Athens Institute for Education and Research
ATINER

ATINER's Conference Paper Series
COM2016-2059

# Organizing Knowledge in the Internet of Things (IoT)

**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.** (2016). **"Organizing Knowledge in the Internet of Things (IoT)",** Athens: ATINER'S Conference Paper Series, No: **COM2016-2059.**

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
22/11/2016

# Organizing Knowledge in the Internet of Things (IoT)

**Thomas Fehlmann**

## Abstract

Combinatory logic lays the theoretical foundations for managing complexity in the *Internet of Things* (IoT). An example of such complexity is the safety of self-controlled cars; another one is making the IoT helpful and enjoyable for humans. Just as once Euclid's geometry, famous work originating at the University of Alexandria, made the transition from agricultural to urban living possible since 300 BC, now is the time to use combinatory logic for the transition from industrial production to value creation by intelligent things. IoT *Orchestras* are collections of sensors, actuators and cloud services. IoT orchestras communicate with each other and interact as a system. This paper explains in very short terms, what a model of combinatory logic is and how IoT orchestras implement such a model in practice. Furthermore, it discusses new approaches based on this theory for predicting strange and unforeseeable conditions, for predicting the behavior of IoT orchestras about safety and security up to some defined level.

**Keywords:** Automated Testing of IoT, Combinatory Algebra, Combinatory Logic, Internet of Things (IoT), Lambda Calculus, Quality Function Deployment (QFD), Six Sigma Transfer Functions.

**Acknowledgments:** Many thanks to my dear colleague Eberhard Kranich who laid the mathematical foundations needed for managing complexity with AHP, QFD and combinatory logic, see (Fehlmann & Kranich, 2016 (to appear)).

## Introduction

Today, we embark on a new conquest of Ilion: *The Internet of Things* (IoT). It was already difficult to learn how to develop software properly. Only recently it had been understood that agile methods are the only ones capable of handling the complexity of developing software against unknown customer requirements. What has paved the way for agile was understanding that the aim of software development is not only the well-engineered code but understanding the needs of the customer and translating them into a language that machines understand.

An even more challenging quest is to master the multitude of intelligent things around us. Things talk to each other, exchange information affecting behavior out of direct human control. Cars block because some internal intelligent network decided the car is out of service. Medicine cupboards deny access because the software cannot authenticate the doctor. Autonomous cars crash into each other because different manufacturers build them. How avoiding that our intelligent things close us out of our homes, decide blocking the fridge because we ate too much and reveal to our consort the birthday gift we secretly prepared?

There is a political way of how to deal with arising problems – ignore, or blame others for them – and there is a scientific way. The scientific way is finding a theory that explains the world of IoT, and applying it to practice (Russo, 2004) . Such a theory is available: it is the theory of *Combinatory Logic* (Engeler, 1995). Some years earlier, Engeler (Engeler, 1981), based on research by Barendregt on constructive mathematics (Barendregt, 1977), published the main theoretical result in one of the shortest papers – four pages only – ever published in the *Algebra Universalis* journal (1981). On the other side of the scientific world, Akao and other Japanese scientists developed around the same time *Quality Function Deployment* (QFD) to make Japanese economy more competitive (Akao, 1990). Many practitioners in consumer goods and automotive industry have put the theory into practice. QFD works well, because the theory is sound, and is now available as an ISO standard (ISO 16355-1:2015, 2015).

This paper is organized by introducing first to QFD, then presenting the combinatory algebra of generalized QFD cause/effect relationships, and applying it to the IoT.

## Quality Function Deployment

*Quality Function Deployment* (QFD) is possibly one of the most successful but least known methods for product design. It aims at designing products based on the customer's needs. Successful industries such as automotive or smartphones use it extensively; however, these industries do not relate much with the method because it touches business secrets. Nevertheless, QFD is an open methodology, indispensable for growing new businesses and startups to become competitive for the world market. QFD became famous when Japan used it 40 years ago for such purpose. Interest for QFD in emerging economies is remarkable.

The QFD method relies on detecting the customer's needs by listening to the customer's voice. Most of these needs remain hidden because customers, when asked, usually mix up solutions with needs. For instance, the need of moving physically between different places required different technical solutions, after the advent of motorized cars and now with the ability to construct self-controlled vehicles. Over all times and technologies, the need to move remained the same. Thus, detecting and understanding customer's needs is much more than just listening to the customer's voice. Voices alone might be misleading. Customer's needs transfer into customer's voice.

The choice of technical solutions that best fit the customer's need is the crucial step in QFD. This is demonstrated with a simple case study, referring to some phone *Call-In Service* needing improvement (

Figure **1** and Figure 2). The question is which conceivable solutions contribute to which customer's needs. In Six Sigma terms, these technical solutions are *Controls*. These needs not be technical solutions in the narrow sense. For instance, branding and other soft factors that come with a product might be part of the controls ensuring complete coverage of the customer's needs.

