COMPLEX SYSTEM TESTING BASED ON AHP AND QFD

Dr. Thomas Fehlmann, Euro Project Office AG, Six Sigma for Software

Giblenstrasse 50, 8049 Zürich, Switzerland. e-mail: thomas.fehlmann@e-p-o.com

Eberhard Kranich, Euro Project Office Germany, Statistical Methods

47051 Duisburg, Germany, e-mail: eberhard.kranich@e-p-o.com

Paper Type: Scientific Paper

Abstract

Purpose: The 4th Industrial Revolution started with the advent of cyber-physical products such as medical instruments or autonomous vehicles. Such products require sophisticated methods of testing and certification for safe public usage. This is less easy than before. Software, which is the predominant system component, is updated and learns new behavior during operations. *Autonomous Real-time Testing* (ART) addresses this problem by testing continuously, even during operation. ART is based on the Analytic Hierarchy Process (AHP), for understanding users' needs, and Quality Function Deployment (QFD), used to select relevant test cases.

Methodology/ Approach: ART relies on combinatory logic for generating test cases, and test automation implemented by Kubernetes for executing tests. Since combinatory logic generates a potentially unlimited number of new test cases, all valid, a method is needed to select relevant test cases by measuring their relevance. Voice of the Customer (VoC), AHP and QFD, as described in the international series of standards ISO/IEC 16355, deliver the necessary metrics.

Findings and Research Limitation: Six Sigma Transfer Functions and QFD yield the criteria needed for executing relevant system tests in real-time. The method can be implemented by Kubernetes using Digital Twins for hardware in the loop.

Originality/ Value: The approach combines software metrics, mathematic logic, and big data respectively *Artificial Intelligence* (AI). It was not realistic just a few years ago. Now, thanks to Kubernetes and the revolution that has occurred in software technology, it has become the key to digitalization of products typical for the 4th Industrial Revolution. Cyber-physical products become testable and can be certified for public, and safe, use.

Keywords: Autonomous Real-time Testing (ART), Analytical Hierarchy Process (AHP), Quality Function Deployment (QFD), Software Testing, Certified Cloud Services, User-centric Product Development, Certifying Cyber-physical Products, Kubernetes.

1. Introduction

Cyber-physical products have initiated the 4th industrial revolution. Steam, electricity, information technology did identify the previous three industrial epochs. Hardware no longer is controlled and activated by itself, or human operators, but rather by software. Software controls

everything: motion, movement, force, energy consumption, orientation, visual recognition, and autonomous communication between various intelligent products.

In this environment, products receive software updates every few weeks, or days, and still are expected to run reliably and safely. A major concern is that current testing strategies are not capable of assessing systems with software sized around a few hundred thousand of function points. Tester do not even measure test size, because functional sizing methods were once designed for predicting cost of developing software, not for testing products. Moreover, today's testing approaches are oriented towards stand-alone products that keep their original behavior stable over time. However, with machine learning and interconnection with other devices, this is no longer appropriate.

2. Short Literature Overview

There is a huge amount of literature about software and system testing; consult the ISO standard (ISO/IEC/IEEE 29119-4, 2015) for references; however, most of it, including standard body of knowledges, available from the international testing associations, refer to testing all kind of features but functionality is just one among many features of a product (ISTQB, 2011). The move to agile software development did not change much on testing (ISTQB, 2014). But with cyber-physical systems, functionality becomes paramount. Testing relevant functionality is something different than testing code because functionality in today's complex systems often originates from the cloud, or from services where code is not available, or irrelevant to a large extend because only a small part of the functionality is used for a specific product (Ebert & Weyrich, Sep 2019).

The modern approach to functional testing of cyber-physical systems is *model-based* (Aerts, et al., 2017). This allows to cope with a testing scope not limited by own code. Models can be finite state machines (FSMs), modeling the transition from inputs to output, or temporal logic, modeling behavior, or state-based models, representing system dynamics (Baldini, 2020). Then, testing covers not only execution of test cases, but also their verification (Donzé, et al., 2013).

However, since we aim at generating new test cases from existing ones, the most attractive approach is *Combinatory Logic* (Engeler, 1995). Engeler has shown how the arrow term model of combinatory logic can be used to understand how the brain thinks (Engeler, 2019). For autonomous testing – not only for the automatic execution of tests – something similar seems useful, explaining how humans assess what a system does (Fehlmann & Kranich, May 2020). For testing medical instruments, this approach is especially promising (Biundo, et al., 2020).

With combinatory logic applied to test cases, there is combinatorial explosion. Uncontrolled, it blasts everything apart. This is where *Quality Function Deployment* (QFD) and especially Saaty's *Analytic Hierarchy Process* (AHP) comes in. They allow to automatically select test cases that are relevant for the user. *Six Sigma Transfer Function* (Fehlmann, 2016) perform that task. The series of international standards ISO/IEC 16355 (ISO 16355, 2015-2019) explain all details. For Six Sigma, consult for instance El-Haik (El-Haik & Shaout, 2010).

