 30% = 50%.
50% Real Improvement = 100% of Goal

Guessing

When all ells fails, make a best guess at what the improvement will be. Even a bad or wrong guess is better than nothing. It will be a starting point that further discussion and Evolutionary delivery will improve upon. In the Advanced Impact Chapter, I will show several ways to express the uncertainty behind the numbers given.

Solutions and the Impact Estimation Table

Let's review the definition of Solution:

a Solution is anything that gets us from where we are, towards where we want to be, under defined conditions and within a defined timeframe.

If what we have listed as a Solution does not do this, there is no positive impact in the Impact Estimation Table, we do not have a Solution. If you still say, "I believe this is a great Solution" or "I know this is necessary, even though it does not give any positive impact in the Impact Estimation Table", you are probably right, let's look at some of the possibilities.

Requirements are missing!

One possibility is that our Solution is targeting Requirements that are not specified, but should be. Evaluate what Requirements this Solution is intended to improve upon, and if it is necessary to add that Requirement. Be careful not to add Requirements that does not have a real Stakeholder. Another possibility is that there is no such need, and therefore the Solution can be ditched.

Low impact, or Great for other things, but not for this!

If the Impact is zero or insignificant and we where expecting this to be a great powerful Solution. One possibility is that the Solution is great for many other things, but our Requirements are tailored specifically to our Stakeholders needs. We need Solutions that are tailored accordingly. Solutions that are not helping meeting our specific Requirements should be ditched.

<<<Picture: Sales man selling a great jacket, that is way to small>>>

Illustration: Salespeople will often tell us that what they are selling is great without first finding out what our needs are.

Negative Impacts

Some times, Solutions that were intended to have great Impacts on one Requirement, seriously sets back another Requirement. This has to be taken into account when evaluating our set of Requirements. Can we live with the negative impact on the other Requirements, if yes, is it worth the sacrifice on one Requirement to get an improvement in another? Sometimes the negative Impact on other Requirements can be so great and unrecoverable that the whole Solution has to be modified or ditched. Sometimes the negative impact on other Requirements can be won back through other clever Solutions.

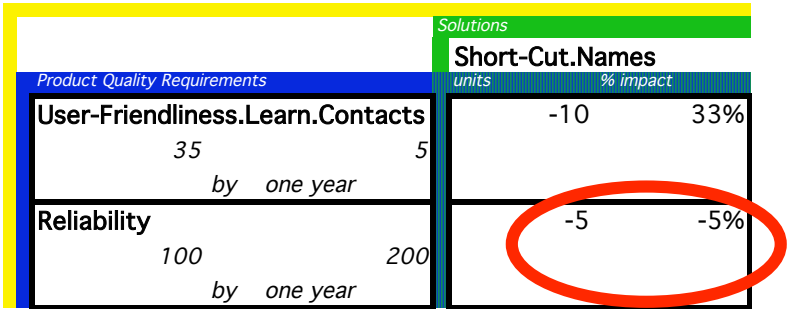


Illustration: Short-Cut.Names was intended to improve on User-Friendliness.Learn, but it has a negative impact on Reliability. Can we win back what is lost with other Solutions, or do we have to throw out Short-Cut.Names from our set of Solutions?

Not complete

Sometimes I see Impact Estimation Tables with a Solution that is eats away on resources, but has little to no effect on the Requirements. When I ask why this Solution is still in there, I sometimes learn that it is a Solution seen as necessary for other Solutions that are held to have great impact on the Requirements. In this case, we must bundle the proposed Solution with the additions that make the Solution powerful.

I have added two Solutions to our Impact Estimation Table. Both of them I believe will do wonders for my project.

	Solutions	2	3	4
	Short-Cut.Name	Buttons.Rubber	Frame	Flash
Product Quality Requirements	units % impact	units % impact	units % impact	units % impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10 33%	-4 13%	0 0%	0 0%
Reliability 100 200 by one year	-5 -5%	10 10%	0 0%	0 0%
Development Resources	units % impact	units % impact	units % impact	units % impact
Project-Budget 0 100000 by one year	10000 10%	10000 10%	40000 40%	10000 10%

Illustration: Notice that neither the Solution Frame nor Flash is helping me achieve any of my Requirements. But they are costing me plenty. As individual Solutions, neither qualify as a Solution as they are not getting us from where we are towards where we want to go. We need to either throw out these Solutions, change them or rearrange them so that they do give a desired effect.

It turns out that **Frame** will do nothing good by itself, but is necessary as a foundation. Flash on the other hand will not work without the **Frame**. What we need to do here is bundle **Frame** and **Flash** together, and submit them to the Impact Estimation table as a package. We can call it **Frame.Flash.**

	Solutions	2	3	4
	Short-Cut.Name	Buttons.Rubb	Frame.Flash	
Product Quality Requirements	units % impact	units % impa	units % impact	% impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10 33%	-4 1 %	-20 67%	
Reliability 100 200 by one year	-5 -5%	10 0%	60 60%	
Development Resources	units % impact	units % im	units % impact	% impact
Project-Budget 0 100000 by one year	10000 10%	10000 1 %	50000 50%	

Illustration: Frame.Flash has now become a strong contender of a Solution.

When estimating the Impact of a Solution, we estimate it as if it was built directly on what we already have. We do not assume other things to be in place before it.

	Solutions	2	3	4
	Short-Cut.Name	Buttons.Rubber	Frame.Flash	???
Product Quality Requirements	units % impact	units % impact	units % impact	units % impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10 33%	-4 13%	-20 67%	0 0%
Reliability 100 200 by one year	-5 -5%	10 10%	60 60%	0 0%
Development Resources	units % impact	units % impact	units % impact	units % impact
Project-Budget 0 100000 by one year	10000 10%	10000 10%	50000 50%	0 0%

Illustration: We could now make a fourth Solution and fill in its estimated interaction with our two Requirements and its claim on Project-Budget.

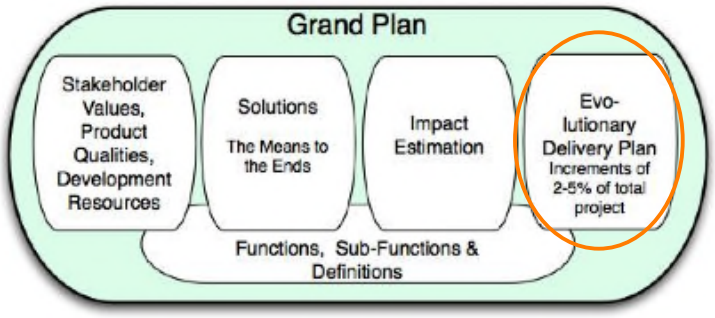
In the chapter on advanced Impact Estimation, we will go through many more elements that can be used in the table. These additional elements will give the project planners and engineers many invaluable insights into their projects. We will also go through several different ways of using the Impact Estimation Table.

IET Summary

IET is short for Impact Estimation Table. It contains two elements, one is the estimation of the effect a Solution have on a Stakeholder Value or Product Quality Requirement, or how much Development Resources the Solution will consume. The other element is making a table displaying all the effects all the Solutions have on all the Stakeholder Values, or Product Quality Requirements, and on the Development Resources.

We use IET to: help us choose one Solution over another, get rid of Solutions that are not efficient at satisfying our Requirements, choose a set of Solutions that together can satisfy our Requirements, to find weaknesses and strengths our chosen set of Solutions have on our Requirements, and as a tool to communicate all this between everyone involved.

Evolutionary Project Delivery or Evolution makes projects fun again



Evolution. Where are you?

Methods in use for developing and delivering projects

Working as a consultant and teacher worldwide, across industries, at all levels and in small to huge projects, I have seen many methods and styles of running projects over the years. Most companies start with one of the standard methods of project management and develop their own method based on those, incorporating their own hard won experience from years in the field.

The problem is that most of these standard methods are fundamentally flawed. They simply do not work. They are working against the realities of how projects work. They ignore the complexities of the world. These standard methods and their variations can lead directly to inflexible, stale, bureaucratic paper methods. The project management method gets in the way of the development team instead of supporting it. The only reasons for still being in business when using such methods is a combination of the competitor using similar methods and that smart people have learned ways to secretly avoid using the methods given to them. I know these are very strong statements, but let us look at some of the problems.

Waterfall, Big-Bang, PERT type & Spiral methods

These methods are the ones that go through one to a few cycles. They start with some kind of feasibility study, goes through planning the requirements, then design, then more detailed design, then implementation, then testing, then delivery, then maintenance. All along there are usually checks that verifies that each Evo Cycle is ready before the next is started. Usually there are strict time schedules as to when these phases are to be completed as to keep the project on schedule.

These methods are sometimes called waterfall methods, because they flow in one direction, Big-Bang, because it will all be delivered at the end, or Spiral, when a big project is divided up into two or three cycles. Tools like PERT planning all build on these principles.

I will collectively refer to all these methods as 'Waterfall methods' as they all have similar detrimental weaknesses.

> Our head is stuck inside a drum, and we think that is the sky. It is so little, our universe. Take your head; lift up---look into that vastness of the creation.
>
> Sri Sri Ravi Shankar, Listen and Celebrate

The Complexity of Reality

Even in a simple and defined game like chess the realities are so complex that any attempt to plan every move at the beginning will be nothing but a poor joke. Yet, that is exactly what is attempted with the waterfall type methods. Somebody plans what to do, and then somebody does it. This is attempted in the complex world of life, not in a simple defined game like chess. Anybody can see that this is impossible. The more complex the realities of our project are, the more hopeless this will become. The more the realities are changing, like new technology, new competitors, changing customers' needs, the more ridicules the waterfall type approach is.

> Project Methods & Chess Principle: Our Project Development methods should be more than good enough to play a game of chess, which is simple and well defined compared to the realities of our projects.

If there are no new technologies, no changing customer needs, no improvement is done over the old, the market is stable, nothing is changing, all is known, then a waterfall type approach can be used. In fact that is how it was developed. It will work for building ten more houses, which are exactly the same as the 10 previous houses you build. Lay it all out on a PERT chart and start building.

Have you ever heard of the concept of freezing the requirements? This is the norm when the requirements are done on paper, and it is decided that it is unacceptable to change them, as that would mess up everything else in the waterfall method. This 'freezing of requirements' proves my point, the methods are developed for situations where all is known and nothing is changing. Yet, that is not the realities of most projects today, so people decide to freeze the requirements artificially. We can decide to freeze the requirements if we like, but the needs of the Stakeholders are changing, we just decide not to fulfill our Stakeholders, our customers, needs. People are working with methods that are working against the realities of their projects, they should be using methods that help them and support them.

The Poor understanding and specification of Requirements

Everything in a Waterfall type method builds on getting the Requirements of the project done correctly first. If they are wrong, the design will be wrong etc. The problem is that there are fundamental misunderstandings about what requirements are and how to specify them. As discussed earlier in this book, most people mix ends and means, they do not consider all of the Stakeholders needs, they just look at the product as if it was to live in isolation. If they did manage to separate the ends and the means, there exists little culture or knowledge on how to specify the ends. So when the whole project is dependent on good stable Requirements the project will inevitably go wrong.

Of cause, the reality of all the projects I have had the pleasure of working with, there were no way of knowing all the requirements or designs up front. So even if I know how to specify them well, I would not know what they all where, and the ones I got right would become wrong as reality changed.

The 'New Document Type' way to make sure it gets done correctly, or The creeping death of an organization.

Because of the complexity of reality and the need to get things right up front when using a Waterfall type project, most companies have developed a sea of documentation types that aim at covering all possible angles. As new problems pop up during the years, new types of document types are developed to try to handle the problems. This will slowly eat away all our productivity. Many times I have worked with groups of people that have put huge amount of time and thought into developing documents that we later have found not to be used for anything or by anyone. I will stop this argument here, because if you are in a situation with too much paperwork, and too many document types, I am sure you are painfully aware of the ridicules length this foolishness can take a company and the troubles that follow. Just know it is not necessary.

Far away from the realities, or, Disconnected

People working in Waterfall type projects are often completely disconnected from the realities of what they are working with. The bigger the project is, the bigger the disconnection is. Some people are part of a team writing the requirements for a project only to pass it on to someone else to develop it. Sometimes there is little or no feedback, and then the people writing requirements start on a new project. After a few years of this treatment, the engineers' lose contact with the realities of what they are doing. They are just working the paper mill. There is no way of knowing if the requirements are great or terrible. Managers start measuring the

performance on how much paperwork is written. Motivation plummets and the quality of their work suffer without them even knowing about it. The people are disconnected from the Stakeholders of their own projects.

Quality control & Inspection methods, or, Get it right now or suffer later.

One of my expertise's is developing and teaching quality control and software inspection processes. These processes have many great advantages and can be very beneficial on all kinds of projects including Waterfall type projects. Because of the enormous expenses of getting something wrong up front in a Waterfall type project, we see the strongest demand on those types of projects. Allot of effort and money is spent on getting things right with these methods, and where people are not spending the time and effort, they pay even more (10 to 100 times more) later on. Quality control and Software Inspection methods are very powerful, but are often used to soothe out the problems of waterfall type methods. There are much better ways of fixing those problems, and quality control and Software Inspection methods have much better uses.

Managing Waterfall type projects, or, Pretending to be nice but I am no fool.

I do not exaggerate when I say the waterfall type projects do not work, they don't! They are a farce and a disaster. It is simply impossible to run a project according to those models. Yet, that is what most people do, so they must work? No!

People do not follow the methods given to them, they find ways around them. People pretend to be following the methods, to satisfy the bureaucracy, and believe me, waterfall leads to massive bureaucracy. They write documents unrelated to what is happening, just to have them filled out.

Microsoft engineers do not use their own PERT like tools as they are based on the waterfall method, they can't, because they have real projects, with real challenges, unknowns, changing requirements, new technology, competitors etc. They use methods that give feedback and accept changes.

It is the norm for people to write the Requirement documentation after the Design documentation has started. It is the norm for people to start building or coding a project before the Design or even the Requirement documentation is started. Project managers end up using their own, undocumented methods to get some feedback, and I support them in that. In fact, that is what smart people do, as a project is doomed if it follows the waterfall type method. Even 'ad hock' methods or 'no method at all' can be better than most waterfall type methods.

Smarter people change the methods, so the methods support the development, and not hinder it.

<<<<<< (NOTE: Below is just ramblings so I remember what I want to get across, all facts are different and will be changed accordingly) Case Study Example: Ericsson Base Station Asia

Mr. .. and Mr. … where given the job of developing and Delivering a Mobile Telephone Base Station for the Asian Market in 12 months, and the normal expected time for this kind of development and delivery was 36 months. Despite threats from the methods people within the organization, they had to abandon their normal waterfall method development and adapt an evolutionary development method. They managed against all expectations to do the impossible and delivered successfully to the Asian market within the timeframe. They have since been promoted many levels within the organization.

Source: About Succeeding >>>>>>

Evolution. Basic ideas and principles

Evolutionary Project Delivery is a project management method used to ensure delivery of defined Product Quality Requirements within time, budget and other resource constraints. It is best described as an ongoing learning cycle. At every cycle it strives to meet more of the Stakeholders Requirements. People involved are learning anything and everything needed to successfully deliver the project. Evolutionary Project Delivery has a uniquely successful track record that no other project management method can claim.

Let us start right of with some guiding principles

Evolutionary Project Delivery top principles are:

1. Learning: Evolutionary Delivery is a Learning cycle. Learn from reality. Learn from experience. Learn what works and what does not.

2. Early: Learn early enough to change what needs changing before it is too late.

3. Small: Small Evo Cycles gives small risk. By keeping the delivery cycle small, little is lost when, not if, a cycle fails to deliver.

4. Simpler: The complex gets simpler and easier to handle by dividing it up into smaller parts.

Learning: Defined: Learning can be defined as a change in behavior.

The very simple explanation of doing a simplified Evolution project

Plan-a: Make a high-level overview plan,

Plan-b: Split that plan into increments. And select one increment to be done first.

Do: Do the first increment. Aim to give some real improvements to Stakeholders within a short period of time.

Study: Measure and study how well the increment did, compare it to the expectations and learn from that experience.

Act: Based on what we learned, keep the increment, throw it out or make the necessary changes!

Then we start all over again, and we continue circling until the project is done.

We divide the process into four main steps that makes a learning cycle. This is known as Deming's Plan-Do-Study-Act cycle.

Feedback: The most fundamental principle of any project management method, or, Feedback or get lost.

Feedback from reality! The earlier the better. The more relevant the better.

We must make sure we are getting some feedback from reality as early as possible. With reality I do not mean that we have another shiny document, I mean some real or simulated Stakeholder has seen some kind of improvement in their/its life.

With relevant I mean real Stakeholders needs. Many people measure many things in their projects, but few people measure the few critical things. Divide our project so we can deliver some part early, and learn from the experience & the results.

To get early feedback, a chance to learn, and control, we divide all projects into smaller parts. Then we can do, deliver and get that valuable feedback early.

Incremental Development divides the whole into smaller parts, and delivers it part by part. Evolutionary Delivery begins with something that is working and improves it systematically.

Evolutionary Delivery aims to get feedback and learn from each cycle, and apply what was learned to the next cycle. Incremental Development does not. Evolutionary Delivery is focused on achieving quantified Stakeholder Value and/or Product Quality Requirements. Incremental Development just builds the parts one by one.

It is important to understand the difference:

Incremental: dividing the development into smaller increments. The increments are not typically useful without the other increments, no or little learning from the increments.

Evolutionary (Evo): Is incremental, but with the intention of the increments to some degree being useful to some real Stakeholders. In each Evo Cycle, the Stakeholder Values and/or Product Qualities improves in the direction of the quantified Goal levels. We learn from the Evo Cycles and its use with Stakeholders, and change the plan (Requirements, Solutions, Evo Cycles, anything) accordingly.
Evo = Incremental + Whole + Steering towards Requirements + feedback, learning and change.

Illustration: My wife Florence, with my daughter Mira-Bai Evolving from age -1 month to 3.5 years.

A human being is not put together using an incremental development process, one part at a time, first the legs, then the body, head etc. Humans evolve in an Evolutionary process, where we are more or less complete with all main parts from an early stage, then in time we grow, learn and our abilities evolve, before we eventually hit the bucket.

Steer it to the Goal, or, The Scud and the Patriot missiles

Evolutionary Delivery is all about learning from experience.

> Many problems have come, many problems have been solved, and each problem has enriched life in some way, has brought up some strength in you.
>
> Sri Sri Ravi Shankar, Listen and Celebrate

In waterfall type projects, the Goals or requirements are developed, then signed as approved, and sometimes frozen. Then all activity thereafter is generated towards meeting those Requirements, but there is no feedback, other than time passed and other development resources used, as to the success of meeting those Requirements.

This is like the Scud missiles, which were programmed by the Iraq military during the Gulf war to hit targets as big as whole cities. Even though the Goals were stable, the cities did not move, it was almost impossible to program the missiles to hit their targets. Once aimed and shot out, there was no way of steering or adjusting according to realities of shifting wind and atmospheric conditions etc.

In Evolutionary projects, the Requirements are carefully developed with an improved understanding of whom they come from (Stakeholders), what they are (ends and means), and how to express them (quantified, measurable and testable). Much less time is spent in the early development of documentation. We accept that we do not fully understand all the Requirements, Solutions & technologies, and the interaction between them. We assume the world moves & needs change. We start building the project and delivering the results with early and frequent feedback on the progress towards the Requirements and constant adjustments to everything.

Evolutionary Project Delivery is very much like a Patriot (heat seeking) missile, capable of hitting flying targets. Little time can be spent on the ground planning the flight path before launching a Patriot, and yet the Patriot can hit the Scud (or another rocket) mid air. This is done by keeping a sharp eye on the Scud (the Goal), constant measuring and feedback (Evo Cycles) on where the Patriot (Solution) is relative to the Scud (the Goal), and constant adjustment to correct the Patriots flight path as necessary to hit the Scud (Learning cycles & Evolutionary Delivery Cycles).

The analogy of feedback and adjustment, or, open your eyes or crash!

Open your eyes! You don't drive cars blindfolded, but some of you seem to be 'driving' your projects blindfolded.

In Evo Project Management,
Stakeholder Values and Product Qualities are the 'destinations',
Meters are the 'eyes',
Evo Cycles ensure that the Meters are used continuously,
Evo Cycles also enable learning and immediate adjustment.

Continuous feedback from our project and its realities is a must for being able to successfully re-adjust the course towards the Requirements.

In order to drive a car successfully, we must not only have eyes but also continuously and immediately use our eyes and adjust our steering accordingly.

In order to 'drive' a project successfully, we must not only have quantified Stakeholder Values & Product Qualities (destinations) and measure with a Meters (eyes) our progress towards them. We must do it continuously (Evo-Cycles) and in real time learn and adjust our steering accordingly (Evo-Learning).

From Complex to Simple, or, How to eat an elephant

Many of today's projects are extremely complex and almost impossible to manage successfully. Using Evolutionary methods we make complex projects less complex and manageable.

By first focusing on what needs to be accomplished, the Stakeholder and Product Requirements, a top level overview is provided, even in the most complex of projects.

By focusing on delivering a few Requirements at a time, simplicity dawns, even in the most unmanageable of projects.

Something that is simple, the mind cannot accept. The ego wants very complicated and difficult things.

Sri Sri Ravi Shankar, The Concept of God

By not having to plan, up front, in detail, by not having all of the Requirements and Solutions and all of the technology and all of the interactions between them; but by keeping the planning only to what is at hand, what is real, what is next; the people working on the project stay close to the action, motivated and efficient.

During integration in waterfall type projects, too many Solutions are all integrated at the same time. It is very complex to understand what goes wrong (it will!). In Evolutionary projects, the next, small Solution, or Evolutionary Cycle is integrated into a working core. When something goes wrong (it will), it is simpler to analyze, understand and fix the problem.

As for the elephant, I do not know, I eat vegetarian food;-).

Evolutionary Delivery Cycles, or, Satisfying Requirements, not building even more…

In Evolutionary delivery projects, the projects are divided into small increments called Evolutionary Delivery Cycles or just Deliveries.

These Evolutionary Delivery Cycles are not Cycles of more pages written on the project plan. Finishing and delivering the Requirement documentation is not an Evolutionary Delivery Cycle, because product documentation delivers no real improvements to any Stakeholder or Product.

In the simplest, most relaxed form of Evolutionary delivery, sometimes called 'incremental delivery', delivering a part of a Solution, but no improvements to the Stakeholders or Product is called a Delivery Cycle. In proper, more-powerful Evolutionary Delivery, delivering a part of a Solution is not valid. Only actual improvements in Stakeholder Values or in Product Qualities are valid Evolutionary Delivery Cycles.

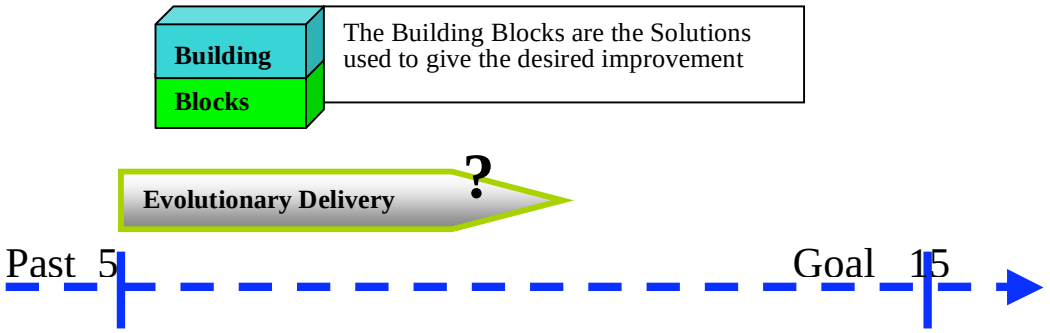


Illustration: The fact that we built and delivered something does not make it an Evolutionary Delivery Cycle. Evolutionary Delivery Cycles are defined by the improvements made, on Stakeholder Value or Product Quality Requirements, from Past (or Status) towards Goal levels

For Evo Cycles to be valid, what we plan is irrelevant, what we build and deliver is irrelevant. Only the actual real improvement to a Stakeholder Value or a Product Quality Requirement is relevant.

Seven steps to steer our projects towards success.

First, we must have 'success' itself clearly defined and planned.

This is done through first identifying our Stakeholders and their needs. Then specifying them quantitatively in a Stakeholder Value Requirement specification.

Second, we must know the difference between, and separate, the Ends and the Means, the Requirements and the Solutions.

The Requirements must be separated from the Solutions. The Requirements are considered holy, we should meet them by any means, any Solutions, possible. The Solutions are the workhorses, and can be changed at any time a better horse comes along.

Third, we must have a method for selecting, at any time, the best Solutions, and the best next Evolutionary Cycle.

There is no single static plan in the beginning, that is followed through to the end. Every current Evo Cycle is evaluated according to the current situation. Use the Impact Estimation Table and the different uses of it, to evaluate which Solution or Evolutionary Cycle to carry out next.

Fourth, we must be able to track and study the results each Evolutionary Cycle is giving us in relation to our Requirements.

Initially, we have ideas about how good Solutions and Evolutionary Cycles are. But reality is usually somewhat different. We measure, collect and learn from the realities. This gives us the ability to see our current situation more clearly. We don't drive our car blindfolded. Yet, that is what we are doing when we manage our projects without frequent feedback from the realities. Let's open our eyes, and see what is happening. We can do it through the Evolutionary project management method, with clearly defined Requirements and early measurements of actual progress towards them. We can track the progress with a 'project management' variation of the Impact Estimation Table.

Fifth, we must be able to learn from current results, and as necessary change anything to help us reach the current Requirements..

It is easy to get stuck on ideas of how things are, or how things should be done. When Evo feedback tells us differently, we must act on that reality.

Sixth, we must stay nimble. We must be alert and proactive to changing Stakeholders' needs.

Satisfying current Requirements is our primary reason for existence. As the needs of the Stakeholders change, so do their Requirements, we must be aware and proactive in dealing with the changing needs. Our set of written Requirements where probably incorrect from the start. We must actively seek to learn what the real Requirements are. We must constantly be monitoring for new and changing needs from our Stakeholders. This is best done through having close contact with our Stakeholders. Whenever possible they should try out, or use, some of our early deliveries.

> The moment you make an effort to love somebody or to be happy, you cannot.
>
> Sri Sri Ravi Shankar, Compassion And Trust

In the physical world, in our projects, effort is needed, we must be proactive.

Seventh, and flexible in our approach to meeting them.

Early on in a project, we can identify Solutions that we think will best satisfy the Requirements. These Solutions give us a place to begin and confidence in that there is a way to satisfy the Requirements. However, normally we can not foresee the intertwined complexity of reality before reality hits, and the initial Solutions will quickly show their insufficiency. To at all meet the Requirements, we have to modify and find new Solutions.

But here, in the middle of development, I encourage you, because now you finally can get the necessary insight, to find the brilliant Solutions. The Solutions that no-one could think of before, they just did not have the reality of the complexity, the Stakeholder feedback, the feedback from the last Evolutionary Cycle, the experience working with the development team etc. Throw out the Solutions you initial had planned to use, and replace them with better ones. Before, during and after each Evolutionary Cycle, keep searching and thinking of Solutions that are better suited to satisfy the Requirements.

