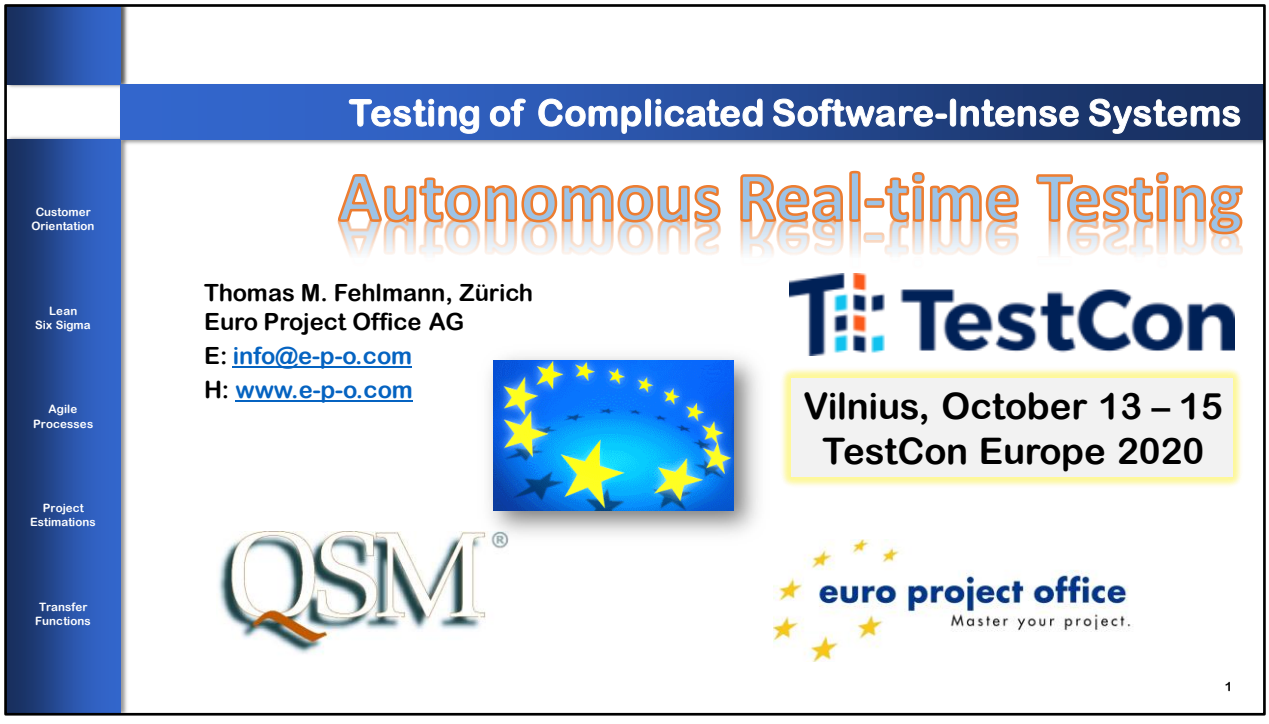


Autonomous Real-time Testing

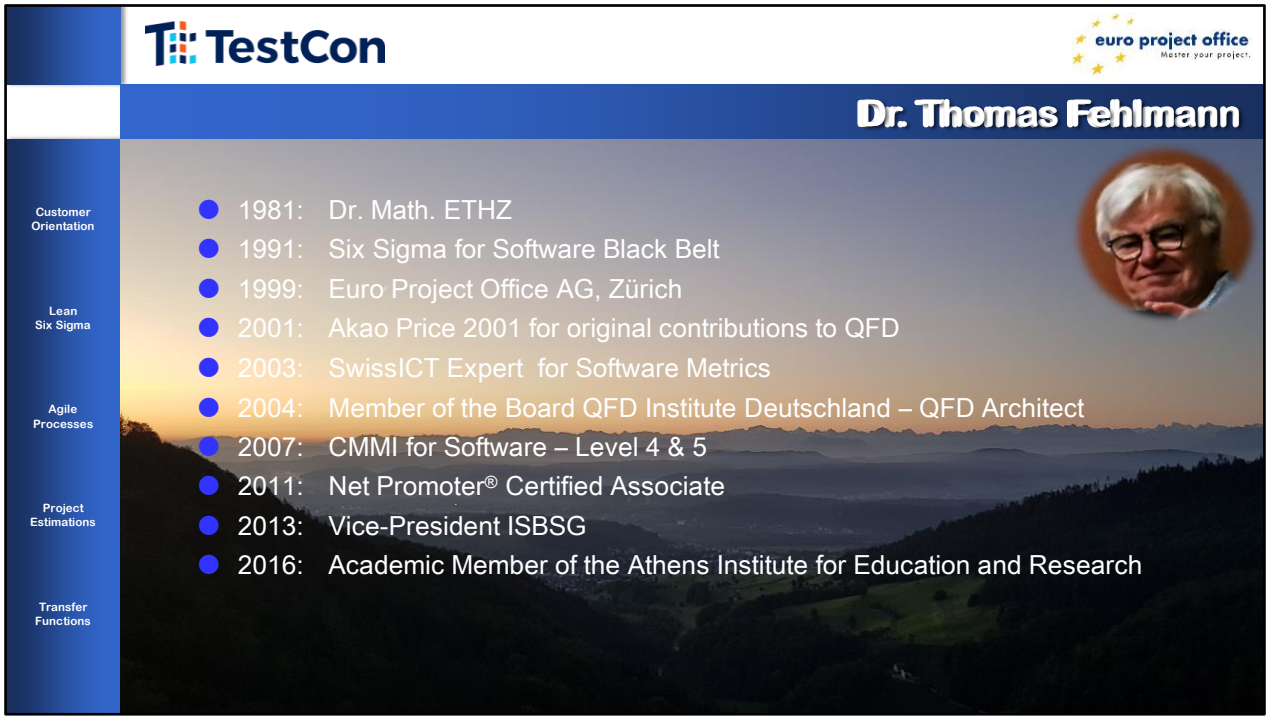
Software testing is becoming increasingly important because more and more products are software-intense. Cars, for example, contain more and more control software (ECUs) that are networked with each other. With new rail vehicles, software problems delay commissioning by months, even years, because the different components are not coordinated with each other. A timely system test would help, but there is a lack of time and resources. The functionality of the whole, quite complex system is simply too great. So, you must automate.

This presentation explains the theory and explains the implementation approach for a framework for Autonomous Real-time Testing (ART) of a software-intense system while in operation.

Thomas Fehlmann

Thomas Fehlmann is a senior expert in software metrics and testing, a Lean Six Sigma Black Belt for lean and agile software development and promoter of customer-oriented product design and testing.

Since 2016, Thomas has been an academic member of the Athens Institute for Education and Research.



1981:	Dr. Math. ETHZ
1982-89:	Manager Software–Development
1990-95:	Senior Consultant – Digital Equipment Corp.
1996-99:	Sales Support Manager – Proposal Center
1999ff:	Euro Project Office AG, Zürich Akao Price 2001 for original contributions to QFD Member of the Board of QFD Institute Germany – QFD Architect Lean Six Sigma for Software Development Quality Function Deployment (QFD) and New Lanchester Theory Net Promoter Score Associate
2001-12	Six Sigma Black Belt for GMC Software AG
2003-18	Chair of SwiSMA, the Swiss Software Metrics Association
2012-19:	Member of the DASMA Board
2013-18:	Vice-President ISBSG
2016ff:	Academic Member of the Athens Institute for Education and Research

TestCon

euro project office
Master your project.

Goals of this Presentation

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

1) *Understand how Testing Integrates with Software Development for Highly Complex Systems*

2) *Explain How to Use Combinatory Logic for Autonomous Testing*

3) *Do Continuous Deployment for Safety-critical Systems, based on Autonomous Real-time Testing*

This Photo by Unknown Author is licensed under CC BY-NC

3

Just to make it clear: the focus on ISO 19761 COSMIC is due to the popularity of COSMIC when sizing embedded software. Any other sizing method can be adapted to the task of sizing tests as well, if allowing for a little imprecision due to lack of compliance to the VIM and the GUM.

The VIM: ISO/IEC Guide 99:2007, 2007. International Vocabulary of Metrology – Basic and general concepts and associated terms (Vocabulaire International de Métrologie – VIM)

The GUM: ISO/IEC CD Guide 98-3, 2015. Evaluation of measurement data - Part 3: Guide to Uncertainty in Measurement (GUM).

The VIM and the GUM are among the oldest standards that the world uses. In simple words: You can add & subtract measurements, like you can measure distance between several points in space, and get the total distance, possible after some trigonometric adjustments.

With COSMIC, measuring tests by measuring data movements is straightforward and easy. With IFPUG-like sizing methods, care must be taken not to double count elementary processes.

TestCon

euro project office
Master your project.

Goals of this Presentation

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

4) *And by-the-way, you cannot manage software nor software testing without measuring functionality*

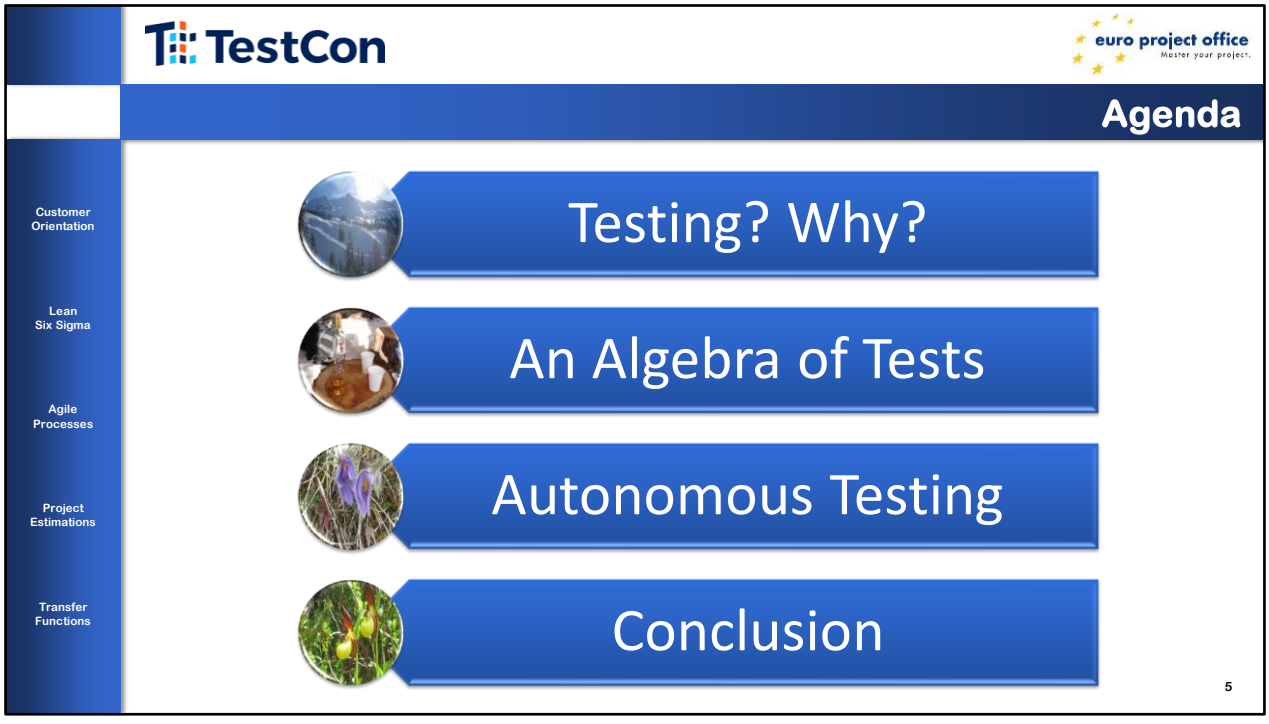
"In God we Trust, all others must bring data"