All this is only possible if there is a measurement method that covers both functionality and test size. This is also an international standard ISO/IEC (ISO/IEC 19761:2019, 2019). Details and sample applications can be found in the COSMIC manual (COSMIC Measurement Practices Committee, 2020). COSMIC can be used for modeling both functionality and tests (Abu Talib, et al., 2006).

Technically, these relevant selection of test cases generated by combinatory logic must be executed by software in dedicated containers. *Kubernetes* (The Kubernetes Authors, 2018) is the current software technique that allows executing such tests, implemented by *Custom Resources*

(The Kubernetes Authors, 2021). When testing cyber-physical systems, hardware in the loop is replaced by *Digital Twins* (KubeEdge, 2021).

The paper is organized as follows: section 3 explains how Six Sigma Transfer Functions select relevant test cases, with reference to two samples; section 4 outlines *Autonomous Real-time Testing* (ART) for non-testers, and section 5 presents some conclusions and suggestions for future work.

3. Six Sigma Transfer Functions

The *Formula that Explains the World* has the form

$$\boldsymbol{Ax} = \boldsymbol{y} \tag{1}$$

where $\boldsymbol{y}$ is the profile of what is the goal of undertaking – e.g., meeting customer needs – and $\boldsymbol{A}$ is the system that is expected to achieve that when applied to the technical solution profile $\boldsymbol{y}$. $\boldsymbol{A}$ is called a *Transfer Function.* If the transfer function is linear, or can be linearized, then the methods of *Linear Algebra* are applicable for solving the world formula (Fehlmann, 2003).

3.1 An Introduction to Transfer Functions

However, for testing, $\boldsymbol{x}$ is the vector profile describing the importance of the test cases for the customer, and $\boldsymbol{y}$ is the vector profile prioritizing the qualitative or quantitative user needs. The transfer function $\boldsymbol{A}$ measures the effects of test cases in view of the user stories that represent the customer's needs and values.

Solving a transfer function means guessing controls that cause a certain response, near enough to the observed or desired response. The number of controls needed, and thus the dimensions of the system $\boldsymbol{A}$ that is supposed to solve the problem, is not known from the beginning.

Sample transfer functions include, but are not limited to

- Six Sigma DMAIC analysis – searching for the cause of an observed failure.
- Search Engines – searching for URLs that match the search criteria
- Software & Systems Testing – searching for test stories that test user stories (FUR)

The problem is how to find controls with a profile $\boldsymbol{x}$, and an approximation of $\boldsymbol{A}$, such that $\boldsymbol{Ax}$ is near enough to the goal profile. If $\boldsymbol{A}$ is linear, the *Eigenvector Method* solves such problems. The *Convergence Gap* decides whether the guess is valid or not.

3.1.1 The Eigenvector Method

To solve (1) for some unknown $\boldsymbol{A}$ and $\boldsymbol{x}$, search for transfer functions $\boldsymbol{A}$ whose symmetric matrix $\boldsymbol{AA}^{\mathrm{T}}$ has eigenvectors $\boldsymbol{y}_E$ such that $\boldsymbol{y} \cong \boldsymbol{y}_E$. Consult Fehlmann (Fehlmann, 2016) for details. For comparison without bias, we need *Profiles*, i.e., vectors of Euclidean length one (Saaty, 2003); thus, a profile vector $\boldsymbol{y}$ fulfils

$$\|\boldsymbol{y}\| = \sqrt{\sum_{i=1}^{i=1} y_i^2} = 1 \tag{2}$$

In equation (2) $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$ is the profile vector with its vector components, and $\|\ldots\|$ the Euclidean norm for the length of a vector. To be compliant with ISO/IEC 16355, profile vectors are required. Otherwise, comparisons would not be based on a ratio scale, but introduce the length of the vector as a bias.

3.1.2 The Convergence Gap

Note that $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{y}_E) = \boldsymbol{A}\big(\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{y}_E)\big) = \boldsymbol{A}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}_E)$. Thus, $\boldsymbol{x_E} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}_E$ is called the *Achieved Solution Profile*, because if $\boldsymbol{y} \cong \boldsymbol{A}(\boldsymbol{x_E})$ the transfer function $\boldsymbol{A}$ and the achieved solution $\boldsymbol{x_E}$ solve the world formula approximatively.

