Athens Institute for Education and Research
ATINER

ATINER's Conference Paper Series
COM2017-2209

The Role of Mathematics for Success in Business

Thomas Fehlmann
Researcher
Euro Project Office AG
Switzerland

An Introduction to ATINER's Conference Paper Series

ATINER started to publish this conference papers series in 2012. It includes only the papers submitted for publication after they were presented at one of the conferences organized by our Institute every year. This paper has been peer reviewed by at least two academic members of ATINER.

Dr. Gregory T. Papanikos
President
Athens Institute for Education and Research

This paper should be cited as follows:

Fehlmann, T. (2017). **"The Role of Mathematics for Success in Business",** Athens: ATINER'S Conference Paper Series, No: **COM2017-2209.**

Athens Institute for Education and Research
8 Valaoritou Street, Kolonaki, 10671 Athens, Greece
Tel: + 30 210 3634210 Fax: + 30 210 3634209 Email: info@atiner.gr URL: www.atiner.gr
URL Conference Papers Series: www.atiner.gr/papers.htm

ISSN: 2241-2891
02/06/2017

The Role of Mathematics for Success in Business

Thomas Fehlmann
Researcher
Euro Project Office AG
Switzerland

Abstract

In old times, kings, emperors and polis states gathered scientists around them to benefit from protruding knowledge about successful warfare and economics. The foundation of the university of Alexandria gave the Ptolemy kings in Egypt a significant advantage in the world of the 3rd century AD. Islamic empires and the Ottomans later profitably supported universities, thus withstanding Christian kingdoms of the west. Later, things turned around, some empires forgot about science, and in Europe leading nations arose based on their superiority in applying scientific results to power. Mathematics played a major role for instance for the artillery, with its ability to predict ballistics. Today, this is still the case, although not always visible to the public. Many modern money-generating businesses rely on mathematics, as well as security measures. But there is more: Who is aware what made the digital storage and distribution of pictures and music possible? What exactly has Google Search in common with Linear Algebra? What is the foundation of Big Data? When was this invented? Was it already Euclid, or did something important happen after the 3rd century AD? Many people today have mathematical skills not superior to Euclid's students, but mathematics in the 20th century possibly made the biggest steps forward ever. This paper presents modern experiences from the last 40 years that made businesses successful with a little bit of advanced mathematics – advanced means, not yet covered by Euclid's geometry.

Keywords: Analytic Hierarchy Process, Combinatory Logic, Quality Function Deployment, Six Sigma Transfer Functions, Voice of the Customer.

Acknowledgments: Many thanks to my dear colleague Eberhard Kranich who investigated the mathematics behind transfer functions, see (Fehlmann, 2016).

Introduction

In the antiquity, the University of Alexandria stood at the origins of almost all technology. They discovered the theory needed not only for siege machines in warfare, but also for more efficient irrigation systems. Examples include the geometry of Εὐκλείδης, and the theory on conic sections, probably documented by Ὑπατία (ca. 370–415) following an interesting conjecture of Russo (Russo, 2004). By the end of the 17th century, the differential calculus developed by Newton and Leibniz preceded the first industrial revolution; calculating the tangential slope that was needed to construct steam machines.

In the 20th century, things accelerated. Since the Seventies of the last century, information technology and software have created a huge range of new business opportunities. Today, the *Information & Communication Technology* (ICT) is probably the most innovative driver in the economy. Among the Forbes' 25 largest companies, Microsoft, Verizon and Samsung represent ICT accurately. Moreover, banks, that count for more than half of the largest companies, have turned from money lending houses into huge ICT enterprises. Some of these businesses grew from small startups to world-leading organizations. ICT companies managed their growth by finding out which qualities they needed for outperforming the competition.

In the Eighties, the decisive quality was the speed of writing reliable software; first as a prototype, then making a product out of it. Programming computers was difficult and time-consuming. Mathematical logic and the development of formal languages opened the way to useful programming languages and compilers, such as Algol, Pascal, and C++.

Times changed; in the Nineties, rapid prototyping was no longer a delighter but became an expectation, in Kano's term (Kano et al., 1984). Now, ICT entrepreneurs were delighted by uncovering customer's need and create technical solutions, based on the *House of Qualities* (QFD) method. At these times, aligning the technical solution to customer's needs was the decisive quality that allowed dominating the market.

Times changed; in the Zeros of this century, writing software has become an engineering discipline, and the race was for features and functions. However, resources were still limited. *Six Sigma Transfer Functions* helped startup companies to concentrate resources on those tasks that customers liked most. It became possible to analyze customer preferences based on the *New Lanchester Theory*, an application of Six Sigma transfer functions, and even predict the evolution of customer's needs, for instance with the *Net Promoter® Score* method, using another application of Six Sigma transfer functions for analyzing the measured score (Fehlmann & Kranich, 2012). One house of quality was no longer good enough. Creating the optimum technical solution became an expectation. *Comprehensive Quality Function Deployment* of *Deming Chains* became instrumental in adding the right new features to the products.

In the Tens of this century, the world is changing at an incredible pace. Digitalization changes the way we do business. Former delighters became

expectations, once again. From a competitive approach, it moves into a collaborative approach, where alliances and the user's involvement transform customers into partners. Decisive quality is no longer primarily linked to reliability of software, because this is with today's frameworks taken for granted; nor is the feature list a decisive quality. Technical features can be obtained from the cloud, as needed. The qualities needed today for economic success have much to do with the ability to let different services cooperate. Google build a business model by bringing different things together. They used *Transfer Functions* to match content controls with expected responses.