Delivering real benefits early or all at once later, or, would you like a drink and an appetizer while you wait for your main meal?

In an Evolutionary project we do not only get increased control, we and our Stakeholders get early benefits.

In a Waterfall type project, our Stakeholders will not get any benefit during the development. In larger Waterfall type projects this can be several years of nothing, except our Development Resources are running.

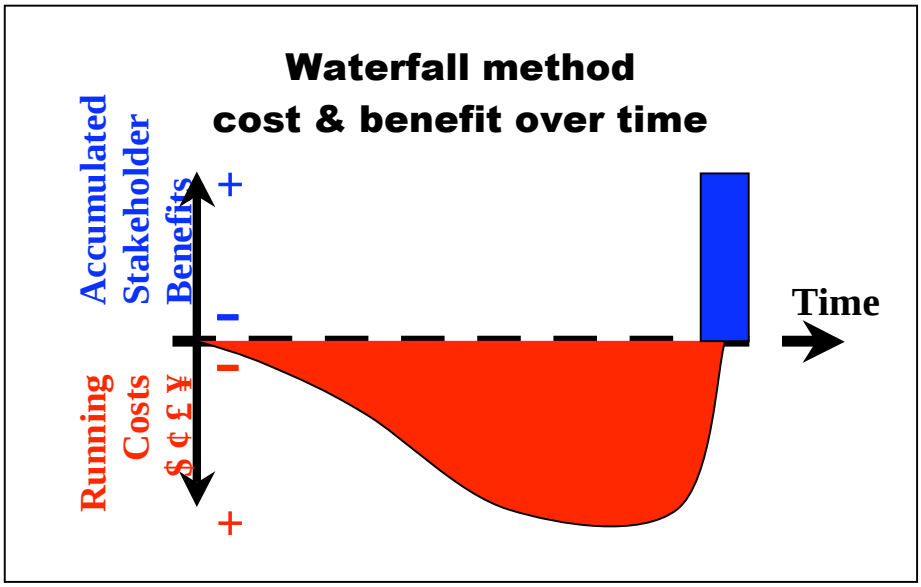


Illustration: In Waterfall type projects, the Stakeholders gain benefits at the very end of the project as illustrated in blue. A typical Development Resource curve of Waterfall type projects is shown in red as integration testing towards the end of the project is exceedingly time consuming and expensive, normally blowing all budgets.

In Evolutionary Projects, each delivery is aiming move us from the Status towards the Goal level of our Stakeholder Values. Some smaller deliveries might not be used by Stakeholders, but many deliveries may be. Often early deliveries are charged for. It is common in the Software Industry to release Betas (early not finished versions) of software, and some charge for the releases as well, making money before the product is finished.

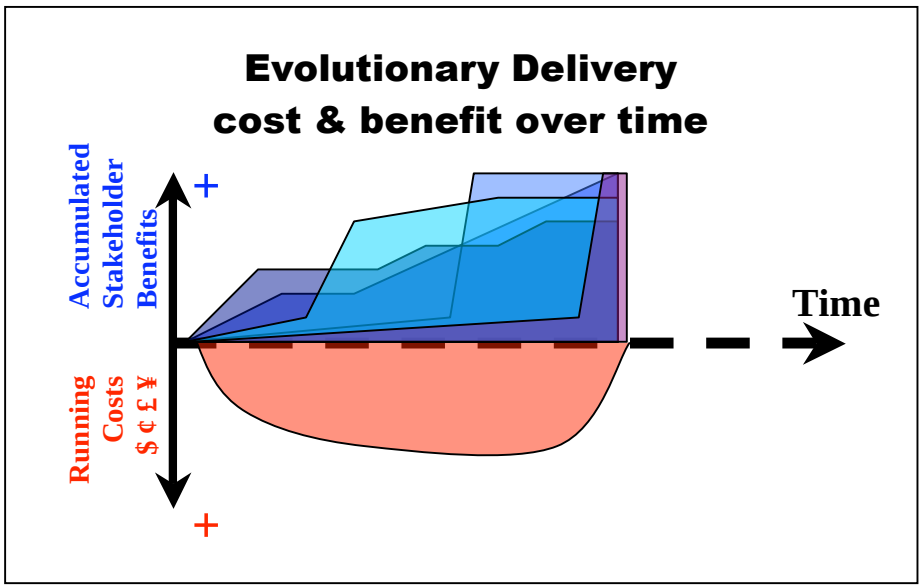


Illustration: In Evolutionary Delivery Projects the Stakeholder gains benefits early as illustrated by bluish colors. The Development Resources curve starts with more investments up front, and then it evens out and ends without exploding Development Resources towards the end of the project.

<<<<<<<kai note: Chris Dale pointed out that one axis is accumulating while the other axis is running cost (rate). Needs a fix? >>>>>>

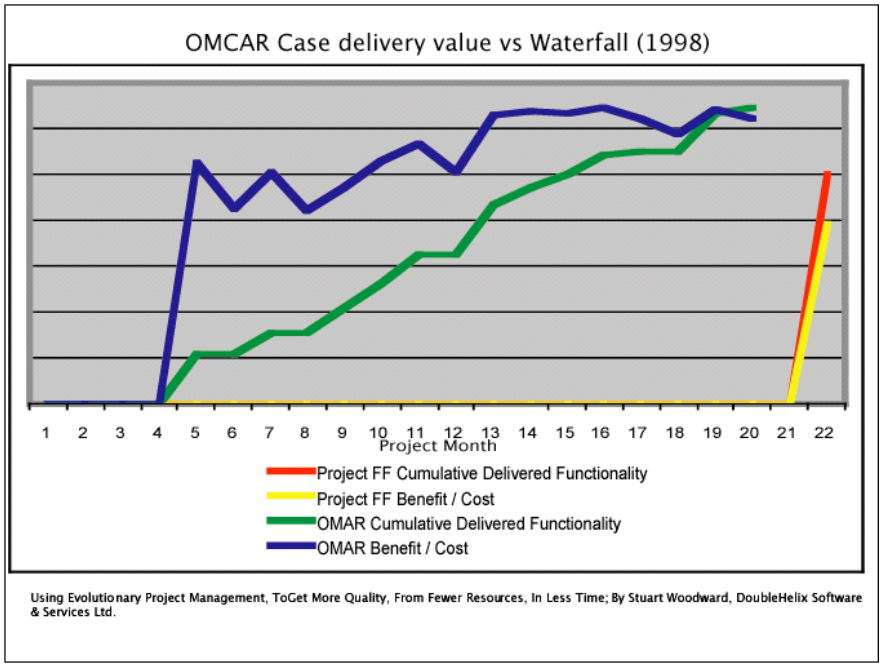


Illustration: A Waterfall Project is plotted on the same graph as a Evolutionary Delivery Project. Notice that the Evolutionary Delivery Project delivered more by month 15th than the Waterfall project ever did. Source: Using Evolutionary Delivery, to get more Quality, from fewer resources, in less time; by Stuart Woodward, Double Helix Software & Services Ltd.

Evolution. Taking you from your previous plan towards the Evo way

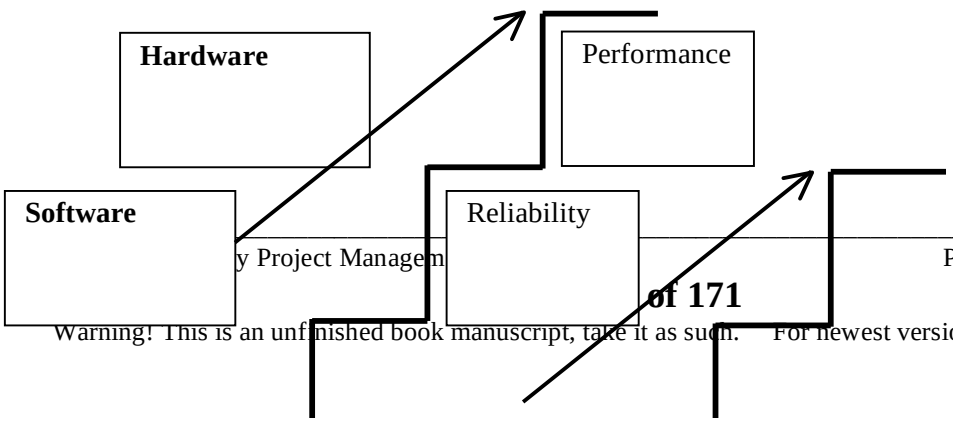
Simple Evo

Let's look at a simplified, yet powerful method of dividing a project into Evolutionary Delivery Cycles. The simplified method can be applied on any project at any time without first shifting the focus to being driven by End-State type Requirements. We have successfully used this simplified Evo methods when called in to help projects recover from disaster at very late stages of development.

The Process: Divide the whole project into smaller parts. Then pick one of those parts and divide that part into smaller parts. Then again pick one of those parts and divide that part into yet smaller parts. Then pick one of those parts, and develop and deliver that part. Then the next part etc. When all the smallest parts are developed and delivered, divide something else up into smaller parts and pick, build and deliver those. Ideally each part should to some degree work.

When we do this simplified Evo process we usually draw staircase steps and divide the bigger parts into smaller steps.

Here is an example of how this might look like applied to our example project of developing a new mobile telephone.



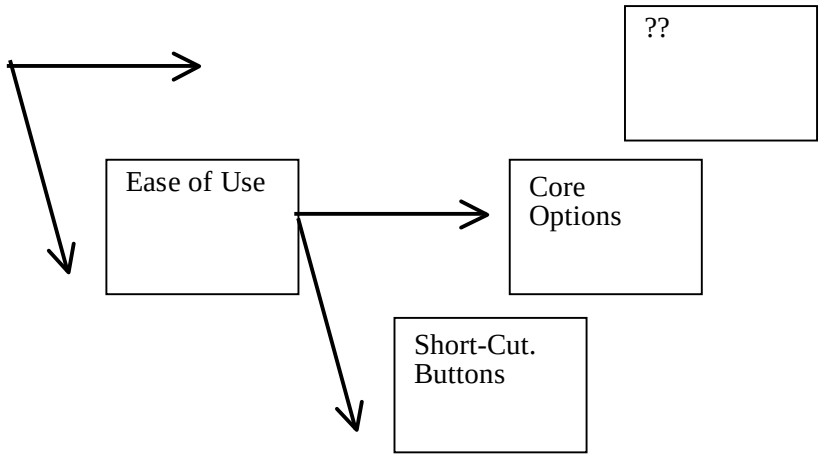


Illustration: The simplified way of doing Evo, divides the whole into smaller parts. When it is small enough to be done within the timeframe of the Evo Cycle, all priority is given to developing and delivering that small part. When that part is completed, the developer examines what was learned and decide on the next Evo Cycle and again begin with detailed planning of that single Evo Cycle, etc.

This is just one example of how this project could be divided up into smaller delivery cycles. Another of potentially many different approaches could be to keep the Hardware & Software together, and divide it up from there.

All parts are not planned in detail, they are left for when their time has come for development. The focus is on the Evo Cycle at hand. Each Evo Cycle is developed all the way from the planning stage to delivery. From every Evo Cycle we learn what works and what does not, and the overall plan is continuously updated accordingly. Then the focus is shifted to the next Evo Cycle. Detailed planning only happens with the cycle at hand.

This bare bone Evo method makes big, complex and unmanageable projects, small, simple and manageable. Following this bare bone Evo method we can easily reap many of the benefits of proper Evolutionary Delivery.

Some weaknesses of the 'simple way'?

With this simple Evo method, intelligent selection and priority for what to develop and deliver is limited. The focus is on the building blocks and not on the Stakeholder Values or Product Qualities. We could risk successfully dividing and delivering the product within Budgeted Development Recourses, but fail to deliver required Stakeholder Values and Product Qualities. With this simple Evo method, both feedback and the use of feedback is limited. Without good feedback, necessary adjustments can not necessarily be done.

Proper Evolutionary Delivery, or, You got to dance the dance to get the full benefit.

Most of us will have to make a shift in our thinking. When managing projects we normally have thought in terms of the thing and the functionality (Functions) we deliver. Proper Evo's main focus is to deliver improvements to Stakeholder Values & Product Qualities. We deliver these improvements with any means imaginable. If we deliver an Evolutionary Cycle, we succeed to the degree the Stakeholder's Value or the Product Quality Requirements are meet.

> If you want to grow in Divine Love, you have got to drop the pride and all of the artificial wall we build between ourselves and others. In the "wall" we keep judging others, and we think others are judging us.
>
> Sri Sri Ravi Shankar

In spiritual development, this "dropping' is the challenging part. To implement the Evo, with all its benefits, we must first drop our waterfall concepts with all its baggage as this baggage hinders our understanding and execution of what is so beautifully simple. Old baggage can often be one of the biggest challenges to successfully implement Evolutionary Delivery methods into a company culture and must be taken seriously by those managing the change.

To get from the previous plan to the Evolutionary Delivery plan, we have to convert our old project as described earlier in this book. This includes understanding the difference between ends and means, separating them, specifying the ends measurable, and preferably setting up an Impact Estimation Table to understand how the Solutions impact the Requirements.

Evolution. The Planguage starts forming

Slice & Dice

There are many ways to slice & dice a project. Let's look at some principles that will help us in making intelligent choices.

Evo Planning Policy (example)
Development Recourses: Evo Cycles must consume less than 2% of any Development Resource, or, Evo Cycles must not cost more than we can afford to lose with a smile.
Time: Evo Cycle time must be less than 2% of the total project time, or, Evo Cycles must be a week or shorter.
Value: Evo Cycles that deliver the most improvements to Stakeholder Values or Product Qualities, for the Development Resources they consume, will normally be delivered first, or, do the juicy bits first!

We can vary the numbers (2%) as we see fit. Experience shows that weekly cycles work well with most projects, but some larger projects have reported great success with one-month cycles, yet other projects have cycles down to one day. If the cycles become bigger than about 5%, we will probably start losing many of the benefits given by Evolutionary Project Management.

In this Evo Planning Policy example, the principles are put into effect with the use of a Policy to be followed by project planners and engineers. A Policy gives authority from a higher level of management to operate within the boundaries of the policies. To break the policy requires approval from the appropriate level of management.

The Size of the Slice: So we can smile

If a Evo Cycle is developed and delivered, and it fails to deliver the expected benefits to the Stakeholders, and the effort for that Evo Cycle is lost! Can we then walk away from that fact with a smile knowing that we learned something, and that we have plenty of time and resources necessary to succeed in the project?

The two first points in the policy state that the project cycles should be smaller than 2% of both the budget and the overall project length. We have found that 2% works well in general. That means if a project takes about one year, a typical cycle is 1 week or less. If it is a four year project, a typical cycle is 1 month or less.

The Size of the Slice: Don't take a bigger bite than we can chew well.

In Evolutionary Delivery projects, we make commitments for each delivery. What is to be delivered within what timeframe. The manager for that delivery must believe they can deliver what they commit to within the

timeframe. If they are un-comfortable with that, they can commit to delivering less, until they feel comfortable committing themselves and their team. The delivery has to be 100% done. It is easy and dangerous to be almost done. What done is has to be specified in the commitment. It usually includes a specific measurable improvement to a Stakeholder Value or a Product Quality or both. It should also have a deliverable level of faults or bugs. Sometimes other supporting elements like documentation or completed change control entries in a database have to be completed as well.

The Size of the Slice: The pressure is on and on and on, and it feeeeels good.

When you studied for your last exam, how much did you really study well in advance? Or did you mostly study last minute. Most of us have done a lot of last minute studying. The same happens in projects lasting several months or years, we take it very easily in the beginning, and work overtime towards the end. Working slowly in the beginning can be boring and stressful, and working overtime towards the end of the project can tire us out and make us sick.

In Evolutionary Delivery the short cycles ensures that we have pressure to deliver all the time, thus improving productivity massively, but the pressure ends up being at a comfortable level. People enjoy the feeling of being productive and doing real things. It is never boring, always something real and important to do.

In general people tend to like one-week 'pressure' cycles. One month seems too long; nobody is feeling the pressure the first week(s).

In one-week 'pressure' cycles people don't develop more than what is required. That is, they don't create nice little extras that the Stakeholders never officially asked for. These seemingly well-intended extras can cost a project a lot of unwanted headaches. For a starter, extras will probably not be tested. They might not integrate well with other parts etc. and, well… no one is paying for them. Projects are known to fail on extras alone.

Evolutionary Delivery projects have been organized in weekly cycles, where the team gets to go home for the week when they are finished with the cycle. That being Thursday, Friday or Saturday. This little game makes people real good at estimating what they can do in a week, and spending their time wisely. It keeps people from developing unwanted extras. -"Sure you can develop this extra thing or talk at length about something interesting, but I want to get home to my family early Friday afternoon."

> Learning is inevitable. By doing things right you learn, and by doing things wrong you also learn. From every situation, from everybody, you learn either what to do or what not to do. Either by mistakes or by doing things correctly, you cannot but learn. Learning is inevitable.
>
> It is only when you sleep that you do not learn. And if you are asleep in your life, there is neither pain nor pleasure nor learning. Most people are in such deep slumber. That is why many people do not even make an effort to get out of pain.
>
> Sri Sri Ravi Shankar – Weekly Knowledge #360 Guru's Tidbits

If you are not learning during development of your projects, if you are not feeling the pain during the development of your project, you are asleep. Evolutionary Project Delivery is the wakeup call!

It is all about learning. Learning what we understand and what we don't, what works and what doesn't, how much time it takes, how much it costs, who the Stakeholders are and what they actually want, about the market, technology, our project team, etc.

The Content of the Slice; Stakeholder Values or Product Qualities

A Evolutionary Delivery is a delivery of some improvement in Stakeholder Value or Product Quality. To deliver this improvement, normally, but not necessarily, some addition to the product that is thought to give this improvement is developed and delivered. Yet, from a project manager's point of view, only the actual improvement in Stakeholder values or Product Quality is important. The Project Manager is using these improvements to maneuver the project towards success.

The Content of the Slice; Functions – No!

This is where most people fail, they Evolutionary deliver new functions. They do get some of the benefits of Evolutionary Delivery, but they miss out on most of them. It can to some degree work on run of the mill products, but will not work on state of the art products. In either case, improved control is achieved by focusing on delivering Stakeholder Values & Product Qualities.

At one level, all the functions are already there, they are just not as good as we like them to be. Lets take the example of a word processor. Some of the nice functions of a word processor are spelling & grammar checking, formatting, cut & paste etc. If we observe the products preceding computers, all these fancy functions where already there. With paper and pencil, scissors and glue, and a trained brain, we had all the same functions available. The computer based word processor does not add any new functions, but it adds qualities like speed, accuracy, consistency, etc. thereby delivering on Stakeholder Values, of being able to write quicker and nicer etc.

When the computer was first introduced to the mainstream, as long as a function was computerized, it was so much faster than doing the same job manually that it was seen as great. As computers continue to mature, computers and programs will be judged on how well it does something, not just that it can do it. If we want to introduce a word processor today, it has to do something better than the other word processors. It can be smaller, faster, easier to use, more compatible, more stable, prettier, cheaper etc., and possibly some variation of a 'new' function, but we will see less and less of that. The 'new' function will be new to computers, not new to Stakeholders. The Stakeholders has already had a way of doing what the 'new' function is doing, but maybe the computerization of the function can save the Stakeholders time & money. Improvements to the lives of the Stakeholders are what we as project managers get paid to deliver.

The Content of the Slice; Functions – well, Yes!

Yet, the improvement in Stakeholder Values & Product Quality is normally delivered thru some new improved function or product. In a car, we can give improved road grip by developing and delivering a better tire, in a phone a better screen, in a word processor, speed, by optimizing the code, or user friendliness thru rearranging the menus etc. The possibilities are endless. Yet, all of them are there only to serve the higher Requirement of improving Stakeholder Values. So yes, we probably have to deliver some fantastic functions, but remember that they are only the means to an end. When we measure them all up against how well they improve Stakeholder Values and Product Quality, we will find that you will be much more effective and flexible in delivering our projects.

Start from what is already there! ,or, A flying start…fly baby fly!

Have you noticed how I always say we must move from the Past or Status to the Goal level, not from nothing to the Goal level? I find people tend to start from nothing, from scratch. In Evolutionary Delivery we start from the Past or Status level, and improve from there. This does not mean that the finished new product needs to have anything remaining from the old. It can be completely replaced.

In one project I worked for, my client had their own software based product that they had already sold to their customer. Their customer had previously developed their own software product to do something related but different than what my client' product did. Then they asked my client to evolve their product to also do what their in-house product was doing. The customer was very pleased with their own software product, but thought it would be even better if they had one product to relate to, instead of two. The customer also wanted to get rid of the responsibility of further developing and maintaining their own product, so they could focus on their core business which was not software development.

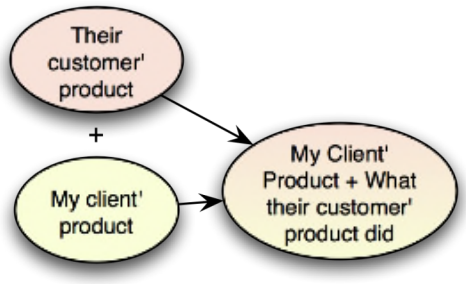


Illustration: My client was asked to expand the Function of their product to encompass the Function of another product. How should they do that Evolutionary?

We started as usual by listing the Stakeholders and the Stakeholder Values, then we identified how good the old product was, using Past levels, and what improvements they wanted, using Goal levels. Then we did the same with the Product Qualities. In isolation their customer was very happy with the product they had, so the critical improvements they wanted was related to having one product instead of two, and from not having to develop and maintain their own product. My client, already a veteran in the use of Evo, was going to start evolving from their own product, adding functions and improving the qualities of those functions until they had a complete product. My client pointed out that they would first have to work hard and long before they would get the Product Qualities up to the same level as the product it was replacing, and then they could eventually improve on them. We suggested they looked at the possibility of getting the code from the clients product, and start evolving from there, towards the new product. The finished new product did not have to contain a single line of code from the old product.

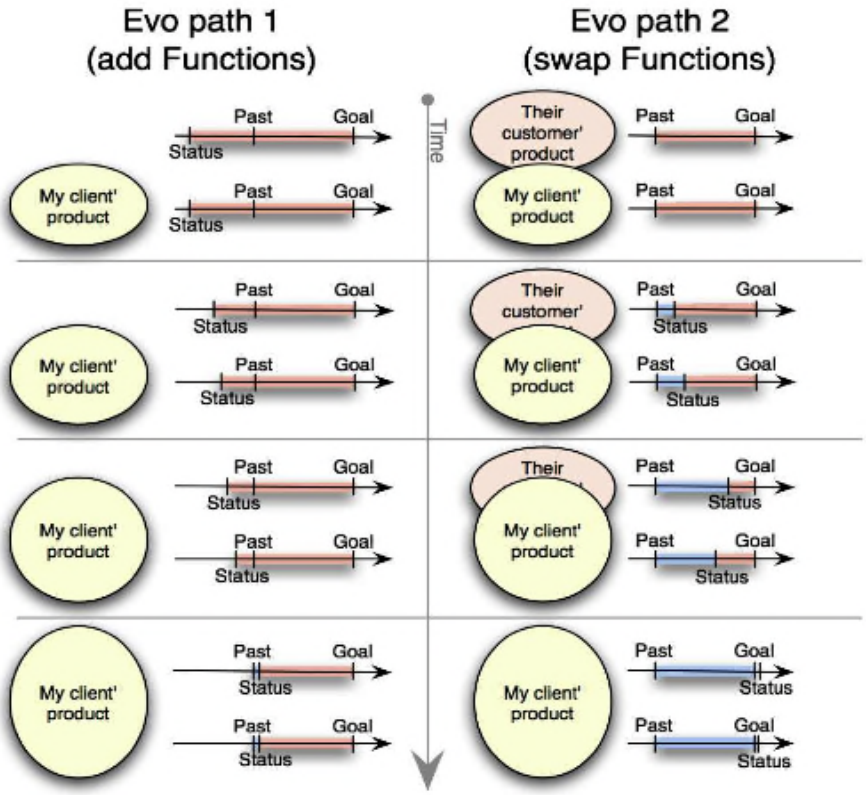


Illustration:

In Evo path 1, my client does not use the system they are replacing, instead they take their own existing product and expands its Functions to include the needed Functions of the product they are replacing. The oval with the text "My client' product" is symbolic of the Functions expanding in time, the two arrows are symbolic of Product Qualities of those new Functions.

In Evo path 2, my client starts using their own product "My client' product" and at the same time the product they will eventually replace "Their customer' product".

If my client chose Evo path 1 they would first have to build into their own product, not only the Functions, but also the Product Qualities already present in "Their customer' product".

If my client chose Evo path 2, they start from the situation their customer is living with today; two systems, all the Functions they need, and current Product Quality (Past levels). They can evolve their existing product to gradually take over more and more Functions of "Their customer' product", and from the very first Evo Cycle evolve the Product Qualities levels from the Past towards the Goal levels.
Using Evo they will also find that much (maybe about 50±20%) of the Functions in the product they are replacing, is not ever used or needed, so they save Development Resources by not developing those Functions into their new product.

Sometimes the old system we want to start developing from is not a product as such, it is just the way things are done today, or more interesting, how well things are done. Adapt their old systems and evolve from there, even if their old "system" is a manual system. The 'how well' Pasts we are looking for are the Product Qualities and the Stakeholder Values that exists now.

Example of using the old, to quickly evolve to the new.

Let's assume we are developing a new mobile phone. We have Requirements and Solutions for a new model. We know approximately what we will end up with.

Start out with a previous model. It gives us Past levels of Stakeholder Values and Product Qualities. We can even start with the physical phone. See if there is some little thing that can be changed and improved within a first cycle.

It could be the instructions in the manual, or the shape of the buttons, or swapping in a new antenna, anything that can be done within the first cycle. Do it, and then repeat. Quickly we evolve from the old model towards the new model.

At times the mobile phone might not be very mobile. Let's say we put in a better antenna or a bigger screen that does not fit in the old housing, it might be hanging out of the old phone just connected by a wire, but we are proving that it works, and we can measure that it works better.

The final product might not have any common parts of software or hardware with the old model, but the old model gave us a working environment that enabled us to start developing and learn immediately.

Yes, but I have no base to start from. How do I get started? I can't get anything out the door within a week! I need at least 6 months do build the foundation…

As a teacher and implementer of Evolutionary delivery, all experience has taught me that these assumptions are as normal as they are wrong. If you have these worries, do not give up, know that there is a base, there is a way to get something out the door early, even if you cant see it yet. There is always some past, we just have to think differently. Normally people think that their product is a new one, so there is no base. That is the wrong baseline, one with focus on the product. The correct baseline should be on the Stakeholder Values & Product Qualities that the Stakeholders are experiencing today. How well the Stakeholder is doing, what we intend to improve upon, with our new product? That is the Past.

Evolutionary Impact

To qualify as an Evolutionary Cycle, each cycle has to positively impact, at least one Stakeholder Value or Product Quality Requirement.

Find Evo Cycles we can complete within one cycle time (normally one week) and that gives maximum impact towards our Goals, and that uses a minimum of development recourses.