Let $\boldsymbol{y'} = \boldsymbol{A}(\boldsymbol{x_E})$. The *Convergence Gap* is defined by the Euclidean norm for the length of the vector difference between goal profile and achieved solution profile

$$\|\boldsymbol{y} - \boldsymbol{A}(\boldsymbol{x_E})\| = \|\boldsymbol{y} - \boldsymbol{y'}\| = \sqrt{\sum_{i=1}^{n}(y_i - y_i')^2} \tag{3}$$

The convergence gap (3) can be used as a quality metric for the transfer function $\boldsymbol{A}$. It tells how well $\boldsymbol{A}$ solves the problem. Therefore, we call $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ the *World Formula*.

3.2 Six Sigma and QFD

Six Sigma has been widely adopted in manufacturing for managing processes and making them capable. QFD is linked to product design, management, and continuous improvement. Both share the use of transfer functions ("matrices") to analyze cause and effect, both solve the "world formula" $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. However, there are some significant differences.

In Six Sigma, matrix cells contain measurements, whereas QFD relies on experts' opinions to build the matrix. Measuring is among the core capabilities in manufacturing, whereas in many QFD domains, measurements are not readily available. In product management, the needs of the customers are difficult to measure and to assess (Pietsch, 2015). The customer, when asked, will tell you solutions, not needs; in most cases, habits obscure the true needs and expectations.

The focus on measurements, capable processes and the use of statistical methods are characterizing Six Sigma, whereas QFD rather focuses on customer needs, and value for the user. However, getting a valid goal profile is difficult, especially when no customer is available that can be questioned, e.g., in product development.

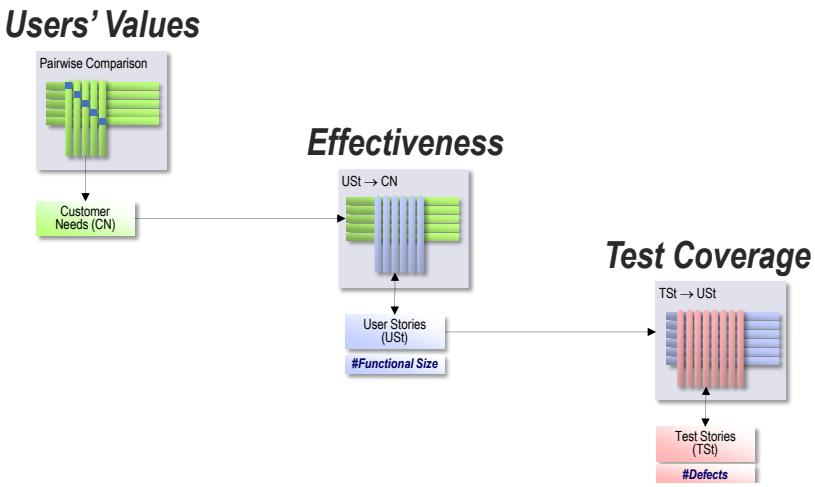

Figure 1: Three Transfer Functions are Needed to Test Users' Values

An excellent method to get a profile for users' values is the *Analytic Hierarchy Process* (AHP). It yields the customer value profile for user stories. Agile teams have a variety of methods for prioritization; however, a simple prioritization list is not enough. A priority scale is not per se a ratio scale. Profiles must get rid of bias introduced by simple linear prioritization.

User values first transform into functional effectiveness, telling whether functionality is what matters for the users' values, yielding a priority profile to user stories. This profile in turn is used to assess relevance of tests in view of users' values. This constitutes another transfer function which is called *Test Coverage*. It effectively measures whether the functionalities defined in the user stories is tested.

The functional effectiveness identifies the data movements used per user story. This traces the requirements. Each data movement is associated with a user story. Thus, the test coverage matrix can be calculated automatically, locating each data movement used in a test case and counting it for the respective cell of the test coverage matrix.

3.3 Measuring Importance and Relevance of Tests

A key element of cause-effect analysis is importance. How much does a certain control feature impact the process outcome? In Six Sigma, a *Design of Experiments* (DoE) is conducted to measure impact by measuring it. In QFD, it is often expert's opinion, not factual measurement.

User stories define requirements for functionality, and functionality can be measured. Today's software is very often embedded somewhere in some cyber-physical system and drives its response. Thus, when doing QFD for software, we expect measurements rather than expert opinions as coefficients in the cell of a matrix. To some extent, Six Sigma for Software is a mix between QFD and Six Sigma.

A *Test Story* is a collection of test cases addressing a common business scope. The transfer function $\boldsymbol{A}$ maps the test story profile $\boldsymbol{x}$ onto the user story profile $\boldsymbol{y}$ such that the convergence gap closes. This transfer function $\boldsymbol{A}$ is called *Test Coverage* and has the form of a linear matrix.

In Agile, finding the relevant user stories becomes the central part of development work. This done, creating a test coverage matrix only requires grouping test cases into test stories. If the convergence gap is close to zero, test cases address the responses that users expect. If the convergence gap opens, some test cases that matter for the users are missing, or superfluous.