In the Twenties, more use of mathematical theory in business is foreseeable. With the Internet of Thing (IoT), everybody becomes programmer and creates applications, for fun, and for breadwinning. Now, security and safety issues become dominant. Orchestrating the configuration and the software in the IoT is no longer planned, designed, then implemented and tested. Software is created guided by *Customer Experience* (CE), and customer reactions lead immediately to functional changes. The roles of customers and consumers merge. Testing is no longer limited to the testing laboratory, as functional unit tests occur the very same day software gets created, while integration- and CE-testing becomes social. It involves groups of early adopters and promoters. The mechanisms used in IoT quality management are still based on Six Sigma transfer functions.

Autonomous things must be able to execute *Autonomous Testing in Real-Time*. Otherwise, suppliers and operators of these things will run into liability problems when their things do decisions that have the potential to physically impact safety and security of humans. The new ISO 16335 standard (ISO 16355-1:2015, 2015) extends Six Sigma transfer functions, and thus QFD, towards a mechanism to continually understand what customers need – and how these needs change in real-time. Features and functions, systems and programs are no longer stable and static; they rather dynamically adapt to new wishes and ideas of its users. *Combinatory Logic* is needed for managing and controlling the IoT testing.

The Eighties – Formal Languages and Rapid Prototyping

In the Eighties, a small company producing industrial color quality instruments and color quality management software made the journey from formal languages to rapid prototyping. In the beginning of computer software products, memory allowing rapid access was expensive and expensive. It was impossible to run large, error-prone code. Code had to be written concise, and formally proving the code was cheaper than debugging and testing. Therefore, the startup used a LISP-dialect for programming its color quality management applications, measuring the visual impact of colors in textile and automotive.

The programming environment supported complicated management tasks with a minimum of system memory. This offered industrial color

quality management to many small and medium enterprises in emerging markets. Writing new applications in the LISP-dialect was fast and easy for people with a mathematical background, or for programmers with ancient Greek or Sanskrit background, but ordinary programmers with nothing else than some understanding of computers, or economics, usually failed.

The startup became the world leader in industrial color quality measurements after they could produce a prototype software within one week that allowed the laboratories of a large German automotive producer to define color quality requirements for its suppliers. Their purchase department made these color specifications compulsory for all their parts supplier, forcing them to buy the color quality measurement equipment and software, and the small company was no longer small.

Formal languages still are predominant; HTML is nothing else than a variant of LISP; only, parentheses were replaced with tag pairs. This made HTML more readable than LISP. It became acceptable for practitioners without mathematical background.

The Nineties – Discovery of Customer's Needs

In the Nineties, rapid programming became standard practice. Customers expected prototype solutions, avoiding specifications in natural language. Software producers detected that customers sometimes have different needs than those anticipated by developers. Microsoft's Office suite became a huge success because Microsoft studied the perceptions of users. Their Office suite became acceptable to the business world.

Japanese engineers had used *Quality Function Deployment* (QFD) for already some time to understand which technical solutions were most valuable to customers. QFD is a Six Sigma transfer function, mapping controls onto responses. Normally the required response is known; the optimum controls are unknown and must be discovered.

QFD, in its most simple form, only coupled customer's needs as the expected response with technical characteristics as controls. When writing project proposals, such QFD proved most successful. Within a large systems integrator in Europe, using QFD in their proposal centers improved the win rate from around 30% to over 80%. Moreover, the long-term success of the projects jumped from the well-known 20% (Standish Group, 2013) to incredible 97%; not only because the proposals were better focused on customer's needs, but also because proposals did rely on effort estimations based on functional size (Hill, 2010), and were only written, and effort spent, if the systems integrator had something to offer that had the potential to persuade the customer. QFD thus not only was used as analytics tool but as well for predicting success with offers.

QFD was not only helpful in setting up projects but also used for product features. However, for instance, DEC Digital Equipment tried to set up their *DecWrite* tools with QFD – and they failed (Schein et al., 2003). In contrary to Microsoft with its successful Office, the DEC analysts did not

ask, or observe, customers what they expect from document creation and publishing tools. They asked their engineers; people who looked at a document as structured data. Secretaries did not do so. Consequently, the market was not inclined to share the engineer's view with DEC.

QFD as a Six Sigma transfer function always was kind of a mathematical tool; however, the way it had been used was more like handcraft, not scientific, and using "bad mathematics" (Mazur, 2016).