(Edward Deming, Grandfather of Lean)

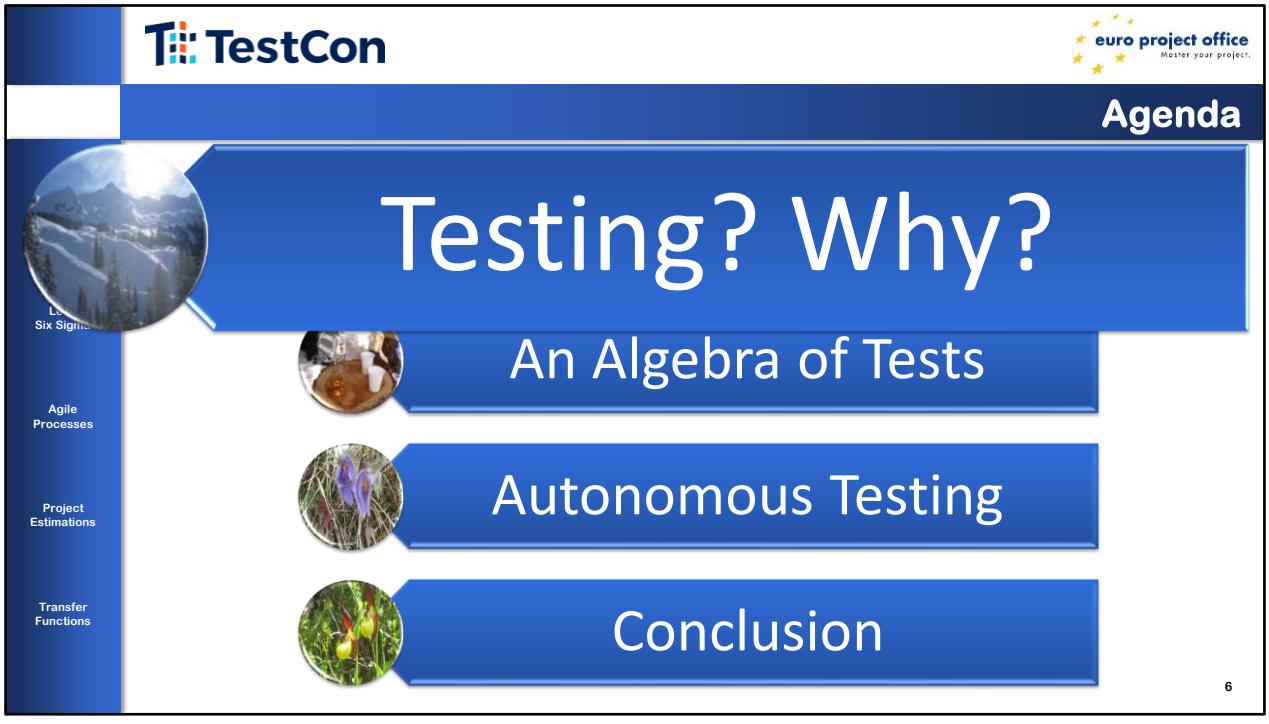
4

As a side-effect, I also want to persuade you that testing without measuring test size and test intensity is nonsense. Deming said it almost a hundred years ago that you cannot provide quality without measurements. And measuring tests means measuring functionality.

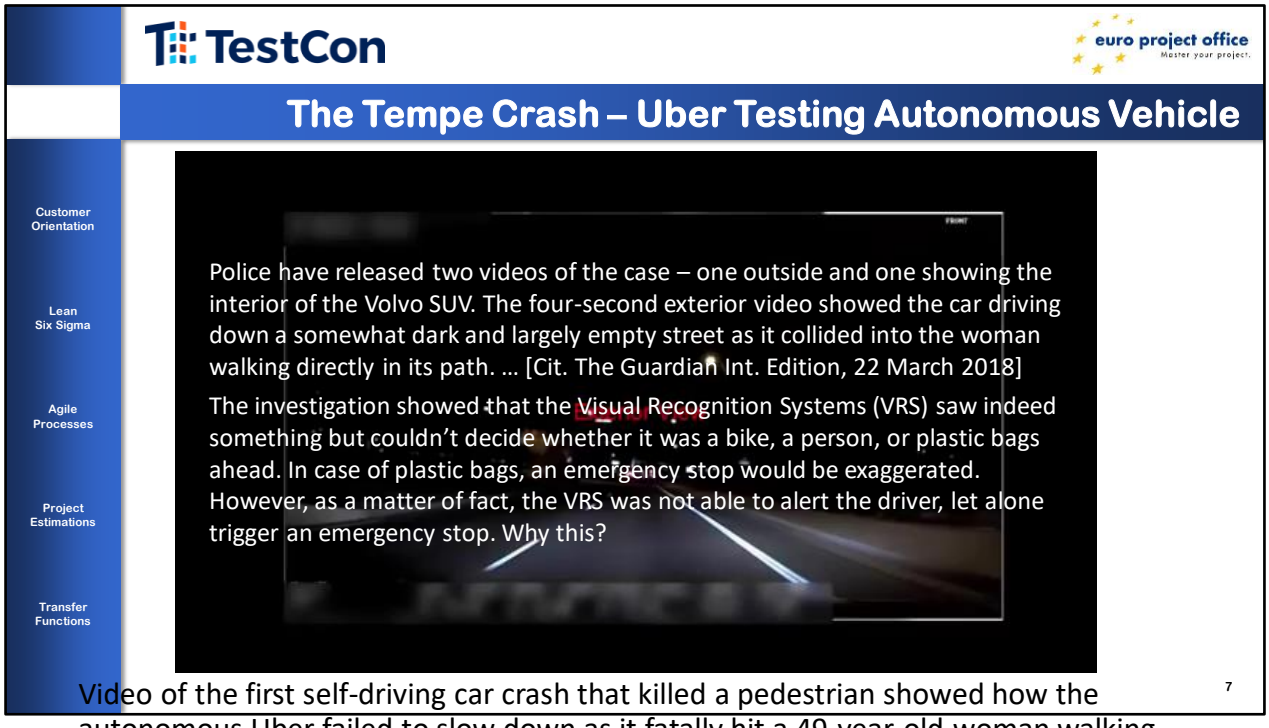
Measuring functionality, however, means measuring functional size with some function point method; something rather unpopular among developers. Maybe because developers somehow feel as being the ultimate creator?



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



Video of the first self-driving car crash that killed a pedestrian showed how the autonomous Uber failed to slow down as it fatally hit a 49-year-old woman walking her bike across the street.

The newly released footage of the collision that killed Elaine Herzberg in Tempe, Arizona, on Sunday night has raised fresh questions about why the self-driving car did not stop when a human entered its path and has sparked scrutiny of regulations in the state, which has encouraged testing of the autonomous technology.

Police have released two videos of the case – one outside and one showing the interior of the Volvo SUV. The four-second exterior video showed the car driving down a somewhat dark and largely empty street as it collided into the woman walking directly in its path.

The 14-second video inside the car showed the operator, identified by police as Rafaela Vasquez, 44, appearing to look at something inside the vehicle and not at the road at the time of the collision. She alternated between looking down and looking forward and appeared shocked at the last minute just as the car failed to stop. … [Cit. The Guardian Int. Edition, 22 March 2018]

The investigation showed that the Visual Recognition Systems (VRS) saw indeed something but couldn't decide whether it was a bike, a person, or plastic bags ahead. In case of plastic bags, an emergency stop would be exaggerated. However, as a matter of fact, the VRS was not able to alert the driver, let alone trigger an emergency stop. Why this?

This was a trial run of an untested autonomous car on a public road!

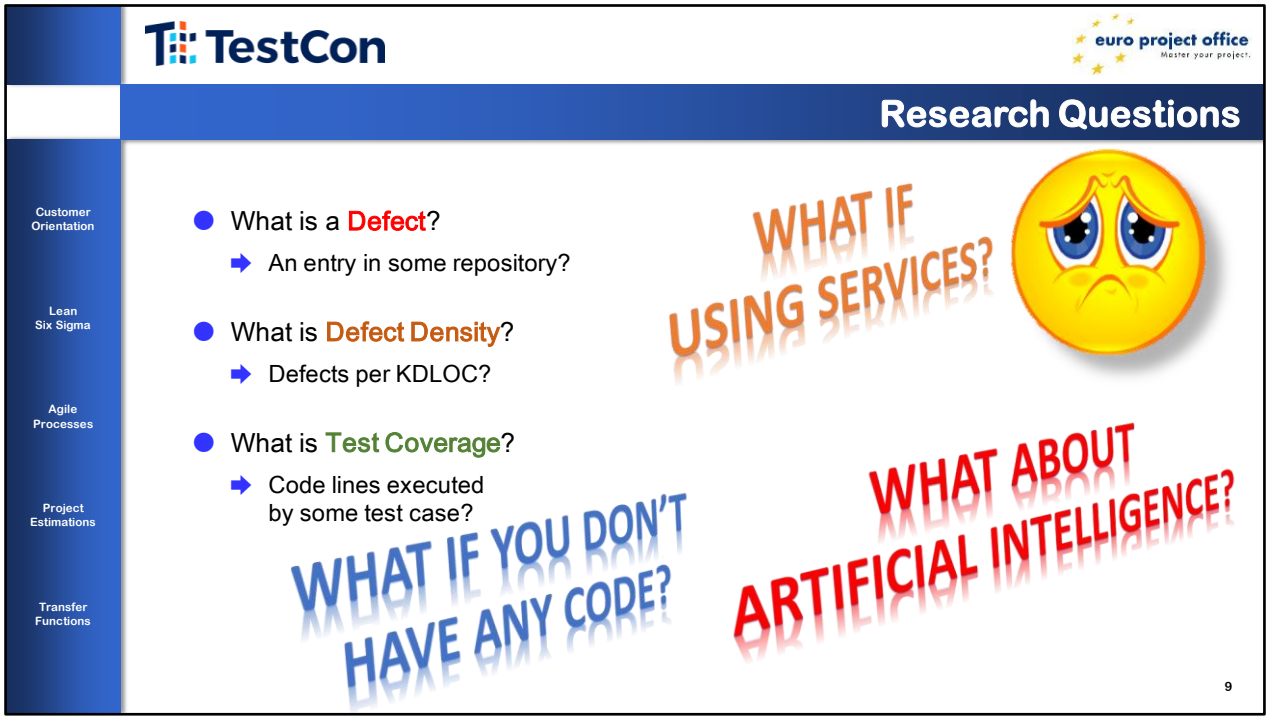
Are they crazy? Running an untested autonomous car on public roads in a trial run?

Unfortunately, Uber is not alone. Swiss observed that a new software used for engine control stopped working correctly when approaching landing strips on a passenger flight.

