

Combinatory Metrics for Software Development

*Aligning Software Development
with Customer's Needs*

Dr. Thomas M. Fehlmann

Euro Project Office AG
Zeltweg 50
CH-8032 Zurich, Switzerland

Phone: +41 1 253 1306
Fax: +41 1 253 1364

E-mail: Thomas.Fehlmann@e-p-o.com

V1.0-6, 22 May, 2002
© 2002 by QFD Institut Deutschland e.V.
and the author

Combinatory Metrics for Software Development

Dr. Thomas M. Fehlmann
Euro Project Office AG, Zurich, Switzerland

Summary

Combinatory Metrics is the science of defining metrics for topics that are not directly measurable. Such topics exist in abundance in the area of software development.

This paper explains how to establish a network of combinators to cover the topics for which we want to get metrics. The basic techniques are the cause – effect combinators known from QFD (“Quality Function Deployment”). The combination technique is taken from Combinatory Algebra, a theory used in computer science to understand the nature of computing with knowledge about topics.

We also present the experiences we made when applying these techniques to a medium-sized software development organization that targets a niche market. People do not necessarily like metrics, but combinatory metrics are very helpful when developing products that satisfy customer’s needs.

*Thomas M. Fehlmann, Dr. sci. Math. ETH, Euro Project Office AG,
Zeltweg 50, CH-8032 Zurich, phone +41 1 253 1306, fax
+41 1 253 1364, E-mail: Thomas.Fehlmann@e-p-o.com*

Table Of Contents

1.	Combinatory Metrics.....	4
1.1	Why metrics?	4
1.2	The problem with metrics	4
1.3	Software	4
1.4	Topics.....	5
1.5	Combinatory Metrics	6
1.6	Quality Function Deployment.....	7
1.7	Combinatory metrics and QFD	9
2.	A Practical Case Study	10
2.1	Metrics for the Software Development Model.....	10
2.2	Listening to the Customer's Voice	11
2.3	How does it translate into software development?	11
2.4	Aligning to the market using combinator.....	12
2.5	Testing software with combinator	13
2.6	Metrics Precision.....	13
2.7	Filling the Gaps	14
2.8	Case Study Metrics Overview.....	14
3.	Conclusions.....	15

Table of Figures

Figure 1:	Sample Software Development Model.....	5
Figure 2:	How much does this feature contribute?.....	6
Figure 3:	A Cause–Effect Combinator with derived metric for solution characteristics	7
Figure 4:	Comprehensive QFD – Sample use of combinator	8
Figure 5:	Sample Lanchester combinator linking competition to business processes	9
Figure 6:	Metrics the for Software Development Model.....	10
Figure 7:	Customer's Needs for a Hotel Reservation System	11
Figure 8:	CMM Assessment Result.....	12
Figure 9:	Sample testing combinator.....	13
Figure 10:	Overview of Combinatory Metrics used for the Case Study	14

1. **Combinatory Metrics**

1.1 **Why metrics?**

When working on complicated matters that are hard to understand, men always tried to use measurements to help assessing where he stands. Measurements were at the very beginning of seafaring, and already the old Greeks in the 2nd century B.C. constructed analog computers to navigate in stormy seas [20].

However, when talking about metrics in software development, enthusiasm is limited and the audience skeptical. So many well-meant approaches failed to yield the benefits people had hoped for. Technology in software development changes so rapidly that before a metric could establish itself, it was already outdated.

1.2 **The problem with metrics**

The main problem with metrics is that you need something to measure. When talking software, it is hard to find anything tangible. One of the best candidates is the Function Points method (IFPUG 4.1). Function Points are independent from the software techniques used [14].

Function Points is a good example that measurements and metrics are not the same. Function Point is an excellent way of measuring the size of software. They are therefore very suitable as metrics for software size and — more general — for the size of ICT (Information & Communication Technology) applications, and can therefore be used easily and successfully for derived metrics such as defect density and system availability and reliability, and thus whatever relates to the size of an application.

However, when we try to use Function Points to estimate the cost of developing the software, we run into a number of problems. Software development cost depend on a variety of things, such as scope definition, availability of requirements and of customers, knowledge of the application domain and — at the same time — of the logic behind software development, and other skills, development tools used, business processes in effect, and much more. At the end, the size of the generated software is then only one cause among others that contribute to development cost¹.

If we want to use Function Points to predict time and effort needed to develop the software, then we must first study the cause – effect relationships between size, time and effort in software development. To do this, we use combinatory metrics.

1.3 **Software**

For practical purpose, we use a slightly different definition for “software”. We define software as an *expert service* that is delivered at a remote location, at

¹ In a recent Web – enabled software product development project, the cost of development split as follows: 85% getting the requirements right; 15% design, coding, testing and packaging.

a time chosen by the user of the service with the help of ICT, without having the experts at hand that once prepared this service (see [6]).

This definition both captures the technical aspect of software, namely that software is running a program (sequence of executables) which manipulates data, as well as what users expect from the software part of a web service: Providing valuable services fast and with almost no dependency from the personal availability of experts, sometimes with the possibility to enter a consequential contractual business agreement². People resources that might be needed to complement such service (e.g. a help desk) are not part of software; neither are hardware and other ICT equipment that is necessary to provide such service. Consequently, *software engineering* describes the techniques needed to define such service, i.e. the various tools for defining, creating and documenting what the service provides such as design, programming, and documentation tools.