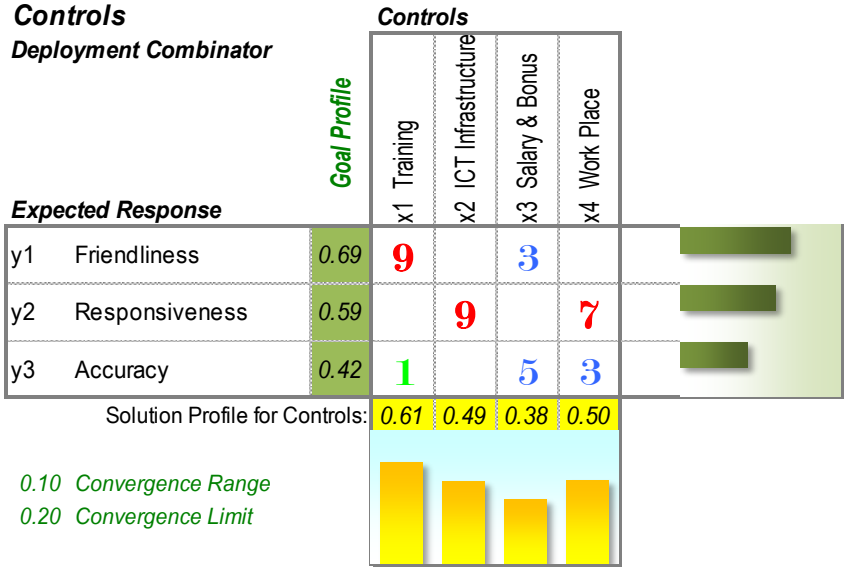
Figure 1 shows an example of the old QFD style. In this *Call-In* service improvement project, the values of the customer had been identified by a profile with three dimensions, y_1: *Friendliness*, y_2: *Responsiveness*, and y_3: *Accuracy*. The importance of these call-in service qualities is identified in the order shown. Four controls, x_1*: Training*, x_2*: ICT Infrastructure*, x_3*: Salary & Bonus*, and x_4*: Work Place* compete for investments; the numbers in the matrix cells stand for their budget part related to the respective goal. For instance, x_2*: ICT Infrastructure* impacts y_2*: Responsiveness* only.

Obviously, the choice is open how much to invest into which control topic, and investments should be optimized for maximum effectiveness. In mathematical terms, if $\boldsymbol{y} = \langle y_1, y_2, \dots, y_m \rangle$ is the response and $\boldsymbol{A} = a_{i.i}$ the system ($i = 1..n$ and $j = 1..m$ denoting the indices for rows and columns), then optimum solution controls $\boldsymbol{x} = \langle x_1, x_2, \dots, x_n \rangle$ must be found such that

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{A}\boldsymbol{x} \\ y_j &= \sum a_{ij} x_i \end{aligned} \tag{1}$$

Both, $\boldsymbol{x}$ and $\boldsymbol{y}$, normally have several dimensions, represented as linear vectors of unit length, i.e., with $\|x\| = \|y\| = 1$ in the Euclidean norm.

Figure 1. *Call-In Service old QFD: Aligning Budget based on Contributions*



The QFD matrices were usually prepared by expert teams agreeing on the coupling between control and responses. Old QFD used the following "bad mathematics" to solve equation (1):

$$\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$$
$$x_i = \sum a_{ji} y_j \qquad (2)$$

where $\boldsymbol{A}^{\mathsf{T}} = (a_{ii})$ is the transpose of matrix $\boldsymbol{A} = (a_{ii})$.

Figure 1 shows $\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$. Obviously, $\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$ is no solution for $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, but this approximation yielded good-enough solutions for the relative importance of controls, in many cases. The reason for this is that the initial step for numerically solving $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is $\boldsymbol{x_0} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$, and that experienced QFD moderators could "read a matrix" to assess whether $\boldsymbol{x} = \boldsymbol{x_0}$ was good enough.

The cell values in QFD matrices are not numbers only. Much more important than the numerical value is what needs to be done in that cell. For instance, in

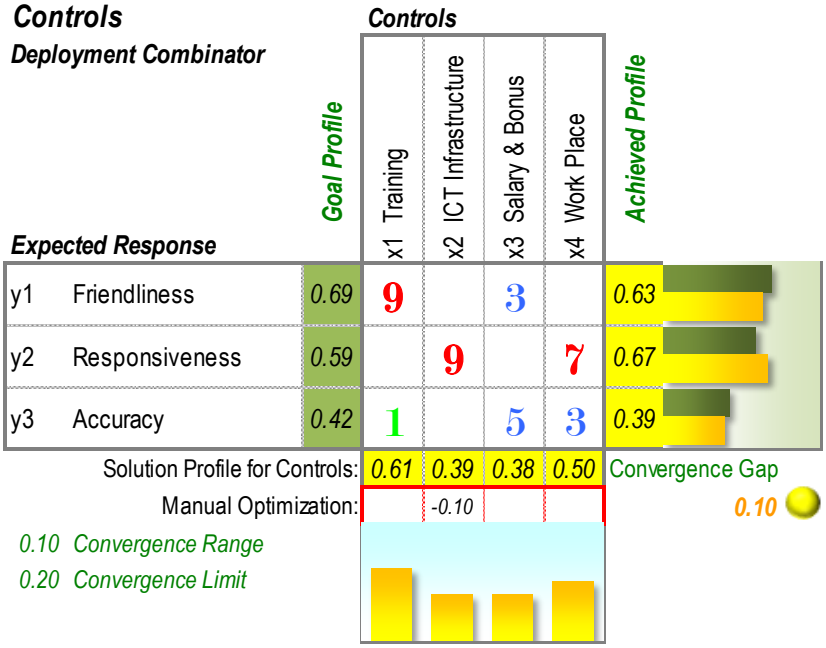
Figure 1, the actions causing the budgeted cost are more important than the numbers only. QFD is a method of designing a system or a product.

The Zero's – Features & Functions

As had already been learned with DEC's *DecWrite* experience, the QFD method of the Nineties did not work for product development; not even for larger and more complicated projects. Akao proposed already in

1980 *Comprehensive QFD* (Akao, 1990), connecting matrices to each other, thus making the controls of the first matrix the expected response for the matrix one level deeper. However, the QFD of the Nineties was not good enough. The quality of controls remained unknown when becoming expected responses. Linear algebra made it better (Fehlmann, 2003).

