Athens Journal of Sciences- Volume 5, Issue 3 – Pages 247-260

Computer Science and Digitalization

By Thomas Fehlmann[*]

How does Digitalization affect Computer Science? What is its impact? Does it change the basics, or is "Digitalization" just another buzzword used to attract attention? First, digitalization is a fact, describing a long-standing trend that for some time was hard to understand by people outside the ICT industry. It describes the growing possibilities to make physical "Things" intelligent, in the sense that they got cheap, little sensors for their physical environment, connect to the Internet, heavily depend on software for operations, and talk to cloud services collecting data, giving instructions and coordinate physical events. Digitalization causes products to become software-intense. This paper investigates the actual trends in product design and improvement. It proposes how to migrate the concepts of the past regarding software quality and quality measurement into the near future, where the software running the product needs to become safe, and safety needs to become measurable, such that consumers can take informed and responsible decisions. Also, privacy is put in jeopardy, much more than with traffic prediction systems that tracked our car's navigator to predict traffic jams.

Keywords: *Autonomous Vehicles, Combinatory Logic, Consumer Metrics, Defect Density, Internet of Things, Metrics for Software, Product Privacy, Product Safety, Software Testing, Test Metrics.*

Introduction

Digitalization has become a buzzword for many things – new, disruptive technologies that change the business model for many enterprises; the use of embedded software to improve features and characteristics of products that otherwise look physical, or mechanical; the move from personal service, for instance in travel agencies, to online channels; but also, the widespread use of tracking technologies when for instance physically visiting a shop. Like in an online shop, visitors are traced using GPS and their smartphones' WiFi, the time is measured they spend at certain offers, the sequence of offers they consider is tracked, and finally compared to their effective purchases.

The main change to computer science is that such software impacts the physical world directly, without a human intermediate. In the past, an engine driver got instructions from a software-based train control system and acted with discernment. Now, the software takes over train control and there is no human in between who is responsible for relevant events. Similar with cars, which in the past were equipped with navigation systems, and now the navigation system's successors take over the steering wheel, for dedicated routes.

Much of such aspects of digitalization remains hidden from the public, giving the notion a somewhat conspiratorial meaning and creating fear and distrust

[*]Senior Researcher, Euro Project Office AG, Switzerland.

https://doi.org/10.30958/ajs.5-3-3 doi=10.30958/ajs.5-3-3

against computer science and modern technology. Conservativism gains popularity because people do not know what digitalization entails.

Some confound digitalization with globalization, yet another term that creates fear and calls for populistic solutions that cause extensive damage to societies and economy. Recently, it has become apparent that today's big data algorithms can impact US elections, and British leavers. People's liberty, and the rule of democracy, is in jeopardy because of these recent technologies.

It is therefore high time that computer scientists, especially when engaged in education, look at the various facets of digitalization and explain what it really means, and what the said algorithms are all about.

The Impact of Software on Personal Liability

There are three major discoveries; all made in the 20th century, that today every single person must understand to cope with digitalization:

- The Russel Paradox;
- The Fast Fourier Transform (FFT);
- Eigenvector Theory and Transfer Functions.

For a self-determined citizen of the 21st century, it is necessary to understand what these discoveries mean, and what impact they have on society. This is within reach of every college-level student and business professional. It is worth the effort, because it is the key for people mastering digitalization.

The Russel Paradox

In 1901, Bertrand Russel detected that any formalization of naïve set theory led to a contradiction (Fraenkel et al., 1973). The impact of this discovery is huge but contra intuitive. In intuitionistic interpretation[1], any system that describes a logic for objects without giving strict construction rules for them contains contradictions. The 20th century, but also the past, is full of vain attempts to construct righteous social systems that ended up in horrible contradictions against what was intended. Lawmakers are still inclined to try even more complicated regulatory systems because they do not understand the impact of the Russel paradox. Rule-based systems need arbitration in case its rules no longer apply or no longer provide the intended results. The arbitration process is more important than the rules itself.

[1]Intuitionism is an approach where mathematics is purely the result of the constructive mental activity of humans rather than the discovery of fundamental principles claimed to exist in an objective reality. That is, logic and mathematics are not considered analytic activities wherein deep properties of objective reality are revealed and applied but are instead considered the application of internally consistent methods used to realize more complex mental constructs, regardless of their possible independent existence in an objective reality. *Intuitionism.* [Online] Available at: https://bit.ly/1NjIEsI. [Accessed 8 March 2018].