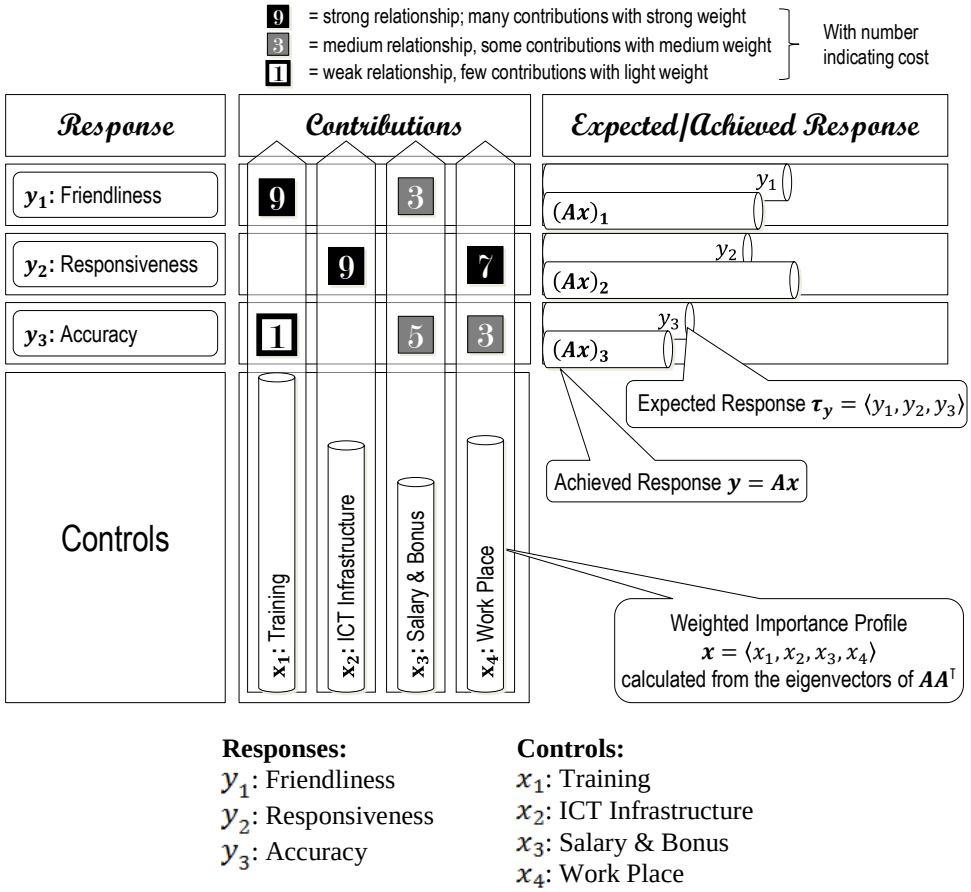
*Transfer Functions for Cause-Effect Relationships*

Transfer functions map control vectors $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$ onto response vectors $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_m \rangle$, for instance design decisions into product features in DfSS, or *Voice of the Engineer* onto *Voice of the Customer* (VoC) in QFD. The response $\boldsymbol{y}$ is known – observable, measurable – whilst the control vector $\boldsymbol{x}$ is uncertain. Transfer functions are useful for detecting the root cause of observed responses of a system under scrutiny. If $\boldsymbol{A}$ denotes the transfer function represented as a QFD matrix, the problem $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ must be solved for obtaining the optimum control profile.

**Figure 1.** *Business Drivers Reflecting Customer's Needs when Calling*

| | Customer's Needs Topics | Attributes | | Weight | Profile |
|---|---|---|---|---|---|
| y1 | Friendliness | Remains cool | Always friendly | 40% | 0.69 |
| y2 | Responsiveness | Understands the problem | Finds a way to solve | 35% | 0.59 |
| y3 | Accuracy | Complete information | Compelling | 25% | 0.42 |

**Figure 2.** *Call-In Service QFD: Aligning Budget based on Contributions*




**Responses:**
$y_1$: Friendliness
$y_2$: Responsiveness
$y_3$: Accuracy

**Controls:**
$x_1$: Training
$x_2$: ICT Infrastructure
$x_3$: Salary & Bonus
$x_4$: Work Place

In the example QFD above, shown in

Figure **1** and Figure 2, y1: Friendliness tops y3: Accuracy by a factor of $0.69/0.42 = 1.7$. If the Call-in Service provider wants to improve its operations, by doing some investments, the question is how much to spend in each of the controls for each response cell of the matrix. Spending to get call-in service improvements thus defines the transfer function in this case cell by cell – as in most cases.

Hence the popularity of QFD. As controls for the process (the $\boldsymbol{x}$), there is $x_1$*: Training* of the Call-in Service employees, $x_2$*: ICT Infrastructure* performance of the ICT equipment, $x_3$*: Salary & Bonus* system, and $x_4$*: Work Place* work environment.

Every item in a matrix cell needs communications and costs effort. The total of the cells' content is proportional to the total cost. Benefit comes from the response only. When doing highly complex decisions, matrices are the preferred visualization method for looking at the interdependencies. The cells of the matrix in Figure 2 contain the work items, as for instance cell $\langle x_1, y_1 \rangle$: *Train Call-in Service staff how to be friendly at the phone* with value 9, or $\langle x_2, y_2 \rangle$: *Invest into faster ICT equipment*, also with value 9.

*The Convergence Gap*

As can be seen visually in
Figure **1**, the achieved response does not match exactly the solution. Usually, it is not possible to achieve an exact match because of various constraints and lack of exact data.