Figure 2. *Call-In Service New QFD: Aligning Budget with Manual Corrections*



Since 2003, the expected and the achieved response were compared using the *Convergence Gap*. This is the Euclidian distance between the expected response, the goal $\boldsymbol{y}$, and the achieved response $\boldsymbol{Ax}$, resulting from the controls $\boldsymbol{x}$, and indicates how well these controls explain the observed response:

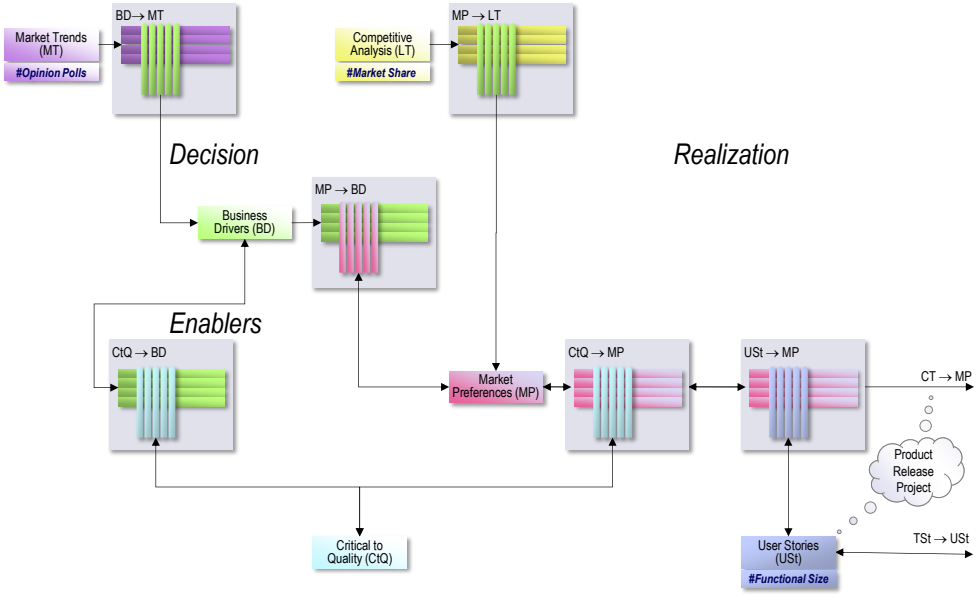
$$Convergence\ Gap = \|\boldsymbol{y} - \boldsymbol{Ax}\| = \sqrt{\sum \left(y_j - \sum a_{ij} x_i\right)^2} \qquad (3)$$

The convergence gap could then be used to optimize controls by iteration, using domain expertise, or by any numerical optimization method.

In Figure 2, the budget for x_2*: ICT Infrastructure* has significantly be reduced. We will see that this goes into the right direction but the reduction is too much; see

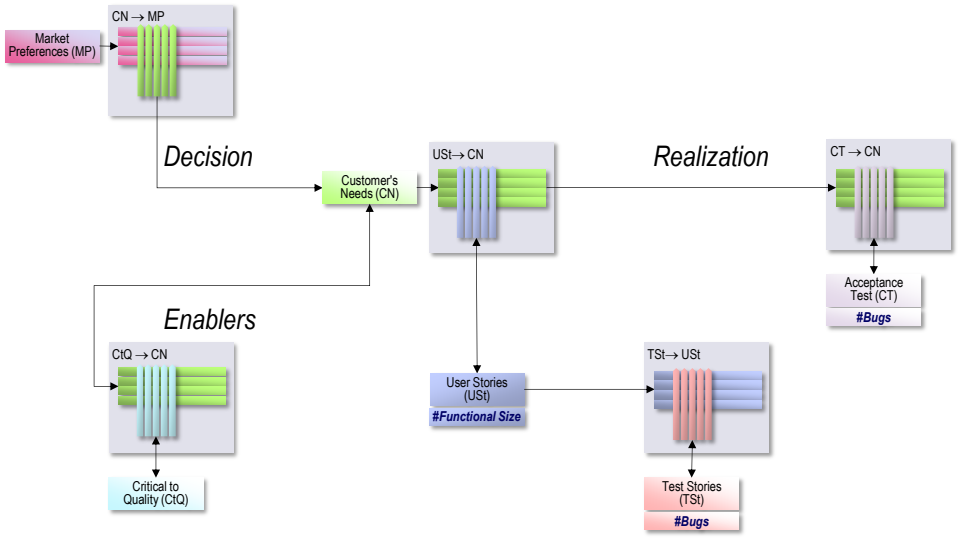
Figure 6. With known convergence gap, it is safe to use the resulting control profile as the target response for another matrix. This allows breaking down customer's needs over several steps in the value chain. Because Deming was the first describing such value chains, we call such comprehensive QFD deployments *Deming Chains* (Deming, 1986). Figure 3 shows a sample Deming chain for product development, yielding market preferences for the product.

Figure 3. *Deming Chain for Some Software Product*



For each release of the product, Figure 4 shows how to derive features out of market preferences.

Figure 4. *Deming Chain for a New Product Release*



In Figure 4 testing forms a bathtub just as the V-Modell. Testing coverage is optimal if the convergence gap in the testing QFD matrices goes towards zero. For a company writing software for large-scale customer communications, a network of up to 27 QFD deployments was necessary to uncover market preferences. This in turn determines which features to implement in each new release. The approach made the company world leader in his field.