1.4 Topics

When creating software, we need to consider various aspects. We call such aspects *topics*. Topics³ are identified by the software development methodology.

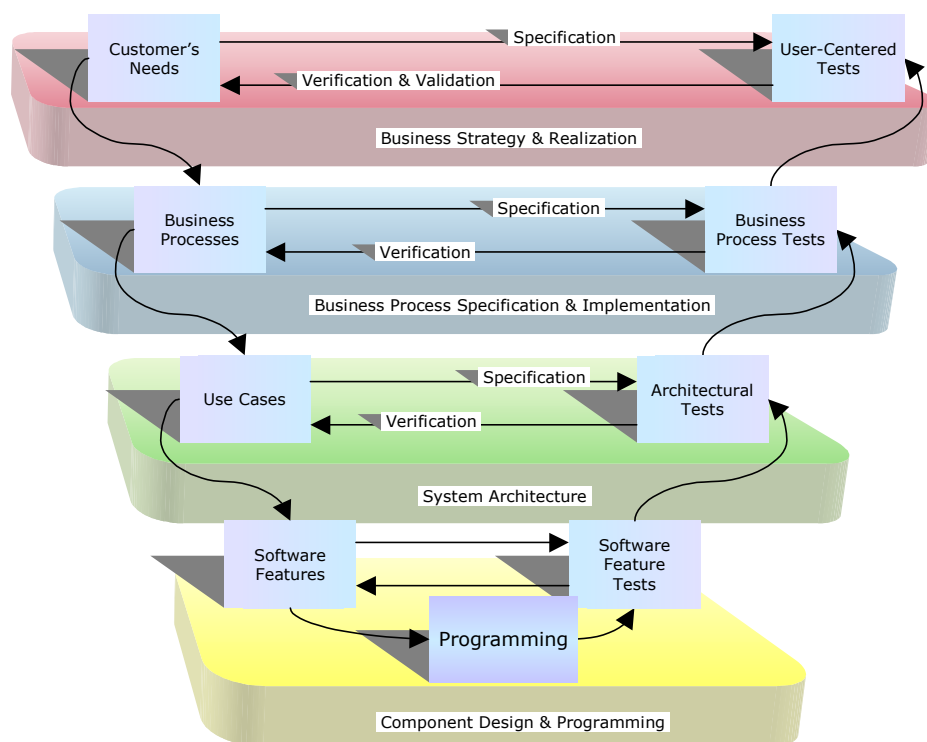


Figure 1: Sample Software Development Model

² Imagine how e-Commerce and other Web-based services could flourish if the providers would start to understand that they must excel in service quality.

³ Sometimes people use the term “views”, such as engineering view or customer’s view, instead of “topic”. However, “view” assumes that the essence of the matter in question is the same, but it is not. For instance, time-to-market and maintainability are two different topics when developing software, not different views.

As an example, we show nine topics that pertain to web software testing in the Figure 1 above.

1.5 Combinatory Metrics

Combinatory Metrics is the science of how to derive metrics for topics that cannot be measured directly, but have an impact on things we would like to know. For instance, we would like to know in advance whether adding a feature to our software would increase competitiveness. Therefore we need a metric for the software feature. Competitiveness is easily measurable: We need to look at the market share, and its variations over time. However, finding a metric that tells us the impact of a given feature on overall success in the market is hard. We have two options: Asking a representative couple of people, or do cause – effect analysis.

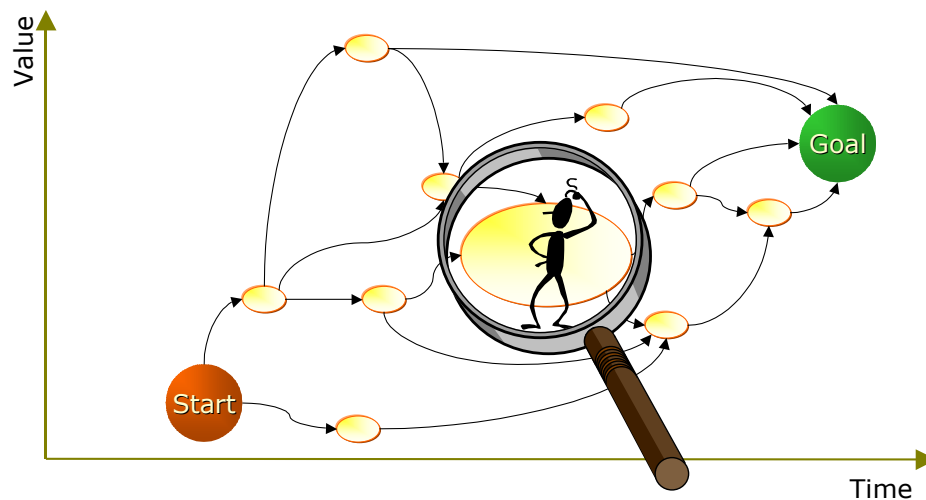


Figure 2: How much does this feature contribute?

Asking people has several inherent problems. First, we may not have a customer readily available, who is responsible and can give us his requirements and answer questions. Second, people often lack the imagination to understand the impact of a proposed feature. This is the reason, why Requirements Engineering is so hard.