Denote the achieved response profile by $\boldsymbol{y} = \boldsymbol{Ax}$ that approximates the predefined target profile $\boldsymbol{\tau_y}$. Then the Euclidian norm

$$\|\boldsymbol{y} - \boldsymbol{\tau_y}\| = \sqrt{\sum_{j=1}^{m} \left((\boldsymbol{y} - \boldsymbol{\tau_y})_j\right)^2} = \sqrt{\sum_{j=1}^{m} \left(y_i - \tau_{y_j}\right)^2} \qquad (1)$$

is termed the *Convergence Gap* and reveals the quality of the approximation of $\boldsymbol{y}$ to $\boldsymbol{\tau_y}$. If this gap fulfills a predefined convergence criterion, then $\boldsymbol{y}$ is sufficiently close to $\boldsymbol{\tau_y}$. In this case, $\boldsymbol{x} = \boldsymbol{A}^{\intercal}\,\boldsymbol{\tau_y}$ calculates the approximate solution profile $\boldsymbol{x}$, solving $\boldsymbol{y} = \boldsymbol{Ax}$.

*The Eigenvector Solution*

For solving $\boldsymbol{y} = \boldsymbol{Ax}$, the Eigenvector method is the method of choice since being introduced by Google (Gallardo, 2007) for its famous search algorithm.

The idea is to revert cause and effect. First, transpose the QFD matrix and calculate the combined symmetrical square matrix $\boldsymbol{AA}^{\intercal}$. According to the *Theorem of Perron-Frobenius*, e.g. (Kressner, 2005), this symmetrical square matrix has a principal eigenvector $\boldsymbol{y_E}$ with $\boldsymbol{y_E} = \boldsymbol{AA}^{\intercal}\boldsymbol{y_E}$.

Using the principal eigenvector, the solution for $\boldsymbol{Ax_E} = \boldsymbol{y_E}$ is $\boldsymbol{x_E} = \boldsymbol{A}^{\intercal}\boldsymbol{y_E}$. $\|\boldsymbol{y} - \boldsymbol{y_E}\|$ is the convergence gap, which according to equation (1); $\boldsymbol{x_E} = \boldsymbol{A}^{\intercal}\boldsymbol{y_E}$ is called the *Eigencontrols* of $\boldsymbol{A}$. If $\boldsymbol{y}$ is near enough to the principal eigenvector $\boldsymbol{y_E}$ of $\boldsymbol{AA}^{\intercal}$, then $\boldsymbol{x_E}$ is an approximate solution for $\boldsymbol{Ax} = \boldsymbol{y}$.

There are many methods available for calculating eigenvectors; e.g., Volpi (Volpi & Team, 2007). For more information, consult the book and web site of Robert de Levie (Levie, 2012). Most statistical packages contain the eigenvector methods; e.g., the *R Project* (The R Foundation, 2015).

The eigenvector solution is not only useful for detecting optimum technical solutions; it also uncovers customer's needs from customer's voice, see Fehlmann and Kranich (Fehlmann & Kranich, 2012).

*The Internet of Things (IoT)*

The *Internet of Things* is a collection of sensors, actuators, and services that connect these hardware elements to software that react on events or collect data for further analyses. Such services are today most often hosted in some cloud, and the term *Web of Things* commonly refers to this. The IoT impacts the physical world over actuators, such as motors, locks, braking and steering controls.

The IoT changes its scope and behavior with every sensor added or removed. Autonomous cars are a relatively simple example of an IoT since within a container; as soon as they start talking to each other, for instance to find out where the other approaching car is heading to, the scope of the IoT changes. Smart homes are intrinsically more complex since they are subject to external controls such as power plants optimizing the power supply over time.

Most IoT components remain small and tiny, and have no great complexity by themselves. A temperature sensor reports actual temperatures on a continuous but limited scale; an actuator might lock doors or continuously dim lights as needed. Their state is relatively easy to describe by terms over the physical world, called *Propositions*. Such propositions in turn are useful to describe testy cases and test responses.

## Combinatory Logics

For a more in-depth discussion of combinatory logic, consult Fehlmann & Kranich (Fehlmann & Kranich, 2016 (to appear)).

### *Introduction*

How to separate topics belonging to the control domain from the response domain? Sometimes, topics go through a process without being affected. In this case, they are not controls but still look like responses. They might originate from a previous value creation step.

*Topics* in the context of this chapter refer to responses and controls of Six Sigma transfer functions. Topics are the domain areas where transfer functions can be defined in between. The transfer functions implicitly define the topics of its domain areas. The control topics of one domain area can become the response topics of another domain area. This creates the need to explain how to combine such topics.

### *Combinatory Logic*

There is a mathematical theory called *Combinatory Logic* (Engeler, 1981) explaining how to combine the topic areas. Combination is not only possible on the basic level; you can also explain how to combine topics on higher levels. Higher level describes knowledge about how to deal with different topic areas. The focus on the basic level is on knowledge about linking causes and effects – or control and responses – looking at propositional statements about the topic areas, using propositional calculus. For foundations of mathematical logic, see Barwise (Barwise, et al., 1977).

Combinatory logic originates from mathematical logic. Combinatory logic addresses issues with the constructive variant of the *Axiom of Choice*. Informally, the axiom of choice says that given any collection of sets, each containing at least one object, it is possible to select exactly one object from each set, without requiring an algorithm saying how the selection is done. In the theory of *Complex Analysis*, such an algorithm seems an unnecessary

condition existence simply matters, and complex analysis proved to be very successful without requiring constructive selection algorithms.

In *Computer Science*, selection algorithms always exist. Nothing exists that matters for a program or process without an algorithm constructing it. Whether such an algorithm ever stops with a result or loops forever, is another question. Interestingly, this constructive variant of the axiom of choice has wide consequences to mathematical logic.