Finding Evolutionary Cycles, or, It's just like cooking!

Sometimes we simply implement a Solution that gives us movement towards some Goal levels. Other times it a) takes to long, b) is to costly, c) gives no impact - no movement towards Goal levels, or d) a combination of a, b & c.

Then we have to start chopping and combining Solutions in such a way that we get the impact we want, within the cost & time constraints.

As in cooking a dish, when finding Evolutionary Cycles, have the Stakeholder Values in mind. Take the ingredients, the Solutions, chop them up, mix them together, and voila, we have a starter, or a first evolutionary Cycle.

Think of the Solutions as the raw ingredients and the Evo Cycles as dishes served to the Stakeholders. Combine parts of Solutions to create an Evo Cycle so they impact our Goals levels within the Evolutionary cycle, within Development Resources.

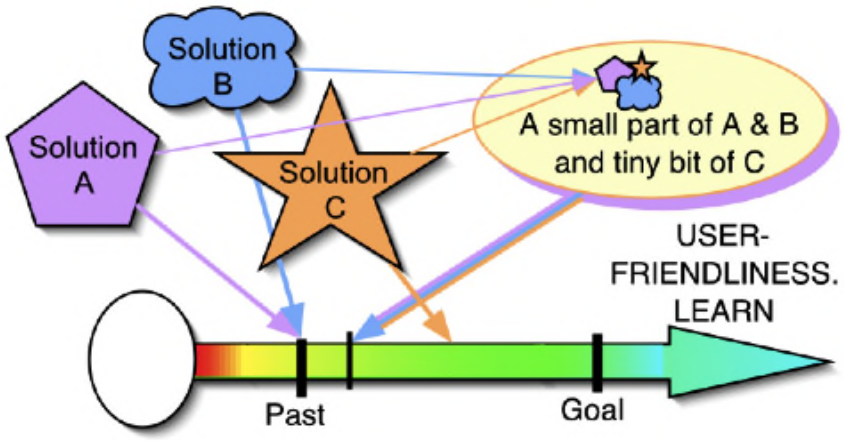


Illustration: Implemented alone Solution A & B gives no improvement towards the User-Friendliness.Learn.Contacts Goal level. Solution C gives the improvement, but would take too much time and would consume too much of the Development Resources to be a valid Evolutionary Cycle. By combining parts of Solution A & B and spicing it a little with Solution C, we can move forward towards the Goal level while still keeping within Development Resource constraints.

Status

Status is a parameter used together with the quantification Scale of a Stakeholder Value or a Product Quality. Status tells us where along the Scale we are right now. It can be viewed as a special category of Past. Where Past is any kind of interesting past reference on a Scale, including old, similar and competitors products, Status is used together with Evolutionary Delivery to say where we are now after the last delivery. It is used to give a current update to where we are, so we can see how much is left to achieve the Goal levels.

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past [Last year]35 min.

Status [Today, cycle 16] **20 min.**

Goal [within 1 year] 5 min.

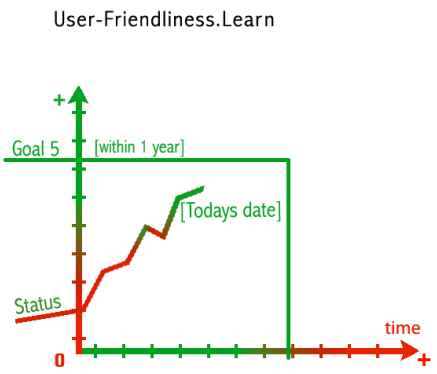


Illustration: The Status levels are hopefully always moving towards the Goal levels. This is especially useful if we are running our projects Evolutionary. We track the improvements over time by keeping track of the past Status levels.

Every time we check, how well the product is dong after an Evo Cycle, we get a updated Status level.

Example: let's assume

Past [ ] 35 min.

Goal [within one year] 5 min.

The Past is our starting point. It takes 35 min. to learn. Then we do some changes to the interface, and apply the Meter and find this;

Status [Evo Cycle 1] 30 min.

Great, it gave us some improvements towards our Goal levels. Then once more we tweak the user interface and apply the Meter to find this;

Status [Evo Cycle 2] 25 min.

Still great, we are getting closer to the Goal level.

Next, we do some changes to the manual, and again apply the Meter. We get this;

Status [Evo Cycle 3] 25 min.

This time we got no improvement, maybe because nobody wanted to use the manual.

The Status level is continuously updated with the latest information about how well we are progressing towards our Goal levels.

The Status level and the Goal level (or Tolerable level) is used in the Impact Estimation Table as a main parameter to describe the movement needed to reach the Goal levels. The job of a developer is always to close the gap between the ever changing, and hopefully improving Status levels, towards the Goal levels. See the chapters on Impact Estimation Tables for more information.

The Evolutionary Project Plan

Evo Cycle Templates

Name Tag:

Type: Evo Cycle

Version:

Stakeholders:

Implementers:

Description:

Name Tag:

Specification Administration

Type: Evo Cycle

Version:

Status:

Owner:

Author:

People

Stakeholders:

Implementers:

Description

Solution Ideas Used:

Implementation Details:

Estimations

Benefits: *(or refer to an Evo Estimation Table)*

Functions Impacted:

Development Resource Budget:

Priority & Risk Management

Priority:

Constraints:

Assumptions:

Dependencies:

Risks:

Issues:

Validation Process:

_____ Defined as:

_____ Defined as:

Actual

Benefits: *(or refer to an Evo Estimation Table)*

Development Resource Used:

Lessons Learned:

Simplified Evolutionary Project Plan

<<<<<<<<<<<

The simple Evo plan is just a list of the Evo Cycles to do

>>>>>>>>>>>>>

Evo Estimation Table

The Evolutionary Project plan can be specified in a table structured similarly to the Impact Estimation Table. Instead of estimating from the Past level, we estimate from the Status level. Instead of estimating the benefits of Solutions, we estimate the benefits of Evolutionary Cycles, but on the same Stakeholder Values or Product Qualities.

In this example, I will choose part of BUTTONS.RUBBER.

Evo Cycle 12

BUTTONS.RUBBER

CURRENT: To the current keypad,

FORMUALA22: exchange the keys with Rubber formula 22 found by our testing lab.

What not to do: Develop a new layout shape and size for the keys.

		Estimate		Actual		Estimate		Actual	
Product Quality Requirements		Cycle 12 Buttons.Rubber				Cycle 13 ?			
Status	Goal	units	% impact	units	% impact	units	% impact	units	% impact
USER-FRIENDLINESS.LEARN	5 by one year	-10	33%						
35									
RELIABILITY	200 by one year	-3	-3%						
100									
Development Resources		units	% impact	units	% impact	units	% impact	units	% impact
PROJECT-BUDGET	100000 by one year	2000	2%						
0									

Illustration: We can use a Evo Estimation Table to estimate the impact of a Evolutionary Cycle on the Stakeholder Values, Product Qualities and Development Resources. In the Evo Estimation Table we substitute the Past level with a Status level. In this table we are using two columns, one for our initial estimate and next to it one for the Actual result as measured at the end of completing the Evolutionary Cycle.

Product Quality Requirements		Estimate		Actual		Estimate		Actual	
		Cycle 12 Buttons.Rubber				Cycle 13 ?			
Status	Goal	units	% impact	units	% impact	units	% impact	units	% impact
USER-FRIENDLINESS.LEARN		-10	33%	-5	17%				
35	5								
by	one year								
RELIABILITY		-3	-3%	-1	-1%				
100	200								
by	one year								
Development Resources		units	% impact	units	% impact	units	% impact	units	% impact
PROJECT-BUDGET		2000	2%	2500	3%				
0	100000								
by	one year								

Illustration: Reality Strikes as we implement the Evo Cycle. We were overly optimistic when estimating Evo Cycle 12's impact on User-Friendliness.Learn and Project-Budget. On the other hand, the negative impact on Reliability does not seem to be as great as previously estimated.

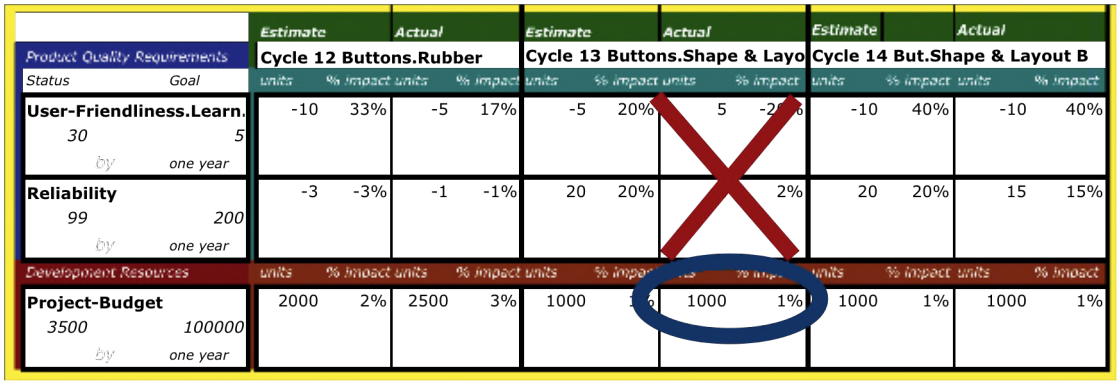
Product Quality Requirements		Estimate		Actual		Estimate		Actual	
		Cycle 12 Buttons.Rubber				Cycle 13 Buttons.Shape & Layo			
Status	Goal	units	% impact	units	% impact	units	% impact	units	% impact
USER-FRIENDLINESS.LEARN		-10	33%	-5	17%	-5	20%		
30	5								
by	one year								
RELIABILITY		-3	-3%	-1	-1%	20	20%		
99	200								
by	one year								
Development Resources		units	% impact	units	% impact	units	% impact	units	% impact
PROJECT-BUDGET		2000	2%	2500	3%	1000	1%		
2500	100000								
by	one year								

Illustration: The Status levels have now changed. We are closer to our User-Friendliness.Learn.Contacts Goal level, a little bit further away from our Reliability Goal level, and we have used some of our Project-Budget.
Using what we learned in Evo Cycle 12, we decide what to do at Evo Cycle 13 and estimate the impact Evo Cycle 13 will have on our Stakeholder Values, Product Qualities and Development Resources.

Product Quality Requirements		Estimate		Actual		Estimate		Actual	
		Cycle 12 Buttons.Rubber				Cycle 13 Buttons.Shape & Layo			
Status	Goal	units	% impact	units	% impact	units	% impact	units	% impact
User-Friendliness.Learn.Contacts		-10	33%	-5	17%	-5	20%	5	-20%
30	5								
by	one year								
Reliability		-3	-3%	-1	-1%	20	20%	2	2%
99	200								
by	one year								
Development Resources		units	% impact	units	% impact	units	% impact	units	% impact
Project-Budget		2000	2%	2500	3%	1000	1%	1000	1%
2500	100000								
by	one year								

Illustration: Implementing Evo Cycle 13 gave us a surprise, it had a very negative impact on User-Friendliness.Learn.Contacts, and only a small impact on Reliability. Our ideas about the shape & the layout of the buttons did not have the effect we thought it would have. Now we either have to keep the old shape and layout, or adjust it so it gives the desired improvements. Evo Cycle 13 gave us some valuable insight about how the Shape and Layout should be, to improve User-Friendliness.Learn.Contacts &

Reliability. So we decided to try out what we have learned with a new variation on the shape & the layout of the buttons.



Product Quality Requirements	Estimate	Actual	Estimate	Actual	Estimate	Actual
	Cycle 12 Buttons.Rubber		Cycle 13 Buttons.Shape & Layo		Cycle 14 But.Shape & Layout B	
Status Goal	units % impact	units % impact	units % impact	units % impact	units % impact	units % impact
User-Friendliness.Learn. *30 5 by one year*	-10 33%	-5 17%	-5 20%	5 -2%	-10 40%	-10 40%
Reliability *99 200 by one year*	-3 -3%	-1 -1%	20 20%	2%	20 20%	15 15%
Development Resources	units % impact	units % impact	units % impact	units % impact	units % impact	units % impact
Project-Budget *3500 100000 by one year*	2000 2%	2500 3%	1000	1000 1%	1000 1%	1000 1%

Illustration: We threw away Evo Cycle 13. The Development Resources used we cannot reclaim. Hopefully we learned something valuable. The Status levels on the Stakeholder Values and Product Qualities stay the same as they did after Evo Cycle 12, but the Development Resources changed. In Evo Cycle 14 we applied the expensive lessons of Evo Cycle 13 and redesigned the Shape and Layout to get the desired effect on User-Friendliness.Learn.Contacts and Reliability. We managed to get a huge improvement on User-Friendliness.Learn.Contacts as we learned much from Evo Cycle 13.

In this simple way we can manage the evolvement towards our critical Stakeholder Values and Product Qualities, while simultaneously controlling the consumption of Development Resources. Notice that the Evo Estimation Tables (EET) contains an impressive amount of information needed by a project manager: Requirements, Budgets, timings, Solutions/Evo Cycles, project progress estimations, actual progress (Status) and actual resource consumption.

The first time people estimate the impact of Evo Cycles, most find themselves having to give very rough guesses with little actual knowledge. After estimating Evo Cycle after Evo Cycle, constantly seeing the difference between the Estimated vs. Actual impacts, they quickly learn the skill of estimation, and they gain confidence in their abilities and the Evo plan. Companies also grow written knowledge of what kind of Solutions are more cost effective towards improving specific types of Stakeholder Values and Product Qualities.

Evo Summary

<<<<< >>>>>

Putting it all together, or, baking the cake

Data Availability; Example from one real paragraph of Requirement specification for a bank.

Original Specification

Data Availability

For Fixed Income reporting, “trade data” assumes the inclusion of calculated sales credits for all trades.

• All required data should be available for query and reporting via Business Objects - USER-A will specify data objects required.

• All data should be available to a single query - users should not have to manually incorporate data from various data locations (unless it is non CNOS data).

• For the purposes of reporting, all trades and static data updates for all Fixed Income sites, should be available by 08:00 GMT on the next working day. After 08:00 GMT, this data should not undergo further updates until 20:00 GMT on the same day.

• For Global End of Day See Section 3.3.

• At any time, users of Business Objects should have access to trades with a Trade Date within the current year and the previous 2 years up to 01/01/(current year-2).

• It should be possible for USER-A to and query on trades with Trade Dates earlier than 01/01/(current year-2) with 1 day’s notice.

Data Availability Analysis

Then I analyzed and tagged the text with these categories; Assumption, Ends, Quality Ends, Means, Comment, Constraint and Function.

Data Availability

Assumption: For Fixed Income reporting, “trade data” assumes the inclusion of calculated sales credits for all trades.

Ends: All required data should be available for query and reporting

All required data: defined as: USER-A will specify data objects required.

Means: via Business Objects

Quality Ends: Time (effort/complexity/skill level) to build reports. See 3.2.1.4 Critical Success Factors

Means: All data should be available to a single query [unless it is non CNOS data].- ***Comment****: users should not have to manually incorporate data from various data locations*

Constraint: [For the purposes of reporting], all trades and static data updates for all Fixed Income sites, should be available by 08:00 GMT on the next working day.

Means: After 08:00 GMT, this data should not undergo further updates until 20:00 GMT on the same day.

Ends: data remain constant

Means: For Global End of Day See Section 3.3.

Ends: Capture all trade and static data from all fixed income sites

Function: (instant online access) At any time, users of Business Objects should have access to trades with a Trade Date [within the current year and the previous 2 years up to 01/01/(current year-2).]

Quality Ends: It should be possible for USER-A to query on trades with Trade Dates earlier than 01/01/(current year-2) with 1 day's notice.

Data Availability - complete rewrite

Then I re-write the specification, the main categories I used were, Product Quality Requirements, Product Functions, Solutions and Assumptions.

Quality Requirements (Ends)

Data Availability

Single.Query:

Gist: All data should be available to a single query [unless it is non CNOS data]. <- USER-A Request

Scale: % of reports that can be produced from a single query.

Past [May 1. this year] 60% <- Guess Sarah ??

Goal [May 1. next year] 90% <- Guess Sarah ??

Comment: users should not have to manually incorporate data from various data locations

Data.Access.Speed

Scale: Time, from USER-A wants access to trades, until they are provided with the information onscreen.

Goal [May 1. next year, MIS, with a Trade Date within the current year and the previous 2 years up to 01/01/(current year-2)] 10 Minutes <- Kai Wild Guess

Goal [May 1. next year ,Trade Dates earlier than 01/01/(current year-2)] < 1 day <- USER-A

Data.Time

Comment: Data.Time may not be a critical success factor

Scale: Average time in Minutes, from USER-A wants to produce a report, until the USER-A has the report in hand.

Past [old way]

Tolerable = Past <- Kai's Guess

Goal [May 1. next year, MIS] 50% less than Past [old way] <- Kai wild guess

Product Function Requirements (Ends)

Data.Query.Reporting:

A set of data objects for query reporting is required by USER-A

Assumption: The set of Data objects will be specified by USER-A

Constraint: Only data available in CNOS data will be provided.

Solutions (Means)

Business Objects: A data query and reporting application (to be confirmed) that will be implemented to facilitate the query of CMIS data and the development of MIS reports.

Supports: Ease of building reports

Global.End: For Global End of Day See Section 3.3.

Data.Freeze: Take a snap shot of the data at 08:00 GMT.

Source: USER-A "After 08:00 GMT, all trades and static data updates for all Fixed Income sites should not undergo further updates until 20:00 GMT on the same day."

Assumptions

TradeData.Inclusion: For Fixed Income reporting, “trade data” assumes the inclusion of calculated sales credits for all trades. Authority: Selina Mitchell

Data.Time: Constraint: [For the purposes of reporting], all trades and static data updates for all Fixed Income sites, should be available by 08:00 GMT on the next working day.

Sunrise Hotel Website Example

We will work with a requirement specification not unlike what some of you sometimes are presented with, and I will demonstrate how I separate the requirements into their logical places, clean them up, make the Stakeholder Values and Product Qualities measurable, and then weed out the Solutions from the Function Requirements. Then I will evaluate if we have the right Solutions to meet the Requirements using an Impact Estimation Table, and finally make an Evolutionary Delivery Plan.

Many of my readers ask about complete worked through examples, but none of my clients are willing to publish theirs in its entirety as it reveals to much of their business. And even if they did publish theirs, it would not be suited for a book format, and it would probably be technically difficult to follow for most readers.

This Requirement specification is made up, and so is the case study. Neither do I have any knowledge about hotel booking systems, so take the technical part of this example as such. This gives me the freedom to show an example in this book that is fairly easy to understand, and to rework it as I like without considering my clients. I hope it is still a useful exercise.

The Requirements as handed to us, or, overcooked spaghetti

Before you look at how I reworked these requirements, you might want to analyze the requirements yourself, and think about what you would do with them.
You suggest spending 10 minutes and
1. Identify the Product Quality Requirements, underline them and mark them with a Q.
2. Identify the Product Functions, underline them and mark them with a F.
2. Identify the technical Solutions, underline them and mark them with an S.

Business requirements for Sunrise Hotel Room Booking System

The Sunrise Hotel website currently has details of the rooms we have available, tariffs and some information about the local area. We need to add the ability for customers to book rooms online through the website using a credit/debit card, and to receive an e-mail confirmation when their booking has been accepted.

We should give prospective customers a '*Book Room'* option, which takes them to a booking screen. In this screen, they should be able to select the type of room they wish to book (single, double, twin, suite), and their arrival and departure dates (we should have some kind of a calendar for them to enter the dates, as it reduces the possibility of entering them incorrectly). We should also allow them to specify if they want breakfast only or breakfast and evening meal, and possibly the ability to tell us of any special requirements they have (baby cot, vegetarian meals etc). We need to ensure that this screen is easy for customers to use, as some hotel booking systems are difficult to navigate.

When the customer clicks '*Calculate Cost'*, we should present them with the full cost of their stay. This will be based on the number of nights, room tariff, meal options, number of adults and children under 12 (charged at _ price), plus the 15% local sales tax. We have a lot of foreign customers, so we need a mechanism that can convert the total cost into their own currency - the converted price doesn't need to be entirely accurate (can we get an approximate rate for the currency conversion, maybe from the previous day?).

Customers will occasionally make two bookings for Sunrise Hotel during their vacation, and travel around the area in between. If we could link these two bookings together, and process them in a single transaction, this would be really useful.

If the customer then clicks '*Proceed'*, they should get another screen where they can enter their credit/debit card details. After these are entered and validated, we should send them an e-mail to confirm their booking. The confirmed booking also to go into our own internal room reservation system, so that the Reservations Team have up-to-date information. The reservations system we use is 'HotelBook', and it does have an 'API' for bookings to be entered automatically.

Before booking the room, we should have some way for the customer to check room availability for their preferred dates. Under no circumstances should we allow a room to be double booked, so customers shouldn't be allowed to spend too long between checking room availability and confirming their booking.

The response time for each step of the process needs to be very fast (at least as fast as our competitors), and we'd like this to be written in Java. We'd like some ability to make changes to this ourselves (there's a possibility that we'll change our promotional material in the near future, and may need to change the Sunrise Hotels logo at the top of the page, and our tariffs change regularly. One of our admin staff has his own website, so could make these changes if they're simple).

The current website is hosted on AOL, and our Marketing Department wants to ensure that the new functionality maintains a consistent 'look-and-feel' with the existing webpages we have.

Before I do anything, I like to make a first stab at who the Stakeholders are, so I can begin to understand the Requirements.

Potential Stakeholders for the Sunrise Hotel room booking website

Sunrise Hotel Management

Sunrise Hotel Front Desk

Sunrise Hotel Restaurant

Sunrise Hotel Maidservice

Sunrise Hotel Marketing

Sunrise Hotel IT

Customers

- National
- International

Banks/ Credit Card Issuers

HotelBook reservation system

Developers

Then I will create meaningful categories.

Stakeholder Function

Stakeholder Values

Product Functions

Product Qualities

Solution Constraints

Solution Ideas

Constraints

Development Resources

Background Notes

And analyze and dump the text into the categories.

Business requirements for Sunrise Hotel Room Booking System

Stakeholder Function

Stakeholder Values

Product Functions

We need to add the ability for customers to book rooms online through the website

and to receive an ~~e-mail~~ confirmation when their booking has been accepted.

they should be able to select the type of room they wish to book (single, double, twin, suite), and their arrival and departure dates

they want breakfast only or breakfast and evening meal, and possibly the ability to tell us of any special requirements they have (baby cot, vegetarian meals etc).

we should present them with the full cost of their stay.

Product Qualities

as it reduces the possibility of entering them incorrectly).

We need to ensure that this screen is easy for customers to use, as some hotel booking systems are difficult to navigate.

so that the Reservations Team have up-to-date information

Under no circumstances should we allow a room to be double booked,

The response time for each step of the process needs to be very fast (at least as fast as our competitors),

We'd like some ability to make changes to this ourselves (there's a possibility that we'll change our promotional material in the near future, and may need to change the Sunrise Hotels logo at the top of the page, and our tariffs change regularly. ... if they're simple).

our Marketing Department wants to ensure that the new functionality maintains a consistent 'look-and-feel' with the existing webpages we have.

Solution Constraints

using a credit/debit card

Solution Ideas

using a credit/debit card

e-mail confirmation

We should give prospective customers a '*Book Room*' option, which takes them to a booking screen. In this screen,

(we should have some kind of a calendar for them to enter the dates,

We should also allow them to specify if

When the customer clicks '*Calculate Cost*',

(Full cost of stay) This will be based on the number of nights, room tariff, meal options, number of adults and children under 12 (charged at _ price), plus the 15% local sales tax.

We have a lot of foreign customers, so we need a mechanism that can convert the total cost into their own currency - the converted price doesn't need to be entirely accurate (can we get an approximate rate for the currency conversion, maybe from the previous day?).

Customers will occasionally make two bookings for Sunrise Hotel during their vacation, and travel around the area in between. If we could link these two bookings together, and process them in a single transaction, this would be really useful.

If the customer then clicks '*Proceed'*, they should get another screen where they can enter their credit/debit card details. After these are entered and validated, we should send them an e-mail to confirm their booking. The confirmed booking also to go into our own internal room reservation system,

Before booking the room, we should have some way for the customer to check room availability for their preferred dates.

(prevent double booking) so customers shouldn't be allowed to spend too long between checking room availability and confirming their booking.

and we'd like this to be written in Java.

One of our admin staff has his own website, so could make these changes

Constraints

Development Resources

Background Notes

The Sunrise Hotel website currently has details of the rooms we have available, tariffs and some information about the local area.

The reservations system we use is 'HotelBook', and it does have an 'API' for bookings to be entered automatically.

The current website is hosted on AOL,

Staring with Product Functions, I give every idea a nametag for identification and further work. I also delete some text, which in my view, did not add anything. It looks something like this.

Product Functions

Room-Type: type of room they wish to book.

Single:

Double:

Twin:

Suite:

I then added a little information on some of the Product Functions, and the whole Product Function specification looks like this:

Product Functions

Book- Rooms: book rooms in Sunshine Hotel

Online: through the website.

Pay: pay or hold the room through the website

Dates: specify

Arrival: date of arrival

Departure: date of departure

Room-Type: type of room they wish to book.

Single: one single bed.

Double: one double bed.

Twin: two single beds.

Suite: one double bed, one single bed, two rooms + bathroom.

Food: specify

Breakfast: only breakfast.

Breakfast- Dinner: breakfast and dinner.

Vegetarian: type of vegetarian meal requested.

Special-Requirements: Specify

Baby-Cot: a bed suitable for children up to 3 years.

Other: any other special requirements.

Sum: present them with the full cost of their stay.

Confirmation: confirm when their booking has been accepted.

The Product Qualities receives the same treatment, starting with giving each idea a name.

Product Qualities

Correct: reduces the possibility of entering them (dates) incorrectly.

Easy: screen is easy for customers to use, as some hotel booking systems are difficult to navigate.

Current: Reservations Team have up-to-date information

Double-Booking: Under no circumstances should we allow a room to be double booked.

Snappiness: The response time for each step of the process needs to be very fast (at least as fast as our competitors),

Updates: ability to make changes to this ourselves

(there's a possibility that we'll change our promotional material in the near future, and may need to change the Sunrise Hotels logo at the top of the page, and our tariffs change regularly. ... if they're simple).

Image.Consistency: our Marketing Department wants to ensure that the new functionality maintains a consistent 'look-and-feel' with the existing WebPages we have.

Then the magic, specifying the Product Qualities in a clear, measurable & testable way.

Product Qualities

Correct

Stakeholders: Sunrise Hotel Front Desk, Sunrise Hotel Management, Customers.

Scale: % of all dates, that are entered into the website correctly, by customers and staff, per month.

Meter: a log of complaints and requests to change dates.

Past [Front Desk] 99.5%

Goal [Front Desk & Website] 99.5%

Easy.Intuitive

Stakeholders: Sunrise Hotel Management, Customers

Scale: average rating, per month, by our customers, on how intuitive the website is, on a rating from 1 to 5, where 5 = perfectly intuitive - 4 - better than normal - 3 = average - 2 = poor - 1 = hopeless.

Meter: customer satisfaction survey.