This holds as well for software. You cannot decide Turing's Halting Problem (Turing, 1937). Consequently, no test suite can exist that, after applying all test cases to the system, will let us unambiguously determine whether the system consistently meets any given specification. The only certainty a test can assert is that all test cases constructed for all test stories found were passed. From this, it is not possible to derive that the system works always correctly.

However, if no system is completely testable, but some systems can do severe harm to humans or things, suppliers and programmers of such systems cannot make liable for all actions of such a system. A consequence of the Russel paradox is that no software-controlled system ever will be able to relieve humans from all responsibility for running such a system. In contrary, these systems will never replace humans but require even more knowledge ability and technical comprehension from its users.

The Fast Fourier Transform

Everybody assumes it for granted that digital phones, radio, music, and video can be transmitted by the Internet's digital channels. But how is it done?

A *Fast Fourier Transform* (FFT) is an algorithm that samples a signal over a period of time, or space, and divides it into its frequency components, or its inverse[2].

By around 1977, the first chip appeared based on an algorithm that transferred analog audio signals to digital signatures, developed already a few years before (Cooley and Tukey, 1964). This algorithm produced a transfer function from digital controls to analog output that was fast enough. Thus, its speed allows for immediate audio rendering when digital to audio transfer functions are calculated. Transfer must go in both directions, for recording and for rendering audio signals.

The digital controls use a functional algebra with sinus functions as vector base. The coefficients for that vector base constitute the digital signature of the audio signal that can be stored on digital devices. The process of finding these coefficients is called *Sampling*. Audio rendering produces sound output at certain wavelengths, and the condition is that the output produced is similar enough to the original input converted into a digital signature.

Eigenvector Theory and Transfer Functions

Eigenvector theory is another achievement of linear algebra that changed the world; however, not before the start of the 21st century. For instance, eigenvector theory is employed in the *Analytic Hierarchy Process* (AHP) developed by Saaty (1990) to solve multi-criteria decision problems, in the football teams ranking method proposed by (Keener, 1993) or, more recently, for powering Google's PageRank algorithm (Langville and Meyer, 2006). The name 'PageRank' refers to Larry Page, one of the founders of Google. The eigenvector theory is now widely in use and paved the way to the Internet age.

[2]*Fast Fourier Transform.* [Online] Available at: https://bit.ly/1KA8K3A. [Accessed 16 April 2018].

To explain eigenvectors and eigenvalues, let $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ be an arbitrary square matrix. A non-zero vector $\boldsymbol{x}$ is called an *Eigenvector* of the matrix $\boldsymbol{S}$, if $\boldsymbol{x}$ and the vector $\boldsymbol{Sx}$ are in the same direction; i.e., eigenvectors are the directions which are invariant under the transformation $\boldsymbol{S}$.

The *Eigenvalue* λ reveals whether the vector $\boldsymbol{Sx}$ remains unchanged ($\lambda = 1$), is reversed in direction ($\lambda < 0$), is shrunk ($0 < |\lambda| < 1$) or stretched ($|\lambda| > 1$). Thus, the fundamental equation to solve is

$$\boldsymbol{Sx} = \lambda \boldsymbol{x} \tag{1}$$

Real transfer functions are not square matrices. Let $\boldsymbol{A}$ be a linear multiple response transfer function $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with a response profile $\boldsymbol{y} \in \mathbb{R}^{m}$. How can equation (1) solve $\boldsymbol{y} = \boldsymbol{Ax}$, to calculate a solution profile $\boldsymbol{x} \in \mathbb{R}^{n}$?

First, transpose the matrix $\boldsymbol{A}$. This yields the transpose $\boldsymbol{A}^{\mathrm{T}}$. Second, calculate the combined symmetrical square matrix $\boldsymbol{AA}^{\mathrm{T}}$, by using matrix multiplication.

Following Fehlmann and Kranich (2011), a response profile $\boldsymbol{y}$ can be determined by solving the following eigenvector respectively eigenvalue problem:

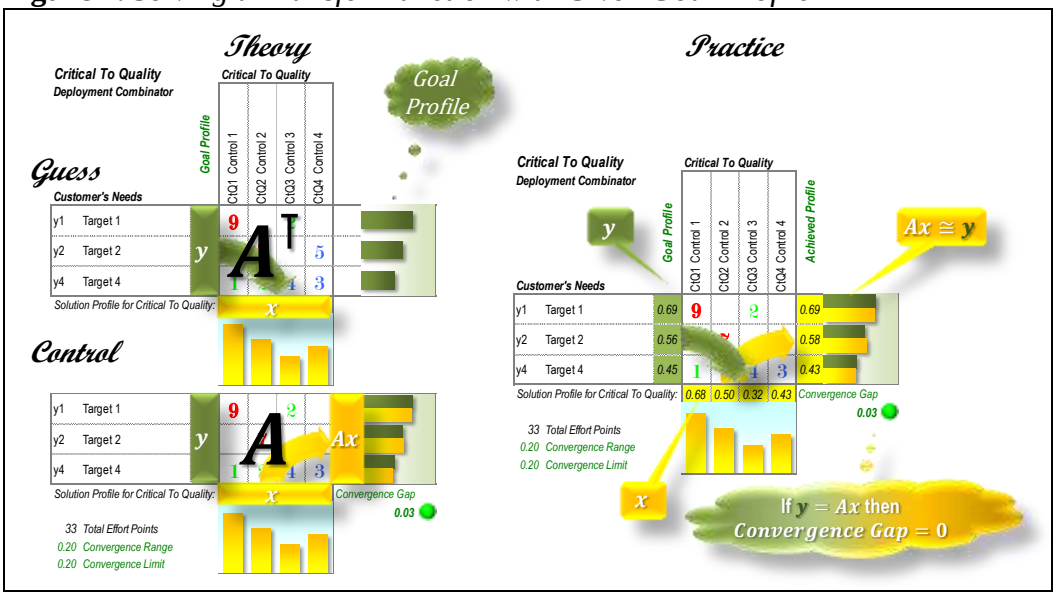
$$\boldsymbol{AA}^{\intercal}\boldsymbol{y} = \lambda \boldsymbol{y} \tag{2}$$

Obviously, the matrix $\boldsymbol{AA}^{\intercal} \in \mathbb{R}^{m \times m}$ is symmetric; i.e. $\boldsymbol{AA}^{\intercal} = (\boldsymbol{AA}^{\intercal})^{\intercal}$. This matrix has exactly m (not necessarily) distinct real eigenvalues. There exists a set of m real eigenvectors, one for each eigenvalue, which are mutually orthogonal and thus linear independent (see Fehlmann, 2016) and the literature; e.g., (Kressner, 2005). If the matrix $\boldsymbol{AA}^{\intercal}$ is positive definite; i.e., $\boldsymbol{x}^{\intercal}\boldsymbol{AA}^{\intercal}\boldsymbol{x} > \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathbb{R}^{n} \setminus \{\boldsymbol{0}\}$ in (2), the *Theorem of Perron-Frobenius* states that $\boldsymbol{AA}^{\intercal}$ has a *Principal Eigenvector* $\boldsymbol{y_E}$ with $\boldsymbol{y_E} = \boldsymbol{AA}^{\intercal}\boldsymbol{y_E}$. The principal eigenvector is all positive; while all others have negative components.

Using that 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*; $\boldsymbol{x_E} = \boldsymbol{A}^{\intercal}\boldsymbol{y_E}$ is called the *Eigencontrols* of $\boldsymbol{A}$. If $\boldsymbol{y}$ happens to be near an eigenvector of $\boldsymbol{AA}^{\intercal}$, $\boldsymbol{x}$ is an approximate solution for $\boldsymbol{Ax} = \boldsymbol{y}$.

Starting with the (green) goal profile in Figure 1, the controls must be guessed using domain expertise. This is the difficult step; the easy step is validating the choice of controls. The (yellow) solution profile can be calculated by the Eigenvector method explained above. The cell elements are measurements, correlating the controls to the targets. Often, frequencies are counted (e.g., for the FFT), or size (e.g., size of tests relating some control to one of the targets). A valid solution is recognized by the small convergence gap. Consequently, finding root causes for observed effects is a matter of trying enough controls, and selecting the best one.

Figure 1. *Solving a Transfer Function with Given Goal Profile*



Every high school graduate and every professional should be able to assess the power of transfer functions, at least in theory. For actual computation, we have computers. Computing the eigenvector is a numerical algorithm available in every statistical or mathematical package (e.g., the *R Project* (The R Foundation, 2015) or in Excel (Volpi and Team, 2007)).

Eigenvector solutions are in wide use today, for search engines such as Google (Gallardo, 2007) or *Big Data* analysis (Li et al., 2015). It is used for analyzing root causes for observed effects. One can observe Social Media and determine the root causes for the posts[3]. Identifying target groups for ad campaigns is easy, even predicting success rates.

A simple application of the principle that can be used for educational purposes is uncovering the business drivers behind *Net Promoter Survey* (NPS) responses (Fehlmann and Kranich, 2014). NPS is the famous two-question survey approach invented by Reichheld, the *Ultimate Question* "Will you recommend?" (Reichheld, 2007).