*The Graph Model of Combinatory Logic*

Let $\mathcal{L}$ be the set of all propositions over a given domain. Examples include statements about customer's needs, solution characteristics, methods used, etc. These statements contain no free variables; i.e., they are propositions about the business domain we are going to model.

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

$$\{a_1, \dots, a_n\} \to b \tag{2}$$

The left-hand side of (2) 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.

In set-theoretical notation, the formal definition is

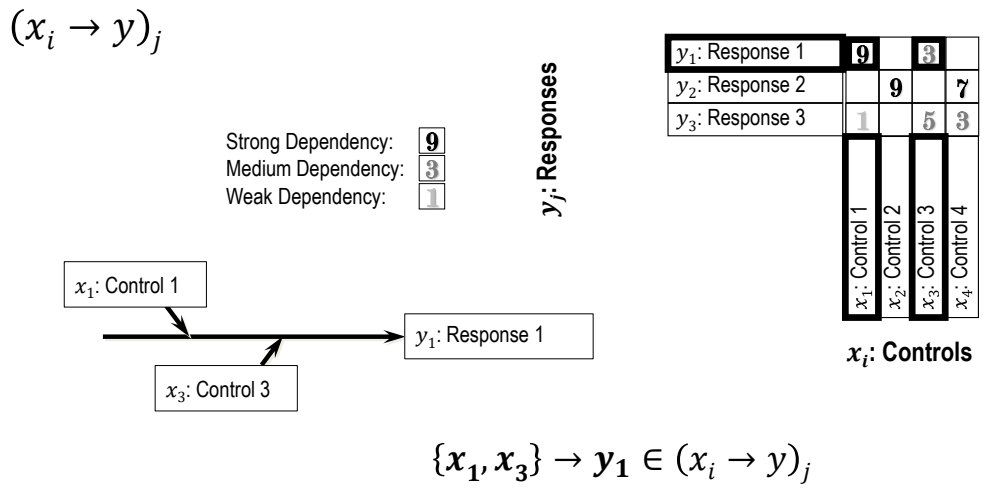
$$\begin{aligned} &\mathcal{G}_0(\mathcal{L}) = \mathcal{L} \\ &\mathcal{G}_{n+1}(\mathcal{L}) = \mathcal{G}_n(\mathcal{L}) \cup \left\{\{a_1, \dots, a_m\} \to b \middle| a_1, \dots a_m, b \in \mathcal{G}_n(\mathcal{L})\right\} \\ &m = 0,1,2,3 \dots \end{aligned} \tag{3}$$

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

$$\mathcal{G}(\mathcal{L}) = \bigcup_{n \in \mathbb{N}} \mathcal{G}_n(\mathcal{L}) \tag{4}$$

The elements of $\mathcal{G}_n(\mathcal{L})$ are arrow terms of level $n$. Terms of level $0$ are *Topics*, terms of level $1$ are *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. The rules correspond to the cause/effect correlations in the QFD–matrix.

**Figure 3.** *Representing QFD Matrices as Rule Sets*




Arrow terms 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 is considered the initial form of QFD matrices. Converting a series of Ishikawa diagrams into a QFD matrix is straightforward, see Figure 3 below.

*Arrow Term Notation*

To avoid the many set-theoretical parentheses, the following notations are applied:

- $a_i$ for a finite set of arrow terms, $i$ denoting some finite indexing function for arrow terms;
- $a_1$ for a singleton arrow terms; i.e. $a_1 = \{a\}$ where $a$ is an arrow term;
- $\emptyset$ for the empty set, such as in the arrow term $\emptyset \to a$.

Note that $a_i$ can be empty. The indexing function cascades, thus $a_{i,j}$ denotes the union of a finite number of $m$ arrow term sets

$$a_{i,j} = a_{i,1} \cup a_{i,2} \cup \ldots \cup a_{i,j} \cup \ldots \cup a_{i,m} \quad (5)$$

With these writing conventions, $(x_i \to y)_j$ denotes a rule set; i.e., a finite set of arrow terms having at least one arrow. Thus, they are level 1 or higher. Each element $x_i \to y$ of $(x_i \to y)_j$ denotes one Ishikawa diagram (Akao, 1990), which is a cause/effect constituent of a QFD deployment and stands at the origins of QFD in Japan. The matrix $(x_i \to y)_j$ represents the QFD deployment. This matrix obviously is a rule set within $\mathcal{G}(\mathcal{L})$. The union of all possible QFD matrices is infinite and therefore a knowledge base in $\mathcal{G}(\mathcal{L})$.

Many elements of $\mathcal{G}(\mathcal{L})$ that do not resemble known QFD deployments, such as

$$\left((x_i \to y)_j \to z\right)_k \quad (6)$$

This is a finite set of arrow terms whose left hands consist of finite rule sets.

Another such example is $x_i \to (y_j \to z)$. This is a cascade of rules.

Subsequently, the association to the right for arrow terms is used:

$$x_i \to y_j \to z = x_i \to (y_j \to z) \tag{7}$$

*The Application Operation*

Let $M, N \in \mathcal{G}(\mathcal{L})$. Application of $M$ to $N$ is defined by

$$M \bullet N = \{a | \exists b_i \to a \in M, b_i \subset N\} \tag{8}$$

In case of $M$ as a rule set, and $N$ as a topic set, this represents the selection operation that chooses those rules $(b_i \to a)_j$ from rule set $M$ that are applicable to the topic set $N$. The definition applies to all higher–level $\mathcal{G}(\mathcal{L})$ terms as well. As an example, consider $M = (b_i \to a)_j$ and another rule set $L = (d_k \to c)_p$, which represent two QFD deployments.