3.4 Functional Size and Test Size

The basic model both for functional size and for test cases is the UML sequence diagram (Levesque, et al., 2008), in a simplified variant that we call *Data Movement Map*.

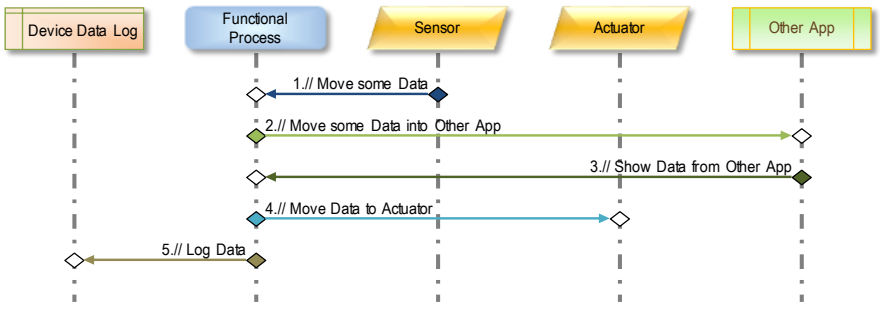

Figure 2: A Sample Data Movement Map with One Sensor and One Actuator Device

Although data movement maps can become large, testers can focus on a selection of relevant data movements. Here only four objects of interest are displayed and only four out of many more data movements. The tester should be able to step through an App by "visiting" every object of interest while executing a functional process. Data movement maps implement the ISO/IEC 19761 COSMIC functional size measurement method. Sizing an application, or a test case, is automatic once you have its data movement map. Size is defined as the number of data movements moving unique data groups. A *Defect* is localized by its respective data movement.

Now we can define *Functional Size*, *Test Size*, *Test Intensity* and *Defect Density*:

- Functional Size: Number of data movements needed to implement functionality
- Test Size: Number of data movements executed in tests
- Test Intensity: Total test size divided by total functional size
- Defect Count: Number of data movements affected by defect detected in a Test Story

It is obvious that a higher test intensity has significant value for the user of a cyber-physical system and that she or he probably is interested in knowing when and how well the systems has been tested, or retested. A display incorporating a testing dashboard would be valuable, for instance before sitting in an autonomous car. Nevertheless, test intensity is not telling much if tests are not relevant for the user.

3.5 Two Sample Test QFD

The author's book (Fehlmann, Jan. 2020, p. 73) contains a sample testing of an *Advanced Driving Assistance System* (ADAS) on the top level – microservice architecture – where it might look relatively simply. Test intensity is increased by combining functionality, for instance the visual recognition of an obstacle with the weather forecast or rain sensors.

3.5.1 An Advanced Driver Assistance System (ADAS)

It is assumed that components such as camera app, visual recognition, the lidar and the navigator work as expected. A *Recommender* app might use deep learning, or a more conventional rule-based engine. The scope of testing is restricted to a data movement map of 37 data movements only, representing the flow of data groups between 12 objects of interest. Figure 3 is the test coverage matrix from the respective test stories into user stories. The transfer function, i.e., the cells of the matrix, is defined by the number of data movements in each test story that pertains to some specific user story. Obviously, a data movement can occur in many test cases, belonging to many test stories, and pertaining to more than one user story. Strictly speaking, matrices with cell values above nine do not comply with ISO/IEC 16355. However, thanks to Saaty's Ratio Scale (Saaty, 2003), such matrices can be scaled down to the traditional 0...9 scale. The solutions to the world formula will not be affected.

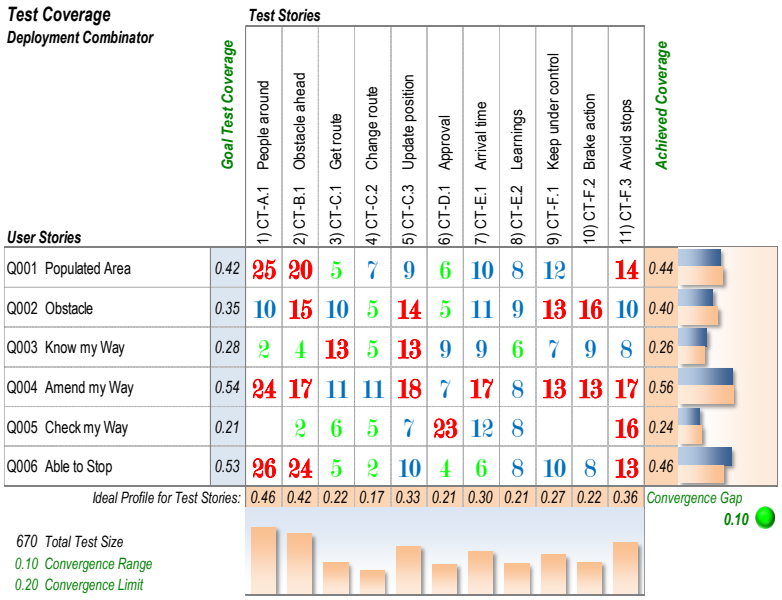

Figure 3: Sample Test Coverage Matrix

Like when analyzing big data, the measurement of the data movements frequency indicates whether test stories cover all user stories in full. The convergence gap measures whether the

tests fit the goal profile of the user stories, derived from the users' values. The test coverage transfer function yields an ideal solution profile for the relative importance of test stories. Thus, irrelevant test stories can be detected and removed.