Eigenvector techniques are not only useful in the context of digitalization; in fact, they were originally invented for other problem. For instance, transfer functions allow measuring controls indirectly that cannot be measured or even observed directly, such as exoplanets' size, orbit, and composite. One can also identify private traits and attributes (Kosinski et al., 2013), even potential voters, based on such analysis[4]. Measuring political tenure from observations and opinions posted in social media is nothing that should surprise a citizen of the 21st century. But it did, because their schools did not prepare them well.

[3]It is hardly conceivable how democracy can survive digitalization without citizens who understand the power of transfer functions.
[4]The now famous (former) Cambridge Analytica Ltd case.

The Need for Risk Containment

Consumers in the 21st century face the problem that more and more functions rely on software and on subsidiary and even on autonomous products and systems that are hard to understand and even harder to control.

Autonomous Vehicles and Automated Driving

To help understanding the systems in use, consumers of autonomous vehicles and other software-intense products need metrics that allow them judging the system's fitness for doing its tasks. These metrics must give confidence to the responsible consumer that his autonomous system has been, for example, tested for protecting her or his privacy and provides the expected safety. Since consumers bear the ultimate responsibility, never the autonomous system, these metrics matter. Such metrics are the premise sine qua non for autonomous vehicles to go into public operation, and even more necessary for automated driving, when drivers still are present in a car but can leave control to the machine intelligence, for a while on certain routes.

Measuring security, or privacy protection, is not simple. Measuring test coverage for a system using cloud services on containerized virtual machines is far from being current practice; traditionally, testers test code and measure test coverage against code. But there is no code available for these interconnected services. Even if it were, only the functionality used matters. For instance, map services for autonomous vehicles are a very specialized part of the overall map functionality available in the cloud.

Being owner of an autonomous vehicle does imply liability for all damage caused by that vehicle. Bearing that responsibility is not attractive if owners or users have no means to assess the risks involved. Liability cannot be with the car's supplier because of the Russel Paradox.

The Internet of Things (IoT)

More and more "things" become "intelligent – this means, they receive IP addresses and start communicating to the Internet. Intelligent systems adapt to our wishes which they can analyze using transfer functions and the Eigenvector theory. Unfortunately, they also can transmit information about our habits, our health and corporal fitness to all kind of interested parties.

Installing an intelligent home is thus rather unwise unless there are no privacy needs, or the system is well under control regarding privacy. This not only implies protection against intruders but also keeping all rights on the data collected and maintained by the IoT concert. Right on data implies the right to know where what data is stored for which use. Sharing health data with trusted parties, such as with home care for elderly, or ill people, might make sense, but data owners must be able to block such data sharing. Without a measurement device indicating actual privacy levels, data owners are lost.

Even more important is that all analysis based on such data is shared with the data owners, and that data owners can consent.

Social Media

Nowadays, the threat to privacy, and even democracy, by data collected with social media has become widely known. The business model of Internet giants such as Facebook and Google consist of collecting data, analyzing it applying Eigenvector theory, and using it to place personalized advertisement and for other marketing activities.

The problem is less with this business model but with society that does not understand how they operate. It is high time that these simple data analysis methods are widely made known to students and professionals. This is the responsibility of educational institutions.

Keeping Intelligent Things under Control Using Metrics

How can a car be kept under control, even if the car is able to drive at much higher speed than permitted? Stupid, it is the tachometer.

In the same manner, software metrics are needed to keep software-intense things under control. Consumers of such products must be able to get relevant information about their risk exposure. But software metrics is not extremely popular at computer science department, and if so, the focus lies on quality metrics such as compatibility, maintainability, usability, or portability.

Quality of Software

The series of international standards ISO/IEC 25010 (ISO/IEC 25010, 2011) define quality models for software. The *Security* characteristics in the product quality model (ISO/IEC 25010, 2011, p. 10) include confidentiality and authenticity, but physical safety is subsumed under *Functional suitability*, part of functional appropriateness. Even if metrics were available to measure all this, this does not meet the needs of society facing digitalization. And what is the use of quality models without metrics? Digitalization affects two major characteristics of software-intense systems: the Privacy and the Safety.

Physical safety has not been much of a concern in the past, for software developers, as safety was mostly a mechanical characteristic of products. This has changed dramatically with digitalization. With sensors, recognition software based on neural networks and deep learning,[5] software and its behavior can have huge impact on the physical world. Programmers need to keep safety into account as mechanical engineers do (Szegedy et al., 2014).