On the other hand, we know quite well how to conduct a customer's needs analysis. Cause – effect analysis does transform such insight into software requirements and design choices. Moreover, the QFD Methods yields a metric for these topics, by assigning a value for its relative contribution to satisfying customer's needs.

Cause – effect analysis is not easy either, but it has the big advantage, that it is not much dependent on time. Neither technology advances, nor people's perception have much impact. Once cause – effect relations are understood, much of the hard work is done for good. Cause – effect analysis can be reused as long as the topics remain the same. What does change indeed, is the relative value of the topics

Thus the first step in cause – effect analysis may be hard, but combining the results is easy.

1.6 Quality Function Deployment

The best-known application of combinatory metrics is QFD.

QFD allows for three basic operations:

1. It describes how a solution approach contributes to solve a problem. Traditionally, the problem has been stated as how to satisfy customer's needs; the solution to it being the optimum choice of solution components that constitute the solution approach. The tools used are correlation matrices such as the famous "House of Qualities"⁴.

We call the items being correlated to each other, such as solution approaches and the problem under question, *Deployment Topics*. The target Deployment Topics we call *Goal Topics* (the left-hand side of the correlation matrix), the means describing the solution approach we call *Solution Topics* (bottom of the matrix).

We refer to these links between the Deployment Topics described by such correlation matrices as *Cause–Effect Combinators*⁵.

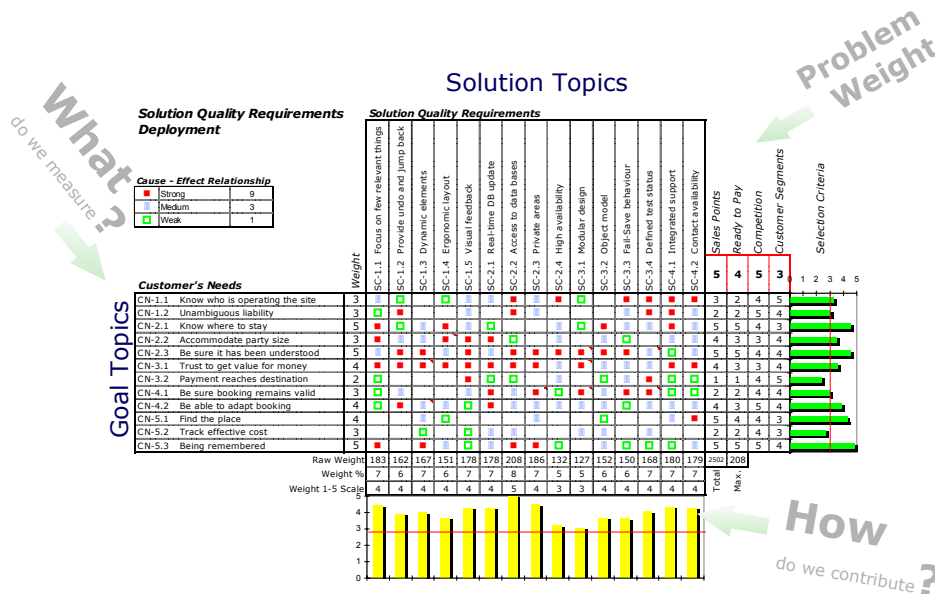


Figure 3: A Cause–Effect Combinator with derived metric for solution characteristics

Figure 3 shows a sample Cause–Effect Combinator used in the ICT project explained below, see section 2 of this article. We derive a metric for the solution topics — in this case the solution quality characteristics — from the goal topic, which consists of the customer's needs.

2. Furthermore, QFD describes how Cause–Effect Combinators $\mathcal{F} * \mathcal{G}$ may be applied to each other. The Solution Topics that support the Goal Topics of the Cause–Effect Combinator \mathcal{F} , become in turn the Goal Topics in Cause–

⁴ The "House of Qualities" is usually the first of a sequence of correlation matrices that describe the quality deployment for a given project, product, or service.

⁵ Note that there exist other approaches to cause–effect analysis that do not provide the capability of being combinators or provide metrics.

Effect Combinator \mathcal{E} . Such combinations are referred as Comprehensive QFD, or QFD in the Broad Sense [2].

Linking topics by combinator allows deriving metrics for each of the topics that contribute to the goal topic.

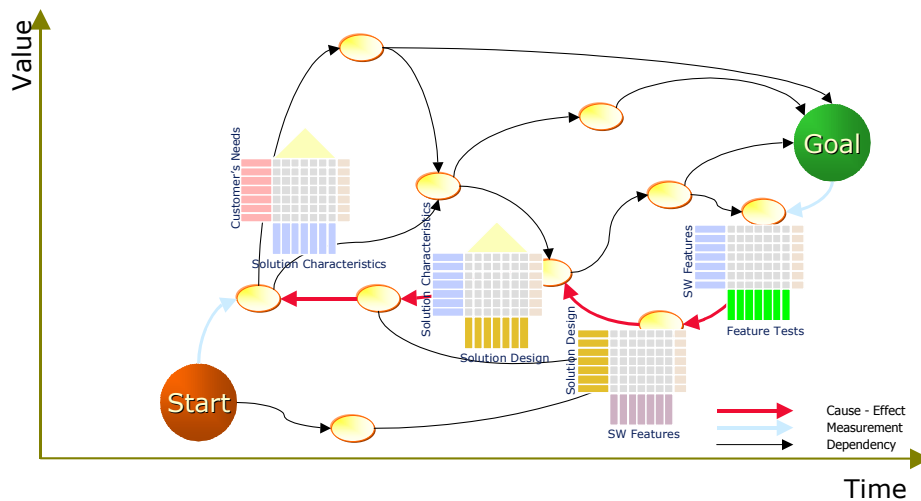


Figure 4: Comprehensive QFD – Sample use of combinator

3. Third, QFD integrates measurement methods for measurable topics. The best-known example is the Voice Of The Customer method, which normally stands at the beginning of every deployment (for example, see [10], [21], [22]).