Past [ ] 2

Goal [ ] 4

Easy.Give-up

Stakeholders: Sunrise Hotel Management, Customers

Scale: % of customers, per month, that intended to book a room on our website, that for one reason or another gave up, and either never booked a room with us, or booked thru other means.

Meter: customer survey, web-log.

Past [ ] 100 %

Goal [ ] 1 %

Easy.Fast

Stakeholders: Customers, Sunrise Hotel Management.

Scale: average time for a new customer to complete his/her booking without help from another person.

Meter: customer survey, web-log.

Past [Website then call Front Desk] 10 min.

Goal [Front Desk or Website] 5 min.

Current

Stakeholders: Sunrise Hotel Management, Sunrise Hotel Front Desk, Customers, Easy Book reservation system.

Scale: number of mistakes done by the Reservation Team or customers using our web based booking system, per year, caused by misinformation regarding current bookings and availability.

Meter: log

Past [ ] 5

Goal [ ] 1

Double-Booking

Stakeholders: Sunrise Hotel Management, Sunrise Hotel Front Desk, Customers.

Scale: number of double bookings per year.

Meter: log

Past [ ] 1

Goal [ ] 0

Snappiness

Stakeholders: Sunrise Hotel Management, Sunrise Hotel Front Desk, Customers

Scale: longest time in seconds, for any individual step in the booking process, from a user press a process button or link, until the user is presented with the next step or information.

Meter: clock it with a stopwatch.

Past [dial up connection, old website] 12 sec.

Past [Broadband connection, old website] 10 sec.

Goal [dial up connection] 7 sec.

Goal [Broadband connection] 3 sec.

Updates

Stakeholders: Sunrise Hotel Management, Sunrise Hotel IT, Sunrise Hotel Marketing.

Scale: average time, for a Sunrise Hotel employee that has received a one day training course,
to make a defined Change too the website,
from the employee has the information prepared on the computer but not on the website,
until the information is updated and integrated consistently and correctly on the website.

Meter: instruct the person to do the change, and time it with a stopwatch.

Past [Change=Tarrifs] 120 min.

Past [Change=Logo] 10 hour.

Past [Change=Promotion Material] 10 hour.

Goal [Change=Tarrifs] 10 min.

Goal [Change=Logo] 1 hour.

Goal [Change=Promotion Material] 1 hour.

Image.Consistency

Stakeholders: Sunrise Hotel Marketing, Sunrise Hotel Management.

Scale: % chance that a customer that has either been to our other web-pages, or has stayed in our hotel before, recognizes the web booking pages as Sunrise Hotels.

Meter: customer survey.

Past [ ] 45%

Goal [ ] 80%

Analyzing all the Solutions, I only found one that I considered a Requirement, a Solution Constraint.

Solution Constraint

Credit-Card: hold booking using a credit/debit card.

And the rest I put in my basket of Solutions that I am free to use or not. I gave each of the Solutions a name. And they looked like this.

Solution Ideas

Confirm-Email: confirmation e-mail

Then I added some information to make the Solutions a little more complete. I did not take the time to specify each Solution in much detail, as I do not know if I will end up using the Solutions as presented. When and if I choose to go forward with some of these ideas, I will work out the detail. Here are all the Solutions so far.

Solution Ideas

Confirm-Email: confirmation through sending an e-mail to the customer.

Book-Room.: give prospective customers a '*Book Room*' option,

Booking-Screen: which takes them to a booking screen.

Date-Fields.: for all fields that require dates,

Calendar: have some kind of a calendar that users can access.

Button-Calculate.: on the screen, have a button '*Calculate Cost*' where the customer can click.

Total-Cost: and they will be presented with the total cost of their booking.

Final-Cost-Calculation: based on; the number of nights, room tariff, meal options, number of adults and children under 12 (charged at _ price), plus the 15% local sales tax.

Conversion.: We have a lot of foreign customers, so we need a mechanism that can convert the total cost into their own currency

Conversion.Accuracy: the converted price doesn't need to be entirely accurate

Conversion.Currency-Rate: can we get an approximate rate for the currency conversion, maybe from the previous day?

Combine: Customers will occasionally make two bookings for Sunrise Hotel during their vacation, and travel around the area in between." If we could link these two bookings together, and process them in a single transaction, this would be really useful.

Proceed-Button: If the customer then clicks '*Proceed*', they should get another screen where they can enter their credit/debit card details.

Email-Confirm: After these are entered and validated, we should send them an e-mail to confirm their booking.

Internal: The confirmed booking also to go into our own internal room reservation system,

Room-Availability: Before booking the room, we should have some way for the customer to check room availability for their preferred dates.

Time-Limit: (prevent double booking) customers shouldn't be allowed to spend too long between checking room availability and confirming their booking.

Language-Java: written in Java.

Staff-Change: One of our admin staff has his own website, so could make these changes.

Giving the background notes identities.

Background Notes

Exists: The Sunrise Hotel website currently has details of the rooms we have available, tariffs and some information about the local area.

HotelBook: The reservations system we use is 'HotelBook', and it does have an 'API' for bookings to be entered automatically.

Host-Current: The current website is hosted on AOL,

Having analyzed, regrouped and reworked all the text in the original specification. I have a better understanding of what the website has to do, the Product Functions, how well it has to do it, Product Qualities, how I have to do it, Solution Constraint, I have a list of potential ways of doing it, Solution Ideas, and considerations, Background Notes.

Reading the Product Qualities, I think there are some important ones missing, I add them.

Product Qualities (additional)

Availability

Stakeholders: Sunrise Hotel Front Desk, Sunrise Hotel Management, Customers. Sunrise Hotel IT

Scale: % of all time, per year, that the system is up and running and able to receive and process bookings.

Meter: system log

Past [ ] 97 %

Goal [2005] 99%

Upgrade

Stakeholders: Sunrise Hotel Management, Sunrise Hotel IT

Scale: average cost, per function, to upgrade the system after its first release, when we at a later stage decide to add or change Functions to the website.

Meter

Past [ ]

Goal [ ]

Capacity

Stakeholders: Sunrise Hotel Management, Sunrise Hotel Front Desk, Customers.

Scale: number of simultaneous users on the website, before the website noticeably slows down.

Meter: using the web-log to see how many people are currently using the system, subjectively judge if it slows down.

Past [ ] 20

Goal [2004] 35

And I add some Product Functions that I find missing.

Product Functions

Smoking: selection of smoking preferences.

No: a room where smoking is not allowed, and has not been allowed for some time.

Yes: a room where smoking is allowed.

No-Preference: no consideration has to be taken regarding Smoking.

Wheelchair: specifies if room must be accessible with a wheelchair.

Special-Offers: can find and book special offers and prices.

Special-Requests: any service or product the customer requests in connection with the room.

What I don't have any information about are the Stakeholder Values, Stakeholder Functions and Development Resources. Usually, and in this case, I find no need to write down the Stakeholder Function (book hotel, etc), but I find it critical to understand the Stakeholder Values.

Stakeholder Values

Booking-Cost

Stakeholders: Sunrise Hotel Management, Customers.

Scale: average cost, in $, to the hotel, per booking.

Past [Call] $ 3.00

Goal [Web] $ 0.25

Booking-Ease

Stakeholders: Customers. International.

Scale: average cost, in $, for a international customer, per booking.

Past [Phone] $10.-

Goal [Web] $0.25

Added-Business

Stakeholders: Sunrise Hotel Management, Customers.

Scale: average % booked rooms per year.

Past [2004] 75%

Goal [2005] 85%

Customer-Satisfaction

Stakeholders: Sunrise Hotel Management, Customers.

Scale: average customer satisfaction rating regarding booking. 1 = dissatisfied and 5 = extremely satisfied.

Past [Phone] 2.5

Goal [Phone, Web] 3

And Development Resources.

Development Resources

Money

Scale: US$ to develop the website and the underlying database, including all connections to the external booking systems.

Past [Sunrise Hotel Booking System Website, so far] $1000.-

Budget [Sunrise Hotel Booking System Website] $25.000.-

At this stage in the process, I would take these specifications back to the client, and to the Stakeholders involved and ask if I have captured the requirements of the project correctly. I would expect that I had gotten several things wrong, and that by separating the individual ideas and putting them into categories and by specifying them in a clear measurable way, that it would be much easier for the client and the Stakeholders to spot the mistakes and to give me useful feedback. I would ask for what I have missed completely and what should not be there at all. I would ask them to read the Scales and Goal levels carefully and to suggest any improvements and adjustments, and I would ask them if I had understood the Product Functions correctly, and if there where some vital ones missing. And I would ask the individual Stakeholders if I had captured their Stakeholder Values correctly.

Then I would take these Requirements to the developers, and ask them to come up with a rough idea of Solutions that they could implement to meet the Requirements, and the cost of those. With this information, together with the developers, I would create one Impact Estimation Table where we estimate how well the product as defined would meet the defined Stakeholder Values, and a second table where we estimate whether we have enough and the right Solutions to meet the defined Product Quality Goal levels.

	Website with defined Product Qualities
Booking-Cost	100%
Booking-Ease	120%
Added-Business	90%
Customer-Satisfaction	100%
Money	85%

Table: Horizontally, I specify the Stakeholder Values, and the Development Resources, and vertically the whole product as described with the intended Product Function & Product Qualities. The estimates tell me that I might not be able to meet the Stakeholder Value of Added-Business. With this information, we can think of what we need to change to satisfy this Requirement. It also tells me that I might meet the other Stakeholder Values.

Not happy with not meeting the Stakeholder Value of Added-Business, I would research what we can do to bring in added business. Speaking with the Hotel Management I discover that they want to get Added-Business through being reachable through Travel Agents worldwide, and by websites that specializes in finding hotels. I realize that we have not specified any such interface, or solution to enable that. I find that there exists a standard hotel booking system and add that Function to the product specification.

Product Function

Global-Distribution-System:

(GDS) a link to the Global Distribution System, that provides the ability for travel agents and external websites to book rooms at Sunrise Hotel directly.

Reading the existing suggested Solutions, I find some ideas I can use, but mostly I see an incomplete list of not well thought out ideas. So I would begin the work of identifying key Solutions needed to deliver the product.

Solution Ideas

Central-DB:

.One: One and only one database where all data related to rooms, bookings, availability, price etc. is stored.

.Direct: all bookings are done directly with this database.

Comment: so all room availability (house booking, our website booking, Global Distribution System bookings), always are up to date and present current information.

Out-Input: input & output of these data shall happen through.

.Sunrise-Hotel-Website: our own website though the internet.

.Global-Distribution-System-Link: travel agents and external websites connected through the Global Distribution System

.Front-Desk: Sunrise Hotel front desk and telephone center over the intranet.

.Hotel-Management: Management terminal on the intranet.

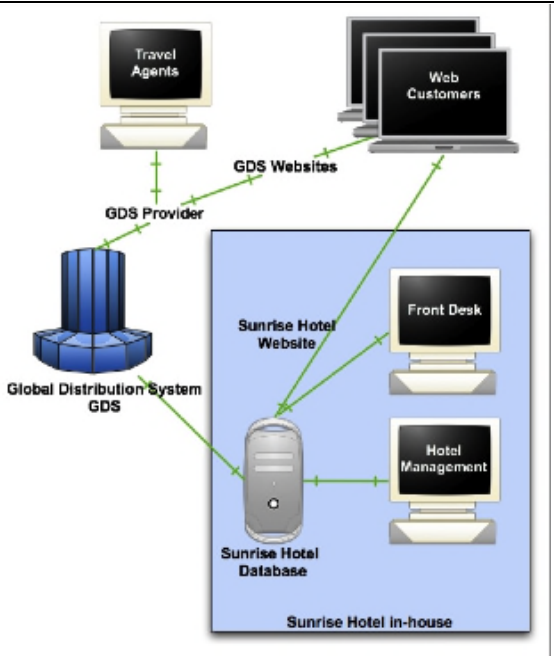


Illustration: Solution Ideas.Central-DB & .Output-Input.

DB-Type:

.MSSQL: use MSSQL database.

.MYSQL: use MYSQL database.

.Filemaker-Pro7: use Filemeker-Pro7 database.

Website.Order-Flow:

.Evo-Cycle1: Click 'Book Room'.

.Evo-Cycle2: Select room type and preferences.

.Evo-Cycle3: Personal & Credit Card information.

.Evo-Cycle4: Conformation.

On Screen: Present a webpage screen.

Printable: that is printable, with minimum graphics.

PDF: with a link to a downloadable PDF of the confirmation.

Email: send an email confirmation.

My intention with this example is not to design a complete hotel booking system, just to show an example. A real system would need more Solutions and Evo-Cycles. Let us take what we have now, and make an Impact Estimation Table.

	Central-DB	*Out-Input*	*DB-Type*	*Website.Order-Flow*	*Conversion*
Correct	0%	0%	0%	0%	0%
Easy.Intuitive	0%	0%	0%	25%	10%
Easy.Give-up	0%	0%	0%	25%	10%
Easy.Fast	0%	10%	10%	25%	10%
Current	80%	0%	0%	0%	0%
Double-Booking	80%	0%	0%	0%	0%
Snappiness	10%	0%	50%	0%	0%
Updates	10%	0%	25%	0%	0%
Image.Consistency	0%	0%	0%	0%	0%
Availability	0%	0%	80%	0%	0%
Upgradability	25%	0%	25%	0%	0%
Capacity	0%	0%	35%	0%	0%
Money	5%	5%	5%	1%	1%

Impact Estimation Table: On the left side I list the names referring to the Product Qualities, and across the top I list the names referring to the Solutions.

The Impact Estimation Table is telling me where I have gaping holes, like the Product Qualities 'Correct' and 'Image.Consistency' have no Solutions that have any effect on them whatsoever. In addition, most of the other Product Qualities are weak, and we need more Solutions to satisfy them as well. Luckily, we have not used up all the 'Money' Development Resources.

I would keep working on existing Solutions and develop new Solutions until I have a good set of them that would satisfy all my Product Qualities. I will not do that now, but rather start working out an evolutionary delivery plan.

Now I will divide the project into areas, which I can do more or less independently.

High level Evo Plan

Front-Desk: Sunrise Hotel front desk booking

Website: Sunrise Hotel booking website

Global: Global Distribution System (GDS) booking

DB: Underlying db

Management: Management and updating

Next, I need to decide where to start, to help me in that decision, I will use an Evo Table.

	Front-Desk	*Website*	*Global*	*DB*	*Management*
Booking-Cost	0%	40	40	10	10
Booking-Ease	0%	40	40	10	10
Added-Busines	0%	20	80	0	20
Customer-Satis	10%	40	10	15	5
Money	20%	20	5	20	20

Evo Table: on the left side, I list the key Stakeholder Values and the Development Resources, and on the top the potential Evo Cycles.

As we already have a front desk booking system in use, 'Front-Desk' scores the lowest, 'Global' scores the highest because it is such an critical Evo Cycle to bring in Added-Business. Also 'Global' is the Evo-Cycle that is estimated to consume the least of the Development Resources. I will start with 'Global', and do 'Front-Desk' later.

High level Evo Plan

Evo-Cycle 1. Global: Global Distribution System (GDS) booking

Evo-Cycle 2. Website: Sunrise Hotel booking website

Evo-Cycle 3. DB: Underlying db

Evo-Cycle 4. Management: Management and updating

Evo-Cycle 5. Front-Desk: Sunrise Hotel front desk booking

I will start using whatever hotel booking system is there already, and build the Global Distribution System (GDS) that enables people and external websites to book a hotel room at our hotel. I will focus on building it in such a way that I can later change the underlying database and keep using the GDS part on the new database.

Then I would divide up the deliveries of Evo-Cycle 1. Global.

Detailed Evo Plan

1. aaaa

2. bbb

I will end the details here, but I would repeat the process I used above to find smaller Evo-Cycles, and decide what to do first.

Then I would do the first Evo-Cycle, and see/measure what happens & learn, fine-tune the plan and repeat.

Prioritization - how to make efficient decisions about where to use limited Development Resources, in a live, changing, dynamic system!
or,
Prioritization - I want it all yesterday and for free, so I will first do that which costs less and delivers the most benefit now.

A large part of project management is about prioritizing, and almost everything presented in this book is optimized to help you prioritize.

The 'New Oxford American Dictionary' definition of prioritizing:

> prioritize verb
>
> designate or treat (something) as more important than other things : *prioritize your credit card debt.*
>
> • determine the order for dealing with (a series of items or tasks) according to their relative importance : *age affects the way people prioritize their goals*

This definition assumes that we do not have unlimited Development Resources. I will like to make that clear. If we had unlimited Development Resources we would not have to prioritize, we could do everything first.

There are two areas of focus when prioritizing in projects;
one is finding out what we like to achieve with limited Development Resources, setting the Stakeholder Values & Product Quality Scales and Goal levels;
the second area is to decide where to deploy the limited Development Resources so as to achieve those Goal levels, selecting Solutions and Evo Cycles to implement.

Almost all methods of prioritizing fail because of two main reasons;

1. the Stakeholder Values and Product Qualities are not well understood or stated.

2. the prioritizing process is not done continuously, but often only once in the beginning of a project.

In any real system, it is observable that prioritizing happens continuously. Yet most management prioritizing methods don't.

Observe your own body as an example of a real live system. You are probably reading in this book because you have a Stakeholder Value of learning. Right now your priority, where you are deploying your resources, are on a Solution we can call reading in the hope that it will effect your Stakeholder Value of learning, but how long will learning stay as your priority? 1 hour? 5 hours? After some time your priorities will change, you will want to rest, socialize, go to the toilet or eat. The needs and wants, the Stakeholder Values and Product Qualities of a human being, as well as for your projects change in time, so our priority methods for project management must be able to observe and adjust as well.

Weighting methods are often used in project management, and are typically not changing according to the actual changes and therefore can be disastrous for the projects that rely on them. If you applied weighing methods to a human, the model could say that learning is the most important now, so we must read. The human would read, but get tired, hungry and in need of a toilet, but the model would not change its weighting. The human would learn enough for purpose, but the model would not change its weighting.

By learning how to specify the Stakeholder Values and Product Qualities quantifiable with a Past, a Status and a Goal level, separating the Solutions, the means, from the ends, by evaluating Solutions and Evo Cycles on an Impact Estimation Table, and by delivering the Stakeholder Values and Product Qualities evolutionary

we have build the foundation for prioritizing effectively. We shall now introduce a few more elements that can be used together with what we already learned about Stakeholder Values and Product Qualities, to the Impact Estimation Table and to the Evo delivery process. We will see how it operates together in a way that will enhance our understanding and practical application of prioritization.

Tolerable

Tolerable is a parameter used together with the quantification Scale of a Stakeholder Value or a Product Quality. The Tolerable parameter identifies the beginning of the Tolerable Range and the end of the Intolerable Range along the quantification Scale. The Tolerable Range ends at the Goal level.

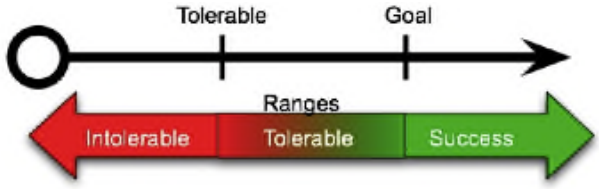


Illustration: The black circle represents the system, with its functions, and the black arrow represents the quantification Scale of one of many Product Qualities. The two points on the Scale, Tolerable and Goal level, marks the dividing line between where the Product Quality is Intolerable, to where it is Tolerable, and from where it is Tolerable to where it is Successful.

The Intolerable Range is a range along the quantification Scale, where the Stakeholders will NOT tolerate the system. No one will buy the product if one of the Product Qualities falls within the Intolerable range.

The Tolerable Range is a Range along the quantification Scale, where the Stakeholders will tolerate the system, but they are not happy with this particular Product Quality. The system is not Successful when within this Range. If a Goal level is contracted for but not reached, partial payment can be agreed upon within the Tolerable Range.

The Success Range along the quantification Scale is a Range where the Stakeholders are happy with this particular Product Quality. If a Goal level is contracted for, and reached, full payment would be expected.

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Status [Today, cycle 16] 20 min.

Tolerable [within 14 months] 18 min.

Goal [within 1 year] 5 min.

Tolerable together with the Scale reads:

It will absolutely **not be tolerated,** if within 14 months, the average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone, is **more than 18 min.**

or

The 18 minutes point on the quantification Scale divides the Intolerable range and the Tolerable range.

The Tolerable level can be set by predicting the level where funding to our project will be cut, or where no one will be interested in buying our product or service. If we cannot get better results than the Tolerable level, the project is simply not worth the money nor the effort.

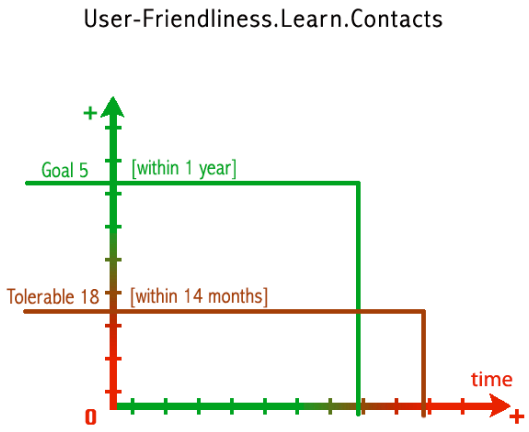


Illustration: Notice that in this example, the date for the Tolerable level is 2 months after the Goal level. If by that date, the time to learn is more than 18 minutes, no matter how well the product is performing in other areas, the project will probably be a failure.

Failing to meet one single Tolerable level on one Stakeholder Value or Product Quality aspect can bring down the whole project. We may have been very successful with many aspects of our mobile telephone, we can have long standby and talk time, great sound qualities etc, but what level of **User-Friendliness.Learn.Contacts** will be so bad, that nobody will choose our product, no matter how great the other Product Qualities are. That level would be our Tolerable level, a level we must get better than.

Consider a live system, like your own body. You will find there are many Tolerable levels that single handedly can bring down your body. Let's say we are loved, we have plenty of food, etc., but we are freezing. What body temperature will shut your system down? When will you hit the bucket? That is the Tolerable level. A level that single-handedly will shut the whole system down. Think of the Tolerable level as a starvation point. If we do not get at least this amount of food, we will starve to death.

If, while planning a project, we don't find any Solutions that will give us better results than the Tolerable level, we should probably not go ahead with the project.

When all critical Product Qualities are above the Tolerable level, it is possible for people to use the system. A project's first order of priority should normally be to get all the critical Stakeholder Values and Product Qualities above the Tolerable levels, then move them to the Goal levels.

Status, Tolerable and Prioritizing, or, The human body model.

A visual example.

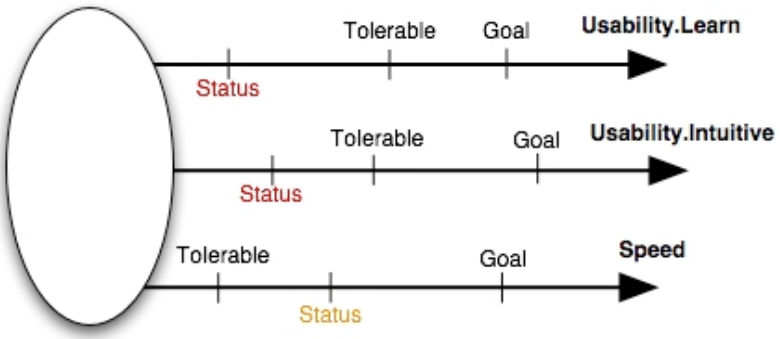


Illustration: Usability.Learn, Usability.Intuitive & Speed represents three critical Product Qualities. Observing the Status, Tolerable & Goal levels, we can see that the current Status of the Product Quality –

Speed, is above the Tolerable Range, while Usability.Learn and Usability.Intuitive is in the intolerable Range.

The first priority is to get all Product Qualities out of the Intolerable Range and into the Tolerable Range. We can prioritize development on improving Usability.Learn and/or Usability.Intuitive.

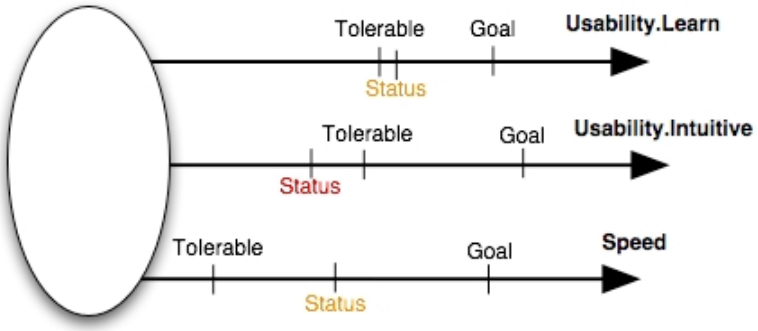


Illustration: For the next Evo Cycle, Usability.Learn was prioritized, now only Usability.Intuitive is in the Intolerable Range.

In the next Evo Cycle we can prioritize Usability.Intuitive.

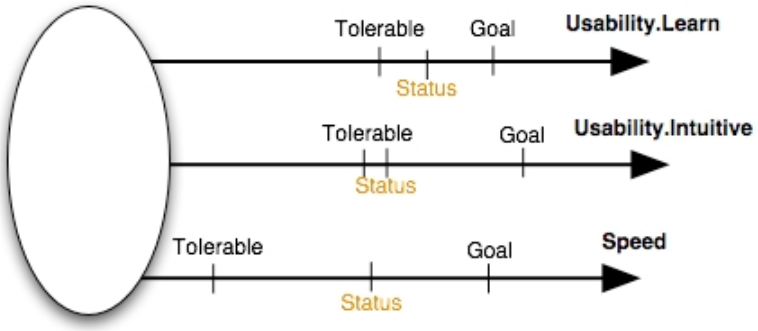


Illustration: Now the Status for all three critical Product Qualities are in the Tolerable Range

The next priority can be to bring all Product Qualities past the Goal levels and into the Success Range. As far as this model is concerned, we can work on improving any of the three Product Qualities.

Evo Estimation Table example

In practice, our clients use Evo Estimation Tables to dynamically prioritize what area needs to be improved next. Just as in the model above, but with real Stakeholder Value or Product Quality Requirements and real Development Resources.

Prooduct Quality Requirements				*Estimated Impact* Cycle 14 But.Shape & Layout		*Actual Impact*		*Estimated Impact* Cycle 15		*Actual Impact*	
Past **Status**		*Tolerable*	**Goal**	*Units*	*%*	*Units*	*%*	*Units*	*%*	*Units*	*%*
User-Friendliness.Learn.Contacts				-10	40%	-10	40%				
55	**20**	25	**5** by a year								
Reliability				20	20%	15	15%				
70	**114**	150	**200** by a year								
Style				0	0%	0	0%				
5	**9,5**	7	**9** by a year								
Development Resources											
Project-Budget				1000	1%	1000	1%				
0	**4500**	140000	**100000**								

Table: In this Evo Impact Estimation Table, we have included the Status and the Tolerable level, and red, yellow and green lights. A red light means that the Status level of the Product Quality is in the Intolerable Range, a yellow light that it has passed the Tolerable level and is in the Tolerable Range, and a green light that the Status level has passed the Goal level and is in the Success Range.