[5]*Deep learning.* [Online] Available at: https://en.wikipedia.org/wiki/Deep_learning [Accessed 17 April 2018].

Safety is always connected to functionality, although closely linked to "functional suitability, the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions" (ISO/IEC 25010, 2011, p. 10). Measuring safety is standard practice in mechanical environments and is easily transferable to software-intense systems.

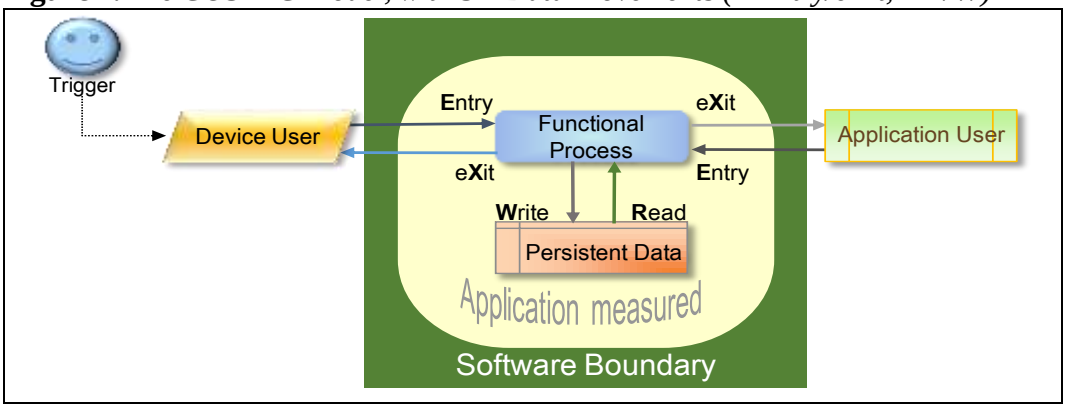
Privacy in turn is a quality characteristic that too relies on some functionality, e.g., for managing encryption keys when moving data from one service to another, but largely is linked to how data is handled when transmitted. Measuring privacy of software in turn is not standard practice and needs consideration.

Both metrics rely on models for the software in use. This is a necessity because software-intense systems rarely consist of one monolithic piece of code alone. Typically, such systems rely on cloud services, embedded software services, mechatronics, and apps interfaces to humans using such systems. Code is only partially available; what matters is functionality.

Software Models

The COSMIC standard (ISO/IEC 19761, 2011) identifies layers (Figure 2). Its boundaries detect the flow of data moving from one object into another; however, the total count does not depend from how boundaries are drawn. Communication between functional processes require typically an **E**ntry and an e**X**it, with a device in between that connects the two processes, regardless of whether data movements cross an application boundary or not. **R**ead and **W**rite relates to moving data in and out of permanent data stores.

Figure 2. *The COSMIC Model, with Six Data Movements (2 **E**ntry/e**X**it, 1 **R**/**W**)*

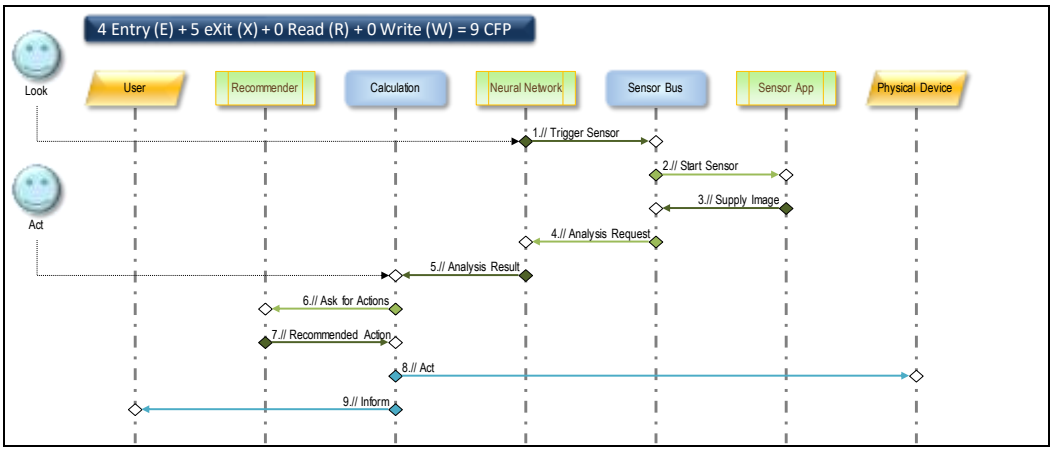


Every data movement transports a *Data Group*, identifying the data moved from one object to another. Obviously, the content of those data groups matters for privacy protection; however, it also can affect safety up to some degree.