Then,

$$M \bullet L = \{a | \exists (d_k \to c)_j \to a \in M, (d_k \to c)_j \subset L\} \tag{9}$$

Moreover,

$$(M \bullet L) \bullet N = \{f | \exists g_j \to f \in M \bullet L, f_j \subset N\} \tag{10}$$

and thus

$$(M \bullet L) \bullet N = \{f | \exists (d_k \to c)_j \to g_i \to f \in M, (d_k \to c)_i \subset L, g_i \subset N\} \tag{11}$$

Therefore, $M$ selects those rules from $L$ that are applicable to the topics in $N$.

Furthermore,

$$L \bullet (M \bullet N) = \{a | \exists b_i \to a \in L, b_i \subset M \bullet N\} = \{a | \exists b_i \to a \in L, \exists (c_j \to b)_i \subset M, c_{j,i} \subset N\} \tag{12}$$

Thus, the result of $L \bullet (M \bullet N)$ are those elements of $N$ that have both a rule $b_i \to a \in L$ and for each $b_i$ there are rules $(c_j \to b)_i \subset M$. Applying a rule set selects those rules that apply to a given topic. Therefore, the application of graph terms represents the combination of the QFD matrices as used in Comprehensive QFD (Akao, 1990). The defined application operation is natural and of practical relevance. In terms of QFD, this operation selects such cause/effect relations that apply to a given topic. Combinatory logic provides a model for applying QFD.

*The Significance of Combinatory Logic*

Since rule sets represent knowledge, combinatory logic is capable to represent knowledge in a mathematically strict way. Knowledge is unlimited and can always be extended; however, knowledge is constructive. It is not something that exists somewhere in the clouds where you can pick it; knowledge rather needs being constructed in some strictly logical way.

**Representing Unlimited Knowledge**

Rule sets represent things that organize themselves such as cars that drive automatically, flying drones that find the way to its target, and smart homes that save energy. These things typically acquire knowledge while they are in operations. Predicting their behavior is ultimately impossible without representing the knowledge acquisition during operations.

Interestingly, agile software development works the same way: exact specifications are unknown at the beginning. While software is developed together with the stakeholders, more and more the ultimate result becomes apparent. Combinatory logic is possibly the way of choice for controlling agile software development.

*Parallel Computing*

Rule sets are of unlimited size but well structured. Moreover, if the base set represents QFD relationship matrices, they carry associated metrics, namely the convergence gap. QFD relies on measuring cause-effect relationship. For software, there are various measures that can be applied: functional size, security, safety, cost, and non-functional metrics such as ease-of-use. The IoT consists of things made intelligent by software, connected by software and acting autonomously by software. This is called an *IoT Concert*. Organizing an IoT concert, called *IoT Concertation*, requires software metrics, enabling combinatory logic for IoT. Based on software metrics, two arrow terms describing the software can be compared with respect to size, to defect density and compared with respect to behavior towards the same goal.

Behavior of an IoT concert changes when the environment changes – adding or removing things might change, or even create totally new behavior. Totally unexpected situations might emerge on streets driven by autonomous cars. The rule set is not completely known at any time; however, directed by metrics, a sufficiently good approximation can be built just when needed.

Implementing a rule set is by constructing an automaton that eventually produces all its elements. The arrow term notation (0) describes the algorithm needed for the automaton. The automatons produce arrow terms in parallel and in any order, without knowing much from each other. To make them useful, the automatons needs guidance.

*The Schurr-Radius*

The trick is combining the strict and well-known structure of a rule set with the convergence gap. The rule set can be constructed by an automaton that produces each element eventually after some time. If that automaton can be directed to produce the arrow terms closing the convergence gap, it is possible to do this in a predictable time (Fehlmann, 1981). The arrow terms arise from asking the components of an IoT concert how they behave in some given circumstances. Asking the right question will do:

$$\{(a_i \to b)_j \| \|b_j - \boldsymbol{\tau}_y\| < \varepsilon\} \quad (13)$$

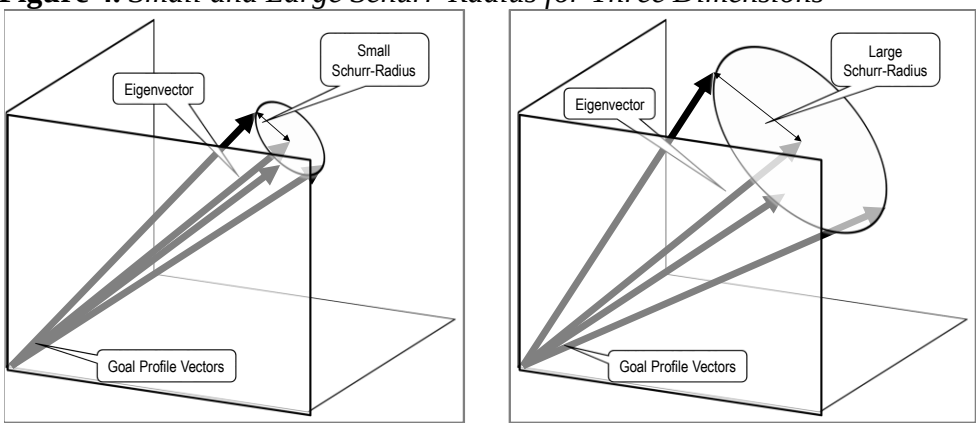
where $\boldsymbol{\tau_y}$ is the *Goal Profile*, representing the target for the circumstances under investigation. For instance, $\boldsymbol{\tau_y}$ could represent the condition that the autonomous car avoids crash. Then, equation (13) represents all crash-free conditions reachable by the automatons.