There exist a variety of methods to measure customer's needs. This topic constitutes in most deployments the starting point for the series of combinator that finally lead to the optimization of the solution approach. Traditional QFD is based on this one measurement.

However, there are many more applicable measurement methods. Topics such as cost, market share can be measured with traditional marketing and controlling tools; scope, defect density, and other metrics related to the size of the generated software use the Function Points method. Such measurements are the crosschecks in the measurement networks.

For instance, the 2nd law of Lanchester [17] allows to link various product characteristics to market share. Using this law, one can predict the outcome of clients' product evaluations and therefore predict the attainable market share. The Lanchester combinator is similar to the cause – effect combinator introduced above, however, it calculates two directions: One is the cause-effect correlation, how well the competition does with satisfying customer's needs, or delivering the wanted solution characteristics. The others are the actual market share measurements.

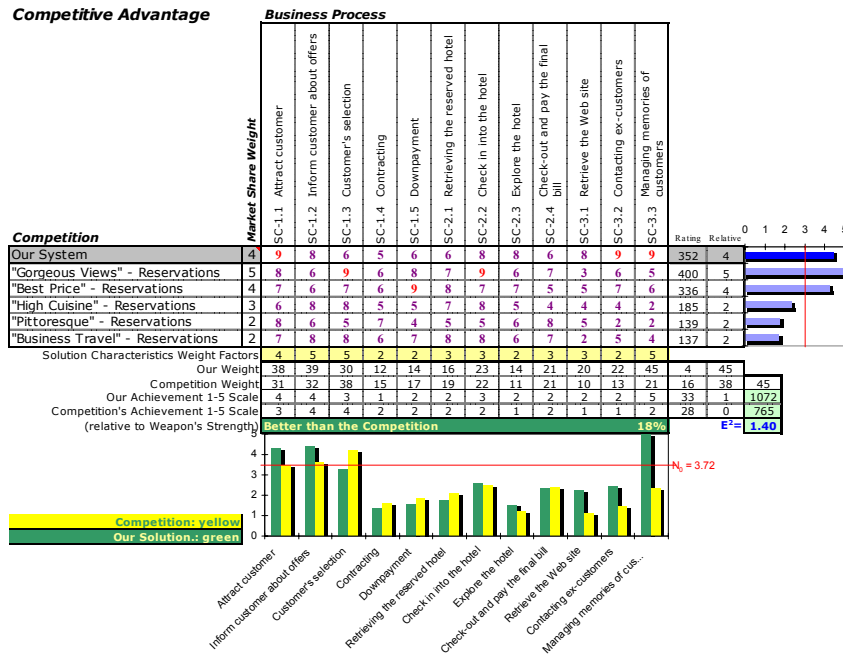


Figure 5: Sample Lanchester combinator linking competition to business processes

Using such measurements, one can establish a metrication scheme for the whole software development project. In the example in section two of this paper, we used four measurement methods to calibrate the combinatory metrics: An assessment of the software development processes according the Capability Maturity Model (CMM) criteria [11], a Voice of the Customer approach at the beginning of product development, a software size measurement using Function Points to define defect density, and two Lanchester combinators for the market share (see Figure 10: Overview of Combinatory Metrics used for the Case Study).

1.7 Combinatory metrics and QFD

Combinatory metric is an extension of QFD in two directions. First, we observe that the combinators do not only work in the direction of the deployment. A deployment answers the question, which solution mix causes the wanted effect. There is the other way round as well: If we have a solution, then the same combinator answers the question, how well this solution supports the goal topic.

This two-way calculation has been repeatedly used for combinators that link solution topics like testing and risk exposure to goal topics such as customer's needs and solution approach. The mappings are not opposite to each other. In general they have a different meaning, as shown in [7].

Another difference is that combinatory metrics are not limited to measuring topics. You can lift the combinators to a higher level and define metrics for how well an organization is capable to apply new topics to new areas [8].

2. A Practical Case Study

The following case study has been altered in business domain, in order to protect sensitive data of the customer.

A medium – sized software house builds software products for a niche market, namely Web – based hotel reservation systems for travel agencies and city tourist offices. It develops a software product for a worldwide market. It has a small but very active and highly skilled sales force.

2.1 Metrics for the Software Development Model

The metrics for the software development model as presented in Figure 1 are defined by the following structure.