Test stories can be enhanced by adding new test cases that incorporate experiences such as driving in heavy rain, or snowfall, and still getting valuable advice by the Recommender. ART selects these additional test cases according to relevance and prepares them for execution.

3.5.2 A Complex Trainset

The second example (Fehlmann, Jan. 2020, p. 103) refers to a trainset for international travel that took eight years for commissioning, because tests were not addressing its complexity.

The starting problem with such complex systems, also called system of systems, is that it is unclear which customer needs are dominant and which are less important across all subsystems. The *Analytic Hierarchy Process* (AHP) answers this question. It allows building a hierarchy for the priorities across all components. However, combining the test coverage matrices for the individual components results in a big matrix where all sub-quadrants are empty except those riding on the diagonal. Each sub-quadrant describes the component tests for one of the many system components, such as traction, door controls, train control, terminology, and many more. Thus, a method is needed to fill in test cases in the empty quadrants, ideally by combining existing test stories with user stories form other components. Since there are so many components, this must occur automatically, using an AI application based on QFD. This challenge calls for *Autonomous Real-time Testing* (ART).

4. Autonomous Real-time Testing

Tests executed when a cyber-physical product is shipped are not adequate for ensuring safe operations. Software updates, undetected defects in sensors, actuators, or added components – such as a compromised smartphone – might change behavior. Such change is detectable by ART. In turn, complex systems often are not tested thoroughly because they are simply too big for traditional testing approaches. Only components are tested. Bug detection is left to some trial phase where it is checked for the first time whether all subsystems work smoothly together. Such checks are not tests. Subsystems often do not even use the same language because their design and inceptions lay decennials apart. ART allows to detect such terminology glitches and produces the missing test cases linking behavior of different components to each other.

Autonomous Real-time Testing (ART) means that tests run

- Anytime
- Anywhere
- In a limited timeslot
- Individually

Test cases must be relevant to users' needs, with respect to users' individual way of using the cyber-physical system. Test data are being captured from logging activities of sensors and actuators.

4.1 The Three Steps Needed for Preparing Art Plus the One Step Needed for Execution

ART requires three steps for preparation of test cases, and one fourth step for preparing execution. Tests are executed with *Digital Twins* (El Saddik, 2018), not with hardware in the loop.

1. Measure functional size and test size with ISO/IEC 19761 COSMIC;
2. Use combinatory algebra to generate test cases for test stories;
3. Use transfer functions for test coverage to select test cases that matter for the user;
4. Execute tests by digital twins.

The framework for *Continuous Integration. Delivery, and Deployment* (CI/CD) is extended by an overlaying activity: ART generates test cases that continuously test software-intense systems, funneled through transfer functions that enforce user values as a selection principle for new test cases. This starts during and in parallel to the CI/CD activities but extends to the full product lifetime. Test results are kept for future learning and improvement.

With Kubernetes, all test instances run in virtual machines, including the generation of test cases using combinatorial logic, and the ART evaluator. ART does not happen in some staging environment; it happens in the real operational environment. This makes a big difference if relevant parts of your system under tests are executing in the cloud. For this, whatever happens to the cyber-physical system and is captured by sensors, or managed by actuators, is collected as history data, and used for enhancing ART test cases. Tests run in the individual user's system, or at least by exactly the software that is used there, not on a staging system, and with individual users' data.