The Swiss Railways now try for four years in a row to put their new Bombardier trainsets into service, and still cannot use them in full.

Boeing had to recall its 737 MAX and now tries to safely return them to service. But they still are not. They are among the many competitors with the new Berlin airport who finishes trials first.

Why don't they test their autonomous vehicles, new train sets, planes, flight engines before commissioning? Why running trials instead? Trial runs are unstructured experiments without knowing the outcome beforehand – at the risk of crashing their enterprise?



Counting lines of code doesn't address today's needs in ICT, maybe except for unit testing, when writing new code. However, you often have neither code nor exact specification when you use some services on the web.

Or, imagine you're using Artificial Intelligence! How would you test AI-based software? Decisions? Do you trust them as if AI originates from some god?

Today's practice in software and system testing is simply a mess. People count entries in bug inventories and mistake this for the number of defects.

Even worse, they look at lines of code and define one defect if they must fix this line – notwithstanding that code can contain many more defects than just one per line.

You cannot compare one "defect density" with another. The metrics used by the testing community are context dependent and almost useless.

Common testing techniques still refer to code – however, code is most often not available if we test our software, and in most cases not for systems that interact with cloud services or rely on Artificial Intelligence components such as a Support Vector Machine (SVM).

Functionality in turn can be assessed and modeled, even if implemented by an SVM. In contrary, code is subject to the programming language, programming environment, and sometimes not even open.

TestCon

euro project office
Master your project.

Why do Test Matter?

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- Health, even life, may depend on the proper functioning of software
  - Software is everywhere
  - Consequences of malfunction might be fatal
- Suppliers of software are liable
  - Continuous Integration – Continuous Delivery
  - Selling cars with Advanced Driver Assistance Systems (ADAS) carries risks for software suppliers
    - Risk growths exponentially with level of autonomy
    - After an accident, car drivers can blame software supplier for bugs
  - Selling autonomous cars is inconceivable without Continuous Testing
- Successful test reports keep customers happy and loyal!

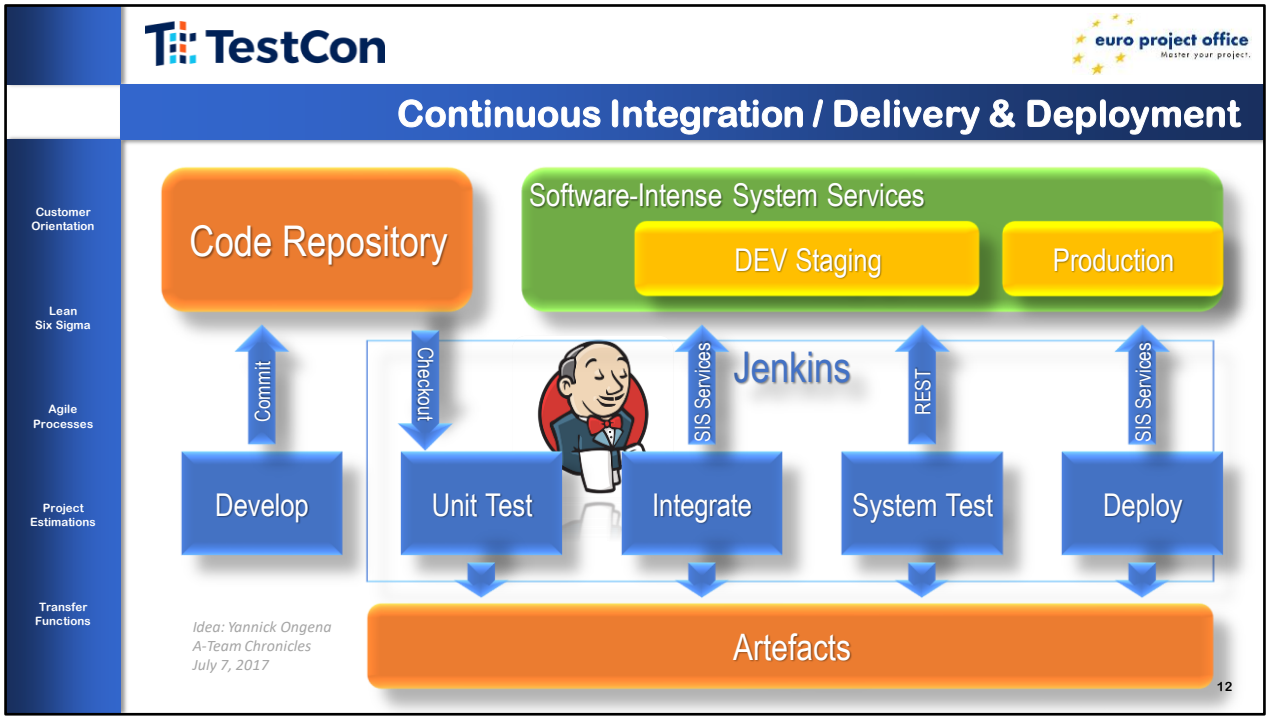
© 2020 Consumer Reports, Inc

11

Software testing is becoming increasingly important as more and more products are software intensive. For example, cars contain more and more control software (ECUs) that are networked together. In new rail vehicles, software problems delay commissioning by months, even years, because the various components are not coordinated. A timely system test would help, but there is a lack of time and resources. The functionality of the software is simply too large. So you must automate.

Automation is not only necessary for the execution of tests, but especially for the generation of suitable test cases. This is possible with combinatorial logic, the Analytic Hierarchy Process (AHP) and Quality Function Deployment (QFD).

When today's cars use map services from the cloud or proprietary sensors for an Advanced Driving Assistance System (ADAS) to make driving decisions, or when in the future one autonomous car meets another, or with truck platooning; or when a new, previously unknown device is added to an IoT network, the original base system expands its functionality. Therefore, a system expanding in this way must be tested again before it can make decisions that have the potential to harm people or things. This is necessary again after each update, after each learning. Testing takes place continuously during operation; it complements continuous delivery and integration.



Tools for Continuous Integration / Continuous Delivery

One of the standard tools used in software development is the open-source software Jenkins. The default interaction model with Jenkins, historically, has been web UI driven, requiring users to manually create jobs, then manually fill in the details through a web browser. This requires effort to create and manage jobs to test and build multiple projects, it also keeps the configuration of a job to build/test/deploy separate from the actual code being built/tested/deployed. This prevents users from applying their existing CI/CD best practices to the job configurations themselves.

The new approach is creating pipelines. Users can implement a project's entire build/test/deploy pipeline in a "Jenkinsfile" and store that alongside their code, treating their pipeline as another piece of code checked into source control.

Similar approaches are advertised by Atlassian, the provider of Jira, the software development tool of choice used by agile teams.

TestCon

euro project office
Master your project.

What is a Test?

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- A Software Test has
  - Several Test Stories
    - Explaining the Value for the Customer
    - Weighted by Customer's Priority for the Test Story
- A Test Story has
  - Many Test Cases
  - Exploiting different aspects – favorable and dismal – of the test story
- A Test Case has
  - Test data and test stubs to run the software under test conditions
  - A known Response
    - Passed: all responses according predefined expectations
    - Failed: at least one test case didn't yield the expected response

13

Testing Apps is a great deal more demanding than testing in traditional environments.

One challenge is that the sheer number of platforms needed to run tests outnumbers capacity even for large organizations.

Another problem is that mobile platforms works under unstable conditions. A communication once established can break down within short and be unavailable for quite some time, without neither the server nor the phone device being able to restore it.

TestCon

euro project office
Master your project.

What is a Test Case?

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- A Test Case has
  - Entry Data ("Test Data") $x_1, x_2, \dots, x_n$
    - Explaining the environment for the test case
    - Typically valid, invalid, borderline data
    - Normal and disturbed communication services
  - A pre-defined expected response y

$$\{x_1, x_2, \dots, x_n\} \rightarrow y$$

  - A sequence of functional processes with its data movements executed by the test case
    - Defining **Test Coverage** and Test Size
      - Test Size
        - » Every Test Case has a size: the number of data movements executed by the test
        - » Total Test Size is the number of data movements executed by all test cases
      - **Test Coverage**
        - » Percentage of data movements covered with test cases

A Test Story creates a predictable environment for executing a sequence of data movements with known expected response.

The test data must be reproducible as well as the environment.

Test Cases are a collection of so-called **Arrow Terms**. These terms, recursively defined, constitute a Combinatory Algebra with excellent properties. They cover all possible calculations; however, the set is not limited. Every finite subset lacks a few test cases, and it is important to know whether these test cases are important.

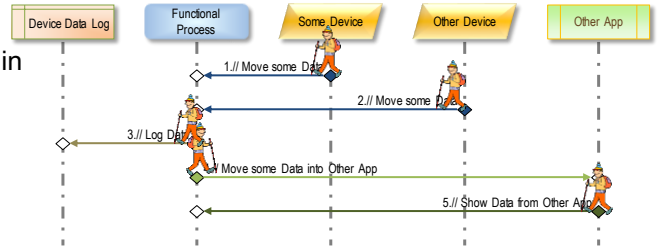
TestCon

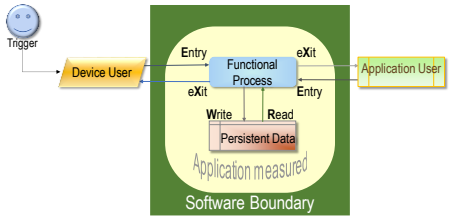
euro project office
Master your project.

Software Testing as a Game

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- Tester sees selected sequences in the Data Movement Map
- Tester can "walk" the data movements when planning or executing tests
  - ➡ Functionality becomes visible and measurable to the development team
- Same Metric:
  - ➡ *ISO/IEC 19761 COSMIC*





15

The basic interface is the sequence diagram, in a simplified variant that we call **Data Movement Map**.

Although Data Movement Maps can become large, you can focus on a selection of data movements only. 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.

You easily recognize in a Data Movement Map the COSMIC model. Sizing an application is automatic once you have its Data Movement Map.

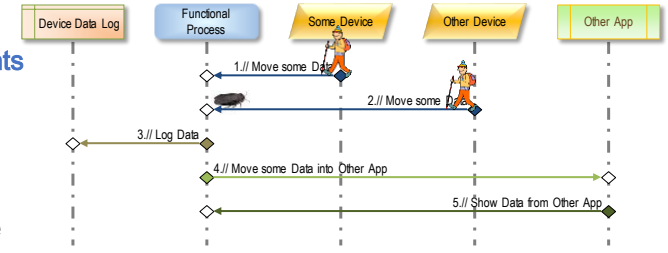
TestCon

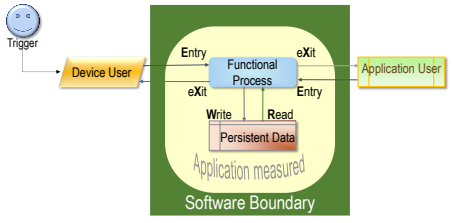
euro project office
Master your project.

Functionality, Defect Size, and Defect Count

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- What happens if Data Movements don't work as expected, have Defects instead?
- Testers mark and count Data Movements where Defects have been detected
- Same Metric:
  - *ISO/IEC 19761 COSMIC*





16

When a defect has been identified, the respective data movement can be visually marked, e.g., by being blocked by a bug.