This went extremely well, until in the late Zero's it turned out that manual optimization, even if based on domain expertise, sometimes missed

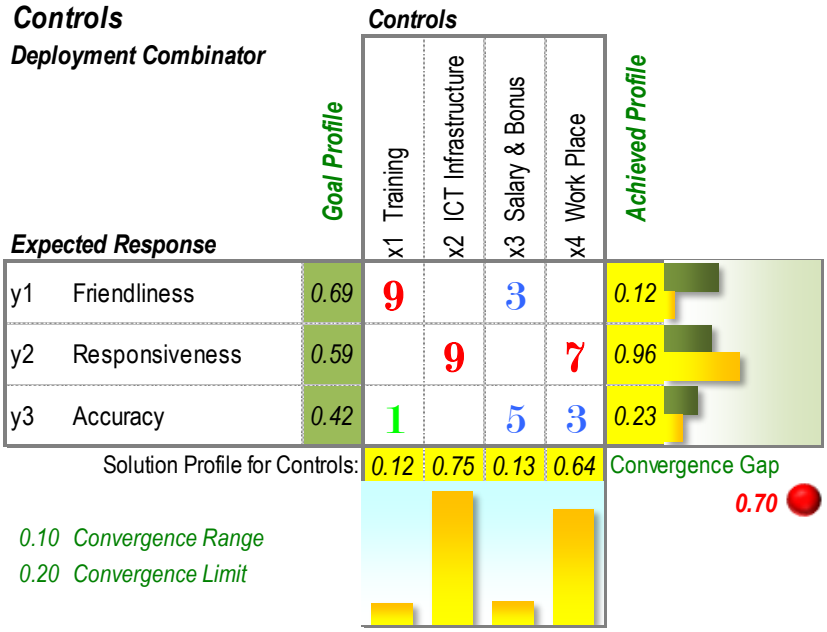
the right solution, although numerical optimization worked correctly. Strange effects were observed and the business target missed. The theory had a weakness.

The weakness was that the controls were not the optimum ones. Optimization of non-optimal controls is feasible but useless. Thus, how to find optimum controls?

Saaty found the answer already some twenty years earlier. For the *Analytic Hierarchy Process* (AHP), he proposed using Eigenvectors to solve decision problems (Saaty & Alexander, 1989). Google used the same method for its PageRank algorithm (Gallardo, 2007). Eigenvectors have the nice property that they level out measurements errors; for instance, if decision teams are not fully consistent in their judgements.

AHP decision matrices are square, reciprocal, and positive, and thus always have a *Principal Eigenvector*, following the theorem of *Perron-Frobenius*. For proofs, see the literature on Linear Algebra, e.g., Kressner (Kressner, 2005), or its compendium in Fehlmann (Fehlmann, 2016). In turn, QFD matrices are rectangular, not square, and connect controls to responses of difference kind and different vector space dimensions.

Figure 5. *Call-In Service New QFD: The Expected Response is No Eigenvector!*



The observation that made eigenvector theory work for QFD is, that in case $\boldsymbol{x} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$ is an exact solution for $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, $\boldsymbol{y}$ is an eigenvector of $\boldsymbol{A}\boldsymbol{A}^{\mathsf{T}}$:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} \tag{4}$$

This fact can be used to find out whether the controls $\boldsymbol{x}$ are suitable for solving $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. In other words, to make Deming chains work, it is

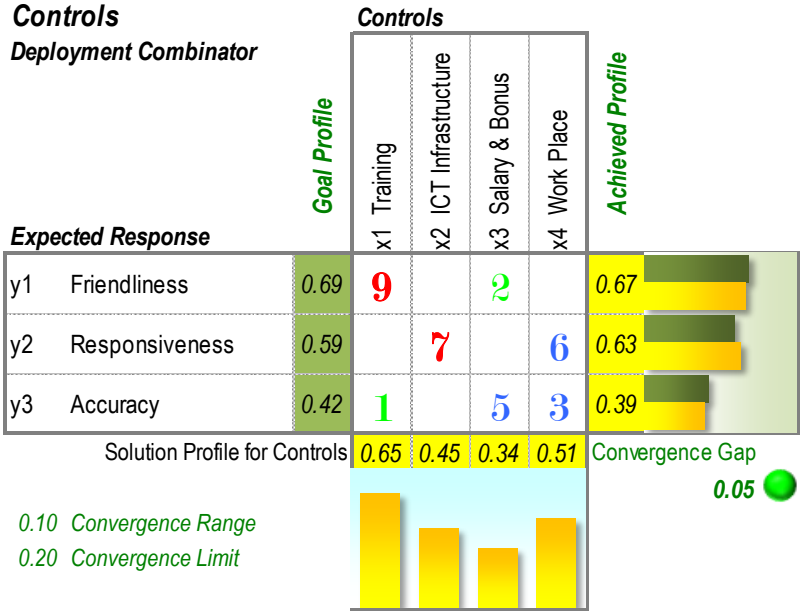
necessary to find QFD matrices $\boldsymbol{A}$ that have the expected response $\boldsymbol{y}$ as the eigenvector for $\boldsymbol{AA}^{\mathsf{T}}$.

Figure 5 makes it obvious that in the call-in example, the expected response $\boldsymbol{y} = \langle y_1, y_2, y_3 \rangle$ is not an eigenvector to the QFD matrix. Thus, the control must be changed, or the budget contributions adapted. In fact, by reducing the part for x_2: *ICT Infrastructure*, and with a few other adjustments, budget allocation reach the goal by the four proposed controls (

Figure 6), with the overall budget reduced. With eigenvectors, the solution profile at the bottom represents priorities, not the total investment budget, as it did in

Figure 1.