In QFD, the terms to consider depend from the goal. It must be known what the goal of the behavior is: doing no harm or minimizing it for autonomous cars, minimizing energy consumption in intelligent homes, avoid crashing for flying drones. On the other hand, testing aims at finding fault conditions.

Rule sets in QFD are also solution topic vectors. They have a convergence gap against the goal response vector. This convergence gap can be used to control the automaton producing the rule set by focusing on the goal profile vector.

Figure **4** (Schurr, 2011) demonstrates the convergence gaps for three dimensions. Higher dimensions are more difficult to visualize but equally simple to calculate.

**Figure 4.** *Small and Large Schurr-Radius for Three Dimensions*




Let $\Delta_1, \Delta_2, \ldots, \Delta_n$ be these differences, namely the convergence gaps between eigenvector and the solution topic vectors in the rule set.

Then the formula resembles the standard deviation $\sigma$, known from statistical methods:

$$Schurr\ Radius = \sqrt{\frac{\sum_{i=1..n} \Delta_i^2}{n-1}} \tag{14}$$

The *Schurr Radius* is the envelope around the convergence gaps, an indicator for the total of variations within rule sets (Schurr, 2011).

## Testing the Internet of Things

An immediate application of combinatory logic is in testing. Test cases have the same structure of arrow terms. The arrow terms represent tests; in $a_i \to b$, the $a_i$ describe the test data and $b$ the test response. Responses can be as simple as the amount of impact on the actuators in an IoT orchestra.

The goal of testing is measuring the Schurr radius. The necessity for test cases produced automatically in IoT is apparent. There are no testers present when users connect a new sensor to their smart home network, or two autonomous cars meet each other. Behavior of the newly connected system still must remain safe. Test case automation is a long-standing need for testers; for IoT concerts, combinatory logic delivers automated test cases almost for free.

*A Simple IoT Testing Case*

The mechanisms in place are shown with a simplified IoT network. Consider a simple data retrieval application. The application meets two functional (FUR) and two non-functional (NFR) with the following goal profile, see
Figure **5**:

**Figure 5.** *Customer's Needs' Priority Profile*

| | IoT Topics | Attributes | | | Weight | Profile |
|---|---|---|---|---|---|---|
| FUR | y1 Extensible | Easy to extend IoT | Device independent | Flexible | 31% | 0.57 |
| | y2 Open | Open Source | Open Interfaces | | 20% | 0.36 |
| NFR | y3 Reliable | Always correct | Always secure | Safe | 39% | 0.71 |
| | y4 Fast | No waiting | | | 11% | 0.20 |

Only three user stories are needed to cover these requirements, see
Figure **6**:

**Figure 6.** *User Stories' Priority Profile for Simple Data Retrieval Application*

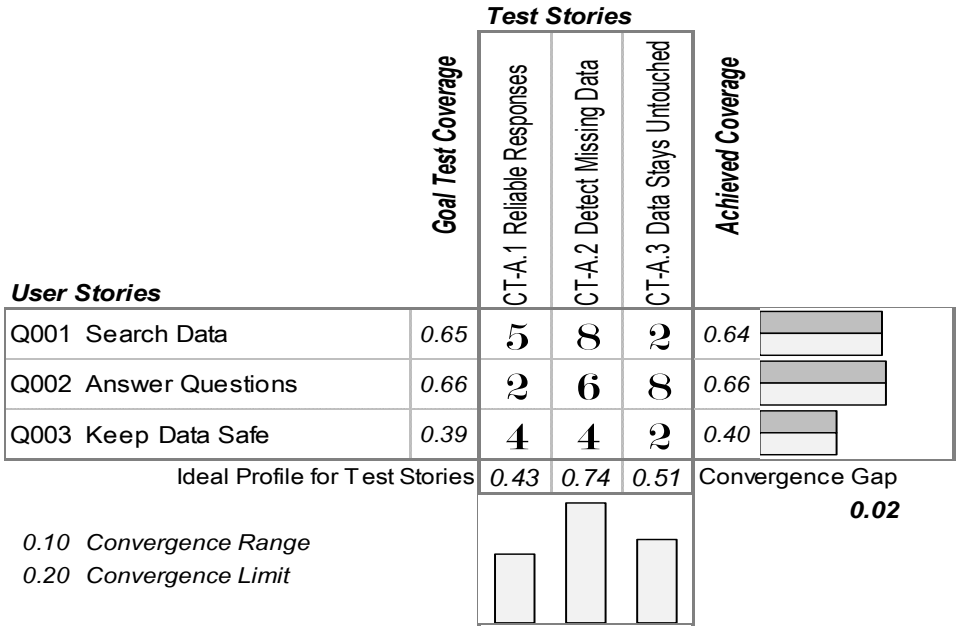
| | User Stories Topics | Weight | Profile |
|---|---|---|---|
| 1) | Q001 Search Data | 38% | 0.65 |
| 2) | Q002 Answer Questions | 39% | 0.66 |
| 3) | Q003 Keep Data Safe | 23% | 0.39 |

The priority profile reflects the number of data movements needed in the software to cope with the user requirements expressed in user stories.