However, such a defect might exist only under defined test data conditions. If test management confirms the existence of such a defect, it is possible to block that data movement for this specific test data or environment.

Now we can define *Test Size* and *Defect Density* based on the ISO/IEC 19761 COSMIC international standard, now available in version 4.0.

Test Coverage is the percentage of Data Movements that are covered with at least one test case.

Test Intensity is Total Test Size divided by Total Functional Size.

Total Test Size is the sum of the sizes of all Test Cases.

All data movements executed by test cases count extra.

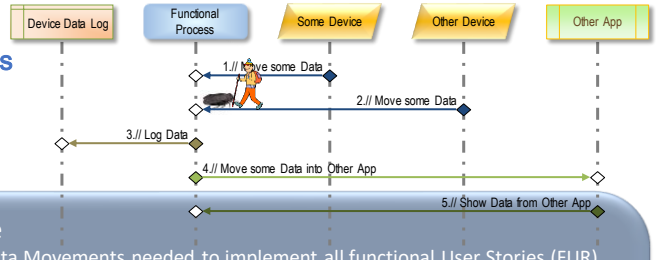
TestCon

euro project office
Master your project.

Functionality, Defect Size, and Defect Count

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- What happens if Data Movements don't work as expected, have Defects instead?
- Testers mark and count Data Movements which have been detected
- Same Metric:
  - *ISO/IEC 19761 COSMIC*



- Functional Size
  - Number of Data Movements needed to implement all functional User Stories (FUR)
- Test Size
  - Number of Data Movements executed in Test Cases
- Test Story
  - Collection of Test Cases aiming at certain functional User Stories (FURs)
- Test Intensity
  - Total Test Size divided by Total Functional Size
- Defect Count
  - Number of Data Movements affected by some defect detected in a Test Story

17

When a defect has been identified, the respective data movement can be visually marked, e.g., by being blocked by a bug.

However, such a defect might exist only under defined test data conditions. If test management confirms the existence of such a defect, it is possible to block that data movement for this specific test data or environment.

Now we can define *Test Size* and *Defect Density* based on the ISO/IEC 19761 COSMIC international standard, now available in version 4.0.

Test Coverage is the percentage of Data Movements that are covered with at least one test case.

Test Intensity is Total Test Size divided by Total Functional Size.

Total Test Size is the sum of the sizes of all Test Cases.

All data movements executed by test cases count extra.

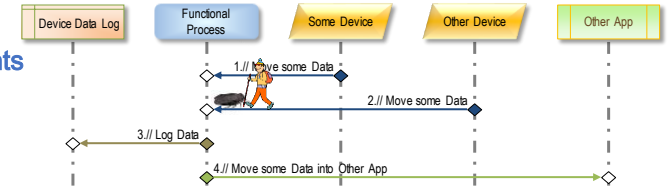
TestCon

euro project office
Master your project.

Functionality, Defect Size, and Defect Count

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- What happens if Data Movements don't work as expected, have Defects instead?
- Testers mark and count Data Movements where Defects have been detected
- Same Metric:
  - *ISO/IEC 19761 COSMIC*



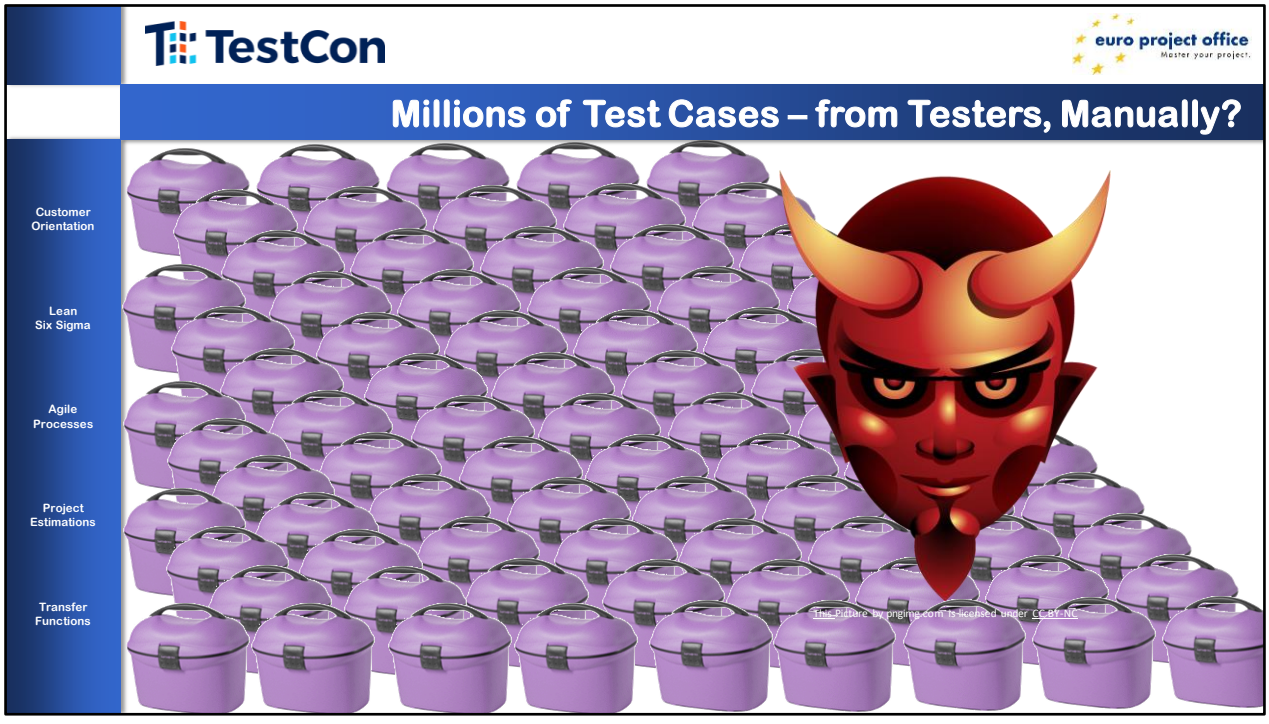
- Typical commercial or administrative software
  - Functional Size ≅ 5'000 CFP; Test Intensity 10
  - Total Test Size ≅ 50'000 CFP
- Safety-critical software, e.g., ADAS
  - Functional Size ≅ 50'000 CFP; Test Intensity 50
  - Total Test Size ≅ 2'500'000 CFP
- Complex Software-intense System (plane, train, …)
  - Functional Size ≅ 500'000 CFP; Test Intensity 10 – 50
  - Total Test Size ≅ 5'000'000 CFP – 25'000'000 CFP

18

If only one defect can be localized per data movement, the total number of defect opportunities is equal to the test size; i.e., the number of data movements per test case, counted with COSMIC.

This allows to identify typical defect densities for typical kind of software. It is obvious that while a test size of 50'000 CFP is manageable, 2'500'000 to 25'000'000 CFP are not, at least not without a little help by my friend, the computer performing with Artificial Intelligence (AI).

In the Nienties, we developped software that was close to 6 Sigma. For instance, Point of Sale software, that's still running today. However, this was rather something in the 500 CFP range and such tests you can manage. Nevertheless, we reached not more than 5.7 Sigma; 6 Sigma proved to become too costly.



But measuring functional size alone does not solve the problem.

Here you see the *Test Cases* needed. They count for many more than the *Unit Tests* that most probably exist, at least for safety-critical software. Functional tests are executed to assert how the different ECU components interact. A few millions test cases probably will do ….

With a functional size of a few hundred thousand, it is impossible to create 10’000’000 test cases manually.

“Houston, we have a problem.”

The creation of test cases must be automated.

The big problem is, where and how to get the millions of test cases that are needed to make meaningful tests for a complex, software-intense system. Not only autonomous cars, or *Advanced Driver Assistance Systems* (ADAS), also IoT used in manufacturing, or trainsets used for mass transit and long-distance, or integrated medical equipment relying on life-critical communicating cannot be properly tested with traditional manual methods.

TestCon

euro project office
Master your project.

The New Approach

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

The Solution is

Autonomous Real-time Testing

- Automated tests run
  - Anytime
  - Anywhere
  - In a limited time slot
  - Individually

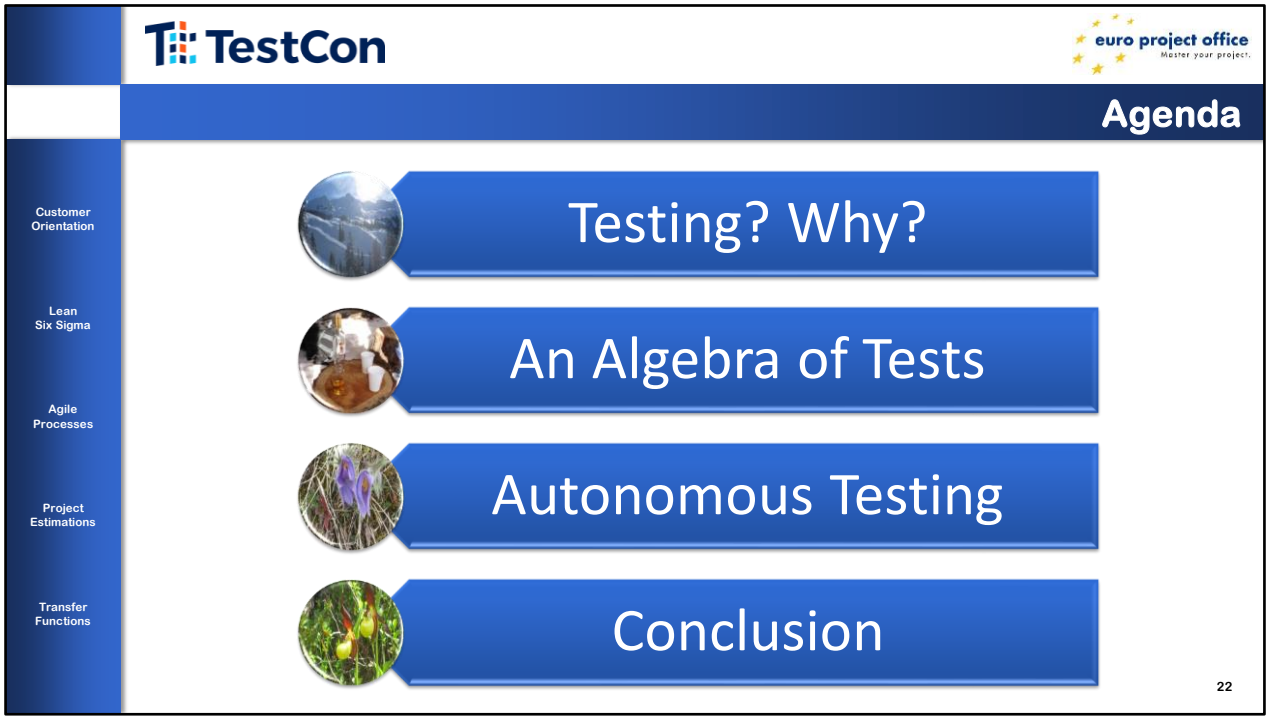
21

Tests run when the product is shipped are probably not adequate for the later state of operations. Software updates, undetected defect in sensors or added components – such as a compromised smartphone – might change the results of tests.