Figure 6. *Call-In Service New QFD: Optimized Controls and Contributions*



Thus, QFD and eigenvectors make experienced QFD moderators all but obsolete. The automated general problem solver producing successful products remains fiction, as already known to logicians (Engeler, 1995). For designing successful products, domain expertise remains essential; however, the Eigenvector method in QFD allows to see whether certain product ideas are promising or not. Therefore, the use of modern QFD based on Eigenvectors is highly recommended before investing money in new product development.

Transfer Functions in Hindsight

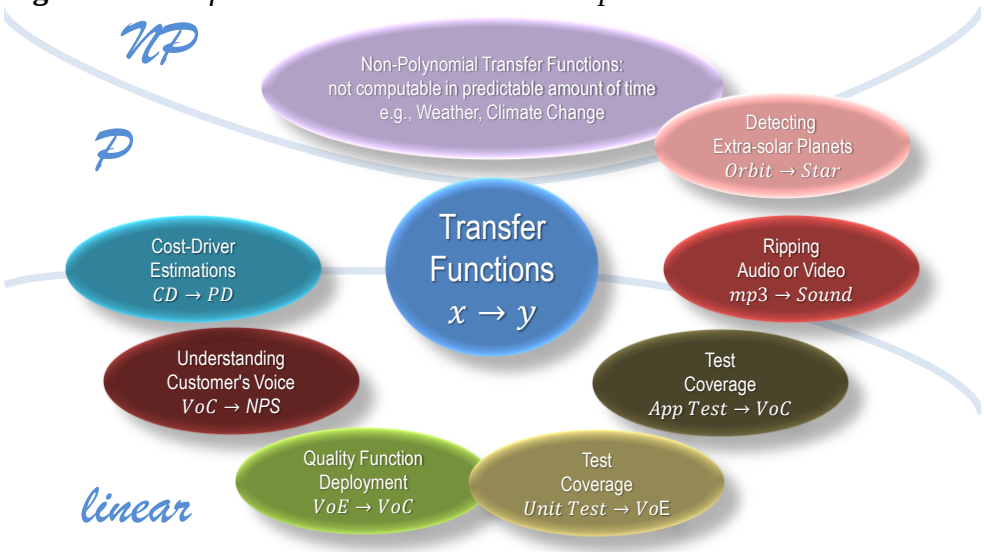
What was the most impressing invention in the 20th century? A few inventions changed the world, much more than cars and railways ever could do. The proof that the *Fast Fourier Transfer* (FFT) is not NP-complete made the deepest impact on humankind probably since the invention of cooking by fire (Cooley & Tukey, 1964). It led to the possibility of transferring audio and video signals in predictable time into digital code. Previously, audio sounds were analogous oscillations of the atmosphere, or of electro-magnetic potentials, caused by beings or loudspeakers.

The FFT describes the original oscillations by digital numbers. Thus, it became possible to build chips that transformed audio signals into storable digital code and later video signals as well.

Today, music and entertainment populates computers, laptops, phones and television sets that connect to the Internet. All this started with FFT.

The development of the FFT procedure is the result of research in *Linear Algebra*. FFT transforms audio signals captured by a microphone as electrical pulses into digital numbers. To be more precise, the FFT algorithm selects the Fourier base functions in the functional vector space that model the analogous signal and represents the signal by the coordinates of the unit vector in this space. This works like a linear vector space. The base units are functions, not points in space. The controls for a transfer function explain the observed response and can reproduce it. This is the essence of the FFT procedure.

Figure 7. *Transfer Functions in Various Disciplines*



There is a large family of transfer functions.

Figure 7 lists the most important of them. Not all transfer functions are linear. Some use polynomial ($\mathcal{P}$), the more complex ones non-polynomial ($\mathcal{NP}$) transfer functions. Eigenvector theory is the method of choice for solving linear transfer functions of equation (1). Solutions to this kind of problem are widely in use, today.

The Ten's – Πάντα Ῥεῖ

Traditionally, QFD matrices only used the three relationship values 1,3, and 9 for the coupling – besides 0 for no coupling. The new ISO standard 16355 (ISO 16355-1:2015, 2015) allows for ratio scale values, both increasing freedom of choice and for using measurements instead of expert team estimates, for the coupling. You can view at expert team judgments as a sort of measurements in their mind. This makes QFD and Six Sigma cause-effect matrices coincide, as Six Sigma transfer functions.

A sample measurement application of Six Sigma transfer functions is for software testing. Test cases transfer test data into expected responses; otherwise the test fails. This gives rise to a QFD matrix whose controls are the test stories, the response being the proof for correct user story implementations. The cell contents measure the number of data movements (ISO/IEC 19761:2011, 2011) needed to execute each of the test cases (Fehlmann, 2016, p. 247ff). The convergence gap is an indication for test coverage. Contrary to traditional QFD, all is automated; no human assessments are needed for the transfer function. Even the selection of controls can be automated; you can automatically generate test cases and determine whether to include them in the test suite based on their contribution to closing the convergence gap.

This opens a new case for autonomous real-time testing. This means that a software system generates test cases when needed; for instance, when the system enters a new environment with new partners and components. It means that such systems behave "intelligent" in the sense that they test the impact of any decisions before they go into execution. For autonomous cars, such tests are a legal requirement for protecting the car supplier against being liable for all kind of hazards that might happen.