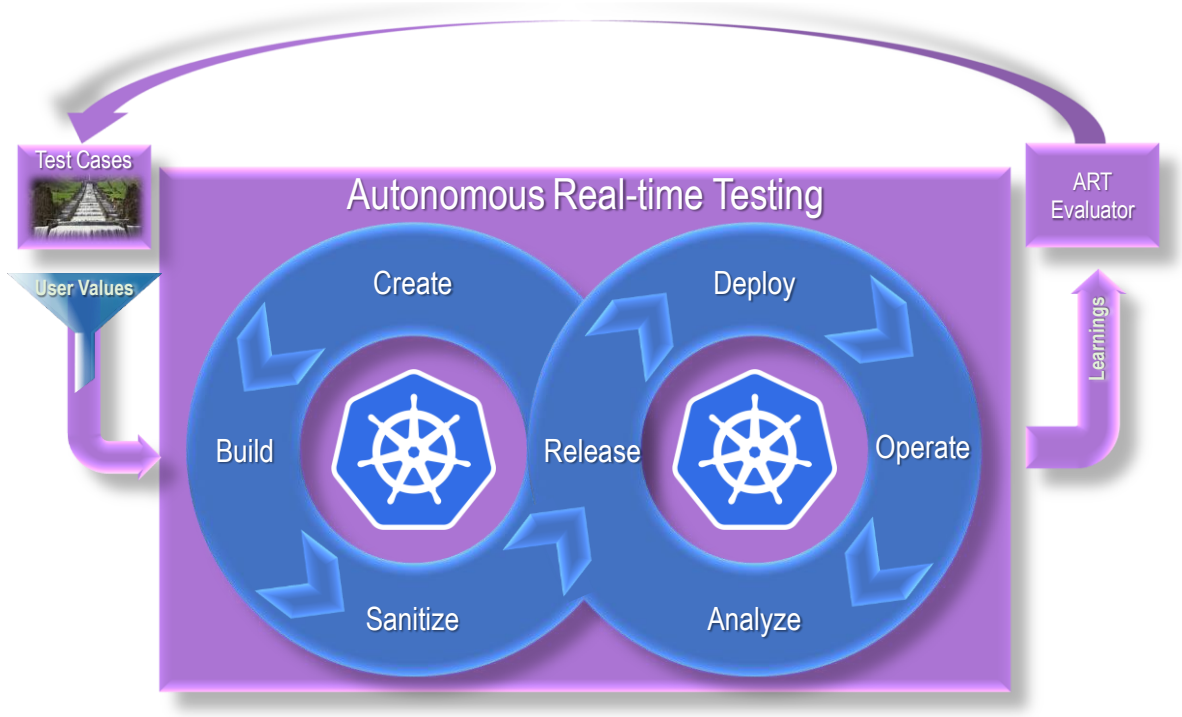

Figure 4: Continuous Integration, Deployment, and Delivery with ART Continuous Testing

The method how to combine test cases is described in (Fehlmann, 2016, p. 319) and (Fehlmann, Jan. 2020, p. 30). The base set $\mathcal{L}$ consists of predicates specifying data groups according to ISO/OEC 19761. The power set $\mathcal{G}(\mathcal{L})$ is the set constructed as follows:

Recursive Definition:

- $\mathcal{L} \subset \mathcal{G}(\mathcal{L})$
- Let $a_1, \dots, a_n$ and b be elements of $\mathcal{G}(\mathcal{L})$. Then the *Arrow Term*

$$\{a_1, \dots, a_n\} \to b \tag{4}$$

is also an element of $\mathcal{G}(\mathcal{L})$.

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

$$M \bullet N = \{a | \exists b_i \to a \in M,\ b_i \subset N\} \tag{5}$$

Equation (5) allows to generate as many test cases as needed, keeping its history of successful execution. Furthermore, it describes tests of tests, e.g., test strategies.

4.2 The Problem with Combinatory Explosion

This combinatorial explosion creates a problem with the "Real-time" part of ART. Test cases must be executed in limited time. Thus, selecting relevant test cases out of the huge number of possible test case combinations requires the additional step 3 in the above list (section 4.1).

The test coverage transfer function linking test stories to user stories is the method for choice for selecting relevant test cases. Only those test cases that keep the convergence gap small enough are chosen and added to the test suite. Thus, we can limit the total test execution time to some reasonable, finite value and still provide maximum value for the user.

This usage of the transfer function combines aspects from QFD and Six Sigma. Clearly it is QFD that guides the requirements tracing done for assessing functional effectiveness; this is a standard practice in QFD, see Schockert (Schockert, 2017) and together with Herzwurm (Schockert & Herzwurm, 2017). However, the test coverage matrix that relies on the QFD for tracing requirements is pure Six Sigma, because ethe test coverage matrix is now measured, not only assessed by requirement, or testing experts.

4.3 Model-based Testing

The implementation method or the code of the software under test is not needed for ART. Analyze user needs instead – use the AHP or use VoC (Mazur, 2014). Draw a data movement map to identify the functionality that matters for the user. The layer of functionality that is being tested is freely selectable and can include the behavior of *Neural Networks* and *Support Vector Machines* that otherwise are not executing calculations and algorithms. AI must be constantly tested before allowed it impacting humanity.

Thus, testing a model instead of code has gained popularity recently. However, it should be noted that selecting a model is far from straightforward. The international standard ISO/IEC 14143 (ISO/IEC 14143-1, 2007) is probably simply not understood well enough. This standard defines which granularity is needed when modeling functionality of software. When selecting a model, the user view is essential. Functional size depends on the selected viewpoint; obviously because functionality is not a universal concept but always dependent from where it starts. For instance, when assessing the size of a software system, nobody wants to include the functional size of the hard disk supporting the operating of software. But the hard disk also is a cyber-physical system; it is no longer a tape, recording bits sequentially. It has a functional size that is non-negligible, but from a technical viewpoint only.