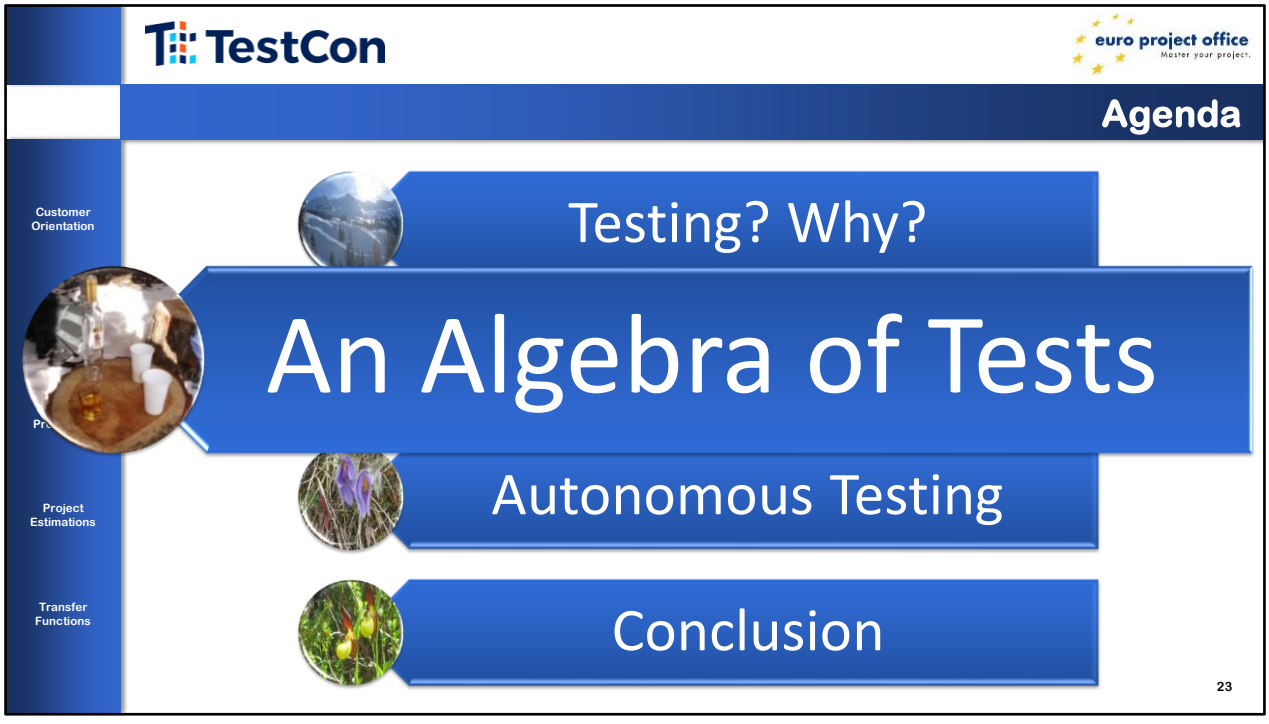
Autonomous Real-time Testing means that tests run

- Anytime
- Anywhere
- In a limited time slot
- Individually

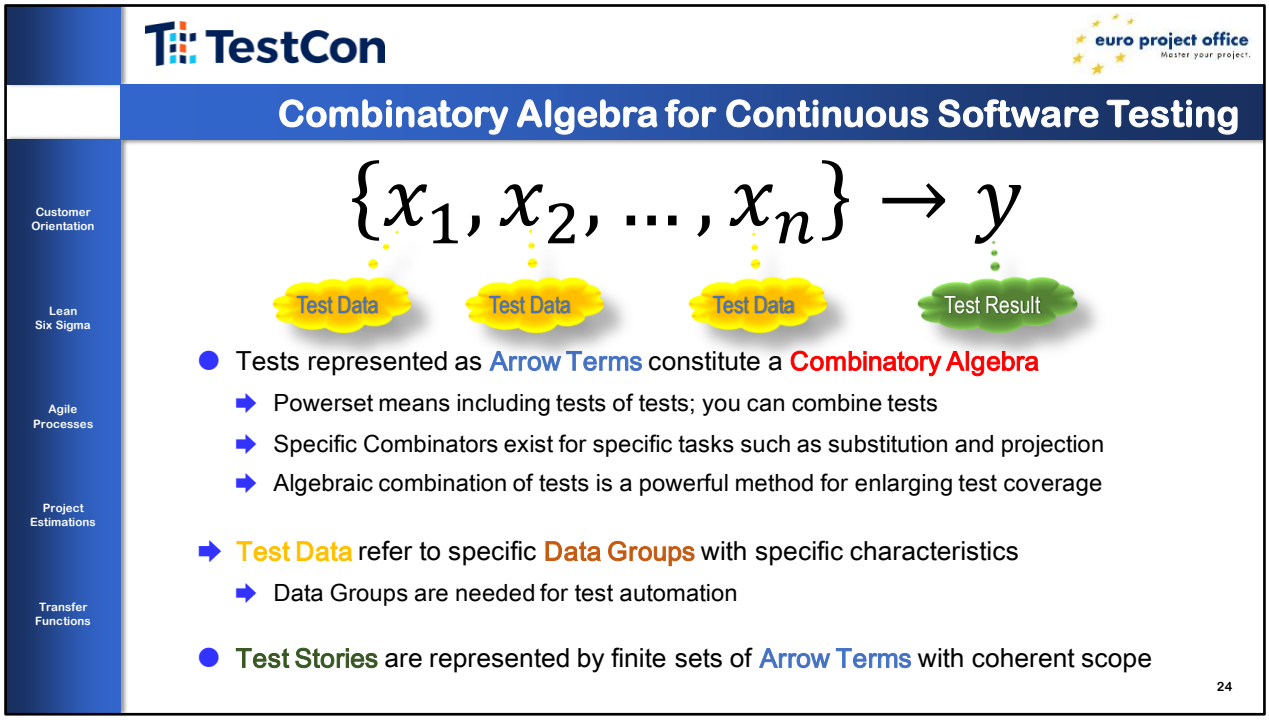
The latter term means that test cases are generated automatically, to adapt to the users needs.



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



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

$$\{a_1, \ldots, a_n\} \rightarrow b$$

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

$$\mathcal{G}_0(\mathcal{L}) = \mathcal{L}$$

$$\mathcal{G}_{n+1}(\mathcal{L}) = \mathcal{G}_n(\mathcal{L}) \cup \{\{a_1, \ldots, a_m\} \rightarrow b | a_1, \ldots a_m, b \in \mathcal{G}_n(\mathcal{L}), m = 0,1,2,3 \ldots\}$$

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

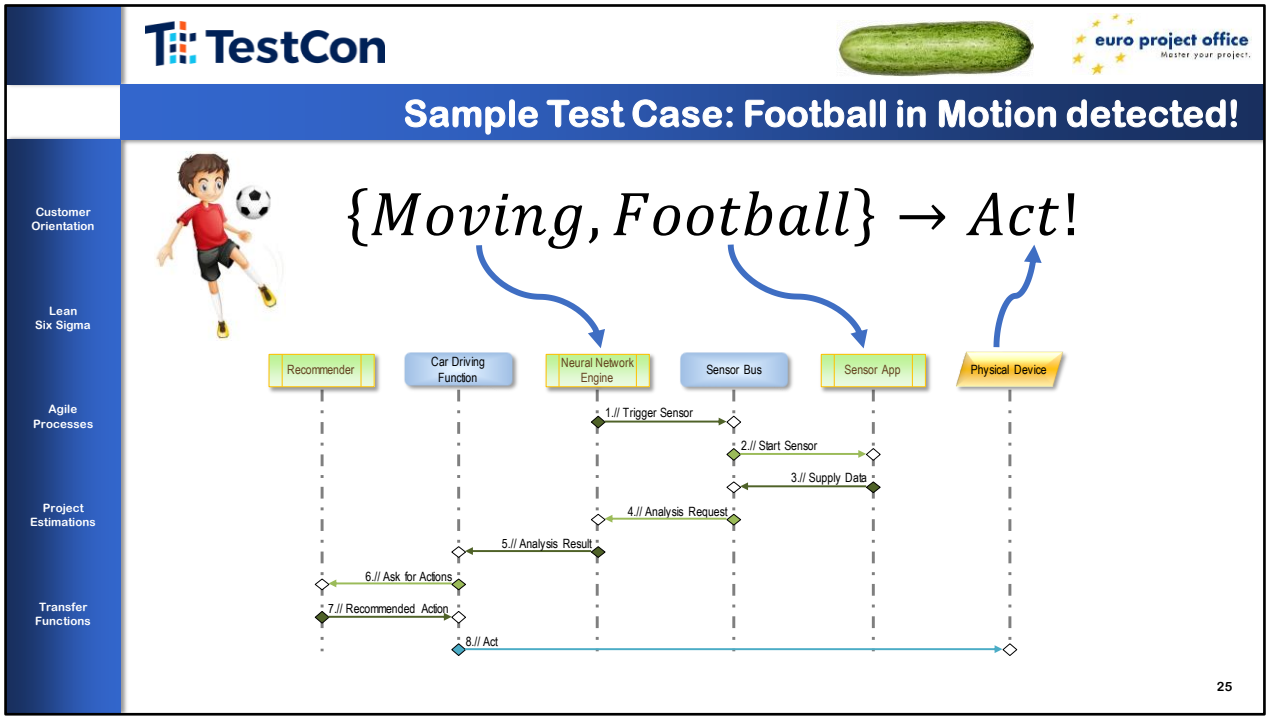
$$\mathcal{G}(\mathcal{L}) = \bigcup_{n \in \mathbb{N}} \mathcal{G}_n(\mathcal{L})$$

The elements of $\mathcal{G}_n(\mathcal{L})$ are arrow terms of level n. Terms of level 0 are *Topics*, terms of level 1 *Rules*. A *Rule Set* is an element of $\mathcal{G}_n(\mathcal{L})$ that consists of level 1 terms only and is finite; if it is infinite, we call it *Knowledge Base*. Hence, knowledge is a potentially unlimited set of rules about topics and rules. This definition is recursive.