First we have the classical QFD Deployment

$$\begin{aligned} & (\text{Business Process } j \rightarrow \text{Customer's Need })_i \\ & (\text{Use Case } k \rightarrow \text{Business Process })_j \\ & (\text{Software-Feature } m \rightarrow \text{Use Case })_k \end{aligned}$$

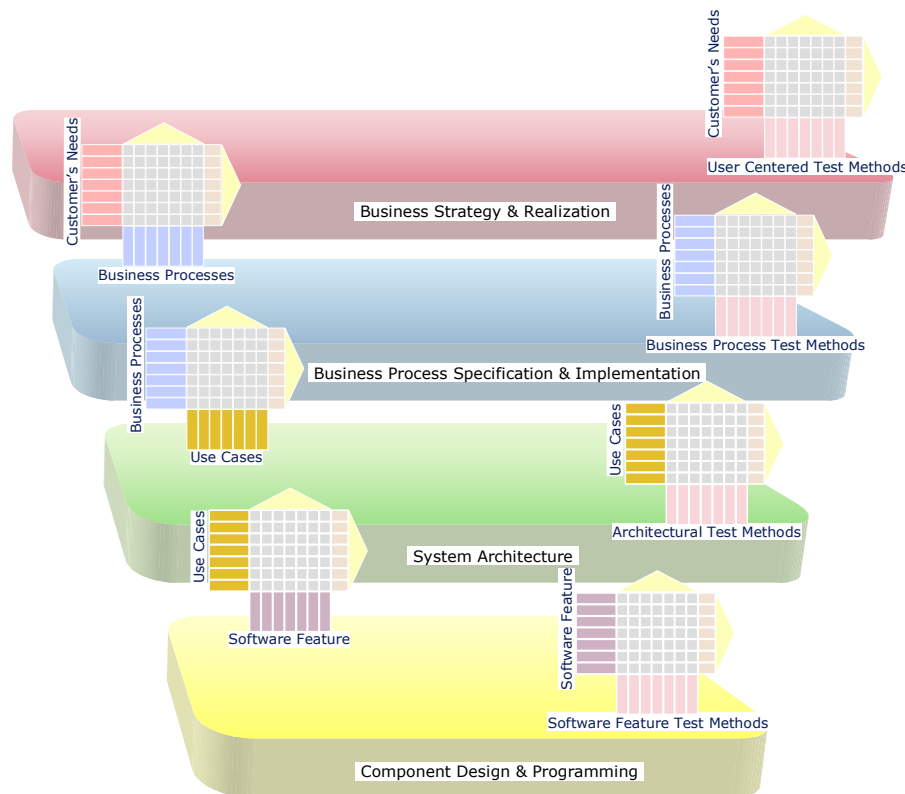


Figure 6: Metrics the for Software Development Model

On the right side, we have the following four test deployments.

$$\begin{aligned} & (\text{User-centered tests } i \rightarrow \text{Customer's Needs })_i \\ & (\text{Application Test } j \rightarrow \text{Business Process })_j \\ & (\text{Integration Test } k \rightarrow \text{Use Case })_k \\ & (\text{Feature Test } m \rightarrow \text{Software Feature })_m \end{aligned}$$

The model starts with “Customer’s Needs” as the Goal Topics of the first Cause–Effect Combinator. Such deployments have been discussed in [10], [12], [16] et.al. Thus we get combinatory metrics for the selection of the Business Processes that we want to support, of the Use Cases that are necessary for the Business Processes, and for the selection of software features. As we have seen, we need to establish calibration measurements to make the metrics useful.

2.2 Listening to the Customer’s Voice

The most useful measurement is the customer’s needs. In this case, we used a questionnaire, developed according the method of Kano[12], to establish the base for the combinatory metrics network. The questionnaire focused on the topics of needs and wants, but did also ask for their preferred solution characteristic. It was not meant to get actual solution ideas from customer, but rather to find out what their unexpressed needs are. The software developers have much better solutions in mind than the customer possibly could imagine. But they did need the voice of the customer to decide what to realize first.

The outcome⁶ of the assessment is shown in the following list:

Customer’s Needs		Explanations	0 1 2 3 4 5
CN-1 Trust	CN-1.1 Know who is operating the site	The need to know who is operating the Web business.	4
	CN-1.2 Unambiguous liability	Know how to act in case of incidents or fraud	4
CN-2 Booking	CN-2.1 Know where to stay	The need to identify the place where the hotel is.	5
	CN-2.2 Accommodate party size	Availability of suitable rooms for all the party.	4
	CN-2.3 Be sure it has been understood	Unambiguously across language and cultural borders.	5
CN-3 Pricing	CN-3.1 Trust to get value for money	Have the opportunity to compare price levels.	4
	CN-3.2 Payment reaches destination	Know where to pay and where the payment goes.	3
CN-4 Traveling	CN-4.1 Be sure booking remains valid	... and you don't arrive at midnight and won't find a bed.	3
	CN-4.2 Be able to adapt booking	Flexibility of coping with change of travel plans.	4
CN-5 Stay	CN-5.1 Find the place	Actually retrieve the hotel upon arrival.	4
	CN-5.2 Track effective cost	Get a detailed bill that identifies cost items.	3
	CN-5.3 Being remembered	Have the feeling to be highly estimated as a guest.	5

Figure 7: Customer’s Needs for a Hotel Reservation System

2.3 How does it translate into software development?

To understand the customer is one condition for success. To be able to use this understanding effectively is the next one. It was therefore felt that there is a need to assess the management processes of the company.

In order to have a second calibration measurement for the combinatory metrics network, we conducted a CMM assessment of the company. To do this, we established additional combinators

$$\begin{array}{c}
 (\text{Solution Characteristics } _j \rightarrow \text{Customer's Needs })_i \\
 (\text{Management Processes } _k \rightarrow \text{Solution Characteristics })_j \\
 (\text{CMM Criteria } _m \rightarrow \text{Management Processes })_k
 \end{array}$$

⁶ The assessment shown here is for demonstration purposes only. These are not the actual results.