In the example above, when choosing what to do in the next evolutionary cycle, which of the Product Qualities would you focus on improving, the **User-Friendliness.Learn.Contacts**, the Reliability or the Style?

The Product Quality – Style, is already in the Success Range, it should have no further priority, no claim on Development Resources.

The Status of the **User-Friendliness.Learn.Contacts** is in the Tolerable Range, we do need to improve this quality level and move it past the Goal level into the Success Range.

Reliability is still below the Tolerable level in the Intolerable Range, it is critical to get it into the Tolerable range and towards the Success Range. The Evo Estimation Table is giving strong advice that Reliability is where we need to prioritize our limited Development Resources.

I will pick an Evolutionary Cycle that can impact Reliability.

Splash.Speaker

Type: Evo Cycle Solution

Description: insert a thin film between the mobile phone speaker and the inside of the housing.

Comment: A common reason for Reliability problems are water getting into the mobile phone through the speaker hole. Splash.Speaker is intended to make the hole for the speaker splash proof.

Prooduct Quality Requirements					*Estimated Impact* Cycle 14 But.Shape & Layout		*Actual Impact*		*Estimated Impact* Cycle 15 **Splash.Speaker**		*Actual Impact*	
Past	**Status**	*Tolerable*	**Goal**		*Units*	*%*	*Units*	*%*	*Units*	*%*	*Units*	*%*
User-Friendliness.Learn.Contacts												
55	**20**	25	**5** by a year		-10	40%	-10	40%	0	0%		
Reliability												
70	**114**	150	**200** by a year		20	20%	15	15%	20	23%		
Style												
5	**9,5**	7	**9** by a year		0	0%	0	0%	0	0%		
Development Resources												
Project-Budget												
0	**4500**	140000	**100000**		1000	1%	1000	1%	1000	1%		

Table: I selected to do Splash next, as Reliability is in the Intolerable Range, and I think Splash can have a ~20% impact on the Reliability Requirement.

the way to come out of ignorance is a definite understanding, a definite knowledge in the mind, that my body is undergoing change all the time, the world is undergoing change all the time, the entire universe is in a state of fluidity and it is all full of change and it is going on on its own, according to its nature.

Sri Sri Ravi Shankar: Eliminating the Cause of Pain

To prevent failures in project planning, we must understand that everything in our project is undergoing change all the time, our stakeholders are undergoing change all the time and everything else is changing all the time. We must plan, learn, study and act accordingly while constantly prioritizing.

Solution Comparison Table, or, Picking the Solution that delivers the most value for the least resources.

The principle 'Value' from the Evo Planning Policy example shown in the Evo chapter will be of great help in deciding where to allocate limited Development Resources:

Value: Evo Cycles which deliver the most improvements to Stakeholder Values or Product Qualities, for the Development Resources they consume, will normally be delivered first, or, do the juicy bits first!

A version of the Impact Estimation Table is used to help select one Solution, or one Evo Cycle, from a selection of many. The Solutions, or Evo Cycles we compare in the table can sometimes be alternatives, where if we choose one, we will not at a later point choose the others in addition, and sometimes we just like to choose what to do first. When used in such a way, we call it a Solution Comparison Table.

In the Solution Comparison Table, we first summarize the estimated impact a Solution will have on reaching our Product Qualities (or Stakeholder Values). Since the individual Product Qualities have different Scales, we have to normalize the number, something we have already done by calculating the impact of a Solution in % of movement between Status and Goal level. This sum gives us a view to how much the Solution impacts the total set of our Product Quality Requirements.

Then, we summarize the Development Resources in the same way, to get a view of how much each Solution eats up of our Development Resources.

Finally, we divide the Sum of Benefits with the Sum of Development Resources. This gives us a number that is relative to each other. A higher number means that we get closer to our Goal levels for the least of our Development Resources.

	Estimated Impact		*Estimated Impact*		*Estimated Impact*		*Estimated Impact*	
Product Quality Requirements	**Splash.Speaker**		**Splash.Keypad**		**Battery.Lock**		**Screen.Scratch**	
Past **Status** *Tolerable* **Goal**	*Units*	*%*	*Units*	*%*	*Units*	*%*	*Units*	*%*
User-Friendliness.Learn 55 **20** 25 **5** by a year	0	0%	0	0%	-1	7%	0	0%
Reliability 70 **114** 150 **200** by a year	20	23%	25	29%	0	0%	10	12%
Style 5 **9,5** 7 **9** by a year	0	0%	0	0%	0,5	0%	-0,5	0%
Sum of Benefits		**23%**		**29%**		7%		12%
Development Resources								
Project-Budget 0 **4500** 140000 **100000**	**1000**	1%	**1700**	2%	3000	3%	2000	2%
Sum of Development Resources		**1%**		**2%**		3%		2%
Benefits / Development Resources		**22,21**		**16,33**		2,12		5,55233

Table: Solution Comparison Table; The Sum of Benefits show that Splash.Keypad has a higher impact on the Product Qualities than Splash.Speaker has (29% vs. 23%). But the cost of developing Splash.Keypad is higher than Splash.Speaker (1700 vs 1000). When dividing the Sum of Benefits with the Sum of Development Resources, Splash.Speaker gives the most value (22,21 vs. 16,33) towards meeting the defined Product Qualities.

Notice that I have scored Battery.Lock's impact of 0,5 on Style as 0% because the Goal level is already reached.

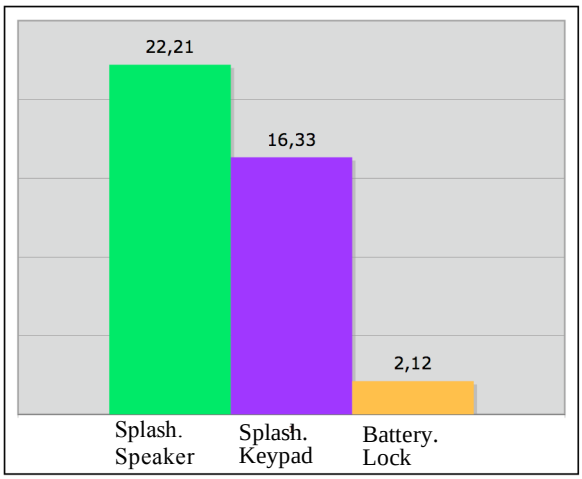


Illustration: Benefits / Development Resources; graphically showing how much benefit each Solution give on the Product Qualities relative to how much they consume of Development Resources.

First focus on getting all quality levels to the Tolerable Range, and then to the Success Range, or, No need for a fancy dining table if you don't have food on the table.

> When there is no peace, forget about God, forget about Truth, forget about Divinity, forget about everything else. The first requirement is peace.
>
> Sri Sri Ravi Shankar: The Five Attributes of Brahman

The first priority in Evo delivery projects is to reach all Tolerable levels. Until all Tolerable levels are reached, we can choose to wait to move any quality to the Goal level. Use a table that shows the % impacts the Solutions have on the requirements from the Status levels to the Tolerable levels and not the Goal levels.

	Estimated Impact		*Estimated Impact*		*Estimated Impact*		*Estimated Impact*	
Product Quality Requirements	Splash.Speaker		Splash.Keypad		Battery.Lock		Screen.Scratch	
Past **Status** ***Tolerable*** Goal	*Units*	*%*	*Units*	*%*	*Units*	*%*	*Units*	*%*
User-Friendliness.Learn.Contacts 55 **20** **25** 5 by a year	0	0%	0	0%	-1	0%	0	0%
Reliability 70 **114** **150** 200 by a year	20	56%	25	69%	0	0%	10	28%
Style 5 **9,5** **7** 9 by a year	0	0%	0	0%	0,5	0%	-0,5	0%
Sum of Benefits		**56%**		**69%**		0%		28%
Development Resources								
Project-Budget 0 **4500** **140000** 100000	**1000**	1%	**1700**	2%	**3000**	3%	2000	2%
Sum of Development Resources		**1%**		**2%**		3%		2%
Benefits / Development Resources		**53,06**		**39,01**		0,00		13,2639

Table: This table scores the impacts the Solutions have on the Product Quality Requirements from the Status levels to the Tolerable levels. In this example Splash.Speaker is still getting the highest Benefit / Development Resources score (53,06 vs 39,01), but prioritizing reaching the Tolerable Ranges first, often changes your desition on what to do in the next Evo cycle.

Notice how the impact of Battery.Lock on User-Friendliness.Learn.Contacts gets no score, even though it will improve the system. This is because User-Friendliness.Learn.Contacts already is in the Tolerable Range, and this table scores from Status to Tolerable. Only Reliability can get a positive score as it is the only quality in the Intolerable Range.

When all Product Quality Requirements are in the Tolerable Range, all color codes are yellow or green, we can track the impact from Status to Goal levels.

Product Quality Requirements				*Estimated Impact* Body.Rough		*Estimated Impact* Body.Shape		*Estimated Impact* Battery.Lock	
Past	**Status**	*Tolerable*	**Goal**	*Units*	*%*	*Units*	*%*	*Units*	*%*
User-Friendliness.Learn				-1	20%	-2,5	50%	-1	20%
55	**10**	25	**5** by a year						
Reliability				12	60%	-3	-15%	0	0%
70	**180**	150	**200** by a year						
Style				0	0%	0	0%	0,5	0%
5	**9,5**	7	**9** by a year						
Sum of Benefits					**80%**		**35%**		20%
Development Resources									
Project-Budget				**500**	1%	**700**	1%	**3000**	3%
0	**6500**	140000	**100000**						
Sum of Development Resources					**1%**		**1%**		3%
Benefits / Development Resources					**149,60**		**46,75**		6,23

Table: In this Impact Estimation Table (IET), all 3 Product Quality Requirements are out of the Intolerable Range. We can now focus on moving them all into the Success Range.

Case Study - Firm

From Waterfall to Evolutionary Development (Evo), or, How to create faster, more user-friendly and more productive software

This case study was written by Trond Johansen, QA & Process Manager at FIRM. It describes how the company made the transition from an Waterfall type project model to an Evolutionary one over a short period of time. It talks about their variant and shows some impressive results.

Email: Trond.Johansen@firmglobal.com

1 About the company

FIRM was established in 1996, and has 70 employees in 4 offices (Oslo, London, New York and San Francisco). FIRM delivers one software product: Confirmit. Confirmit is a web-based application that enables organizations to gather, analyze and report key business information across a broad range of commercial applications. Confirmit can be applied to any information-gathering scenario. Its three main data sources are: Customer Feedback, Market Feedback and Employee Feedback.

The FIRM R&D department consists of about 20 people, including a Quality Assurance department of 3 people where I work. We are mainly involved in product development of Confirmit, but we also do custom development for clients who fund new modules of the software.

2 Development background & history

In the very beginning, when FIRM only had a couple of clients, our development was very ad-hoc and customer driven. We didn't follow a formal development process. Based on client feedback, the software was updated nearly on a daily basis. In some way, we were practicing one important element of Evo; early deliveries to stakeholders.

This ad-hoc development resulted in nice features for the few dedicated clients we had, but it also resulted in a lot of defects, long stressful nights, and little control over what our product was developing into.

As our client base grew, we felt a need to introduce more-formal processes in order to increase our quality standards. Larger clients started to ask questions regarding our development processes.

We formalized the development process according to a waterfall model, and started climbing the CMM ladder. The reason for choosing the waterfall model was that it was the only development process we knew about.

After a few years with the waterfall model, we experienced aspects of the model that we didn't like:

- Risk mitigation was postponed until late stages.
- Document-based verification postponed until late stages.
- Attempts to stipulate unstable requirements too early: change of requirements is perceived as a bad thing in waterfall.
- Operational problems discovered too late in the process (Acceptance testing)
- Lengthy modification cycles, and much rework.
- Most important; the requirements were nearly purely focused on functionality, not on quality attributes.

Our experiences are backed up by statistics

a. In a study of failure factors on 1027 IT projects in the UK, scope management related to waterfall practices was cited to be the largest problems in 82% of the projects. Only 13 % of the projects didn't fail. (Thomas, M.2001. "IT project Sink or Swim," British Computer Society Review)

b. A large study showed that 45 % of requirements in early specifications were never used (Johnson, J. 2002. Keynote speech, XP 2002, Sardinia, Italy)

3 The shift of focus: from waterfall to evolutionary development

Peter Myklebust, FIRM CTO, and I heard Tom & Kai Gilb speak about evolutionary project management (Evo) at a software conference autumn 2003. We had just released a new version of our software that contained a lot of new nice features, but it had limitations with respect to usability, productivity and performance (e.g. throughput and response time). We found the ideas very interesting, and Tom and Kai Gilb offered to give a more detailed introduction to the concepts. They spent one day in our offices, giving a very compressed introduction to Evo. We saw that Evo attacked many of the flaws in our waterfall process; most importantly the high focus on quality attributes that we felt could have been better in our latest release.

We decided to do an Evo pilot with a development phase of 3 months. We decided to do a literature study ourselves and then use Evo as best as we could for the next release (Confirmit 8.5), without further Evo courses.

3.1 FIRM's interpretation of Evo: Basis for the 3 month trial period

Evo is in short: Quickly evolving towards stakeholder values & product qualities, while learning through early feedback. The beauty lies with the simplicity of the method, combined with advanced methods of measurement and control.

After the one-day crash course with Tom and Kai Gilb and a literature study ("Competitive Engineering" by Tom Gilb and other material on the subject), our overall understanding of Evo was this:

· Find stakeholders (End users, super-users, support, sales, IT Operations etc)

· Define the stakeholders' real needs, and the related product qualities

· Identify past/status of product qualities and your required goal level (how much you want to improve).

· Identify possible solutions for meeting your goals

· Develop a step-by-step plan for delivering improvements via the identified solutions, with respect to Stakeholder Values & product quality goals:

And most importantly:

· Deliveries of measurable stakeholder-valued results every week (every Evo cycle)

· Measure weekly: are we measurably moving towards our goals?

3.2 Working with requirements the Evo way

With Evo, our requirements process changed. Previously we focused mostly on function requirements, and not on Product Quality Requirements. It is the Product Quality Requirements that really separate us from our competitors. E.g. spell checker in MS Word, why was this a killer application? There was no new functionality; authors of documents have been able to spell check with paper dictionaries for ages. The real difference was superior product qualities: speed of spell checking and usability.

We tried to define our requirements according to a basic standard:

· Clear & Unambiguous

· Testable

· Measurable

· No Solutions (Designs)

· Stakeholder Focus

Example taken from our requirements in Confirmit 8.5:

Usability.Productivity
Scale: Time in minutes to set up a typical specified Market Research-report (MR)
Past: 65 min.
Tolerable: 35 min.
Goal: 25 min. (end result was 20 min.)
Meter: Candidates with knowledge of MR-specific reporting features performed a set of predefined steps to produce a standard MR Report. (The standard MR report was designed by Mark Phillips, an MR specialist at our London office)

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.

After one week we had defined nearly all the top level quality and performance requirements for the next version of Confirmit; and we were ready to start on our first Evo step. We decided that one Evo step should last one week; because of practical reasons, even though we violate the general Evo policy of not spending more than about 2 % of project schedule in each step. The rationale behind the 2% rule is not to spend more time than you can afford to loose. After one week, you'll find out whether you are on the right track (by getting feedback from stakeholders).

3.3 Find Solutions that takes you closer to your goals

For every Product Quality Requirement we looked for possible Solutions

E.g. for Product Quality Requirement: Usability.Productivity we identified the following Solutions: (identified by their name, not their description here)

- Solution.Recoding
- Solution.MRTotals
- Solution.Categorizations
- Solution.TripleS
- ..and many more

We evaluated all these, and specified in more detail those we believed would add the most value (take us closer to the goal level)

3.4 Working evolutionary, the FIRM Evo week

We organized the week in a special way.

On Friday we plan deliverables for version N, at the same time as we build and deploy version N-1 on the test server. Monday to Thursday is dedicated to design, code and test. During the week, the project collects feedback from stakeholders, based on the previous Evo step/week.

	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 1: FIRM Evo week

3.5 Evolutionary project planning

We collected the most promising Solutions and included them in an Evo plan (expressed by using an Impact Estimation Table: IET. See example below). The solutions were evaluated with respect to *value for clients* versus *cost of implementation*: choosing the ones with the highest value first. Note that value can sometimes be defined as *removing risks* by implementing technically challenging Solutions early.

The IET is our tool for controlling the qualities, and delivering improvements to real stakeholders: or as close as we can get to them. (E.g. support people, using the system daily, acting as clients)

	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		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
5					Usability.Replacability (feature count)						
6		1,00	1,0	50,0	2	1	0				
7					Usability.Speed.NewFeaturesImpact (%)						
8		5,00	5,0	100,0	0	15	5				
9		10,00	10,0	200,0	0	15	5				
10		0,00	0,0	0,0	0	30	10				
11					Usability.Intuitiveness (%)						
12		0,00	0,0	0,0	0	60	80				
13					Usability.Productivity (minutes)						
14		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
15					Development resources						
20			101,0	91,8	0		110	4,00	3,64	4,00	3,64
21											

Figure 2: Example of Impact Estimation Table: IET for MR Project – Confirmit 8.5. Solution: Recoding:

Description: Make it possible to recode variable on the fly from Reportal.
Estimated effort: 4 days

4 Impacts on our product, experiences and conclusions

4.1 The method's Impact on Confirmit product qualities

The method's impact on Confirmit product qualities are *not* measured statistically, by doing a scientific correct large-scale survey, although we are currently considering this. The impacts described in this paper are based on internal usability tests, productivity tests, performance tests carried out at Microsoft Windows ISV laboratory in Redmond USA, and direct customer feedback. Only highlights of the impacts are listed here. No negative impacts are hidden.

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 1: Improvements to product qualities

These leaps in product qualities would not have been achieved without Evo. We have received many pleasant emails regarding these quality improvements from our customers:

> "I just wanted to let you know how appreciative we are of the new "entire report" export functionality you recently incorporated into the Reportal. It produces a fantastic looking report, and the table of contents is a wonderful feature. It is also a HUGE time saver."

4.2 Feedback from developers and project managers within FIRM R&D

Evo has resulted in increased motivation and enthusiasm amongst developers because it opens up for empowered creativity. EVO and Continuous Integration is a vehicle for innovation and inspiration. The developers get their work out on test servers, and receive feedback, every week.

Even though the developers embraced the method, there are parts of Evo they found difficult to understand and execute at first:

· Defining good requirements can be challenging.

· It was tricky to find meters (ways of measuring numeric qualities) which were practical to use, and at the same time measure real product qualities.

· Sometimes it takes more than a week to deliver something of value to the client.

· In order to start the next step, some tests were postponed, and some of the postponed tests were never done.

4.3 Lessons learned with respect to the method

Some of the lessons we learned after the trial period are:

· We will have increased focus on feedback from clients. We will select the ones that are willing to dedicate time to us. Internal stakeholders can give valuable feedback, but some customer interaction is necessary.

· Demonstrate new functionality with screen recording software or early test plans. This makes it easier for internal and external stakeholders to do early testing

· Tighter integration between Evo and the test process is necessary

· "Be humble in your promises, but overwhelming in your delivery

4.4 Conclusions

The method's positive impact on Confirmit product qualities has convinced us that Evo is a better suited development process than our former waterfall process, and we will continue to use Evo in the future.

What surprised us the most was the method's power of focusing on delivering value for clients versus cost of implementation. Evo enables you to re-prioritize the next development-steps based on the weekly feedback, what seemed important at the start of the project may be replaced by other solutions based on gained knowledge from previous steps.

The method has high focus on measurable product qualities, and defining these clearly and testable requires training and maturity. It is important to believe that everything can be measured and to seek guidance if it seems impossible.

One pre-requisite related to the method for using Evo is an open architecture.

Another pre-requisite is management support for changing the work process, and this is important in any software process improvement initiative.

The concept of Continuous Integration (CI)/daily builds was valuable with respect to deliver new version of the software every week.

Overall, the whole organization has embraced Evo. The release of Confirmit 8.5 showed some of Evo's great potential, and we will work hard to utilize it to the full in the future. In June 2004 we had Tom and Kai Gilb for a 4 days course for the whole R&D department and related resources and we hope the next version of Confirmit will prove that we have matured in our understanding and execution of Evo.

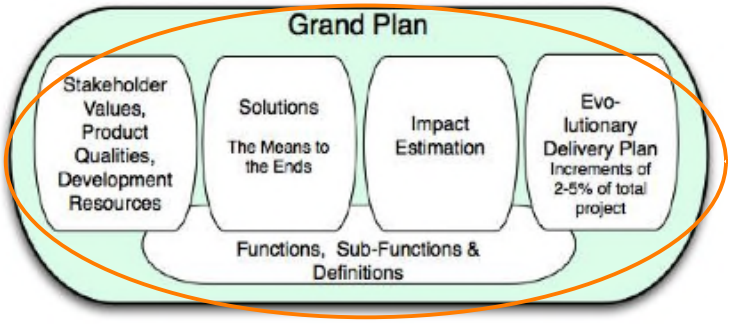
Advanced Chapters

<<<We have now covered all the fundamentals. In the advanced chapters that follow, I will present additional Planguage notation, and cover the theory in more depth.

First, we will look at additional notation you can use to express your plans, starting with notation you can use anywhere, then going into specifics for Stakeholder Values & Product Qualities, Functions, Solutions, Impact Estimation and Evolutionary Delivery.

Then…>>>

Advanced Notation – Everywhere



Building on what we learned in the previous chapters.

We will go through.

How to write so readers understand our intent (Defined as:)

How and when to specify sources of information (<-Source).

How to write comments or notes of any kind, so it does not get mixed up with our core ideas.

How to specify variations in results (± Variance)

How to clearly mark areas where more work is needed (<Angle-brackets>)

How to clearly mark areas where we don't have a factual answer, and are guessing. (?? and ??? and SWAG)

Definitions. Defining Terms, Globally & Locally, or, Everybody knows what it means! Not!

All statements, whether they are part of Stakeholder Values, Product Qualities, Functions, Sub-Functions, Solutions, etc. will be useful only to the degree that they are unambiguous and clear to the intended reader as intended by the author. One powerful yet simple tool to assure clarity is to define words and acronyms thoroughly. Usually, I use the same formatting to indicate that a word is defined as I use for naming statements, I write the name in **bold** letters, with the First Letters Capitalized. I connect words with a dash "-" and indicate a hierarchy with a dot ".".

Local Definitions

If a word in need of a definition is used only in one specification, we can use what I call a Local Definition. For a Local Glossary, we can define the word immediately below the word or acronym used.

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to Learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past 35 min.

Goal [within 1 year] 5 min.

Learn: Defined as: users can, after 24 hours, unassisted by either humans or manuals, perform the task.

Global Definitions

If a word that needs definition is used several places throughout the project, we can define it in what I call a Global Definition Glossary. I recommend my clients to develop a Global Definition Glossary with defined words that apply for the whole organization or project. A Global Definition Glossary can be made easily available for everyone reading the document, through an intra net web site, or equivalent.

If a Scale reads:
Scale: % of all transactions traded by the trading desk …….

If 'trading desk' is a specific word with a specific meaning, it needs to be defined in the Global Definitions Glossary, and it needs to be indicated that it has a specific definition.

Scale: % of all transactions traded by the **Trading-Desk** …….

Global Definitions Glossary
Trading-Desk: Defined as: the central location on the trading floor where …

Once **Trading-Desk** is defined in the Global Definitions Glossary, everyone that needs to refer to that specific meaning can just use the word **Trading-Desk,** without having to explain it in more detail.

As it seems essential for making oneself understood correctly, my clients define a high percentage of the words they use. Those clients that use a Specification Quality Control process(see www.Gilb.com) to enhance clarity and reduce ambiguousness typically define even more words.

<- Source Arrow

The <-Source arrow is a reference to where the information in a statement preceding it comes from, the source of that statement.

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 min. **<- Internal Journal, 12/Last Year p. *12***

Past [Marketing, CS1, Europe, March Last Year, TTM-13 months] 35 min. **<- Internal Journal, 12/Last Year p. *12***

Status [CS2-Alpha0.12, Norway, Today, Cycle 16] 18 min. **<- Bench April Last Year**

Tolerable [Marketing, CS2, Europe, June Next Year] 35 min. **<- Internal Doc. Noa 4/Last Year**

Goal [End-User, Support, CS2, Europe, April Next Year] 6 min. **<- Contract a22 of 3/Last Year**

Goal [Marketing, CS2, Europe, March Next Year] 5 min. **<- Internal Doc. Noa 4/Last Year**

The notation <- is short for Source. If you prefer, just spell out the word "Source:" instead of using the <- arrow. Most of my clients prefer to use the <- arrow. We give source to almost all statements, so people quickly learn what the <- arrow means. The <- arrow gives a visual clue, is quick to write and takes less space than the words. Some clients use a combination, like this <-Source:

Here is a guideline for the use of <-Source:

GEN.SORCE.UNIQUE: Each unique statement must contain detailed references about their exact sources.
GEN.SORCE.VERSION: Each source must contain the version or a date of the source.

All statements should have its own source statement, which should itself be as specific and detailed as possible, taking the reader from an individual statement directly to another sentence, paragraph, table or person. With this practice, it will be easy for the readers to verify the correctness of the information given to them, and if the source is important, the statement will carry the same importance.

I do not write up a list of source documents at the end of a document, as very few people would go thru the trouble of going thru all those documents to find a source.

When we state the <-Source, anyone who question the statement can easily go to the source. They can look for better information and sources. With <-Sources we have the basis for a logical discussion. Instead of discussing what we think, we state the Source of the information. If someone doesn't agree with the information they can back their argument up with updated or better more credible <-Sources.

For the future levels of Tolerable and Goal where we have a contract or agreement that decides the Tolerable and Goal levels, then that contract is the source. Or it is set internally by a marketing or management, then a document from them is normally stated as the <-source. Sometimes the author of the document is the originator of the idea, then they simply write in their own name as the <-source. Often the <-Source is a specific document, contract or speech. We can then simply state the document name, date, heading, paragraph, or however the document is structured. Readers can then immediately verify the correctness of the statement.

Many people have found themselves spending a lot of time and money on misunderstandings. Stating <-Sources gives confidence in the plan, in its accuracy, and its validity. It prevents us from spending resources on what someone wrote that may be incorrect or that nobody wants.

Today in most tools used to write projects, there is a hyperlink system. It is useful if the <-Source is hyperlinked so the reader can simply click on the hyperlink to be taken directly to the <-source statement.

To often, I have seen well intending authors, adding an idea here and one there, not fully aware of the vast additional Development Resources those little idea can add, and that no customer had actually asked for or is willing to pay or wait for that feature. Do not allow people to write project plans without specifying the sources for every statement.

Example Use of <-Source statements

Security.Hack

Type: Product Quality Requirement

Scale: % chance that one hacker can get access to critical client information within one month of trying.

Meter: give a reputable hacker the job to steal information from one of our systems.

Past [Dec. last year] 50% **<- Internal Security Report, Jan. this year.**

Goal [Jan. next year] 10% **<- Contract number 13, Jan. this year.**

IET

	Encryption
Security.Hac	80% ± 20% <- http://en.wikipedia.org/wiki/Advanced_Encryption_Standard Security, Aug. 2005.