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

$$M \bullet N = \{a | \exists b_i \rightarrow a \in M, b_i \subset N\}$$

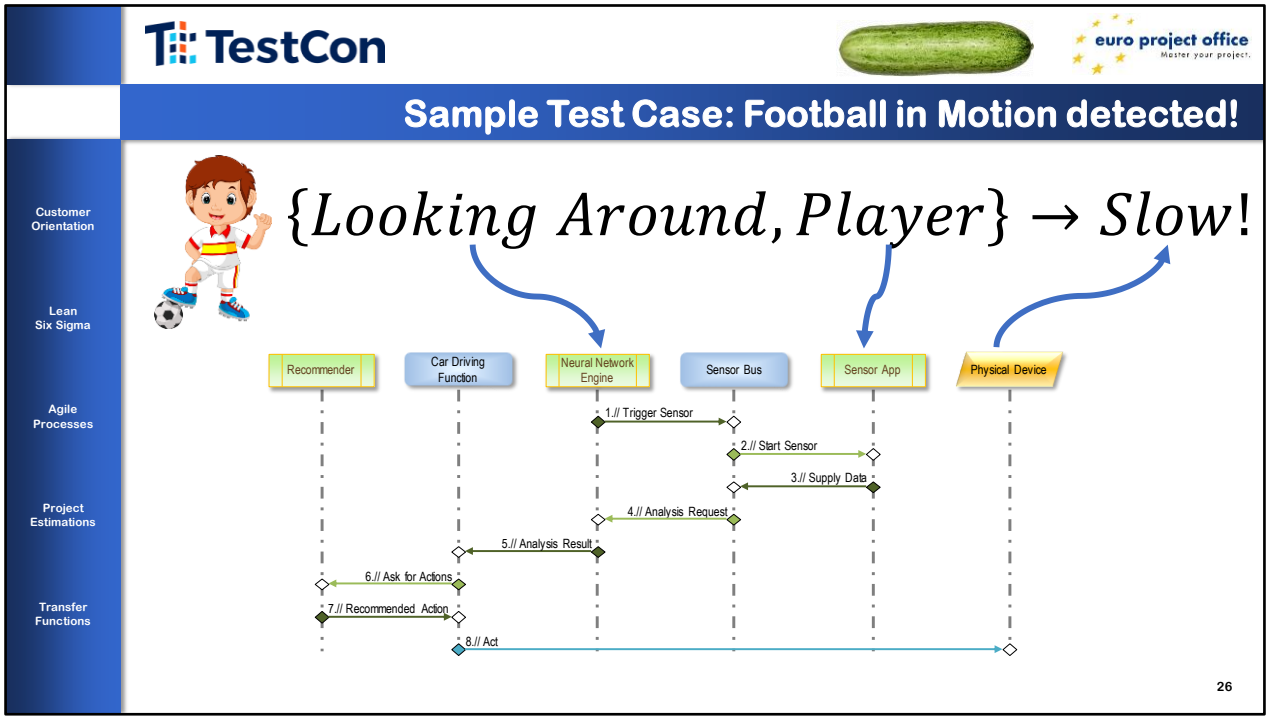
In case of M as a rule set, and N as a topic set, this represents the selection operation that chooses those rules $(b_i \rightarrow a)_j$ from rule set M that are applicable to the topic set N. The definition applies to all higher–level $\mathcal{G}(\mathcal{L})$ terms as well.



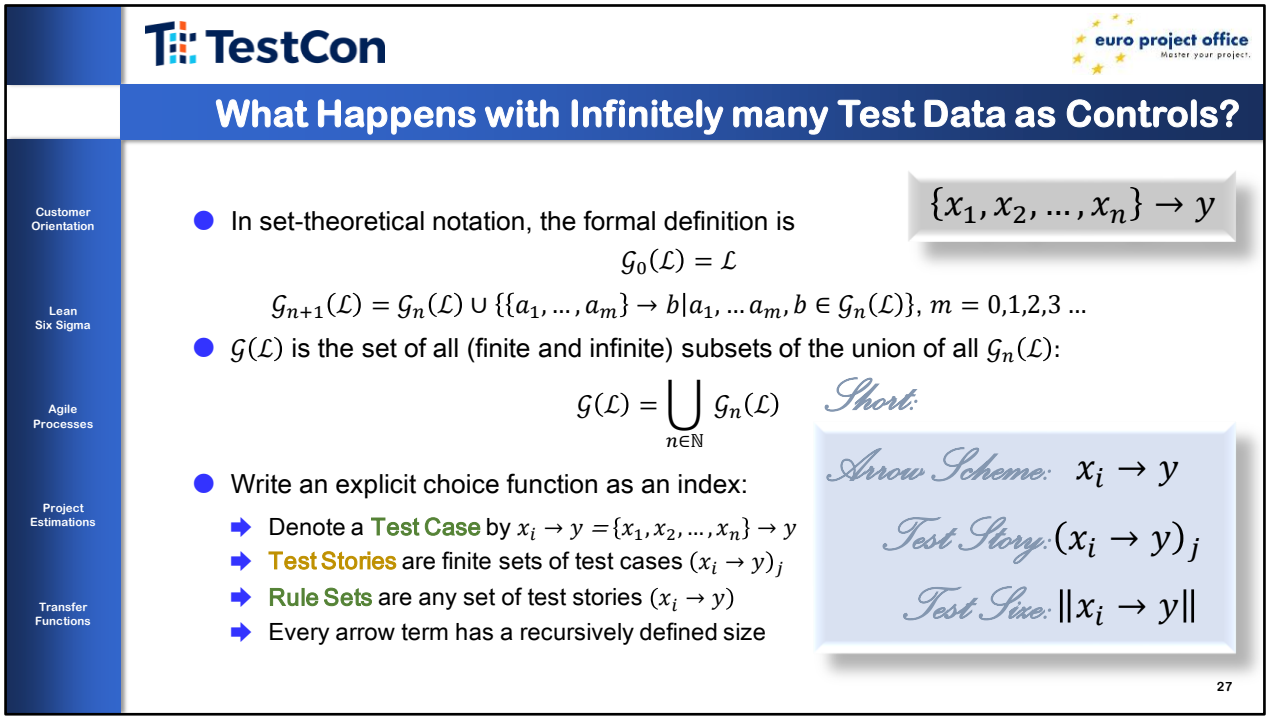
The test data for testing the reaction of the car upon detecting a child football player runs through the usual data movement map that controls all kind of car driving operations (well, very much simplified…); the response might depend upon the actions recommended by the Recommender application, which in turn might also be a learning system but should always recommend actions of the safe side.

And not broadcast all its recommendations to the whole world!

Implement it using Cucumber.



The same data movement map produces different responses with different controls. If children obviously look at traffic, the car might go cautiously ahead, not blocking the traffic behind. Thus, response depends from the controls, images captured by the Sensor App, its interpretation by the Neural Network Engine, causing different recommendations and thus different reactions of the car.

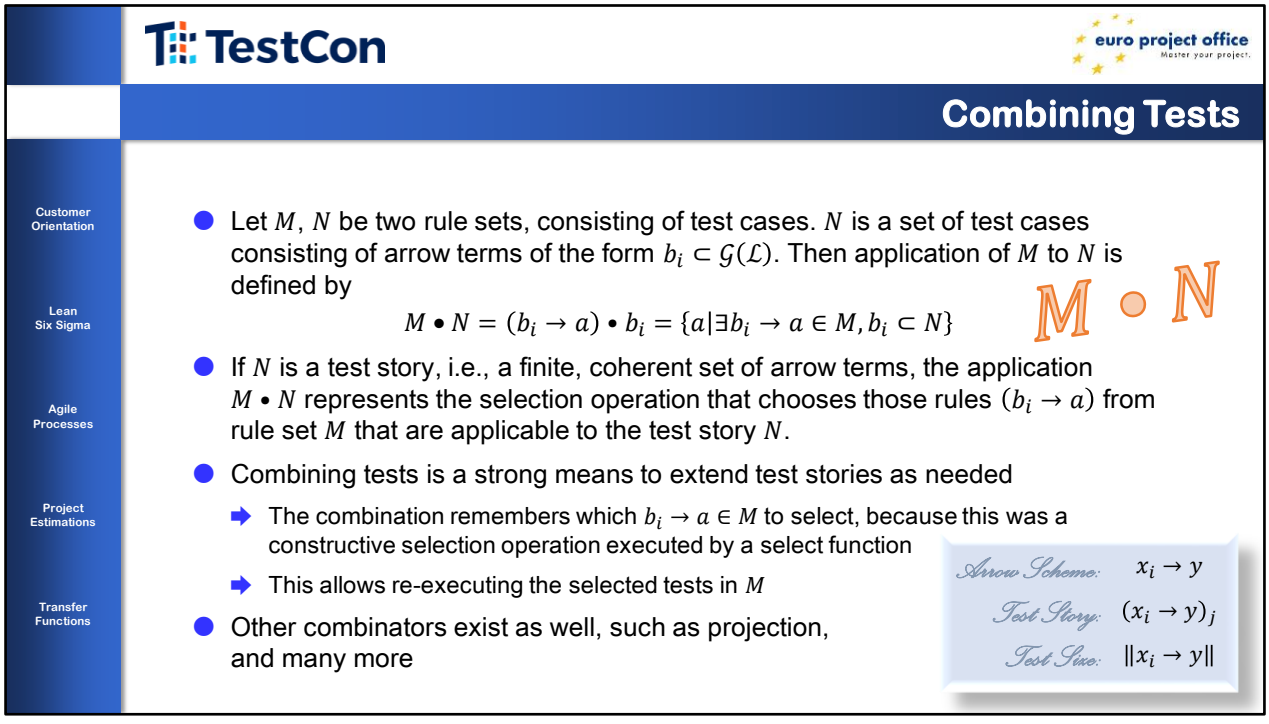


Denote by $\mathcal{G}(\mathcal{L})$ the power set containing all *Arrow Terms* of the form

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

The left-hand side of this term 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.

To avoid too many set-theoretical parenthesis (curly brackets), we use a kind of Einstein-Notation, denoting finite set by its choice function written as an index (subscript), and round parenthesis indication a set consisting of arrow terms of similar structure. If the set is finite, this is indicated by yet another subscript index. Double bars indicate size, as usual.



The application rule for M and N make arrow terms, and thus tests, a Combinatory Algebra. Combining tests is a strong means to extend test stories as needed. The application rule simply says you can combine tests if you have tests that yield the required test data. However, you need them constructively, as tests for the left–hand side of arrow term test cases.

Rule sets – potentially infinite set of test cases – represent things that organizes themselves such as cars that drive automatically, flying drones that find the way to its target, 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.

TestCon

euro project office
Master your project.

The SK Combinators and the Lambda Theorem

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- Combinatory algebras describe formal languages like assembler a program
- The reason for this is that the two combinators **S** and **K** fulfilling

$$\mathbf{K} \bullet P \bullet Q = P \text{ (Projection)}$$

$$\mathbf{S} \bullet P \bullet Q \bullet R = P \bullet Q \bullet (P \bullet R) \text{ (Substitution)}$$

  are enough to describe any kind of construction needed in a logical program

- The Lambda Theorem says that whenever there exists a combinator $M(x)$ with some subterm x, there exists a combinator $\lambda x.Mx$ that fulfills

$$(\lambda x.Mx) \bullet N = M(N)$$

  thus, by replacing x by N in M, this yields a combinator too

  ➡ It means we can use the Lambda theorem to construct test schemata

$\lambda x.Mx$

29

There is a mathematical theory about Combinatory Algebras (Engeler, 1995) that explains quite generally how to combine topics in areas of knowledge. Combination is not only on the basic level possible; you can also explain how to combine topics on the second level; sometimes called meta-level. Intuitively, we would expect such a meta-level describing knowledge about how to deal with different knowledge areas.

Combinatory algebras are models of *Combinatory Logic* (Curry & Feys, 1958) and (Curry, et al., 1972). These are algebras that are combinatory complete. This means that there is a combination operation $M \bullet N$ for all elements M, N in the combinatory algebra and the following two *Combinators* **S** and **K** can be defined as follows

$$\mathbf{K} \bullet P \bullet Q = P$$

and

$$\mathbf{S} \bullet P \bullet Q \bullet R = P \bullet Q \bullet (P \bullet R)$$

where P, Q, R are elements in the combinatory algebra.