This yields a metric for the importance of the CMM criteria for this particular company.

The result of the assessment was as shown in Figure 8. Note that the combinator (CMM Criteria $_k \rightarrow$ Management Processes) $_m$ is used both ways. One way is to derive the needed CMM – level for the management processes such that the market expectations can be met.

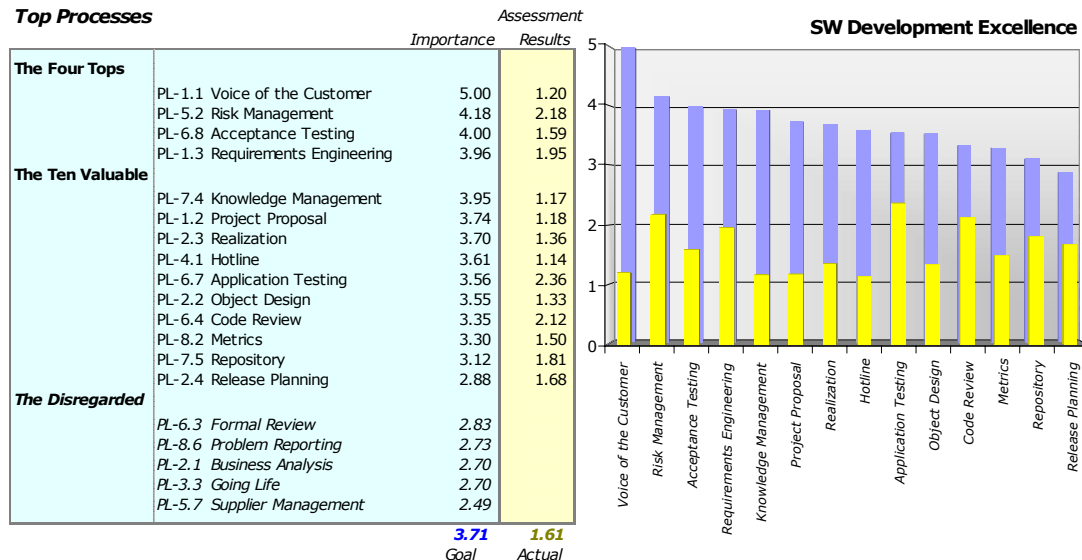


Figure 8: CMM Assessment Result

On the other hand, the same combinator is used to assess the actual performance. The standard Software CMM assessment questions had been used[17]; however the assessment was limited to the most important key process areas, due to cost constraints.

2.4 Aligning to the market using combinators

To understand the impact on the market, we used New Lanchester theory to construct the combinator that links the Business Processes to the actual and the predicted market share of competition. It came out as shown in Figure 5.

Note that the New Lanchester combinator

$$(\text{Business Processes } _k \rightarrow \text{Competition }) _m$$

is used again in both directions, comparing the Web software with existing market share. According this comparison, a market share gain of 18% can be expected, provided all planned features were implemented and the competition remained on the actual state.

At the time of this writing, it is not yet known if the New Lanchester predictions will become true. There exist other case studies showing that such predictions are amazingly correct [5]. However, we cannot yet speak about statistically sound experience involving New Lanchester and combinatorial metrics. It would pay off to learn more about this technique.

2.5 Testing software with combinator

Finally, the testing combinator, discussed in detail in [9], provide a means to optimize the testing effort such that customer’s needs, and the features needed for the planned market share gains, are the focus of the testing effort. A sample testing combinator is shown in Figure 9.