An example may show how this model describes a simple *Look and Act Example*, a simplified variant of a visual recognition and guidance system for an autonomous vehicle. Figure 3 shows how COSMIC models software delivering functionality for analyzing an image captured by a sensor. The analysis is performed by a *Neural Network* (Van Gerven and Bothe, 2018). Actions are taken based on a *Recommender* application. The data movements reflect the messages

exchanged. This model works well if the apps run for instance in a *Docker* container (Singh et al., 2018).

Figure 3. *Using Neural Network to Analyze Image, and eventually Act upon Recommendations*



If those apps, functional processes, and devices run on dedicated virtual machines, container apps, they become invulnerable against attacks. The message exchange vulnerabilities in turn can easily be controlled by either using an encryption key management service for encrypting those messages,[6] or even *Blockchain* to additionally ensure traceback for all message exchanges.[7]

Metrics for Privacy & Safety

As examples for the society's needs regarding metrics for software, we present two closely related example using software models. Metrics for privacy can be defined based on the degree of encryption protecting those data movements involved. Obviously, if data is public, no protection is needed. but sensitive data should be encrypted when being moved across objects residing in different containers, or even moved over a data bus, or on media connected to the public Internet.

For consumers, metrics must be intuitive and nevertheless transparent and correct. We therefore propose the following privacy protection categories (Table 1).

[6]For an example, see https://www.kleverkey.com/en/.

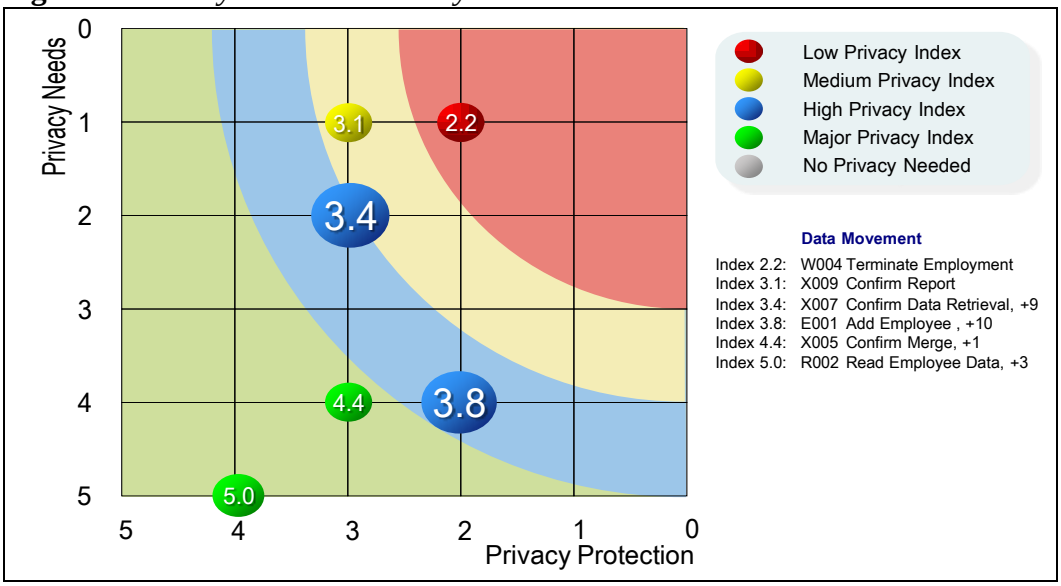
[7]*Blockchain.* [Online] Available at: https://en.wikipedia.org/wiki/Blockchain [Accessed 17 April 2018].

Table 1. *Privacy Assessment Categories*

Privacy Needs	Privacy Protection
Value = 0: No privacy. It's public.	Value = 0: No encryption. It's public.
Value = 1: Disclosure is inconvenient	Value = 1: Weak encryption
Value = 2: Disclosure can be harmful	Value = 2: Strong encryption
Value = 3: Disclosure costs money	Value = 3: Two-way encryption
Value = 4: Disclosure makes guilty	Value = 4: Data never leaves system
Value = 5: Disclosure threatens lives	Value = 5: Container-protected data

Figure 4 shows a proposal how the privacy assessment results for the data movements can be visualized for consumers. The value pair defines the coordinates in the grid, the privacy index is the distance to the <0,0> point. Distance is distorted to avoid privacy threads displayed in the upper left grid square where no privacy needs exist.