Therefore, the user point of view is paramount for testing, whatever size or complexity the systems exhibit. It is therefore hard to understand how testing has ever been thought sensible without adopting ISO/IEC 16355. Sure, it is always possible to adopt a holistic position and believe that assumptions can replace facts based on sound knowledge. But then it is belief not science.

4.4 Selecting Relevant Test Cases

The selection process for additional test cases, generated using combinatory logic, requires searching a large set of valid leaves of the search tree using a hash that evaluates the newly generated candidate. The hash is the convergence gap (3). New test cases are added to the test suite only if they make the convergence gap smaller. This search process is quite tedious and requires significant computing effort when automated. It is not immediately clear which test cases lead to a smaller convergence gap. It might be just a try-and-error algorithm.

There are a few ideas how to speed up the selection of relevant test cases. The most promising is to use linear algebra to identify those cells in the matrix that should contribute more, or less, for closing the convergence gap. Sensitivity analysis could serve as heuristics, indicating where

in the matrix to generate new candidates, and whether the newly generated candidate is promising. This has been investigated by the authors (Fehlmann & Kranich, 2021). However, this is not easy because when adding, or deleting, a test case from the test suite affects more than just one cell in the test coverage matrix. Each addition or deletion of a test case has side effects that are difficult to predict, because each test case is associated with a data movement map with a size usually larger than one.

Nevertheless, skilled QFD moderators do such things in every QFD workshop. They seem to have a sensory for detecting promising spots in a linear matrix. Sometimes they call it "reading the matrix", but usually they cannot explain clearly what that means exactly. It seems human brains use some guidance to successfully identify where to improve a process and its transfer function. Thus, such sensitiveness can be trained, and chances are, that they can even be trained to a *Support Vector Machine* (SVM) (Pupale, 2018). In general, such AI machines can be trained to exhibit similar behavioral skills like human experts. This would allow to fully automatize ART.

4.5 Is Full Automation Needed for ART?

Maybe full automation is not needed. If ISO/IEC 14143 and ISO/IEC 19761 are used to correctly identify user' viewpoints, and AHP to cut complex problems in suitable components, human expertise is probably fast enough, much cheaper and more reliable than AI automation. Experts – in this case both for QFD and the domain under test – can "read the matrix" and identify test stories that need more emphasis. By adding test cases to some specific test story, for instance by variating test data, some test stories add weight where it affects the convergence gap favorably.

Since typically data movements belong to different user stories, adding a test case to a test story, without changing the underlying data movement map, might cause adverse effect. Test cases can contain data movements that spread over several user stories. Sometimes it looks more promising to connect test stories with new user stories by adding dedicated test cases. Creating new test cases within existing test stories needs expertise with the business domain.

If this fails also, it is always possible to add new test stories, or delete existing test stories when they seem obsolete. Then, new columns are added or deleted in the test coverage matrix. This is again manual work, but it can be supported by AI, and domain expertise is crucial.

Thus, full automation is possible, but not needed immediately. To start with ART, finding a QFD expert might be good enough. However, the 4th industrial revolution is not yet at its end, and cyber-physical systems might grow to sizes currently not yet known. The clue is that relevant test cases should refer to relevant user's viewpoints and thus need not to grow into dimensions that cannot be handled neither by humans nor by machines.

5. Conclusions

Tests must be ongoing and continuously because learning systems keep changing. Cyber-physical systems, systems that communicate with each other cannot be tested or certified before release. Continuous testing requires digital twins, emulating hardware in the loop.

ART is a new concept that addresses the needs for building confidence in cyber-physical products. Metrics for functionality, for test size and test intensity build confidence and replace to a large extend the role that metrics like horsepower had delivered for traditional mechanical products of the past century. AHP and QFD, and all the tools mentioned in the international series of standards ISO/IEC 16355 are paramount for ART. It is not enough to produce more test cases than the competition. It is much more important to understand the values of users of cyber-physical products. Only then ART delivers a process that authorities can certify as sufficiently

well guaranteeing safety, privacy, and overall security. But without such a certified testing process, following ISO/IEC 14143, 19761, and 16355, cyber-physical products will not be able to replace traditional mechanical, human-controlled products.

References

Abu Talib, M. et al., 2006. Scenario-based Black-Box Testing in COSMIC-FFP: A Case Study. *Software Quality Professional – Journal of the American Society for Quality,* June, 8(3), pp. 22-33.

Aerts, A., Reniers, M. & Mousavi, M., 2017. Chapter 19 - Model-Based Testing of Cyber-Physical Systems. Dans: *Cyber-Physical Systems.* Cambridge, MA: Academic Press, Elsevier Inc., pp. 287-304.