Obviously, this is an extension of what we used already for Deming chains in product development. It means that products adapt themselves to changing environments; the product itself changes features and behavior. Although we still refer to products, we must acknowledge that products nowadays are mainly characterized by the software they contain. However, an autonomous car is not a "software product" as the good old Microsoft Office suite still is. It is a complex system that connects with other cars and depends from the information it gets by GPS traffic services and road maps.

Luckily, there is a mathematic theory available to help us with the challenge of products that adapt themselves to new environments. This is *Combinatory Logic*, see Engeler (Engeler, 1981) and (Engeler, 1995).

To understand what combinatory logic can contribute to business success in the futures, maybe the Twenties of this century, we look at a model of combinatory logic, the *Arrow Term* model. Arrow terms are formal terms constructed over propositional logic. Our example is concerned with autonomous real-time testing; the propositional logic describing test cases and test results.

Instead of writing matrices, we look at the following power set constructed over propositional logic describing tests. Let $\mathcal{L}$ be the set of all propositions over test descriptions; test data and test results. These statements contain no free variables; i.e. they are propositions about tests in our business domain.

Denote by $\mathcal{G}(\mathcal{L})$ the power set containing all *Arrow Terms* of the form

$$\{a_1, \dots, a_m\} \to b \tag{5}$$

The left-hand side is a finite set of arrow terms and the right-hand side is a single arrow term. This definition is recursive; thus, it is necessary to establish a base definition saying that every proposition itself is considered an arrow term. The arrows of the arrow terms are distinct from the logical imply that some authors also denote by an arrow. The arrows denote cause-effect, not logical imply.

The formal, recursive, definition, written in set-theoretical language, is

$$\begin{aligned} \mathcal{G}_0(\mathcal{L}) &= \mathcal{L} \\ \mathcal{G}_{n+1}(\mathcal{L}) &= \\ \mathcal{G}_n(\mathcal{L}) &\cup \left\{ \{a_1, \dots, a_m\} \to b \middle| a_1, \dots a_m, b \in \mathcal{G}_n(\mathcal{L}), m = 0,1,2,3 \dots \right\} \end{aligned} \tag{6}$$

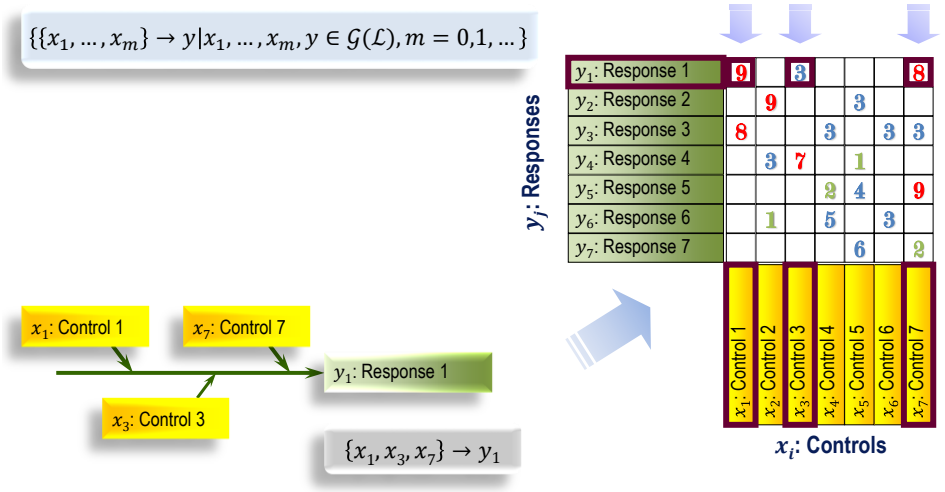
$\mathcal{G}(\mathcal{L})$ is the set of all (finite and infinite) subsets of the union of all $\mathcal{G}_n(\mathcal{L})$:

$$\mathcal{G}(\mathcal{L}) = \bigcup_{n \in \mathbb{N}} \mathcal{G}_n(\mathcal{L}) \tag{7}$$

The elements of $\mathcal{G}_n(\mathcal{L})$ are arrow terms of level n. Terms of level 0 are *Topics*, terms of level 1 *Rules*. A *Rule Set* is an element of $\mathcal{G}_n(\mathcal{L})$ that consists of level 1 terms only and is finite; if it is infinite, we call it *Knowledge Base*. Hence, knowledge is a potentially unlimited set of rules about topics and rules. This definition is recursive, as before. As shown in Figure 8, the rules correspond to the cause/effect coupling in the QFD matrix.

Arrow sets represent Six Sigma transfer functions in a way originally described by Ishikawa. The *Ishikawa Diagram* (Ishikawa, 1990) describes the cause-effect relations between topics and are considered the initial form of QFD matrices. Converting a series of Ishikawa diagrams into a QFD matrix is straightforward, see Figure 8 below.

Figure 8. *Representing QFD Matrices as Rule Sets*



$\mathcal{G}(\mathcal{L})$ extends a single test suite for a rule set $\mathcal{L}$ by a possibly infinite sequence of tests, still on the same domain. The same tests are applicable on different test data and even on different testable objects.

This allows extending existing test suites to cover an extended system. For instance, if a test suite tests the software needed for platooning trucks, and additional trucks are added to the platoon, with different software releases or even truck of different making, the extension to the new platoon allows the platooning software to conduct extended tests that yield the same meaning and provide the same level of security and safety as with the original software for the original sample platoon. However, the tests cover the actual platoon with all its special characteristics and software variants.