Figure 4. *Privacy Needs vs. Privacy Protection*



Distance of the bubbles in the grid (Figure 4) is measured from the starting point $(0,0)$. The *Privacy Index* is in range 0 – 5. Five (5) is the index for maximum privacy; Zero (0) privacy means public data; no privacy granted, or no privacy needed. The privacy index should provide equal length for equal protection. The size of the bubbles represents the number of data movements that lie within this privacy index range.

Safety risks are less difficult to represent. According classical risk management theory (ISO 31000, 2018), risks can be assessed by

- Identifying the risk catalogue
- Classify impact, usually on a scale 0 – 5
- Assigning the probability of risk incurrence

For identifying safety risks in road vehicles, the series of international standards ISO 26262, see (ISO 26262-1, 2011), provide guidance. Currently, the new SOTIF[8] version of the ISO/IEC 26262 is under development. These standards can be used for assessing risks of critical parts; not only mechanical, but also data movements moving critical data groups.

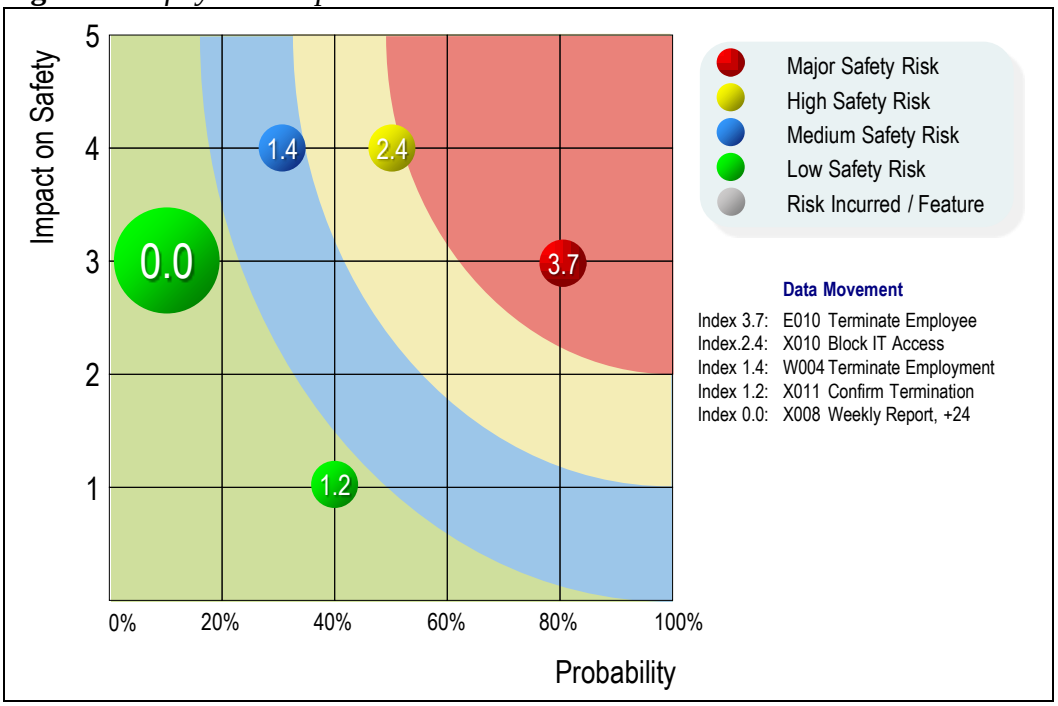
For the graphical representation, we use our distorted Euclidian length again for the positioning of the bubbles. Because distance in the risk grid is measured starting from the $\langle 100\%, 5\rangle$–Point, both grid indices will be mirrored at the grid size value 5, since 100% indicates highest probability and 5 highest impact. Colors should remain the same for the consumer.

Table 2. *Safety Assessment Categories*

Incurrence Probability	Impact
Value = 0%: No risk. It's safe.	Value = 0: None
Value = 20%: Small probability	Value = 1: Low
Value = 40%: Low medium probability	Value = 2: Little
Value = 60%: High medium probability	Value = 3: Medium
Value = 80%: Very high probability	Value = 4: Quite
Value = 100%: Risk incurred already	Value = 5: High

The *Safety Index* graphical representation for consumers looks as follows.

Figure 5. *Safety Risk Exposure*



[8]SOTIF = Safety of the Intended Functionality

Because distance in the risk grid is measured starting from the $\langle 100\%, 5 \rangle$–Point, both grid indices will be mirrored at the grid size value 5, since 100% indicates highest probability and 5 highest impact. Colors should remain the same for the consumer.