Encryption

Type: Solution Build

Description:

Do: Encrypt all client information <- IET

AES: according to the Advanced Encryption Standard (AES) <- **http://en.wikipedia.org/wiki/Advanced_Encryption_Standard**

"Comment"

All forms of comments, notes, hints, examples etc. that does not form a core part of the project shall be clearly marked with "quotation" marks surrounding the comment. We can write comments anywhere, in Requirements, Functions, Solutions, Definitions anywhere, as long as we write quotation marks around it.

User-Friendliness.Learn.Contacts *"We should be known in the industry as the company that makes the easiest-to-use Mobile Phones."* <- Presidents Speech March Last Year

Anywhere we need to add a comment, in the middle of the Requirements statement or in the Solution statement, anywhere, do it, but in such a way that nobody will confuse it with a statement that has to be implemented in any way.

Here is a guideline for the use of comments.

GEN.COM (Comments)

> All manner of comment, notes, suggestion or ideas which are not themselves the actual engineering specification, but merely background, shall be clearly distinguished as such by suitable devices.
>
> *Suggested Devices: italics, "double quotes", Note:…, Comment:..., use of footnotes, use of separate commentary pages.*

Often, after the name of a Requirements, we start right of with a comment that summarizes the Requirements in plain language.

This comment can typically be a wordy summary explanation of the more numeric technical material to come afterwards. It can also be a quote from someone, or something with authority that might be the root of this Requirements coming to life. It can be a customer statement, or a speech given by the CEO, the source of inspiration.

Many projects has gone wrong because someone suggested an idea that was implemented costing additional Development Resources. Later to find out that this idea was never meant to be taken literary without further consideration. Someone (like the organizations boss) just wrote it down thinking at the time of writing that it was a good idea to be considered.

Make sure everyone we work with has one standard way of separating comments from core project decisions. We don't have to use "quotation marks", as long as we all agree on one common way to write "Comments".

Often my clients simply write comments like this:

Comment: I did this to ……. End Comment

When I help my clients re-write their documents into Planguage, we mark all comments, ideas, examples, notes etc. with *italic*. Often we find that 80% to 99% of the text is *comments* of some kind, and only a small percentage is new core ideas.

± Variance

The ± gives the variance of results given or expected on any number used.

User-Friendliness.Learn.Contacts

Past [Marketing, VX2, Europe, Jan. Last Year] 55 **±5** min. <- Internal Journal, 12/Last Year p. 12

Status [CS2-Alpha0.12, Norway, Today, Cycle 16] 18 min. ± 2 min. <- Bench April Last Year

Goal [Marketing, CS2, if titanium is available, Asia, March Next Year] 3.5 **±1** <- Internal Doc. Noa 4/Last Year

± Variance like 55 ±5 is indicating the variance in the number. The number 55 ±5 means the result are anything from 50 to 60. This is different than ?, ?? and SWAG, and Uncertainty in that we might have lots of experience and high certainty, and we know the result will be varying from 25 to 75. ± simply explains the variation in the result we get or expect to get.

All numbers in a plan should indicate their variance. This will help us understand variation of results and consequently risk.

In Impact Estimation Tables (IET) it is especially healthy to use ± variation as we rarely have exact numbers, and not showing the ± variation can lead the reader to think otherwise.

The Sunrise Hotel Website Example, with ± estimations.

	Front-Desk	*Website*	*Global*	*DB*	*Management*
Booking-Cost	0% ± 5%	40% ± 15%	40% ± 30%	10% ± 2%	10% ± 10%
Booking-Ease	0% ± 5%	40% ± 20%	40% ± 5%	10% ± 2%	10% ± 7%
Added-Busines	0% ± 0%	20% ± 5%	80% ± 40%	0% ± 10%	20% ± 15%
Customer-Satis	10% ± 5%	40 ± 10%	10% ± 5%	15% ± 3%	5% ± 5%
Money	20% ± 7%	20% ± 2%	5% ± 3%	20% ± 1%	20% ± 2%

?? and SWAG

?? and SWAG means that this is a guess or have great uncertainty. Many people might have a problem writing a number down on anything without knowing that it is scientifically absolutely correct. To remove this fear, and allow people to use numbers to communicate, we can use question-marks, or SWAG like this: 95??, or ??100?? or 50 SWAG, clearly indicating that the accuracy of this number is questionable, maybe in need of further investigation.

SWAG; defined as: Scientific-Wild-Ass-Guess

<Angle-Brackets>

We use the <angle-brackets> to indicate that the statement in-between the <angle-brackets> need more work, is not complete.

<Trend [Market, Europe, March Next Year] 6 ±2 min. <- Big Journal, 11/Last Year p. 30>

Using <angle-brackets> can take away some fear of writhing something that we are not 100% sure of. We can write some idea down without worrying about its correctness, and worrying about somebody taking that information and making decisions based on it..

Use <angle-brackets> around anything that we acknowledge need more work. It both works as a reminder to the author, that this part is not finished, and it warns the reader of the same, preventing them from taking the information within the <angle-brackets> to seriously.

Administration

For each Requirement, each Solution, each Evo Cycle, etc. I give them a Type, Version, Author, Owner and a Status.

Name-Tag:

Administration:

Type: <Stakeholder Value Requirement, or Product Quality Requirement, or Product Function Requirement, Solution Constraint, or Solution, or Evo Cycle, etc.>

Version: <date and time of last revision>

Author: <name of who wrote the specification>

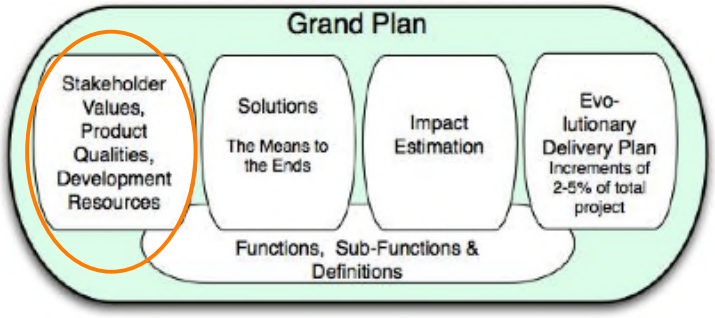
Owner: <only person authorized to make any changes to the specification>

Status: <quality control status, (draft, approved, exited) > <-

By Typing each specification, a reader can immediately see if they are looking at a Stakeholder Value Requirement, Product Quality Requirement etc., even if it is shown in isolation. I can also find and sort for specific types.

By giving each specification a Version, instead of one version for the whole document, <<<>>>

Advanced Notation: Stakeholder Values & Product Qualities



You now know a basic way to write and communicate clear Stakeholder Values & Product Qualities using Planguage. In this chapter we will go through many more techniques to express our ideas. We can use a few of these, or all of them, as needed to express what we want to express.

Building on what we learned in the previous chapter.

We will go through.

How to move from wishes (Wish) to Goals.

How to specify who are the key stakeholders for the Requirement. (Stakeholders)

How to specify several Past levels and future targets depending on different time, space, and conditions ([Qualifier]).

How to use benchmarks, to better understand the challenges of meeting our own targets, and how well we are doing compared to others (Record).

How to specify trends in the Marked or for Product (Trend).

How to show how much impact the Solutions used in the Impact Table have on this Requirement. (IET-Impact)

How to demand a Safety Level to be used in the Impact Estimation Table. (IET-Safety)

Note: Don't expect to understanding the detail of **User-Friendliness.Learn.Contacts** on this page now, it is only an overview. Explanations start on the next pages.

User-Friendliness.Learn.Contacts *"We should be known in the industry as the company that makes the easiest-to-use Mobile Phones." <- Presidents Speech March Last Year*

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Record [SIMO3, Japan, March Last Year, TTM-12 months] 4 min. <- Big Journal, 11/Last Year p. 30

Record [Marketing, Van2, Japan, March Last Year, TTM-11 months] 9 min. <- Internal Journal, 12/Last Year p. 12

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 ±2 min. <- Internal Journal, 12/Last Year p. *12*

Past [Support, Europe, March Last Year, TTM-12 months] 5 ±1 min. <- Internal Journal, 12/Last Year p. 12

Past [Marketing, CS1, Europe, March Last Year, TTM-13 months] 35 ±3 min. <- Internal Journal, 12/Last Year p. *12*

Past [Marketing, VX2 version1.23, Europe, Jan. Last Year, TTM-17 months] 55 ±5 min. <- Internal Journal, 12/Last Year p. 12

Status [CS2-Alpha0.12, Norway, Today, Cycle 16] 18 min. ± 2 min. <- Bench April Last Year

<Trend [Market, Europe, March Next Year] 6 ±2 min. <- Big Journal, 11/Last Year p. 30>

Tolerable [Marketing, CS2, Europe, June Next Year] 35 ±4 min. <- Internal Doc. Noa 4/Last Year

Goal [End-User, Support, CS2, Europe, April Next Year] 6 ±1 min. <- Contract a22 of 3/Last Year

Goal [Marketing, CS2, Europe, March Next Year] 5 ±1 min. <- Internal Doc. Noa 4/Last Year

Goal [Marketing, CS2, Asia, March Next Year] 4 ±1 min. <- Internal Doc. Noa 4/Last Year

Goal [Marketing, CS2, if titanium is available, Asia, March Next Year] 3.5 ±1 <- Internal Doc. Noa 4/Last Year

Wish [within 2 years] 3 min. <- Stakeholder x, Nov. Last Year.

IET-Safety [End-User, Support, CS2, Europe, April Next Year] = Goal * 2 <- The President

IET-Impact [CS2, Europe, Jan. Next Year] 3 min.

Wish

User-Friendliness.Learn.Contacts

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Goal [within 1 year] 5 min.

Wish [within 2 years] 3 min.

Wish levels are points on the Quantification Scale. While Goal levels are committed or promised targets along the defined Quantification Scale. The Wish levels are not committed nor promised. They are just used to capture in writing a wish, request or even an expressed requirement from a Stakeholder, but one that you have not committed to yet.

In theory, all Goal levels first start out as Wish levels. After careful evaluation we can commit to some Wish levels, and thereby make them Goal levels.

It can be important to both acknowledge and to write down a Stakeholder request, without having to commit to delivering it. Often marketing or sales people know about Stakeholder wishes, but for some reason they do not communicate it to the rest of the team. This can among many reasons be because they have no place to write it down, or because they believe it is not feasible to deliver that wish yet. Having a Wish level enables us to write it down in the Requirement specification, without it becoming a requirement, and thereby communicating better with the rest of the team. Who knows, maybe it is possible to reach the Wish level after all;-)

Stakeholders

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Past 35 min.

Goal [within 1 year] 5 min.

All Requirements have somebody or something that want or require that Requirement. After the parameter Stakeholders, specify that group, person or thing that has a stake in this particular Requirement. When the Requirement is required not from a person or group, but from a thing, than we can list the thing as a Stakeholder. I use the Stakeholders parameter for all levels and types of Requirements; at Stakeholder, Product or Sub-Product level, and with Scalar, Function, Development Resources and Solution Constraints types.

Remember my definition for Requirements? Requirements are; anything Stakeholders require. Therefore, per definition, to qualify as a Requirement, it must have at least one Stakeholder requiring it. If it is not required by a Stakeholder it is not a Requirement, it is something else. At the level of Requirements, even seemingly small ideas, just a few innocent words that sounds like good ideas, but that are not Required by Stakeholders, have been known to become very expensive in development, integration, testing, installation and maintenance. In fact, if you allow requirements writers to specify anything they want into the Requirement specifications, it will probably quickly kill your profit margin.

I have often challenged so called Requirements, asking: “who is the Stakeholder requiring it?”. When I get unclear answers back, someone mumbling something about it being a good idea. I clarify to them that it is not a Requirement unless some Stakeholder is requiring it. And ten we yank it out of the Requirement specification.

By listing the Stakeholders, we can also take the Requirement required by a specific stakeholder to them and ask them for correctness and acceptance. During the Evo Cycles we can deliver it to them and measure improvements, we can get feedback and learn from the Stakeholders, learn what the real requirements are.

Example 1: We develop a mobile phone, and the End User of the phone requires two weeks standby-time. Then, in the Standby-Time requirement, list the End-User as a Stakeholder.

Example 2: We develop a battery for a mobile phone. The mobile phone requires certain stability in voltage from our battery. Then, in the voltage stability requirement, list the mobile phone as a Stakeholder.

Illustration: Behind every Requirement are one or more Stakeholders.

Example:

Cost.Total
Type: Product Function
Stakeholders: Customer, Sales Clerk.
Description: display total cost.

With my clients, I insist that all Requirements detail the Stakeholders, this ensures that “good ideas” are kept out, keeps the Requirements specification lean, avoids extra costs, and gives us a reference to communicate with.

[Qualifier]

In the previous chapters we used the qualifier with a date; *[within 1 year]*. The [qualifier] is used to express specific conditions that apply for a specific statement like a Requirement.

Lets expand on this and find many additional ways to use the [qualifier] to express the details of our Stakeholder Values and Product Qualities.

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past **[End-User, CS1, Europe, March Last Year, TTM-15 months]** 35 min.

Past **[Support, Europe, March Last Year, TTM-12 months]** 5 min.

Past **[Marketing, CS1, Europe, March Last Year, TTM-13 months]** 35 min.

Past **[Marketing, VX2 version1.23, Europe, Jan. Last Year, TTM-17 months]** 55 min.

Goal **[End-User, Support, CS2, Europe, April Next Year]** 6 min.

Goal **[Marketing, CS2, Europe, March Next Year]** 5 min.

Goal **[Marketing, CS2, Asia, March Next Year]** 4 min.

Goal **[Marketing, CS2, if titanium is available, Asia, March Next Year]** 3.5

I use a qualifier to specify the condition of the various levels; Past, Status, Tolerable, Goal etc.. I also use the [qualifier] with Functions or Solutions that have condition for validity.

Write the qualifying conditions, the qualifier, inside [square brackets].

The qualifier typically contain any combination of the following:

The Stakeholder

Specifies the Stakeholder to which this statement applies to. It can be useful to think about two categories of Stakeholders, Internal and External.

Internal:

Goal **[Marketing]** 5 min.

> When a Goal level, or any other level is an internal plan, specify the internal Stakeholder in the [qualifier]. This will allow us to express Goal levels that are different from what is contracted, or those our external Stakeholders require.

External:

Goal **[End-User]** 6 min.

> When the requirement applies to / is required by an external Stakeholder, write down that Stakeholder in the [Qualifier]

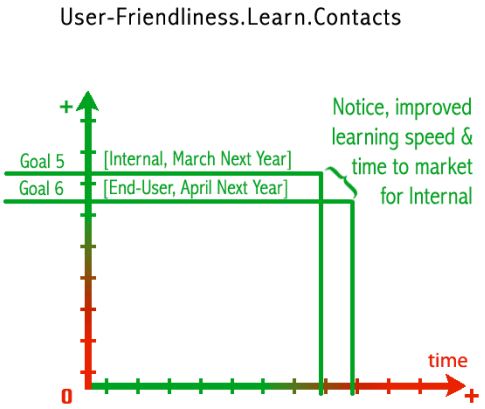


Illustration: We have expressed an internal Goal level that has both higher quality and will be finished earlier than the External End-User Goal level. This gives us the ability to express both what the Stakeholders expects, and what we aim for. Planning an Internal Goal level that exceeds the External expectations, we can both give ourselves a little headroom incase we need that, and we can aim to beat the external expectations, and surprise our Stakeholders.

The what;

Past **[VX2 version1.23]** 55 min.

Goal **[CS2]** 6 min.

We can specify the name of what we refer to in the [Qualifier]. This is usually a product name.

[]

The condition;

There might be changing conditions that make a specific target valid or not. I use **if** before the condition inside the [qualifier] to clarify that this is a condition not a Goal level.

Goal [plastic] 500

says that we plan to release a product with/for plastic at a specific quality level.

Goal [**if** plastic] 500

says that if plastic is used we plan to deliver the specific quality level. If it is not available, the Goal level is invalid.

One example of using the condition fruitfully is when we do not know what products our competitor or supplier will announce, or what technology will be available later in the product cycle.

Goal **[if titanium is available]** 3.5

says that if titanium is available then the Goal level is to be at 3.5, if titanium is not available, then this Goal level is not valid.

One time when we helped an International Aid organization specify their Requirements, we ran into a situation where one Goal level was valid if there was war in the country, and another Goal level was valid if there was no war in the country. This situation fluctuated sometimes from month to month. Using a condition in the [Qualifier] like this:

Goal **[if war]** 55

Goal **[if peace]** 105

gave us dynamically valid Goal levels that was immediately adopted as the country fluctuated between peace and war.

Illustration: This use of condition allows we to make decisions, even if we don't know what will happen in future events. It will automatically be valid or canceled depending on the shifting conditions in the market.

The Scope;

Goal **[Asia]** 4 min.

Goal [**Movie Audiences**] 7 seconds.

[Scope] defines the extent of the area or subject matter that something deals with or to which it is relevant. In this example the geographical area. This allow us to focus on one area or subject matter at the time.

This can prevent us from trying to satisfy the highest demand for all areas. Maybe we must deliver 4 min. to Asia, but only 6 to Norway, then it is possible that we can deliver and sell a product in Norway before we can have it ready for Asia. This not only allows us to make money earlier, but it will also give us valuable experience early on. Many projects have failed because they have tried to deliver the high qualities needed one to everyone. Using Scope, will allow this distinction together with other parameters like Stakeholders.

The Date & Time to Market;

There are two types of dates normally used. The first one is the **delivery date**. For future references the date which the Product Quality or Stakeholder Value is to be reached, and in the case of Past references, when did it deliver. The second type of date is used on historic references, and specifies the **Time to Market** in the Past.

The first; when did it happen, or, when it is valid.

Past **[March Last Year]** 35 min.

Goal **[March Next Year]** 4 min.

Specify when we are meant to reach the Goal levels. Specify when a Past level was reached. If our Goal level is to deliver on March Next Year, or if a Past level was reached at a specific time, specify this in the [qualifier],

The Second; used with Past references only, specifies how long it took to develop and deliver to that level.

Past [March Last Year, **TTM-13 months**] 35 min.

TTM is short for Time To Market. This [qualifier] says that it took 13 months to get the product out to market on March Last Year at a quality (see Scale) level of 35 min.

If we want to get 20 times better in something but we don't know when we have to deliver, then we cannot logically tell anything about the resources required, the means we need to achieve it, or if it at all is realistic. The Goal levels actually have no meaning at all without a timeframe. Do we have time to do worse for the next 100 years or must we get 20 times better by tomorrow? "Improving the air quality 3 times" means nothing if we don't have a date specifying when it must be accomplished.

[qualifier] with Scale

The [qualifier] is also used in combination with the Quantification Scale to reuse the Scale systematically.

Instead of the Scale as we have used it -

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

We can write –

Scale: average time in minutes, to learn defined [Task] for defined [Users].

Goal **[Task= how to program contact names and telephone numbers into the memory of the phone, Users=never used our brand phone before,** Marketing, CS2, Asia, March 2004**]** 4 min.

Goal **[Task=to call someone from the internal name database**, **Users=upgrading from a older model of our brand,** Marketing, CS2, Asia, March 2004**]** 1 min.

<<<<<<<<<<< give many more examples as this is often used>>>>>>>>>>>>

Other [qualifiers]

This set of qualifiers is not meant to restrict, but as examples of what my clients are actually using. If you need to invent some qualifying conditions to express/limit the ideas in the projects you are working in, I support you in that.

Other uses.

The [qualifier] can be used on the Meter. We can specify two or more Meters to be used in different situations.

Meter **[During development]** use a stopwatch

> **[During operation at customer site after final delivery]** automatic timing, recording and reports with 1/10 of a seconds accuracy using X.

We can use the [qualifier] on any idea specified in the project plan. For example, use the [qualifier] when specifying Solutions or Evolutionary Cycles.

Is the Goal level valid?

Goal **[Marketing, CS2, if titanium is available, Asia, March Next Year]** 3.5

This Goal level is only valid if all the conditions in the qualifier are met simultaneously. The Qualifier also gives the Goal level a unique name to refer to. When we have many Goal levels or many Past levels we refer to the specific Goal or Past level by including its [qualifier]

Summary: List of often used [Qualifier] types.

Time/Date; *[Date: 24.12.2045], [Date: Release v2], [Date: During Development];* when did it happen, or, when it is valid.

Stakeholder; *[Stakeholder: Marketing], [Stakeholder: End-User]*

Internal: Goal level, or any other level that is an internal plan. This will allow us to express Goal levels that are different from what is contracted, or those our external Stakeholders require.

External: When the requirement applies to / is required by an external Stakeholder

What/Product; *[Product: VX2 version 1.23];* what we refer to, typically a product name/version.

Condition/if; ***[if** contract with government];* if condition is true, the statement is true.

Scope; *[Scope: Asia], [Scope: Movie Audiences];* defines the extent of the area or subject matter that something deals with or to which it is relevant.

TTM / Time to Market; *[March Last Year, **TTM-13 months**]*; used with Past references only, specifies how long it took to develop and deliver to that level.

Scale Qualifier; *Scale: average time in minutes, to learn defined [**Task**] for defined [**Users**]. Goal **[Task**= how to program contact names and telephone numbers into the memory of the phone, **Users**=never used our brand phone before**]** 4 min.* Combined with the Quantification Scale to reuse the Scale systematically.

Trend Internal; *[Trend: Internal]*; only used with Trend, is an estimate of how well our product will perform in time.

Trend Market; *[Trend: Internal]*; only used with Trend, is an estimate of how well the whole market including Stakeholders expectations and competing products will perform in time.

Record

Record level is a special kind of Past level, the best Past of any similar project or system we know of. It's the world Record on the Scale we are working with. The Record level is a benchmark. Benchmarking the industry or our own company, as the best ever achieved.

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Record [SIMO3, Japan, March Last Year, TTM-12 months] 4 min.

Record [Marketing, Van2, Japan, March Last Year, TTM-11 months] 9 min.

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 min.

Past [Support, Europe, March Last Year, TTM-12 months] 5 min.

Past [Marketing, CS1, Europe, March Last Year, TTM-13 months] 35 min.

Past [Marketing, VX2 version1.23, Europe, Jan. Last Year, TTM-17 months] 55 min.

Tolerable [Marketing, CS2, Europe, June Next Year] 35 min.

Goal [End-User, Support, CS2, Europe, April Next Year] 6 min.

Goal [Marketing, CS2, Europe, March Next Year] 5 min.

Goal [Marketing, CS2, Asia, March Next Year] 4 min.

Goal [Marketing, CS2, if titanium is available, Asia, March Next Year] 3.5 min.

Wish [within 2 years] 3 min.

The first Record together with Scale reads:

The best related product we know about on this Product Quality or Stakeholder Value, [is the SIMO3, in the Japanese market, delivered March Last Year and it took 12 months to market] had an average time of 4 minutes to learn how to program contact names and telephone numbers into the memory of the phone.

The other Record is the best achieved by our company. *Record [**Marketing**]*

The Record can be the Record within our organization or one set by others like our competitor.

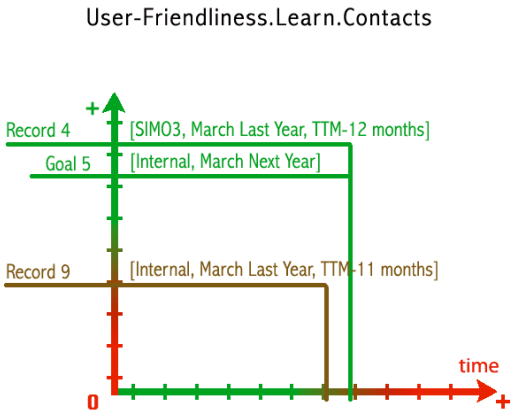


Illustration: In the diagram we can see how our Internal Goal level measure up against what we have achieved before, Record level [Marketing], and what somebody else has achieved before, Record level [SIMO3]. We can also see the Time to Market they used, and compare it to our Goal level. With this information, we can better understand if our Goal levels are reasonable.

Why find Records?

If we set Goal levels without knowing what other people have achieved in the Past, the Record, it is difficult to know anything about the probability of success and the Development Resources it will consume.

If we where to approve a project for improving Mr. Pers high jumping ability. The world record is 2.3 meter. If Mr. Pers Goal level is 3.0 meter within 1 year, and Mr. Pers Past is 1.5 meter, would we approve the funding for this project? Of cause not!

If we have a Goal level that goes beyond the Record, then we are actually not only planning projects but we are doing research. Normally, to beat world Records is a long term evolutionary process. If we know how to achieve world Record Stakeholder Values or Product Qualities, we are likely to get customers or funding. If our Goal level is to beat the world Records on many Stakeholder Values or Product Qualities within the same project, I would be suspicious of the reality of the project.

To go beyond current Records we can count on sky-high Development Costs, or some really bright ideas on different ways of doing things.

Knowing the Record can give encouragement. We know that what we are trying to do is possible, in fact someone has done it before us. We can even do some research and find out how they did it, and reuse many of the ideas.

If we think we are leading edge in our field. We should know the Record in our area. Do Olympic high jumpers know the current world Record in their discipline? Yes! Do you know the world Record in yours?

Illustration: We have a tradition of recognizing Records, here my daughter Mira-Bai and her cousin Jade are awarded with medals for skiing (yes Norwegians are born with skies on;-).

In project management, to be able to compete, both in being the best, and in controlling the costs, we need to know the internal and external Records in our field. If you are a security expert, or a usability expert, or a strength expert, or a reliability expert, you should know the Records numerically.

Trend

The Trend is an estimate, a projection, of how a Product Quality or Stakeholder Value will become in the future.

I use two kinds of Trends,

Internal, is an estimate of how well our product will perform in time.

Market, is an estimate of how well the whole market including Stakeholders expectations and competing products will perform in time.

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Record [SIMO3, Japan, March Last Year, TTM-12 months] 4 min. <-

Record [Marketing, Van2, Japan, March Last Year, TTM-11 months] 9 min. <-

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 min.

Past [Support, Europe, March Last Year, TTM-12 months] 5 min.

Past [Marketing, CS1, Europe, March Last Year, TTM-13 months] 35 min.

Past [Marketing, VX2 version1.23, Europe, Jan. Last Year, TTM-17 months] 55 min.

Trend [Market, Europe, March Next Year] 6 min.

Tolerable [Marketing, CS2, Europe, June Next Year] 35 min.

Goal [End-User, Support, CS2, Europe, April Next Year] 6 min.

Goal [Marketing, CS2, Europe, March Next Year] 5 min.

Goal [Marketing, CS2, Asia, March Next Year] 4 min.

Goal [Marketing, CS2, if titanium is available, Asia, March Next Year] 3.5 min.

Wish [within 2 years] 3 min.

Trend together with Scale reads:

In the European Market, by March Next Year, we estimate 6 minutes will be the average time to learn how to program contact names and telephone numbers into the memory of mobile telephones.

As we can't know the future, Trend is an estimate or projection of how things will be in the future. We can have any number of Trends that give us insight into our own project, the Market and our competitor.