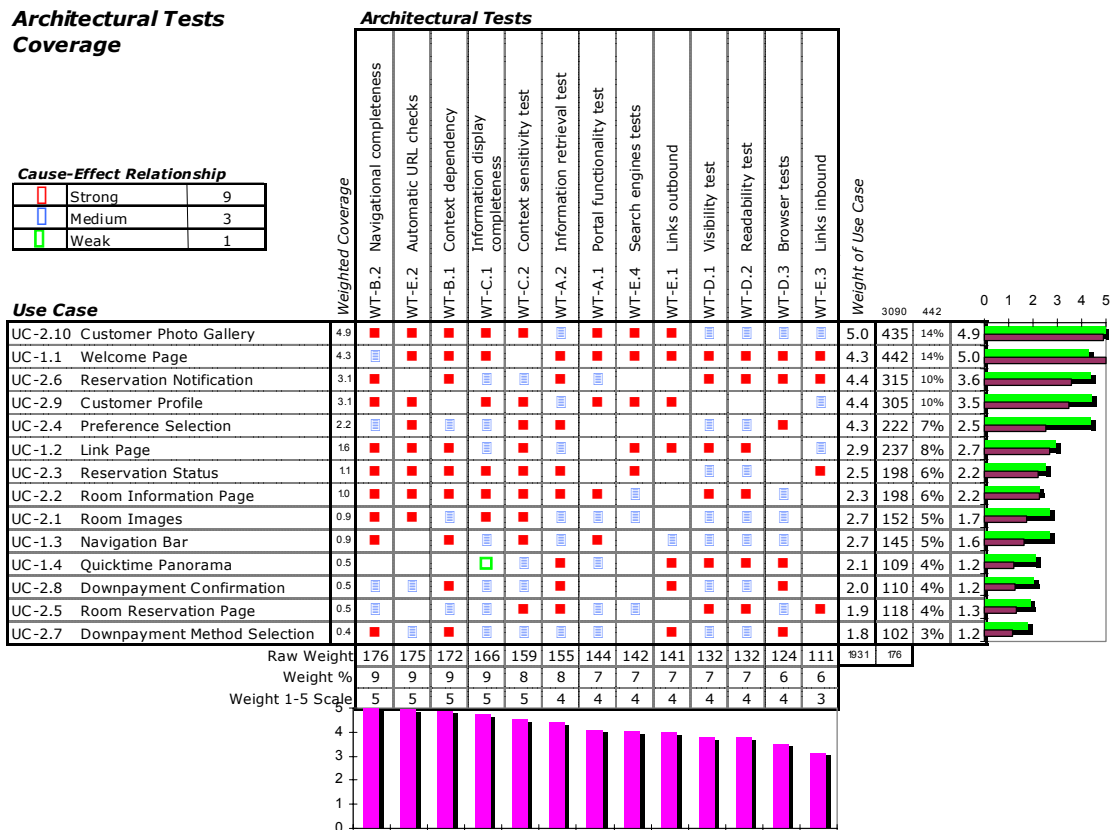


Figure 9: Sample testing combinator

The testing combinator compares actual test coverage (dark brown profile, on the right side of the matrix) with the importance regarding customer’s needs (light green profile). The test case metrics shows their respective importance regarding customer’s needs (which includes the market ‘s expectations according the New Lanchester Strategy).

2.6 Metrics Precision

It is an often-raised question how reliable such a combinatory metric network actually is. At first, it depends from the accuracy of the cause – effect matrices in use. Thus they depend from the domain know-how of the people who did the analysis. However, there is also a statistical method of assessing its dependability. This is in studying the effects induced by variations of the cell values coupling causes to effects. The usual method applied by the author is to change each value by one in each direction, and check if the order of importance changes. It is known that using the mathematical properties of this linear mapping between topics could yield more reliable results, however for all practical purposes so far it was sufficient to do such statistical sensitivity analysis.

However, it is much more important to identify the topics correctly, and keep them separate. It requires common understanding of the domain scope between the people in charge. It is therefore suggested to call such common understanding “*Organizational Knowledge*”. Using combinatory metrics, we may measure organizational knowledge by its effect on the company’s financial results.

2.7 Filling the Gaps

The main advantage of Combinatory Metrics is that the combinators are usable in both directions. One direction is the cause – effect analysis used in QFD to deploy goal topics like the customer’s needs in all branches of the delivery process. The other direction allows assessing the actual performance. In general, such gaps exist and they are very important to know for successful operations.

2.8 Case Study Metrics Overview

Figure 10 shows a complete overview over the combinatory metrics used for the “Hotel Reservation” system software delivery organization, the topics assessed and measured, and all topics for which metrics have been defined. The gaps in the development processes became immediately apparent in the early stage and we could start software development process improvement where there was still time to fix those shortcomings that has most impact on the planned software development.

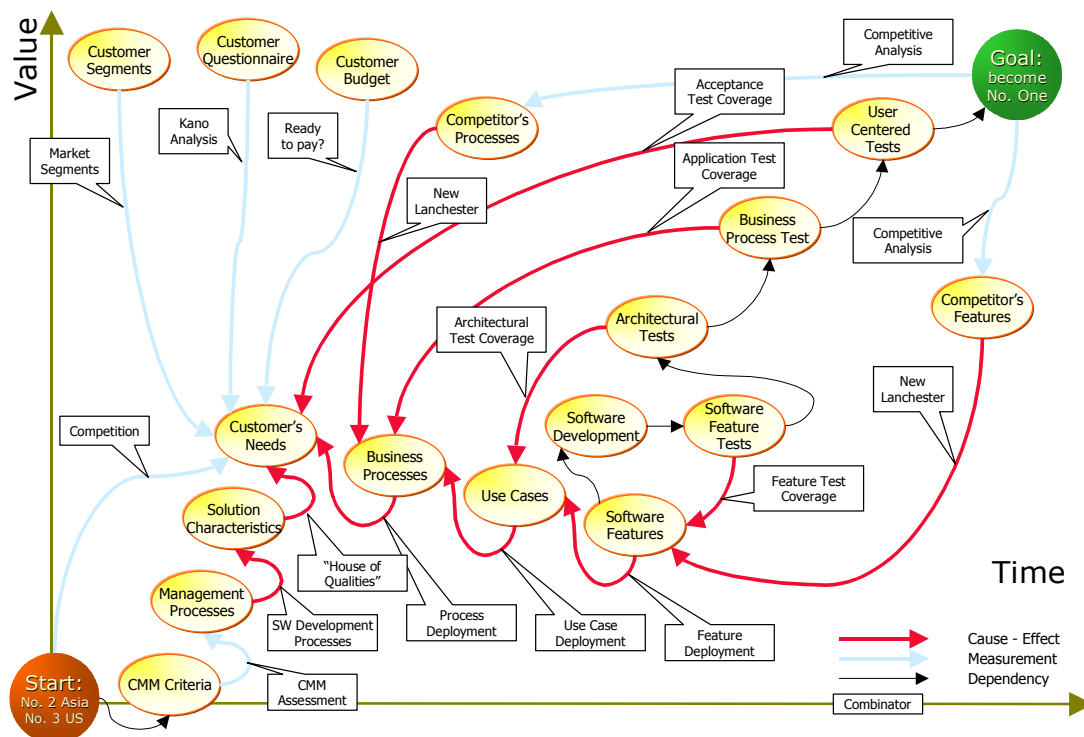


Figure 10: Overview of Combinatory Metrics used for the Case Study

Setting up the combinators was not a major challenge; there was enough domain knowledge and software expertise at hand to do it efficiently. We

used an Excel tool to record the combinatorics and to calculate Combinatory Metrics.