Baldini, G., 2020. *Testing and certification of automated vehicles including cybersecurity and artificial intelligence aspects,* Luxembourg,: Publications Office of the European Union.

Biundo, E. et al., 2020. *The socio-economic impact of AI in healthcare,* Belgium: Deloitte, MedTech Europe.

COSMIC Measurement Practices Committee, 2020. *COSMIC Measurement Manual for ISO 19761 – Version 5.0 – Part 1-3,* Montréal: COSMIC Measurement Practices Committee.

Donzé, A., Ferrère, T. & Maler, O., 2013. *Efficient Robust Monitoring for STL.* Lecture Notes in Computer Science éd. Berlin, Heidelberg: Springer.

Ebert, C. & Weyrich, M., Sep 2019. Validation of Autonomous Systems. *IEEE Software,* 36(5), pp. 15-23.

El Saddik, A., 2018. Digital Twins: The Convergence of Multimedia Technologies. *IEEE MultiMedia (Volume: 25 , Issue: 2 , Apr.-Jun. 2018),* 25(2), pp. 87 - 92.

El-Haik, B. S. & Shaout, A., 2010. *Software Design for Six Sigma – A Roadmap for Excellence.* Hoboken, NJ: Wiley&Sons, Inc..

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

Engeler, E., 2019. Neural algebra on "how does the brain think?". *Theoretical Computer Science,* Volume 777, pp. 296-307.

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

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

Fehlmann, T. M., Jan. 2020. *Autonomous Real-time Testing – Testing Artificial Intelligence and Other Complex Systems.* Berlin, Germany: Logos Press.

Fehlmann, T. M. & Kranich, E., 2021. *A Sensitivity Analysis Procedure for Linear Multiple–Response Transfer Functions,* Duisburg: In Progress.

Fehlmann, T. M. & Kranich, E., May 2020. Intuitionism and Computer Science – Why Computer Scientists do not Like the Axiom of Choice. *Athens Journal of Sciences,* 7(3), pp. 143-158.

ISO 16355, 2015-2019. *Applications of Statistical and Related Methods to New Technology and Product Development Process,* Geneva, Switzerland: ISO TC 69/SC 8.

ISO/IEC 14143-1, 2007. *Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts,* Geneva, Switzerland: ISO/IEC JTC 1/SC 7.

ISO/IEC 19761:2019, 2019. *Software engineering - COSMIC: a functional size measurement method,* Geneva, Switzerland: ISO/IEC JTC 1/SC 7.

ISO/IEC/IEEE 29119-4, 2015. *Software and systems engineering — Software testing — Part 4: Test techniques,* Geneva, Switzerland: ISO/IEC JTC 1.

ISTQB, 2011. *ISTQB - Certifying Software Testers Worldwide.* [Online]
Available at: http://www.istqb.org/downloads/category/2-foundation-level-documents.html
[Accessed 24 April 2017].

ISTQB, 2014. *Agile Tester Extension Syllabus.* [Online]
Available at: http://www.istqb.org/downloads/send/5-agile-tester-extension-documents/41-agile-tester-extension-syllabus.html
[Accessed 24 April 2017].

KubeEdge, 2021. *DeviceTwin.* [Online]
Available at: https://kubeedge.io/en/docs/architecture/edge/devicetwin/
[Accessed 18 October 2021].

Levesque, G., Bevo, V. & Cao, D., 2008. *Estimating Software Size with UML Models.* Montréal, Canadian Conference on Computer Science & Software Engineering, pp. 81-87.

Mazur, G., 2014. *QFD and the New Voice of Customer (VOC).* Istanbul, Turkey, International Council for QFD (ICQFD), pp. 13-26.

Pietsch, W., 2015. *Augmenting Voice of the Customer Analysis by Analysis of Beliefs.* Hangzhou, China, International Council for QFD (ICQFD).

Pupale, R., 2018. *Support Vector Machines (SVM) - An Overview.* [En ligne]
Available at: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989
[Accès le 28 Mar. 2019].

Saaty, T. L., 2003. Decision-making with the AHP: Why is the principal eigenvector necessary?. *European Journal of Operational Research,* Volume 145, pp. 85-91.

Schockert, S., 2017. *Agiles Software Quality Function Deployment.* University of Stuttgart, Germany: EUL Verlag.

Schockert, S. & Herzwurm, G., 2017. *Agile Software Quality Function Deployment.* Tokyo, 23rd International QFD Symposium, ISQFD'17, pp. 134-148.

The Kubernetes Authors, 2018. *Kubernetes.* [Online]
Available at: https://kubernetes.io
[Accessed 15 Dec 2018].

The Kubernetes Authors, 2021. *Use Custom Resources.* [Online]
Available at: https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/
[Accessed 18 October 2021].