The total functional size according to ISO/IEC 19761 COSMIC is 6 CFP, i.e., six data movements only; thus this is a very small and simple application. The user stories' profiles reflect customer's needs for the IoT Topics shown in
Figure **5** by transfer functions as introduced in section 0. User stories' priority profile is simply calculated by counting the number of data movements needed per user story to meet the customer's needs' priority profile.

The test coverage transfer function in
Figure **7** is defined by the number of data movements in a test story delivering user stories.

**Figure 7.** *Test Coverage for Simple Data Retrieval Application*




Coverage is fine with a convergence gap of 0.02. The total number of tested data movements per cell never exceeds eight.

*Connecting IoT Devices to the Database Application*

Connecting IoT devices to a simple data retrieval application adds not only a continuous flow of searchable data but also considerable complexity. By adding one type of sensor and one type of actuator, the functional size almost triples and becomes 22 CFP. Security and safety risks increase with every data movement added to the IoT concert, as they can be misused or hacked, and cause unwanted and unsafe behavior.

Customer's needs remain the same – for simplicity, we do not consider additional needs that arise with IoT operations. Also, user stories remain the same, although data now refers not to static but to dynamic data and the priority profile now changes towards higher importance for *Q003: Keep Data Safe*. Test stories too remain the same but must cover many more data movements. Consequently,

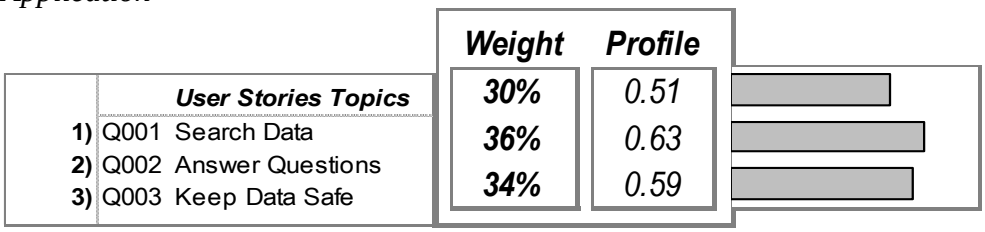
Figure **5** remains valid while

Figure **6** changes its priority profile after connecting the database to the IoT concert because of the additional data movements between devices, database, sensors and actuators.

Figure **6** transforms into

Figure **8** with more focus on *Q003: Keep Data Safe:*

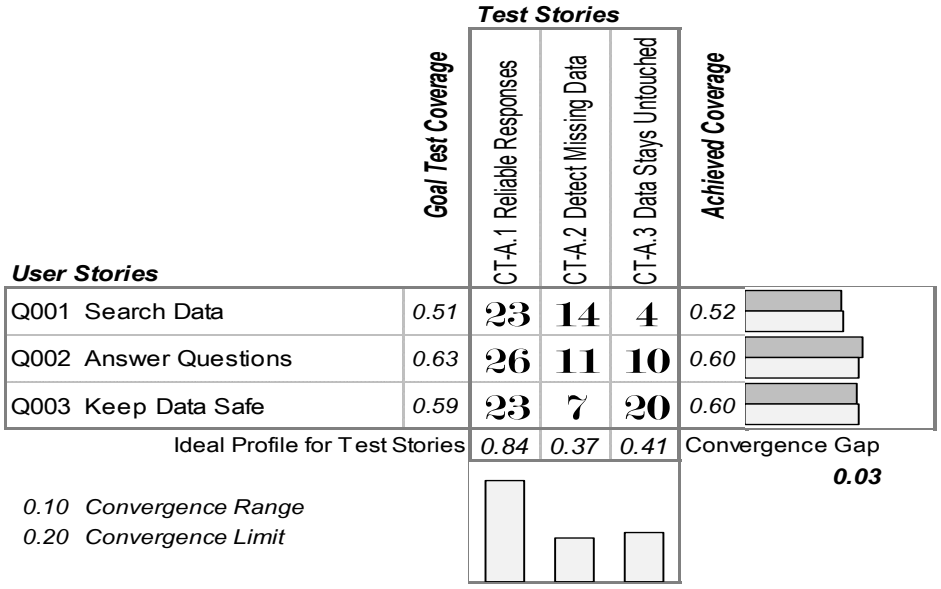
**Figure 8.** *User Stories' Priority Profile for Full IoT Data Retrieval Application*




Consequently, test cases increase in number. For instance, to keep data safe (*Q003: Keep Data Safe*), data transmissions to sensors and actuators must be tested against loss of data, or data transmission interference, e.g., by hackers. This increases test size but not the number of test stories.

The resulting test coverage (Figure 9) is expected to remain the same although test size increases considerably. This means that many more data movements are now under test; however, the test structure remains the same. The knowledge for testing the IoT is inherited from the original tests for the simple data retrieval test scenario.

**Figure 9.** *Test Coverage for Full IoT Data Retrieval Application*




Clearly, both test coverage transfer functions remain within a safe Schurr Radius. Adding more types of IoT devices causes the cell counts grown in the test coverage matrix while the convergence gap remains within the Schurr Radius limits. This is what Combinatory Logic predicts. Thus, the original data retrieval application test serves as a model for the full IoT test. Adding more devices to the IoT concert makes the counts grown even further, but the matrix remains and the testing coverage convergence gap remains within the Schurr Radius. Only one rule set has been applied so far: $(x_3 \rightarrow y)_3$. If the IoT concert covers more user stories, say $j$, then this becomes $(x_3 \rightarrow y)_j$; what in turn most likely requires $i$ more test stories: $(x_i \rightarrow y)_j$.