Combinatory Metrics is especially useful for requirements engineering and testing. On every level combinatorics metrics relate requirements and test cases to customer 's needs and market expectations. We used this metrics to keep the development plans focused on the market needs. Every engineering task has a priority index attached to it, derived from customer's needs.

During development, the market and the economic environment did evolve. Using the cause – effect combinatorics, we keep the product features in line with the evolving needs of the market.

The pilot application is now on the market in spring 2002 and the initial reaction was as expected. We expect to have numbers by end of the year.

The effort to set up this metric network was little compared with the standard quality assurance activities; it was well below 1% of the total development effort. Quality assurance (reviews and tests) counted for 20 – 25%.

3. Conclusions

The main achievement of applying combinatorics metrics in the case study was, that after years of constantly missed schedules, the new product is now on the market just in time and with the quality required (status May 2002).

Combinatory Metrics is a very promising approach to manage complex systems using locally applicable metrics. In the experience gained so far, there was not a single failure. Combinatory metrics deserve an in-depth exploration of its capabilities and characteristics. Its computer science background metrics is very compelling too [3], and more research on this techniques and the enabling theory should be performed. The payoff for complex systems will be tremendous.

References

- [1] Yoji Akao et.al.: Quality Function Deployment (QFD); Productivity Press 1990
- [2] Yoji Akao, QFD and the international standard ISO 9001, 7th International QFD Conference, October 2001, Tokyo, Japan
- [3] Erwin Engeler, The Combinatory Programme, Birkhäuser 1995
- [4] Thomas Fehlmann, Quality Function Deployment for the Full Business Life Cycle, EOQ Proceedings Vienna, April 1999.
- [5] Thomas Fehlmann, Measuring Competitiveness in Service Design, in: QFD Institute (Ed.): 12th Symposium On Quality Function Deployment; June 2000 Novi, MI
- [6] Thomas Fehlmann, Christian Hauri, Measuring Project Management Excellence, in: 3rd European Conference on Software Measurement and ICT Control, FESMA – AEMES, Oct. 2000, Madrid, Spain
- [7] Thomas Fehlmann, Risk Exposure Measurements on Web Sites, in: 4th European Conference on Software Measurement and ICT Control, FESMA – DASMA, May 2001, Heidelberg, Germany
- [8] Thomas Fehlmann, QFD as Algebra of Combinators, 7th International QFD Conference, October 2001, Tokyo, Japan
- [9] Thomas Fehlmann, Business-oriented testing in E-Commerce, in: Software Quality and Software Testing in Internet Times, ed. Dirk Meyerhoff, SQS AG, Köln 2001
- [10] Georg Herzwurm, Sixten Schockert, Werner Mellis: Qualitätssoftware durch Kundenorientierung. Die Methode Quality Function Deployment (QFD). Grundlagen, Praxisleitfaden, SAP R/3 Fallbeispiel. Vieweg-Verlag Braunschweig – Wiesbaden 1997
- [11] Managing the Software Process, Watt S. Humphrey, Addison-Wesley, 1989
- [12] Norikai Kano et.al., Attractive Quality and Must-be Quality, 12th Annual Meeting of the Japanese Society of Quality Control, 1982
- [13] Glenn Mazur, QFD for Service Organizations, by Japan Business Consultants, Ltd., 1993
- [14] IFPUG 4.1.1, International Function Point User Group, Function Point Counting Practices Manual, Release 4.1.1, Princeton Junction, NJ, April 2000
- [15] Shigeru Mizuno, ed. 1988; Management for Quality Improvement: The 7 New QC Tools, Productivity Press, 1988
- [16] Shigeru Mizuno and Yoji Akao, ed. 1994; QFD: The Customer-Driven Approach to Quality Planning and Deployment, translated by Glenn Mazur, Tokyo: Asian Productivity Organization, 1994
- [17] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model for Software, Version 1.1", Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993
- [18] Nobuo Taoka: Lanchester Strategy – An Introduction, Lanchester Press Inc, 1997
- [19] Ernest Wallmüller: Software – Qualitätsmanagement in der Praxis, Carl Hanser Verlag, May 2001
- [20] The apparatus of Antikyra. Technical Museum at Thessaloniki, exposed in: World Exhibition Hannover 2000
- [21] Richard E. Zultner: Quality Function Deployment (QFD) for Software: Structured Requirements Exploration. In: Schulmeyer/McManus (ed.): Handbook of Software Quality Assurance. 2nd Ed., Zurich 1992, S. 297-319
- [22] Richard E. Zultner: Blitz QFD: Better, Faster, and Cheaper Forms of QFD. In: American Programmer. October 1995, S. 25-36