User-Friendliness.Learn.Contacts

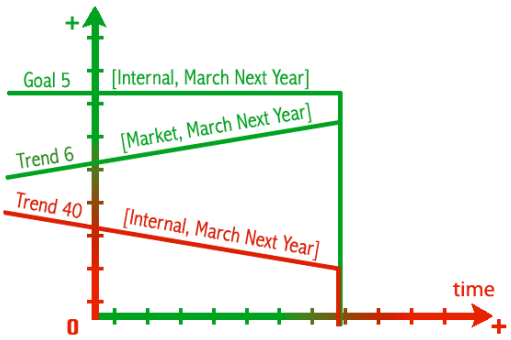


Illustration: In this diagram we used one [Market] and one [Internal] Trend. The [Market] Trend is showing what we expect the Market to be March Next Year. Our Goal level is better than the [Market] Trend. Our existing product is degrading as we can see in the [Internal] trend.

Trend Examples:

Trend [**Internal**, CS1, Europe, 2020] 99

This Internal Trend is giving us a projection on how our current product, CS1, will perform in Europe, 2020. This is especially interesting if the Product Quality or Stakeholder Value is getting worse over time. Let's say we have a product on the market, and it is getting more and more users, the load is increasing, and with that the performance is decreasing, or reliability is decreasing. We can use Trend [Internal] to communicate this and relate it to our new project.

Trend [**Market**, Africa, 2020] 5 min.

The Market Trend is an estimation or projection to where the Market is heading. Let us assume we are developing a product to be delivered in Africa in 2020. We have this information:

Past [Africa, Last Year] 25 min.

Trend [**Market**, Africa, 2020] 5 min.

Goal [Africa, 2020] 10 min.

Here we can see that we are not only competing with the Past levels, but also, we have to estimate the Market Trends, to be able to deliver a competitive product.

Trend [Market, Competing product X, 2020] 2 min.

Not only do we have to beat the competitors current products, but the products they will introduce around the same time as when we introduce our products.

IET-Safety

The parameter IET-Safety, can be used to specifies the Safety Level required in the Impact Estimation Table. The Safety Level in the IET is the over-design done to make sure we have enough Solutions to satisfy the Goal levels. See the sub-chapters 'Sum of Impacts' and 'Safety Level' for more information on and examples of using Safety Factor with IETs.

Goal 6 min.

IET-Safety Goal * 2

In this example a IET-Safety factor of "Goal level * 2" is given, so if the Goal level is 6 minutes, we have to design the projects with a set of Solutions that together adds up to 200% in the Impact Estimation Table.

or

IET-Safety 3 min.

IET-Safety 3 min. means that even though we have set a Goal level of 6 min., we require the Solutions in the Impact Estimation Table multiply so the sum equal 3 min.

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 min.

Trend [Market, Europe, March Next Year] 6 min.

Status [CS2-Alpha0.12, Norway, Today, Cycle 16] 18 min.

Goal [End-User, Support, CS2, Europe, April Next Year] 6 min.

Wish [within 2 years] 3 min. <- Stakeholder x, Nov. Last Year.

IET-Safety [End-User, Support, CS2, Europe, April Next Year] = Goal * 2

*IET-**Impact***

The IET-Impact displays the 'Sum of Impacts' from the Impact Estimation Table. The 'Sum of Impacts' shows us how well our current set of Solutions, as stated in our Impact Estimation Table will meet our Stakeholder Values & Product Qualities Goal levels (if multiplied together). It can also show us if we are meeting the IET-Safety levels specified.

The 'Sum of Impacts' is taken from the Impact Estimation Table. It will change as the project evolves and needs to be updated. With software tools like spreadsheets and data bases, it can be updated automatically. It shows the readers of the Requirement specification how solid our set of Solutions are relative to the Stakeholder Value or Product Quality Goal level.

The 'Sum of Impacts' together with the updated Status level, are the highlights from the design phase done with the IET. IET-Impact is redundant, in that it is information actually developed in the IET, and as such can be left out. Sometimes the reader of the Requirement specification will not go into the design phase and look at the IET, then the IET-Impact gives them critical insight.

For more information on 'Sum of Impacts', see sub-chapter 'Sum of Impacts'.

User-Friendliness.Learn.Contacts

Stakeholders: End-User, Support

Scale: average time in minutes, to learn how to program contact names and telephone numbers into the memory of the phone.

Meter: time 5 people who have not had a mobile telephone before, as well as 5 people who has, use the average.

Past [End-User, CS1, Europe, March Last Year, TTM-15 months] 35 min.

Trend [Market, Europe, March Next Year] 6 min.

Status [CS2-Alpha0.12, Norway, Today, Cycle 16] 18 min.

Goal [End-User, Support, CS2, Europe, April Next Year] 6 min.

Wish [within 2 years] 3 min.

IET-Safety [End-User, Support, CS2, Europe, April Next Year] = Goal * 2

IET-Impact [CS2, Europe, Jan. Next Year] 3 min. <- *Impact Estimation Table*

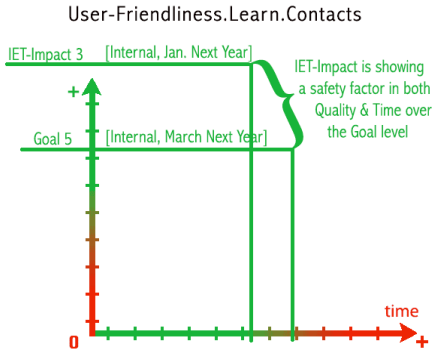


Illustration: We do not need to reach the IET-Impact of 3! When the Goal level is actually reached, its priority falls away. If the IET-Impact falls short of the Goal level, it suggests a weak area that needs more attention. We must remember that Solutions do not normally add together as we add them in the Impact Estimation Table. Therefore, a high IET-Impact is only suggestive that the Product Quality or Stakeholder Value have enough timely Solutions. More importantly, if IET-Impact falls short of the Goal level, we have identified a likely weak point in our plan.

Templates - Stakeholder Values & Product Qualities

It can be very useful with blank templates to fill out. Here are three examples. Feel free to make your own templates, adding or deleting the notation you need.

Name Tag:

Stakeholders:

Scale:

Meter:

Status [ ] <-

Tolerable [ ] <-

Goal [ ] <-

Name Tag:

Version:

Type:

Stakeholders:

Scale:

Meter:

Past [ ] <-

Past [ ] <-

Status [ ] <-

Tolerable [ ] <-

Goal [ ] <-

Goal [ ] <-

Wish [ ] <-

Name Tag:

Administration:

Type: <Stakeholder Value Requirement or a Product Quality Requirement>

Version: <date and time of last revision>

Author: <name of who wrote the specification>

Owner: <only person authorized to make any changes to the specification>

Status: <quality control status, (draft, approved, exited) > <-

Stakeholders:

Scale:

Meter:

Past [ ] <-

Past [ ] <-

Trend [ ] <-

Record [ ] <-

Status [ ] <-

Tolerable [ ] <-

Tolerable [ ] <-

Goal [ ] <-

Goal [ ] <-

Wish [ ] <-

Wish [ ] <-

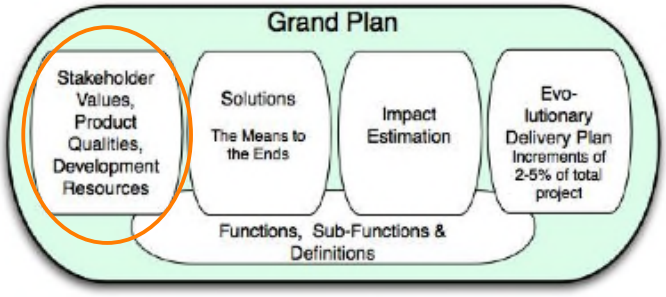
IET-Safety [ ] <-

IET-Impact [ ] <-

_____ : defined as:

_____ : defined as:

Advanced Stakeholder Value & Product Quality Requirements



Critical few Stakeholder Values & Product Qualities

Illustration: Any system has what seems like unlimited Product Qualities. We cannot specify and deal with them all, not even hundreds of them, yet, they are the ones that threaten to kill our systems and the ones that can make our systems great.

How many Product Qualities exists for your System? 10 or 1000, or infinite. I don't know if a limit exists, but even the simplest of systems have hundreds or thousands of Product Qualities. It would take to much time for someone, at one level of a system, to write up, evaluate, measure and track, more than a few dozen Product Qualities.

In practice, we limit ourselves to a critical few Requirements, the ones that can make or break the project. I like to operate with a maximum of 7 per responsibility or Project Level. If you can specify, evaluate Solutions against, measure Status levels during Evo Cycles, learn from and evaluate progress towards Tolerable and Goal levels, for the top 7 critical Stakeholder Values or Product or Sub-Product Qualities, you will probably be beating the competition hands down.

Evo can help detect if any Product Quality of our project is getting out of control and threatening our project. This could be one of the Product Qualities we planned for, or the one we did not plan for.

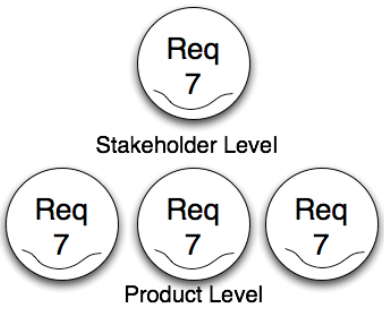


Illustration: in a small project, we might operate with 2 Project Levels, a Stakeholder Level that have 7 Stakeholder Value Requirements, and a Product Level that is divided into 3 sub-products, each with 7 Product Qualities.

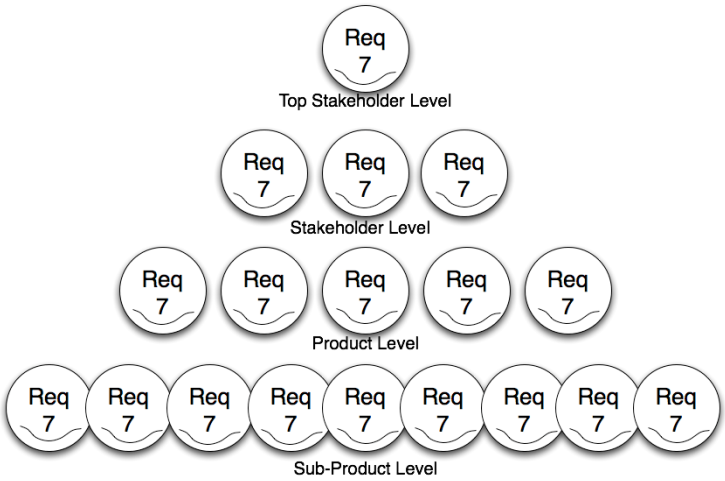


Illustration: a more complex project might have several Project Levels, each level with a set of sub-projects, that again have a set of sub-projects. One group is responsible for the 7 Requirements at their Project Level and for their sub-project. So no one is responsible for more than about 7 Requirements, but in total the team is responsible for 18 * 7 = 126 Requirements.

A little case study in hierarchy of Requirements.

I was facilitating rewriting Requirements for a desktop application that users use to control content on a consumer electronics device. Some of the core functions where; an online shop, installation of items from the desktop application to the device, and backing up and restoring the content on the electronic device. An desktop application already existed that did the core functions, but a new user interface was envisaged intended to make the desktop application more user-friendly. At the Product Level we divided it into two, a Server Side (where the backend to the shop and other updates resided), and a Client Side (the desktop application itself that consumers install on their PC). We further divided the Client Side into two; we grouped the user-friendly related Requirements in one group, and all the other Product Quality Requirements in another. It looked something like this.

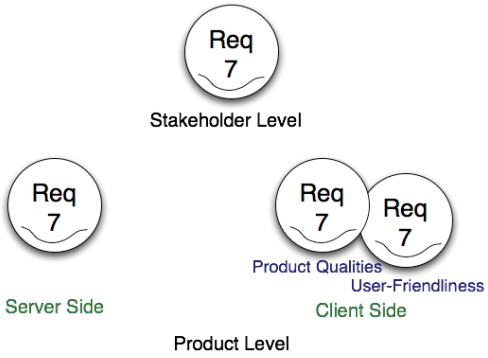


Illustration: Together the Product Qualities of the Server Side and Client Side helped satisfy the Stakeholder Values. The Client Side had about 7 Product Quality Requirements, one was called User-Friendliness. User-Friendliness was further broken down into 7 User-Friendliness qualities that was owned, specified and developed separately than the other Client Side Product Qualities.

User-Friendliness Breakdown and Summary

The main reason they developed a new version of the product was to improve User-Friendliness. One team was responsible for it, and they needed all 7 aspects of User-Friendliness to successfully engineer it into the product. They then reported their progress to the person responsible for the whole product. Below is an example of how this was done.

User-Friendliness: consists of: {**User-Friendliness.Learn**, **User-Friendliness.Intuitiveness + about 5 other not specified here)**

User-Friendliness.Learn

Scale: Average time, for defined [User], to Learn how to do, defined [Task]

Past [Task=Backup, User=Inexperienced] 5 min.

Status [Task=Backup, User=Inexperienced] 3 min.

Goal [Task=Backup, User=Inexperienced] 1 min.

User-Friendliness. Intuitiveness

Scale: % chance, that a defined [User], can correctly figure out, within 7 seconds, how to execute the next command they want to execute, for defined [Task].

Past [User=Inexperienced, Task=Backup] 60%

Status [User=Inexperienced, Task=Backup] 65%

Goal [User=Inexperienced, Task=Backup] 90%

Learn: defined as: figuring out how to do a Task for the first time, including the time to do the task.

Backup: defined as: from a user is in front of personal computer, our product is running, electronic device in hand, but not connected to a personal computer. Until the user has successfully made a backup of the electronic device onto the personal computer.

The lady responsible for User-Friendliness, reported progress to the person responsible for the total set of Client Side Requirements. When grouping **User-Friendliness.Learn** and **User-Friendliness.Intuitiveness**, we see that, time to learn, and, % chance, don't add well. In this case, she decided to report: “the % achievement towards all the 7 User-Friendliness Goals”.

To find this number, first, on each Requirement, she added how far the Status level had moved; from the Past level towards the Goal level. Second, she added the numbers up, and divided the number by the number of Requirements.

When the Status Level of **User-Friendliness.Learn** had moved from Past 5 to Status 3 min. towards Goal 1 min., and Status of **User-Friendliness.Intuitiveness** had moved from Past 60% to Status 65% towards Goal 90%, she could report that her team had moved 45% towards all User-Friendliness Goals.

User-Friendliness.Learn: Goal 1 min. - Past 5 min. = 4 min. Status 3 min. of 4 min. = 75%

Friendliness.Intuitiveness: Goal 90% - Past 60% = 30%. Status 5% of 30% = 17%

(75% + 17%) / 2 User-Friendliness Requirements = 45%

User-Friendliness

Scale: % movement from Past Levels to Goal Levels on **User-Friendliness.Learn**, and **User-Friendliness.Intuitiveness**,

Past [at project start] 0%

Status [during the project] **45%**

Goal [at final delivery] 100%

The team working on User-Friendliness, needed each of the 7 requirements (2 shown) to develop it. They used their own IET to evaluate a long list of potential Solutions to meet their 7 User-Friendliness Requirements. The person responsible for the whole Client Side, could manage his part with the summary of User-Friendliness Requirements, and 6 other Requirements.

Availability Breakdown and Summary

On the Server Side (where the backend to the shop and other updates resided) a system breakdown would disturb thousands of users, therefore Availability was identified as critical. Availability (% uptime) can be computed from Reliability (how often it breaks down) and Maintainability (how long it takes to fix it after it breaks down).

Sys-Availability

Scale: % of time when the server is available to all users and working properly.

Past [server] 97,78 %

Goal [server] 99,73 %

Sys-Availability can be broken down into

Sys-Reliability

Scale: Mean time between system fails to be available to all users or fails to work properly.

Past [server] 60 days

Goal[server] 182 days "about 2*per year."

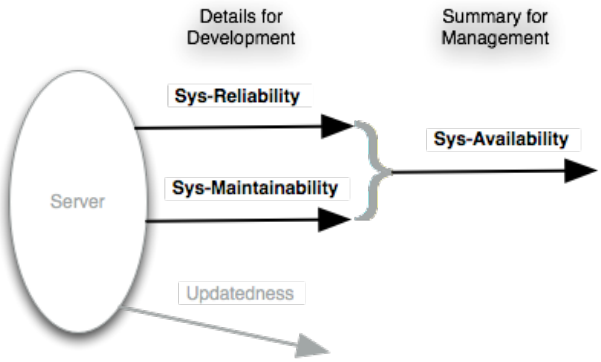
Sys-Maintainability

Scale: Mean time to repair after Sys-Reliability failure.

Past [server] 2 days

Goal[server] 0,5 day

To compute Sys-Availability,
I find the Total-Time by multiplying the two Goal levels.
182 + 0,5 = 182,5 days Total-Time.
Divide Sys-Maintainability Goal level (downtime), with Total-Time to get % Downtime.
0,5 / 182,5 = 00,27% Downtime.
Sys-Availability = 100% - 00,27% = 99,73%.



If I need to improve Sys-Availability with just 1,95%, I need drastic improvements in either or both of Sys-Reliability and Sys-Maintainability. In this example 300% improved Sys-Reliability and 400% improved Sys-Maintainability makes up the 1,95% improvement from Past to Goal levels of Sys-Availability.

Needing to improve Sys-Maintainability 400%, I would divide Sys-Maintainability into distinct tasks and analyze the clock time they consume.

Sys-Maintainability breakdown (Past level):

Time from problem occurs	
Time to detect that the problem has occurred	1 day
Time to report problem to maintenance	30 min.
Time to administer maintenance friends	3 hours
Time to collect tools for repair	30 min.
Time to find the problem	1 hour
Time to analyze the cause of the problem	2 hour

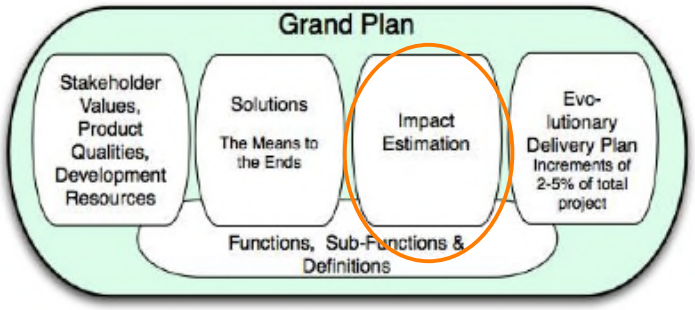
Time to fix and implement fix	1 hour
Time to quality control the fix	1 hours
Time to re-start the system	15 min.
Time to make sure the system is rid of the problem	30 min.

I find that many developers focus only on the technical task (in this example that would be from "Time to find the problem"), but often the technical task is only a small part of the effective time from a User or other Stakeholder perspective (in this example 5.75 hours out of 2 days). Often I find big wins for little effort in the non-technical tasks, sometimes because the people before me were mainly focused on the technical part. When I set Requirements, including other types than Availability, I have the Stakeholders in mind, see it from their perspective.

With the time breakdown of Sys-Maintainability, I can analyze what takes up more time, and where we have the biggest improvement potential. Then I can make a practical Evolutionary plan to improve Sys-Maintainability, thereby also improving Sys-Availability. My first Evolutionary cycle could be to attack the "Time from problem occurs, until, Time to detect that the problem has occurred = 1 day."

And for readers that have a hard time seeing what to do during the first Evo Cycles, when the new system does not even have a foundation, "nothing exists", notice that I could attack this task, without first building the new server. It is a perfect example of a Product Quality that can be improved from day 1, on the very first Evo Cycle.

Advanced IET



So far, I have shown the use of Impact Estimation Tables, IETs, using fundamental structure and information.

	Solutions	2	3	4
	Short-Cut.Name	Buttons.Rubber	Frame.Flash	???
Product Quality Requirements	units % impact	units % impact	units % impact	units % impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10 33%	-4 13%	-20 67%	0 0%
Reliability 100 200 by one year	-5 -5%	10 10%	60 60%	0 0%
Development Resources	units % impact	units % impact	units % impact	units % impact
Project-Budget 0 100000 by one year	10000 10%	10000 10%	50000 50%	0 0%

IET: an example of a basic Impact Estimation Table. Product Qualities and Development Resources on the left side, and Solutions horizontally on the top, with the impacts of the Solutions on the Product Qualities in the middle expressed as a percentage from Past or Status levels to Goal levels. This structure is more or less always the same, but more information can sometimes be usefull.

For many situations, the basic IET is all that is needed to communicate and get the necessary insights to make intelligent development choices. With the fundamental IET structure in place, you can make it even more insightful by adding information and by doing simple calculations on the information.

Choosing the optimum Solutions or Evo Cycles, or, Bang for Bucks

In the Evo chapter, I showed how we use the IET as an Solution Comparison Table for selecting the Evo Cycle that gives the most Bang fro the Bucks. We can Use the IET to compare all sorts of Solutions, not just Evo Cycles.

Each Solution impacts each Product Quality, either positively, no impact or negatively. The Product Qualities (or Stakeholder Values) are expressed using different units in their Scales. We can not directly add the benefits from two Scales with different units, one Scale using minutes another using transactions per seconds.

We can add the benefits of the % impact from Past to Goal levels. Having normalized the impacts, in % improvement from Past to Goal levels, we can add the % benefits one Solution has across Product Qualities with different units.

	Solutions		2		3		4	
	Short-Cut.Name		Buttons.Rubber		Frame.Flash		???	
Product Quality Requirements	units	% impact	units	% impact	units	% impact	units	% impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10	33%	-4	13%	-20	67%	0	0%
Reliability 100 200 by one year	-5	-5%	10	10%	60	60%	0	0%
Sum of Impacts on Product Quality Requirements		% impact		% impact		% impact		% impact
		28%		23%		127%		0%
Development Resources	units	% impact	units	% impact	units	% impact	units	% impact
Project-Budget 0 100000 by one year	10000	10%	10000	10%	50000	50%	0	0%

IET: Sum of Benefits: Short-Cut.Names has a 33% impact on User-Friendliness.Learn.Contacts and a -5% impact on Reliability.
33% + -5% = 28% total impact on my specific Product Quality Requirements. It helps me achieve 28% / 2 Product Quality Requirements = 14% of the way towards my set of Product Quality Goal levels.

When I compare the 'Sum of Impacts on Product Quality Requirements' each Solution gets, I see that **Frame.Flash** gives me the most forward motion towards my Product Quality Goal levels.

Since Development Resources are critical in my evaluation, I can do the same thing with them.

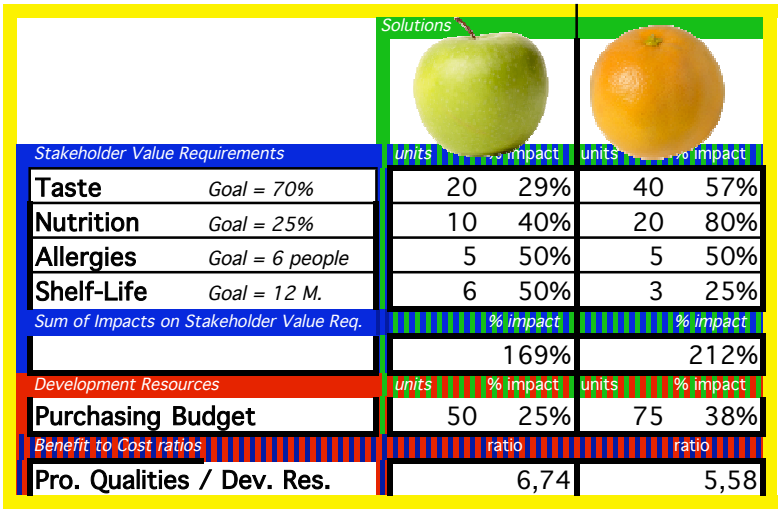
Then I divide the 'Sum of Impacts on Product Quality Requirements' on the 'Sum of Drain on Development Resources' and get a ratio comparing the Solutions. The higher the ratio a Solution gets, the more that Solution satisfies my Product Quality Requirements compared to how much Development Resources it consumes.

	Solutions		2		3		4	
	Short-Cut.Name		Buttons.Rubber		Frame.Flash		???	
Product Quality Requirements	units	% impact	units	% impact	units	% impact	units	% impact
User-Friendliness.Learn.Contacts 35 5 by one year	-10	33%	-4	13%	-20	67%	0	0%
Reliability 100 200 by one year	-5	-5%	10	10%	60	60%	0	0%
Sum of Impacts on Product Quality Requirements		% impact		% impact		% impact		% impact
		28%		23%		127%		0%
Development Resources	units	% impact	units	% impact	units	% impact	units	% impact
Money-Budget 0 100000 by one year	10000	10%	10000	10%	50000	50%	0	0%
People-Budget 0 20 by one year	0,1	1%	0,5	3%	2	10%	0	0%
Sum of Drain on Development Resources		% impact		% impact		% impact		% impact
Development Resources		11%		13%		60%		0%
Benefit to Cost ratios		ratio		ratio		ratio		ratio
Product Qualities / Development Resource		2,70		1,87		2,11		1,00

IET: Benefit to Cost: Even though Frame.Flash clearly gives the most impact towards my Product Quality Requirements (127% of 200%), it also consumes a lot more Development Resources (60% of 200%) than the other two Solutions. 127% / 60% gives me a ratio of 2.11.
Short-Cut.Names, only gives me 28% impact towards my Product Quality Requirements, but it consumes only 11% (of 200%) of my Development Resources. 28% / 11% gives me a ratio of 2.70. Short-Cut.Names gives me the most for my Bang for my Buck, followed by Frame.Flash then Buttons.Rubber.

Notice that if you only have one Product Quality Requirement, the total % impact to reach your Goal level = 100%. If you have two, it = 200% etc. The same is true for the Development Resources. If one Solution consumes 100% of one Development Resource (Money), but you have one more Development Resource (People), you still have 100% (of 200%=50% of total, or 100% of People) of your Development Resources left.

How to compare Apples and Oranges.



Stakeholder Value Requirements		Solutions			
		units	% impact	units	% impact
Taste	Goal = 70%	20	29%	40	57%
Nutrition	Goal = 25%	10	40%	20	80%
Allergies	Goal = 6 people	5	50%	5	50%
Shelf-Life	Goal = 12 M.	6	50%	3	25%
Sum of Impacts on Stakeholder Value Req.			% impact		% impact
			169%		212%
Development Resources		units	% impact	units	% impact
Purchasing Budget		50	25%	75	38%
Benefit to Cost ratios			ratio		ratio
Pro. Qualities / Dev. Res.			6,74		5,58

IET: Apples & Oranges: Solution Comparison Table: When buying fruit, one evaluates, in ones own head, the Product Qualities of the fruits (Solutions), selecting what best satisfy the needs of the family members (Stakeholder Values). In the above evaluation, I have done this evaluation using an IET. Oranges gives me more movement towards my families Stakeholder Value Requirements than Apples does (212 vs. 169). However, Oranges also costs substantially more, consequently Apples delivers a little more Stakeholder Value compared to the drain on my Purchasing Budget. I will have 7 of each thank you!

We certainly can compare different products or Solutions based on their qualities and how well they satisfy a set of Stakeholder Value Requirements.