Thus, the combinator **K** acts as projection, and **S** is a substitution operator for terms in the combinatory algebra. **S**-**K** terms become quite lengthy and are barely readable by humans, but they work fine as a foundation for computer science.

TestCon

euro project office
Master your project.

Arrow Terms – A Model of Combinatory Logic

Customer Orientation

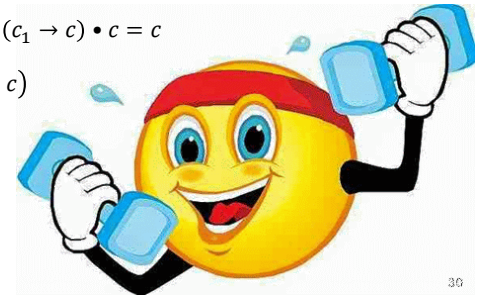
Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- The following definitions demonstrate how arrow terms implement the combinators S and K
  - $\mathbf{I} = a_1 \to a$ is the Identification; i.e., $(a_1 \to a) \bullet b = b$
  - $\mathbf{K} = a_1 \to \emptyset \to a$ selects the 1st Projection:
    $$\mathbf{K} \bullet b \bullet c = (b_1 \to \emptyset \to b) \bullet b \bullet c = (\emptyset \to b) \bullet c = b$$
  - $\mathbf{KI} = \emptyset \to a_1 \to a$ selects the 2nd Projection:
    $$\mathbf{KI} \bullet b \bullet c = \big((\emptyset \to c_1 \to c) \bullet b\big) \bullet c = (c_1 \to c) \bullet c = c$$
  - $\mathbf{S} = \big(a_i \to (b_j \to c)\big)_1 \to (d_k \to b)_i \to (a_i + d_{j,i} \to c)$ is called Substitution
- Therefore, the algebra of arrow terms is a Model of Combinatory Logic



30

The proof that the arrow terms' definition of **S** fulfils equation (2) is somewhat more complex. The interested reader can find it in Engeler (Engeler, 1981, p. 389). With **S** and **K,** an abstraction operator can be constructed that builds new knowledge bases. This is the *Lambda Theorem*; it is proved along the same way as Barendregt's Lambda combinator.

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 must pick it; knowledge rather needs being constructed in some strictly logical way.

TestCon

euro project office
Master your project.

More Combinators as needed

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- There are many more combinators
  - Including a Fixpoint Combinator $\mathbf{Y}$, represented in Lambda calculus for instance by
    $$\mathbf{Y} = \lambda f.\left(\lambda x.\, f(x\ x)\right)\left(\lambda x.\, f(x\ x)\right)$$
    where f and x are variables that can be replaced by the substitution combinator
  - The fixpoint combinator $\mathbf{Y}$ solves
    $$\mathbf{Y}\ g = g(\mathbf{Y}\ g)$$
  - Evaluating the Lambda term above becomes somewhat complicated, like assembler
    $$\mathbf{Y} = \mathbf{S} \bullet \left(\mathbf{K} \bullet (\mathbf{S} \bullet \mathbf{I} \bullet \mathbf{I})\right) \bullet \left(\mathbf{S} \bullet (\mathbf{S} \bullet (\mathbf{K} \bullet \mathbf{S}) \bullet \mathbf{K}) \bullet \left(\mathbf{K} \bullet (\mathbf{S} \bullet \mathbf{I} \bullet \mathbf{I})\right)\right)$$
- It is currently unknown how the fixpoint combinators work on test cases
  - I hope to be able to present more at the 2021 ATINER spring conference in Athens

31

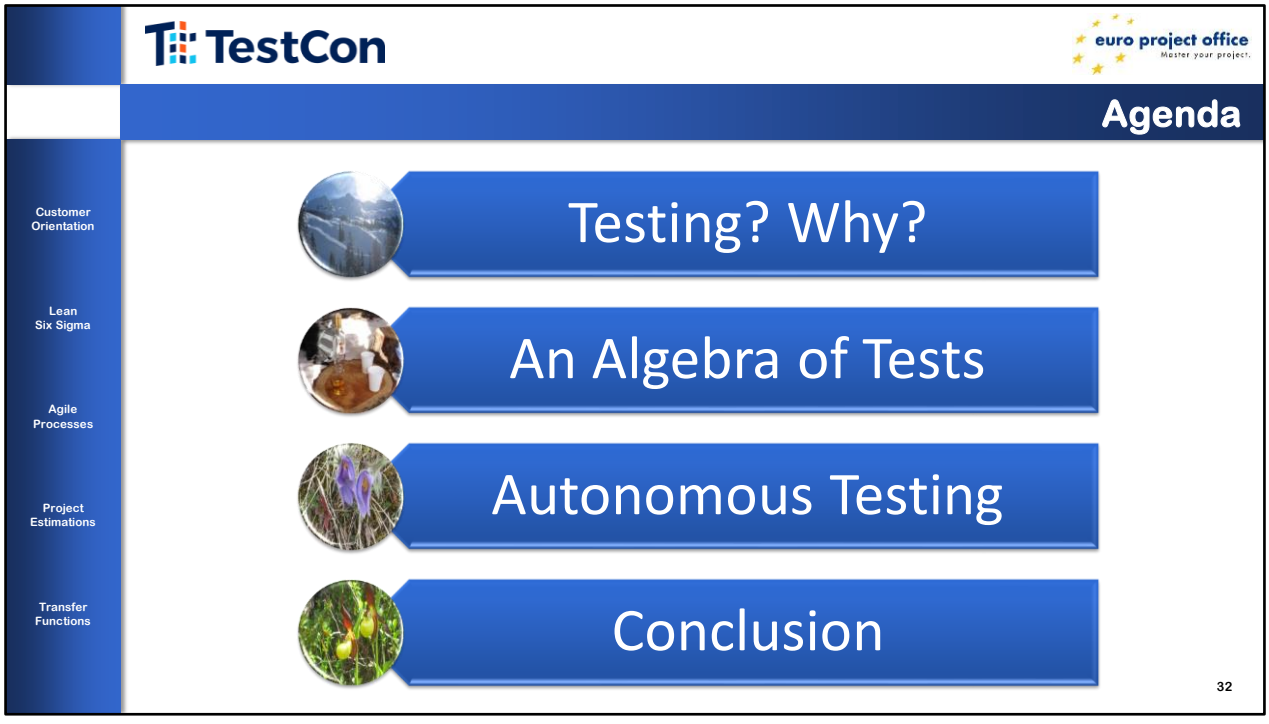
In the classical untyped lambda calculus, every function has a fixed point. A particular implementation is Curry's paradoxical combinator $\boldsymbol{Y}$, represented by

$$\boldsymbol{Y} = \lambda f.\left(\lambda x.\, f(x\ x)\right)\left(\lambda x.\, f(x\ x)\right)$$

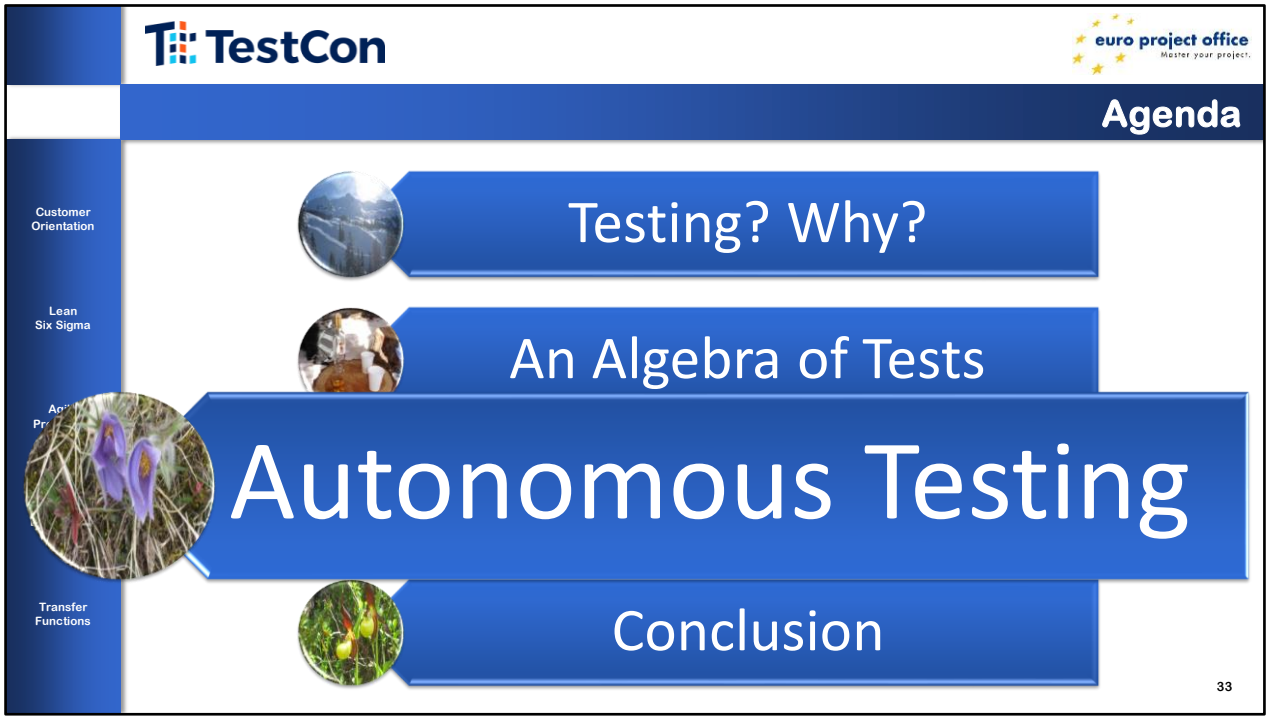
In functional programming, the $\boldsymbol{Y}$ combinator can be used to formally define recursive functions in a programming language that does not support recursion.

Applied to a function with one variable, the $\boldsymbol{Y}$ combinator usually does not terminate. More interesting results are obtained by applying the $\boldsymbol{Y}$ combinator to functions of two or more variables. The second variable may be used as a counter, or index. The resulting function behaves like a “while”-loop in an imperative language.

Used in this way the $\boldsymbol{Y}$ combinator implements simple recursion. Recursion may be achieved by passing in a function as a parameter. The $\boldsymbol{Y}$ combinator demonstrates this style of programming.



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion

TestCon

euro project office
Master your project.

Now We Have Our Fountain Generating Test Cases!

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

Things to Do in Palanga, Lithuania, Musical Fountain, Photo by Tripadvisor

- Our Test Case Generator now works!
- By combining previously defined test cases
- By extracting new test cases from existing test cases
- However, how many???

34

Our arrow term test case generator fountain is now is now gushing with new test cases.

The arrow terms serve primarily as a grammar for test cases, but the properties of a combinatory algebra allow for much more. Test can be combined, using the equation for combining arrow terms or any other combinator. This allows to generate as many test cases as we want and need for achieving full test coverage.

However, a testing environment that produces test cases without end is not very practical either. A selection algorithm is needed to direct the test case generator towards the relevant tests.

TestCon

euro project office
Master your project.