The importance of the original three test stories changed between the data retrieval application and the full IoT concert, see

Table **10**.

**Table 10.** *Test Priority Change when Adding IoT Concert*

| Test Story | Data Retrieval Weight | Data Retrieval Profile | | Full IoT Concert Weight | Full IoT Concert Profile | |
|---|---|---|---|---|---|---|
| CT-A.1 Reliable Responses | 25% | 0.43 | 11 | 49% | 0.84 | 72 |
| CT-A.2 Detect Missing Data | 44% | 0.74 | 18 | 21% | 0.37 | 32 |
| CT-A.3 Data Stays Untouched | 30% | 0.51 | 12 | 24% | 0.41 | 34 |

While test priority initially focused on *CT-A.2: Detect Missing Data*, focus moved to *CT-A.1: Retrieve Previous Responses* after adding sensors and actuator to the system. This is reflecting the additional effort needed to protect data movements between sensors and database from interferences, e.g., data loss or compromise.

The following table (Table 11) shows a comparison of test sizes between the original data retrieval application test, and the full IoT concert test.

**Table 11.** *Simple Data Retrieval Application Test Size vs. IoT Test Size*

| Data Retrieval | | Full IoT Concert | |
|---|---|---|---|
| Test Size in CFP: | 41 | Test Size in CFP: | 138 |
| Test Intensity in CFP: | 6.8 | Test Intensity in CFP: | 6.3 |
| Defect Density: | 17% | Defect Density: | 18% |
| Test Coverage: | 100% | Test Coverage: | 100% |

The key indicator for tests is the *Test Intensity*, the ratio between *Test Size* and *Functional Size*. *Defect Size* in turn is the percentage of defective data movements in the software. All size measurements are according the international standard ISO/IEC 19761 COSMIC (ISO/IEC 19761:2011, 2011). There are no limits for neither functional size nor test size.

*Automated Test Case Generation*

Thanks to the test priority goal profile, derived from the original customer's needs and carried through user stories to test stories finally, it is possible to generate test cases automatically because the convergences gap serves as guidance which tests cases to add. The principles behind artificial intelligence are heuristics, i.e., metrics telling which search branches to follow and which to avoid.

Because of the guidance, artificial intelligence adds only test cases that pertain to the functionality of the implemented user stories, notwithstanding whether the IoT concert now features additional but untested functionality. The data retrieval approach does not cover additional requirements that might come with the IoT concert, such as whether window stores close when sun shine is strong, or avoid car collisions when needed. It is therefore necessary to extend the model, for instance from the simple data retrieval application to something more sophisticated such as a smart home, or autonomous cars. Then, combinatory logic allows the extension of the test

suite from the model that still remains relatively simple, to the full-blown reality covering a much wider range of functionality and that is much more difficult to manage.

Automated testing is a must for IoT systems especially for autonomous cars. For them, allotted testing time can be very small. For instance, in case of an encounter with another car from a different manufacturer that wants to connect and whose behavior is hardly predictable, testing time allowance might be reduced to a few milliseconds.

*Conclusions and Next Steps*

We demonstrated with an example how testing scenarios carry over from simple applications to complex IoT concerts, using the original test cases as testing patterns for automatically extending the test to the full IoT application. Using combinatory logic, testing scenarios designed for the original model carries over to its extended IoT implementation, and this is already an important saving, enabling safe IoT concertation.

Combinatory logic paves the way to testing complex IoT concerts and networked systems, based on the solid ground of existing testing experiences. It has been shown how the quality of testing can be maintained even after moving to automated testing. For testing the IoT, this approach offers significant savings; however, the full potential of combinatory logic in organizing knowledge is significantly greater.

Using combinatory logic for testing obviously is a high hurdle for testers that neither understand eigenvectors nor combinatory logic. Moreover, the approach relies on using customer's needs profiles and software metrics for sizing and evaluating tests. Metrics are not common practices in today's software communities. To help with this, tools must be constructed that make the theoretical background available for practitioners. Such a tool has been presented in a workshop at the QA & Test conference in October 2016 in Bilbao, Spain. Other application of the theory might become apparent since Information & Communication Technology (ICT) undergoes currently a period of cosmic inflation, opening new fields and expanding new practices at an astonishing pace.

## References

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

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

Barwise, J. et al., 1977. *Handbook of Mathematical Logic.* Studies in Logic and the Foundations of Mathematics ed. Amsterdam, NL: North-Holland Publishing Company.

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., 1981. *Theorie und Anwendung der Kombinatorischen Logik,* Zürich, CH: ETH Dissertation 3140-01.

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. & Kranich, E., 2016 (to appear). *Managing Complexity - Uncover the Mysteries with Six Sigma Transfer Functions.* Berlin: Logos Verlag.

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

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

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.

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

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

Levie, R. d., 2012. *Advanced Excel for Scientific Data Analysis.* 3rd Edition ed. Orrs Island, ME: Atlantic Academic LLC.

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.

Schurr, S., 2011. *Evaluating AHP Questionnaire Feedback with Statistical Methods.* Stuttgart, Germany, 17th International QFD Symposium, ISQFD 2011.

The R Foundation, 2015. *The R Project for Statistical Computing.* [Online] Available at: http://www.r-project.org [Accessed 16 April 2015].

Volpi, L. & Team, 2007. *Matrix.xla.* [Online] Available at: http://bit.ly/2gF5IJa [Accessed 29 March 2015].