Table 2 and Figure 5 have enough similarities to help average consumers understanding the meaning of both indices, such that they can look at both representations together and get a correct impression.

Both metrics are helpful for consumer to master digitalization with products hey use, such as autonomous vehicles or the IoT. They can be used to label software just as today any product is labelled for instance regarding energy efficiency.

Autonomous Real-Time Testing

One of the critical success factors for digitalization is also that learning and evolving systems remain continuously under test. Software tests are no longer done in labs alone; after an update, or after a neural network learned to distinguish new images, it must be tested whether previous functionality still work, and previous images are still recognized correctly. Especially neural networks suffer from strange effects like human neural disorder and retesting them is necessary after each deep learning event (Szegedy et al., 2014).

Consumers who understand the Russell paradox will also be able to correctly assess the need for autonomous real-time testing (Fehlmann and Kranich, 2017). Autonomous real-time testing addresses suppliers' liability issues.

Conclusions

Computer science education has a huge challenge. After decennials of missed opportunities and neglect, time has become tight and skillful teacher are missing. Society is threatened by destructive forces blocking freedom of thought and self-determination, and the lack of understanding of the main drivers of digitalization is just helping them. Or does anybody remember populistic parties asking for better education and more money for schools? They have good reason to block peoples' lives and make them dependent from their big data analytics.

Google, Facebook, and the other IT giants in turn need well-educated people and cannot prosper in a "Good Fellas" environment. Digitalization needs well-educated people that are up to date with their century. They need not an explanation what the Russel paradox is and are aware of big data analytics.

Acknowledgments

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

References

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.

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

Fehlmann, T. M. & Kranich, E., 2011. *Transfer Functions, Eigenvectors and QFD in Concert.* Stuttgart, Germany, QFD Institut Deutschland e.V.

Fehlmann, T. M. & Kranich, E., 2014. *Uncovering Customer Needs from Net Promoter Scores.* Istanbul, Turkey, 20th International Symposium on Quality Function Deployment.

Fehlmann, T. M. & Kranich, E., 2017. *Autonomous Real-time Software & Systems Testing.* Göteborg, s.n.

Fraenkel, A., Bar-Hillel, Y. & Levy, A., 1973. *Foundations of Set Theory.* Amsterdam et.al.: Elsevier.

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

ISO 26262-1, 2011. *Road vehicles - Functional Safety - Part 1: Vocabulary,* Geneva: ISO/TC 22/SC3.

ISO 31000, 2018. *Risk management — Guidelines,* Geneva, Switzerland: ISO/TC 262.

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

ISO/IEC 25010, 2011. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,* Geneva, Switzerland: ISO/IEC JTC 1/SC 7.

Keener, J. P., 1993. The Perron-Frobenius Theorem and the Ranking of Football Teams. *SIAM Review,* 35(1), pp. 80-93.

Kosinski, M., Stillwell, D. & Graepel, T., 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences of the United States of America,* 110(15), pp. 5802-5805.

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

Langville, A. N. & Meyer., C. D., 2006. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton, NJ: Princeton University Press.

Li, K.-C., Jiang, H., Yang, L. T. & Cuzzocrea, A., 2015. *Big Data: Algorithms, Analytics, and Applications.* Boca Raton, FL: CRC Press.

Reichheld, F., 2007. *The Ultimate Question: Driving Good Profits and True Growth.* Boston, MA: Harvard Business School Press.

Saaty, T. L., 1990. *The Analytic Hierarchy Process – Planning, Priority Setting, Resource Allocation.* Pittsburgh, PA : RWS Publications.

Singh et al., 2018. *Docker overview.* [Online] Available at: https://dockr.ly/2sZTBJS. [Accessed 9 April 2018].

Szegedy, C. et al., 2014. *Intriguing properties of neural networks.* [Online] Available at: https://bit.ly/2PNYi3L [Accessed 8 March 2018].

The R Foundation, 2015. *The R Project for Statistical Computing.* [Online] Available at: https://bit.ly/19WExR5 [Accessed 27 May 2016].

Turing, A., 1937. On computable numbers, with an application to the Entscheidungs problem. *Proceedings of the London Mathematical Society,* 42 (Series 2), p. 230-265.

Van Gerven, Marcel; Bothe, Sander, 2018. *Artificial Neural Networks as Models of Neural Information Processing,* Lausanne: s.n.

Volpi, L. & Team, 2007. *Matrix.xla.* [Online] Available at: https://bit.ly/2gF5IJa [Accessed 27 May 2016].