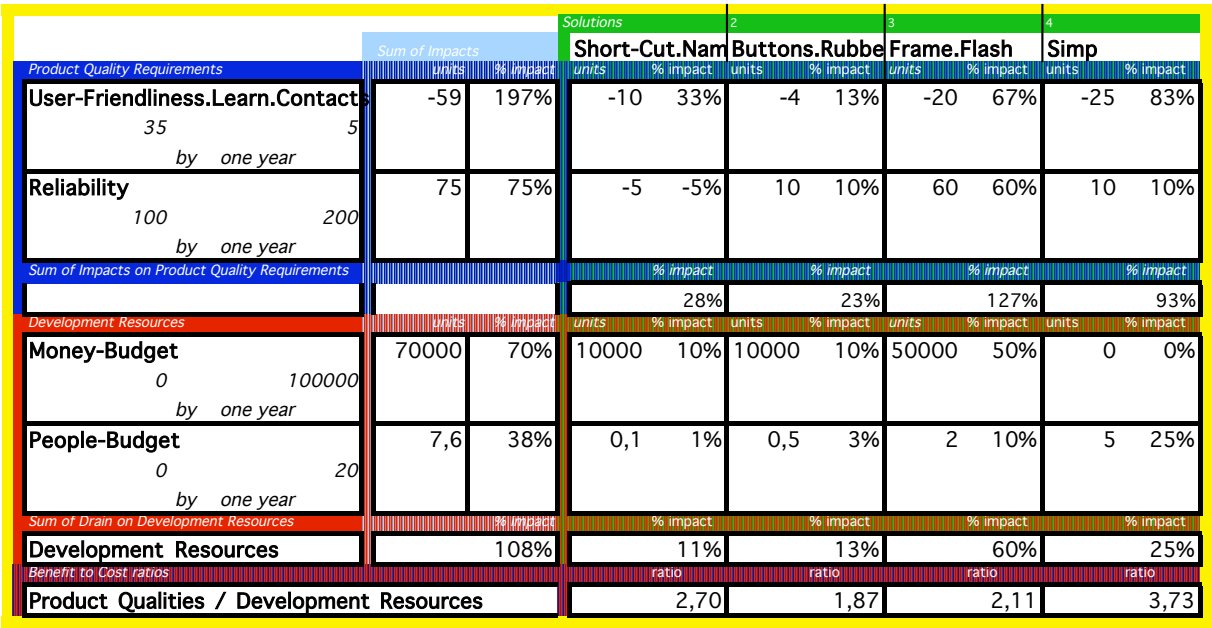
During an Evolutionary Delivery project I constantly use IETs to compare and choose which Solutions to choose and which Evolutionary Cycles to do next.

When my clients needs to compare and choose solutions/strategies/products/directions/designs/suppliers/outsourcing companies etc. I usually help them evaluate their different options using IETs, a process that forces them to agree and write down their Stakeholder Value or Product Quality Requirements in a clear unambiguous way (read quantified), and think systematically about their options and how it effects their Requirements. Usually, unparalleled clarity results from this process.

‘Sum of Impacts’, or, With a set of Solutions, are there weaknesses? Which Product Quality Requirements will not be met? A balancing act.

If you have 30 Solutions to satisfy your seven Product Quality Requirements, you will want to make sure that all your seven Product Quality Requirements gets met.

I use the IET to put together a set of Solutions, that together give a balanced impact on the Requirements. Making sure no Requirement gets too little or unnecessary much impact. We add together the impacts the set of Solutions have on each Requirement, and come up with a number I call ‘Sum of Impacts’.



	Sum of Impacts		Solutions		2		3		4	
			Short-Cut.Nam		Buttons.Rubbe		Frame.Flash		Simp	
Product Quality Requirements	units	% impact	units	% impact	units	% impact	units	% impact	units	% impact
User-Friendliness.Learn.Contacts 35 5 by one year	-59	197%	-10	33%	-4	13%	-20	67%	-25	83%
Reliability 100 200 by one year	75	75%	-5	-5%	10	10%	60	60%	10	10%
Sum of Impacts on Product Quality Requirements			% impact		% impact		% impact		% impact	
				28%		23%		127%		93%
Development Resources	units	% impact	units	% impact	units	% impact	units	% impact	units	% impact
Money-Budget 0 100000 by one year	70000	70%	10000	10%	10000	10%	50000	50%	0	0%
People-Budget 0 20 by one year	7,6	38%	0,1	1%	0,5	3%	2	10%	5	25%
Sum of Drain on Development Resources		% impact	% impact		% impact		% impact		% impact	
Development Resources		108%		11%		13%		60%		25%
Benefit to Cost ratios			ratio		ratio		ratio		ratio	
Product Qualities / Development Resources				2,70		1,87		2,11		3,73

IET: Weaknesses: When adding up the impacts our set of four Solutions have on User-Friendliness.Learn.Contacts, we get the number 197%, listed under 'Sum of Imapcts'. This does not indicate that we will get a result twice as good as our Goal level, but gives an indication and some confidence that we have some ideas for how to satisfy that Requirement.
When adding up the impacts our set of four Solutions have on Reliability, we get the number 75%. This can be cause for attention. Can we tune the current Solutions to better satisfy Reliability, or do we need additional Solutions to satisfy it? We still have additional Development Resources, Money-Budget (70% used) and People-Budget (38% used), to invest on the Solutions.

When adding up the impacts of a set of Solution towards one Requirement, we get a number I call 'Sum of Impacts. In reality, the impacts from Solutions, does not add together. If one Solution takes us 50% of the way towards the Goal level, and a second Solution also takes us 50% of the way, we will not get 100% of the way to the Goal level. In reality there are a whole set of positive and negative synergies as well as overlaps. Reality might be that the two Solutions combined gives us 50%+50%=3% or =300% towards the Goal level. Nevertheless, as long as one is aware of the limitations, the 'Sum of Impacts' gets us an indication of strength and weaknesses of our set of Solutions. I find it especially useful in finding weaknesses.

If one Requirement's 'Sum of Impacts' adds up to 50%, and another to 450% it is indicative of an unbalance, too many of the Solutions are satisfying one Requirement.

To somehow compute the actual combined impact two or more Solutions will have on a Requirement is at best extremely time-consuming, and more realistically impossible. To best deal with the reality of how Solutions interact, I recommend implementing the Solutions with Evo, and measuring the effects. It is cheaper, faster and more accurate than any calculation I know of.

Safety Level

A Safety Factor used with IETs is how much over 100% the 'Sum of Impacts' adds up to.

That is, a Safety Factor of 2 or 3, means the 'Sum of Impacts' should add up to at least 200% or 300%.

Having Solutions in your IET does not mean that you need to implement them at great costs. Let's say you have Solutions that together add up to 300% towards meeting a Performance Goal. When the Performance Goal level is actually reached, the Solutions you had listed in the IET will score 0%, they will loose their priority and never have to be implemented. If you on the other hand implement a few Solutions that you thought would take you to 100% of Goal satisfaction, but they only took you to 67%, then you will appreciate that you have some more Solution ideas that can take you the rest of the way.

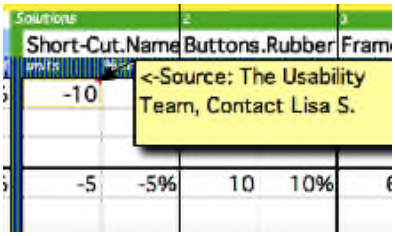
Someone, like an architect, a project manager or managers through a policy, can demand a Safety Factor on the 'Sum of Impacts'. It can be set across all Requirements, and/or separate Safety Factors can be set for each Requirement. See subchapter IET-Safety for how a Safety Factor can be specified for each Requirement.

Many of my clients use a safety factor of 2 or 3.

What is the Source of the estimate? Do we have any Evidence?

<-Source:

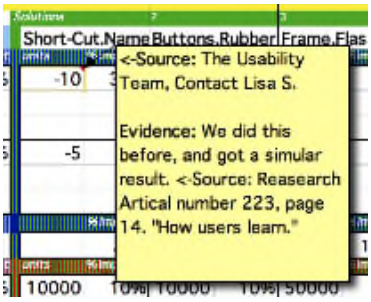
Source is information about where a number came from. Just like any other statement in a project plan, I highly recommend specifying the source of an estimated impact or any other number. It can be a person, team, or a written document. It is important to give enough detail so it is practical for a reader to find and quality control the number. Referring to a document with tens or hundreds of pages is almost as useless as stating no source at all. Give the version number, page number or unique heading or a person with contact information. The minimum level of source should be the person writing the number.



IET: Source: In MS Excel, one can add a "Comment" to each cell. I use this "Comment" field to display the Source of the impact. When I use paper or another tool that does not have such a comment field, I simply specify the Sources on a separate page.

Evidence:

Evidence is factual, what actually happened, not what somebody believes will happen. We want a reference to what actually happened, when we or somebody else, did a similar Solution. We might not have any evidence, and then we should state that.



IET: Evidence: Evidence can be placed together with the source. It is natural that the evidence itself has a source, the source of the evidence.

More Information about the Impact - Experience Level and ± Variation

The ± Variation is an estimate for how large the spread of the impact will be. The Experience Level is a rating on how much experience exists of using similar Solutions and its impact on similar Requirements.

± Variation indicate the expected variation the impact a Solution has on a Requirement. One might for example estimate an impact to be 10%±10% or 10%±1%.

We can rate the level of experience we have with each specific Solution, and its impact on each specific Requirement. The resulting table gives us a good view of how much experience we have, or do not have, related to the Solutions we have chosen.

Example Experience Level Table

Rating Meaning

0.0 Wild guess, no experience.

0.1 We know it has been done somewhere.

0.2 We have one measurement somewhere.

0.3 There are several measurements in the estimated range.

0.4 The measurements are relevant to our case.

0.5 The method of measurement is considered reliable.

0.6 We have used the method in-house.

0.7 We have reliable measurements in-house.

0.8 Reliable in-house measurements correlate to independent external measurements.

0.9 We have used the idea on this project and measured it.

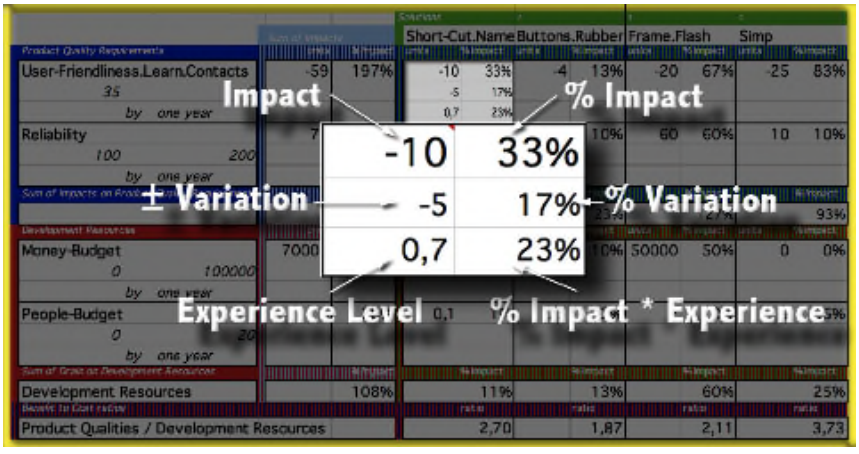
1.0 Perfect experience, we have rock solid, contract guaranteed, long-term, credible experience with this idea on this project and, the results are unlikely to disappear.

Product Quality Requirements	Average experience level	**Short-Cut.Names** experience level	**Buttons.Rubber** experience level	**Frame.Flash** experience level	**Simp** experience level
User-Friendliness.Learn.Contacts	0,425	0,7	0,5	0,2	0,3
Reliability	0,375	0,3	0,5	0,4	0,3
Average of Impacts on Product Quality Requirements	0,4	0,5	0,5	0,3	0,3

Development Resources	experience level	experience level	experience level	experience level	experience level
Money-Budget	0,6	0,7	0,5	0,7	0,5
People-Budget	0,325	0,7	0,3	0,2	0,1
Average of Drain on Development Resources	0,4625	0,7	0,4	0,45	0,3

IET: Experience: In this table, I have rated the Experience Level of each impact. It clearly tells me that the team has little facts or experience behind the estimated impacts of Simp and Frame.Flash. Short-Cut.Names is the Solution that have the highest Experience Level.

The Experience Level gives information about one type of risk, the experience or lack thereof. It is informative by itself. It can also be multiplied with the estimated % Impact to give a conservative estimated % impact.



IET: Experience & Variation: It is estimated that this Solution will move the Requirement 33% from the Past level towards the Goal level. The ± Variation is estimated to be 17%. The Experience Level is set at 0,7. Multiplying the Impact of 33% with the Experience Level of 0,7, a conservative Impact of 23% is achieved.

Typically, with a low Experience Level, comes a high ± Variation. A Solution with a high Experience Level (we have done and measured it many times before), might also have a high ± Variation (each time we did it, we got widely different results).

For each impact, it can be useful to state, the expected ± Variation, as well as the Experience Level, and to multiply the Experience Level with the % Impact.

		Sum of Impacts		Solutions 1 Short-Cut.Name		2 Buttons.Rubber		3 Frame.Flash		4 Simp	
Product Quality Requirements		units	% impact	units	% impact	units	% impact	units	% impact	units	% impact
User-Friendliness.Learn.Contacts		**-59**	**197%**	**-10**	**33%**	**-4**	**13%**	**-20**	**67%**	**-25**	**83%**
35	*5*	-42	140%	-5	17%	-2	7%	-15	50%	-20	67%
by	*one year*	0,4	68%	0,7	23%	0,5	7%	0,2	13%	0,3	25%
Reliability		**75**	**75%**	**-5**	**-5%**	**10**	**10%**	**60**	**60%**	**10**	**10%**
100	*200*	21	21%	-1	-1%	-5	-5%	20	20%	7	7%
by	*one year*	0,4	31%	0,3	-2%	0,5	5%	0,4	24%	0,3	3%
Sum of Impacts on Product Quality Requirements				% impact		% impact		% impact		% impact	
	Sum Impact				**28%**		**23%**		**127%**		**93%**
	Sum ± Variation				16%		2%		70%		74%
Average Experience	Sum Conservative Impact			0,5	22%	0,5	12%	0,3	37%	0,3	28%
Development Resources		units	% impact	units	% impact	units	% impact	units	% impact	units	% impact
Money-Budget		**70000**	**70%**	**10000**	**10%**	**10000**	**10%**	**50000**	**50%**	**0**	**0%**
0	*100000*	24000	24%	1000	1%	3000	3%	20000	20%	0	0%
by	*one year*	0,6	93%	0,7	13%	0,5	15%	0,7	65%	0,5	0%
People-Budget		**7,6**	**38%**	**0,1**	**1%**	**0,5**	**3%**	**2**	**10%**	**5**	**25%**
0	*20*	5,4	27%	0,1	1%	0,3	2%	1	5%	4	20%
by	*one year*	0,3	70%	0,7	1%	0,3	4%	0,2	18%	0,1	48%
Sum of Drain on Development Resources			% impact	% impact		% impact		% impact		% impact	
	Sum Impact				**11%**		**13%**		**60%**		**25%**
	Sum ± Variation				2%		5%		25%		20%
Average Experience	Sum Conservative Impact			0,7	14%	0,4	19%	0,5	83%	0,3	48%
Benefit to Cost Ratios				ratio		ratio		ratio		ratio	
Sum Benefit / Sum Resources					2,7		1,9		2,1		3,7
(Sum Benefit - Sum ±) / (Sum Resources + Sum Res. ±)					1,1		1,3		0,7		0,4
*(Sum Benefit * Credibility) / (Sum Resources * Credibility)*					1,9		2,3		1,4		3,7
*(Sum Benefit * Credibility - Sum±) / (Sum Res. * Credibility + Res.±)*					-0,2		1,1		-0,6		-1,7

IET: Detailed: In this IET, there are a lot of details, too much for many people. I recommend to first use the basic IETs, and then, in time, add more information and calculations to their IETs as they find it useful. Only the white fields needs manual input and adjustments, all other fields are calculation or in the case of the Requirements, linked. During a project, the numbers are constantly changing. It is therefore highly recommended that a spreadsheet application like Microsoft Excel or StarOffice™ Calc is used to do all the calculations for you.

When I read this IET, I see that:

1. it is unlikely that we will meet my Reliability Requirement with this set of four Solutions. The Estimated Impacts add up to 75±21%, and with an average Experience Level of 0,4.
2. User-Friendliness.Learn.Contacts first look good with 197% estimated impact, but the ± Variation of 140% indicates that it might be as bad as (197%-140%) 57%. Then factor in the Experience Level of 0,4 and I am not so confident about it anymore.
3. The good news is that we still have Development Recourses to our disposal, some Money-Budget and especially People-Budget. We can use these Development Resources to develop additional Solutions.
4. The Experience Level regarding the drain on Money-Budget is relatively high, but for People-Budget it is relatively low.
5. If all goes well, and our non adjusted estimates goes approximately according to plan, Simp seems like a winning Solution with a ratio of 3,73, then Short-Cut.Names, Frame.Flash and Buttons.Rubber follows.
6. The conservative estimations reveal that the Solution Simp has a large ± Variation.
7. It is estimated, that the Solution Simp, will not consume any Money-Budget Development Resources.

Formulas for advanced IET.

% Impact = Impact / (Goal – Past)

% Variation = ± Variation / (Goal – Past)

Conservative Impact Estimation for Stakeholder Values & Product Qualities = % Impact * Experience Level

Conservative Impact Estimation for Development Resources = -((Experience Level-1)-1)*% Impact

Where can the use of Impact Estimation Tables be useful?

To make decisions about Solutions on one level, based on the Requirements from the level above, I use Impact Estimation Tables (IET)

If I have a project, that begins with the company stockowners requirements at the first level, and I need to make decisions about what Stakeholder Values at the next level down (Users etc.) I will satisfy, I will use an IET to help me see and evaluate the connection between the two levels.

If I have Stakeholder Values, like Users, and need to make decisions about what Products, and Product Qualities to best satisfy the Stakeholder Values, I will use an IET to help me see and evaluate the connection between the two levels.

And this decision process might go on from one level to the next, between each level I would use an IET. When I am at a level where I will no longer want to make decisions about what to do at the next level, I just want to actually do it, I no longer need an IET.

To select one of many potential Solutions, I use a comparative IET.

Often decisions has to be made, whether to use one technology or another, one supplier or another, one process or another, to satisfy one customer or another, to use one Solution or another, or to pick one Evolutionary cycle or another. When complex decisions are made, I often recommend the use of an IET to evaluate the choices up against the set of Requirements they are intended to satisfy. I view IET as a fantastic decision making tool for complex issues.

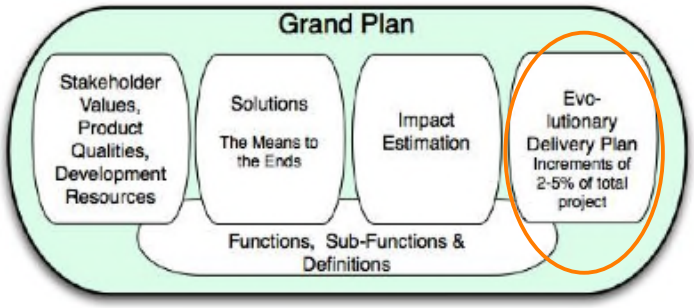
To estimate and track progress in Evolutionary Projects I use Evo IET.

Once I have found the next Evolutionary Cycle to implement, I use an Evo IET to estimate and track progress towards the Requirements. In addition to spelling out the benefits that is expected from each Evo Cycle, the Evo IET also keeps track of the actual results achieved. An Evo IET is a great tool that can teach us which Solutions in actual fact work in practice and which don't.

> The seers, the seen, and the process of seeing all merge. The knowledge, the knower, and the known, they all merge, become one---that is Divine Love.
>
> Sri Sri Ravi Shankar, Directing All Passions to the Divine

Using the Impact Estimation Table, the Ends, the Means, and the Development Resources all merge. The Stakeholders Requirements, the engineers Solutions, and the accountant Development Resources all merge in an understandable, logical, simple straightforward way. That is divine project management.

Advanced Evo



Back-room front-room, or, You want apple pie, sure just wait a while.

Some things might take more than one Evo Cycle to develop, but that does not stop anyone from delivering value to Stakeholders early. We can distinguish between Evolutionary Delivery Cycles, the time frequency of delivering improvements to Stakeholders, and Evolutionary Development Cycles, the time it takes to develop something, that later will be part of a Evolutionary Delivery Cycle.

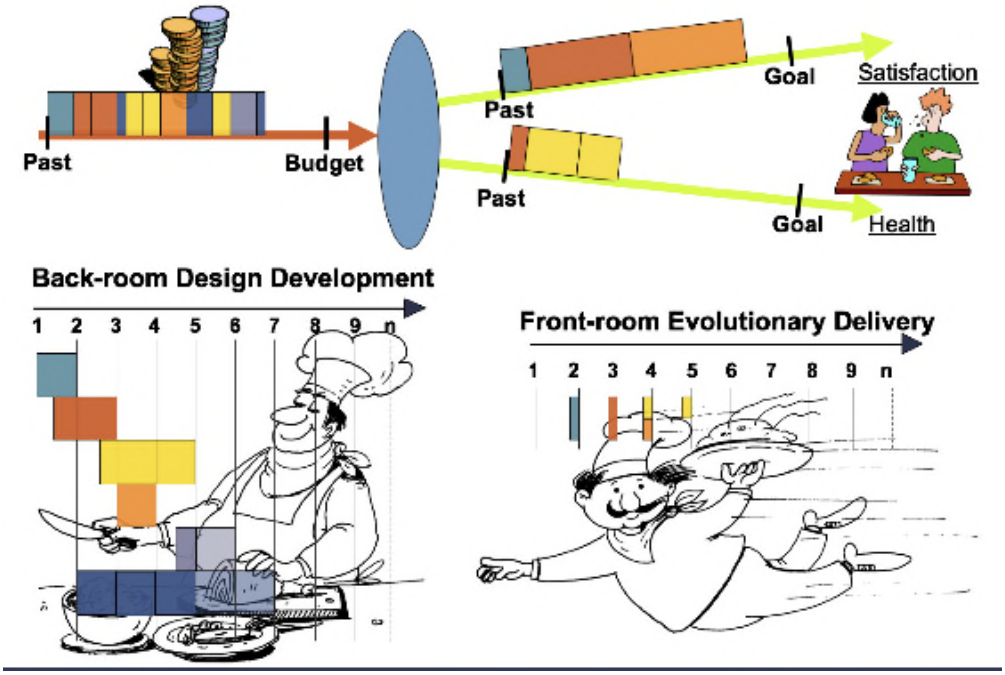


Illustration: Evolutionary Delivery Cycles (Evo Cycles) focuses on delivering something (Satisfaction & Health) every cycle. In many projects I work with, the Delivery Cycle contains the Development Cycle. But in some projects, there might be development that is seen as either impossible or as impractical to divide into short cycles. Then a back-room development cycle that is longer than the delivery cycle can be used. Much like food takes a while to grow and cook, but delicious dishes can be delivered quickly to a waiting customer.

The risk with the use of the back-room, is that many of the benefits that can be gained from forcing oneself to find shorter Development Cycles are lost. If a back-room development cycle takes 3 months, and it is a failure, have that project lost more than it can recover? Will the project still be able to deliver the Requirements within the Development Resources. If the answer is “no!” I would somehow find a way to divide the back-room activity into smaller Development Cycles. People tend to give up to easily on the challenge of finding a way to break the Development Cycle into smaller cycles. In general, I find that it is

always possible. Only when the development is easy, and contains very little risk would I consider a long Development Cycle.

Most systems can be viewed as having several back-room Development Cycles, but usually then they also have several Delivery Cycles. When developing an airplane, many companies develop sub-systems to that airplane. Each sub-system developer can be seen as a back-room development for the whole airplane.

Cycles in Cycles, or, boy it is stormy

<<So we have learned to divide anything into small deliveries, but we have some things that we just don't want to deliver to Stakeholders before they are more mature! Fine, ……..

Regular Stakeholder deliveries that goes to select Stakeholders, i.e. Every week!

Internal builds that need more time to develop and maturity, can be set up on its own internal delivery cycles, and after maturity can be one regular stakeholder delivery. The mistake often done is to not divide the internal delivery into small cycles, thereby falling back to old habits with poor results.

> In every little mind, different, different thoughts come; and different, different moods are there. But, when we sing, what happens? All the minds have the same rhythm, the same song, same waves, same frequency, and there is so much more joy.
>
> Sri Sri Ravi Shankar, Listen and Celebrate

Our projects starts from within our minds, different thoughts and experiences. In the Requirement specifications we synchronize our Ends. With the Impact Estimation Table, we synchronize our thoughts of the effect our Means will have on the Ends. And in the Evolutionary Delivery phase, we synchronize our efforts to deliver the Ends. Just like singing synchronizes our minds, Planguage synchronize the minds and the action of the people involved in our project.

> Let's assume we are creating a new mobile phone, even better than the one we are currently selling. To get better qualities in the phone we have concluded that we need to switch the software architecture to a newer better one.
>
> In this example many people would incorrectly assume that they have to spend a long time to create the foundation before they can start any kind of Evolutionary development.
>
> ……..

Let's assume we are creating a completely new mobile telephone like device with video, sound smell and touch. What we do is specify the Stakeholder values

> I'm not saying, "Don't be active." I'm saying, observe the spontaneity of action.
>
> Sri Sri Ravi Shankar, Listen and Celebrate

Evolutionary Delivery gives us the freedom to act spontaneously to reality, as is so much hindered in waterfall type projects.

"I agree in principle, but it will not work on my project!"

Next Slice

---

1. Don’t take a bigger bite than you can chew.
2. Don’t serve yourself more than you can eat
3. When served new food, eat a small sample first.

What is missing?

> When you listen, listen to something more. Not just to the sound, but to the silence, too.
>
> Sri Sri Ravi Shankar Listen and Celebrate

When we manage our projects, use Evolutionary Delivery to listen, not only to what we have done, but to what we have achieved, what have changed. The use of measurable Requirements is a great way to listen. Listen also to our Stakeholders, is there some missing Requirements?

> Your forgiveness should be such that the person who is being forgiven does not even know that you are forgiving them. They don't even feel guilty of a mistake. That is the right type of forgiveness.
>
> Sri Sri Ravi Shankar

Our Evolutionary Cycle should be so small that if it goes wrong, we get nothing from it, it does not matter, as we have plenty of time to get it right.

A unexpected Requirement is threatening the project!

Examples

Medtronic

Ericsson

Kirkens nødhjelp

This book development

The imagined perfect model (Environmental)

UK Highways

Broadway play

Cray

NCR

Concept Glossary

Evolutionary Delivery Cycle (Evo Cycle)

Evolutionary Development Cycle

Function

Sub-Function

Project Development Resources

Project Level: Stakeholder Level, Product Level, Sub-Product Level

Product Qualities

Stakeholder Values

Sub-Function see Function

Notes on the development of this book manuscript

This book started its life Friday 15 of September 1995 @ 12:14 midnight.

Book development team

Tom Gilb: Inventor of method. My teacher.

Gerard Janssen: gjanssen@xebia.com. Feedback.

+ Many other friends have given feedback at various levels.