Such tests allow the suppliers of platoons and autonomous cars to assume liability in case somethings unexpected happens with the extended system. Obviously, testing metrics must be used that can assure an appropriate test density, and the suppliers needs proof that the tests have been executed and passed. In fact, it is very unlikely that autonomous cars will ever be admitted to public streets without this kind of *Autonomous Real-Time Testing*.

Further Research

The above sample application of combinatory logic to business is possibly nothing but the very start of it all, because combinatory logic, and its arrow term model is a model of *Combinatorial Algebra*; i.e., it describes all computable operations on knowledge bases (Barendregt, 1977). It is yet open what the benefit of this *Universal Property* (Engeler, 1995, p. 8) is when applied to business topics. For instance, combinatory algebra is a means to transform existing comprehensive QFD deployments into new ones, which may fit better into new market environments.

When analyzing *Big Data*, the mathematical methods used are *Graph Theory* (Meyerhenke et al., 2009) and again Eigenvector solutions for

solving linear equations (Andersen et al., 2006). Graph theory is another major achievement of the 20th century, although the paper written by Euler on the Seven Bridges of Königsberg and published in 1736 is regarded as the first paper in the history of graph theory (Biggs et al., 1986). The impact of these techniques on society and business is yet unknown.

References

Akao, Y., ed., 1990. *Quality Function Deployment - Integrating Customer Requirements into Product Design.* Portland, OR: Productivity Press.

Andersen, R., Chung, F. R. K. & Lang, K. J., 2006. Local Graph Partitioning using PageRank Vectors. *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS),* p. 475–486.

Barendregt, H. P., 1977. The Type Free Lambda Calculus. In: J. Barwise, ed. *Handbook of Mathematical Logic.* Amsterdam: North-Holland, pp. 1091-1132.

Biggs, N., Lloyd, E. & Wilson, R., 1986. *Graph Theory, 1736-1936.* s.l.:Oxford University Press.

Cooley, J. W. & Tukey, J. W., 1964. Cooley, James W.; Tukey, An algorithm for the machine calculation of complex Fourier series". Math. Comput. 19 (90): 297–301. *Mathematics of Computation*, 17 August, Volume 19, pp. 297-301.

Deming, W., 1986. *Out of the Crisis.* Center for Advanced Engineering Study ed. Boston, MA: Massachusetts Institut of Technology.

Engeler, E., 1981. Algebras and Combinators. *Algebra Universalis,* Volume 13, pp. 389-392.

Engeler, E., 1995. *The Combinatory Programme.* Basel, Switzerland: Birkhäuser.

Fehlmann, T. M., 2003. *Linear Algebra for QFD Combinators.* Orlando, FL, International Council for QFD (ICQFD).

Fehlmann, T. M. & Kranich, E., 2012. *Using Six Sigma Transfer Functions for Analysing Customer's Voice.* Glasgow, UK, Strathclyde Institute for Operations Management.

Fehlmann, T. M., 2016. *Managing Complexity - Uncover the Mysteries with Six Sigma Transfer Functions.* Berlin, Germany: Logos Press.

Gallardo, P. F., 2007. Google's Secret and Linear Algebra.. *EMS Newsletter,* Volume 63, pp. 10-15.

Hill, P., ed., 2010. *Practical Software Project Estimation 3rd Edition.* New York, NY: McGraw-Hill.

Ishikawa, K., 1990. *Introduction to Quality Control.* Translated by J. H. Loftus; distributed by Chapman & Hall, London ed. Tokyo, Japan: JUSE Press Ltd.

ISO/IEC 19761:2011, 2011. *Software engineering - COSMIC: a functional size measurement method,* Geneva, Switzerland: ISO/IEC JTC 1/SC 7.

ISO 16355-1:2015, 2015. *ISO 16355-1:2015, 2015. Applications of Statistical and Related Methods to New Technology and Product Development Process - Part 1: General Principles and Perspectives of Quality Function Deployment (QFD), Geneva, Switzerland: ISO TC 69/SC 8/WG 2 N 14,* Geneva, Switzerland: ISO TC 69/SC 8/WG 2 N 14.

Kano, N., Seraku, N., Takahashi, F. & Tsuji, S., 1984. Attractive quality and must-be quality. *Journal of the Japanese Society for Quality Control (in Japanese),* April, 14(2), p. 39–48.

Kressner, D., 2005. Numerical Methods for General and Structured Eigenvalue Problems. *Lecture Notes in Computational Science and Engineering*, Volume 46.

Mazur, G. H., 2016. *ISO 16355: A Quality Approach to New Product Development.* Helsinki, EOQ 2016 Congress.

Meyerhenke, H., Monien, B. & Sauerwald, T., 2009. A new diffusion-based multilevel algorithm for computing graph. *Journal of Parallel and Distributed Computing*, 69(9), p. 750–761.

Russo, L., 2004. *The Forgotten Revolution - How Science Was Born in 300 BC and Why It Had to Be Reborn.* Berlin Heidelberg New York: Springer-Verlag.

Saaty, T. & Alexander, J., 1989. *Conflict Resolution: The Analytic Hierarchy Process.* New York, NY: Praeger, Santa Barbara, CA.

Schein, E. H., Kampas, P. J., Sonduck, M. & DeLisi, P., 2003. *DEC is Dead, Long Live DEC: The Lasting Legacy of Digital Equipment Corporation.* San Francisco, CA: Berett-Koehler Publishers.

Standish Group, 2013. *CHAOS Report*, Boston, MA: CHAOS University.