The Main Problem

Customer Orientation

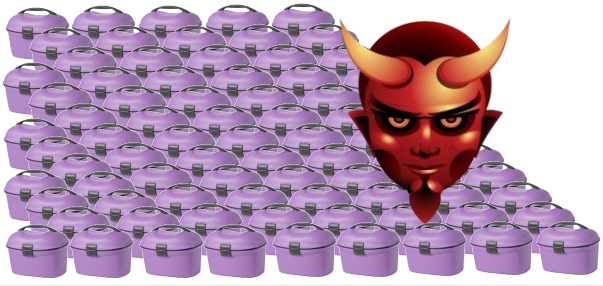
Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- How to select those test cases that are relevant to the purpose of the tests?
  - We cannot execute all possible tests
  - That would possibly exceed the lifetime of the universe



35

Thus, we need an additional mechanism that allows to distinguish relevant test cases from unnecessary ones.

TestCon

euro project office
Master your project.

Savior for the Main Problem

Customer Orientation

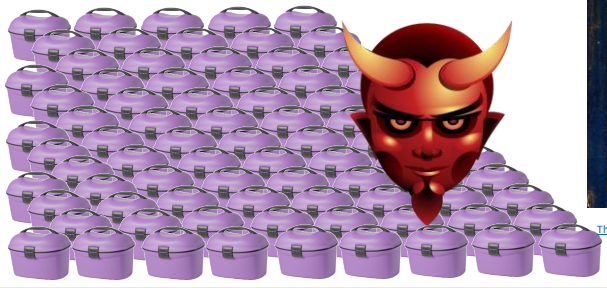
Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

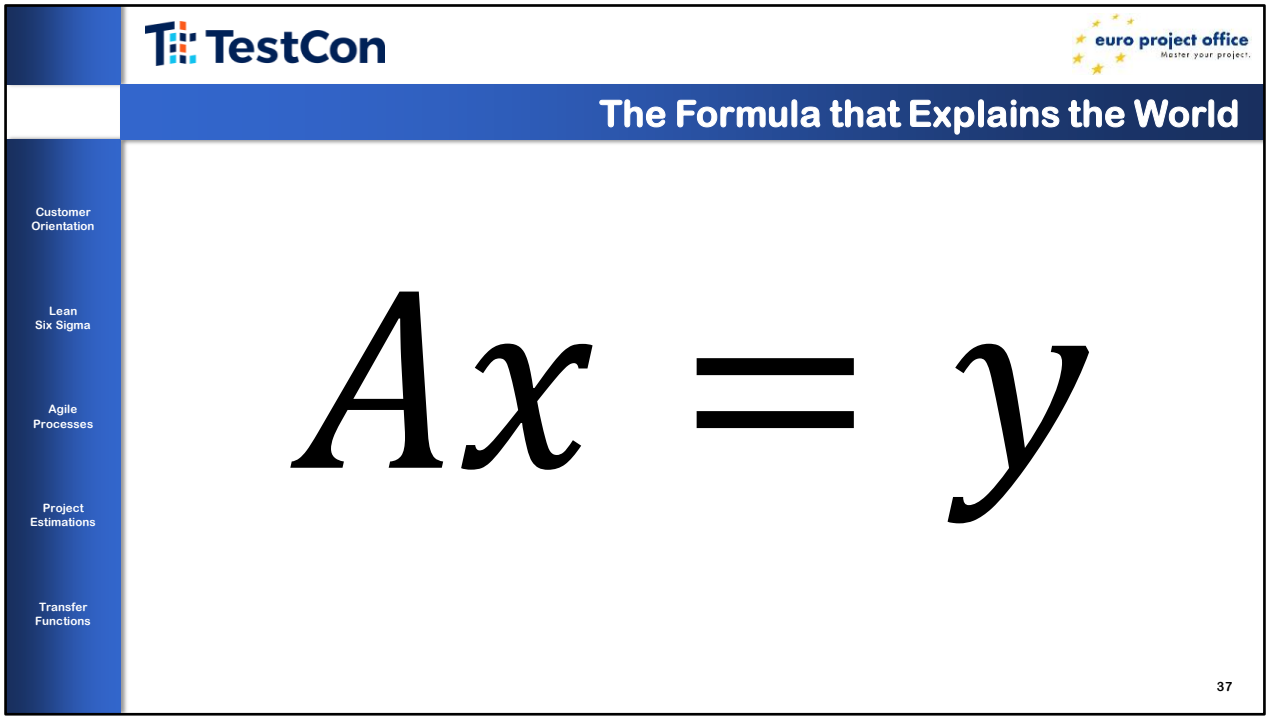
- How to select those test cases that are relevant to the purpose of the tests?
  - We cannot execute all possible tests
  - That would possibly exceed the lifetime of the universe



36

Probably, Angel Gabriel will do it for us.

However, we must understand how angels accomplish their duties.



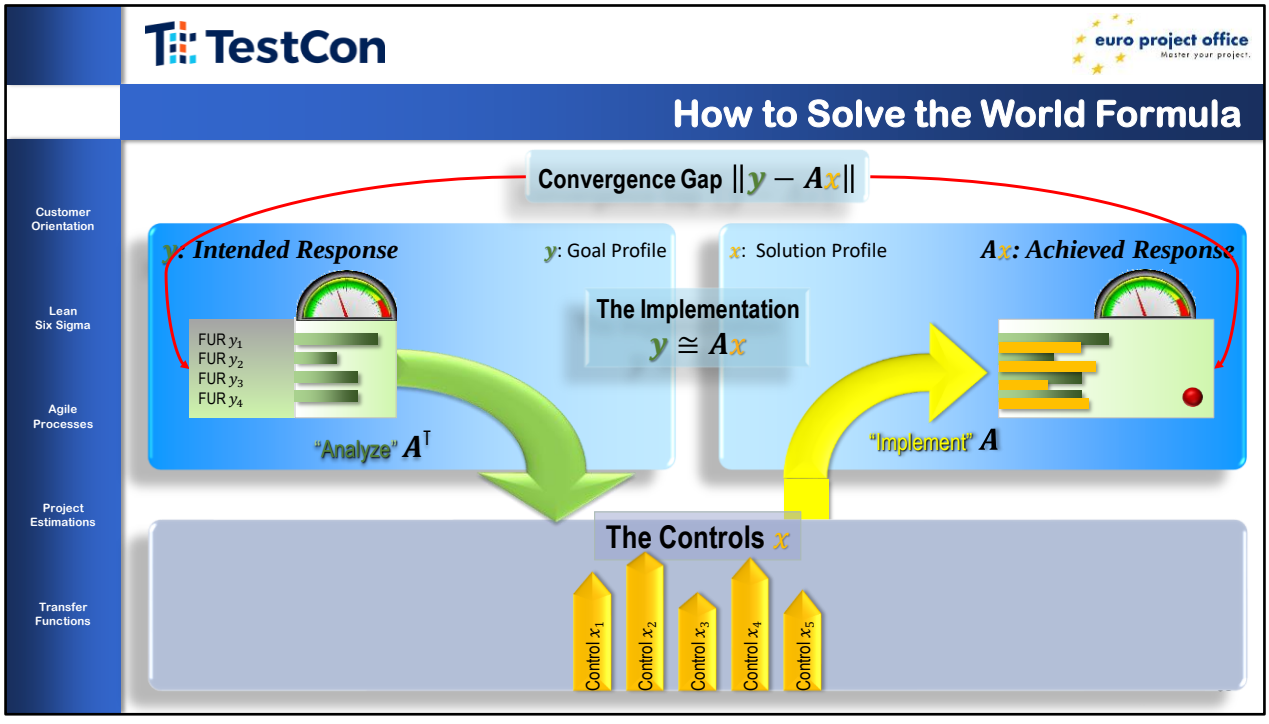
This can be achieved with the help of *Transfer Functions* that map the selection of test cases back onto customer values.

Transfer functions are mappings defined by equations of the form

$$\boldsymbol{y = Ax}$$

where $\boldsymbol{x}$ is the vector describing the importance of the test cases for the customer and $\boldsymbol{y}$ is the vector representing 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.

Now, we're not sorcerers but scientists. This is the formula of the enchantment. Solving the formula for an observable y and the unknown x is what matters for transfer functions. A is the transfer function that we need for the oracle, and that we want to explore. Transfer functions always help if you know what you want but you have no idea how to achieve it.



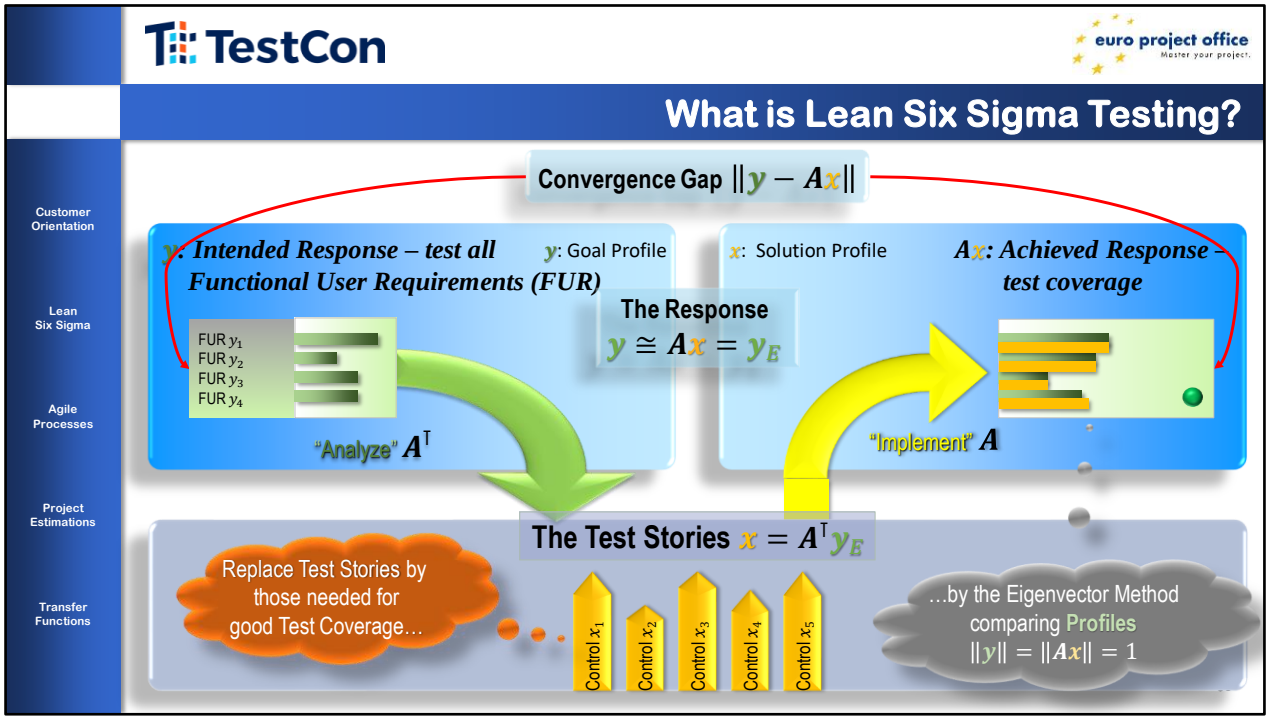
The principle of transfer function is guessing Controls that implement a certain response, be it observed or desired.

You can use this principle for

- Six Sigma DMAIC analysis
- Search Engines
- Software Testing

The problem is how to find Controls with a profile x, and an approximation of A fast enough, such that Ax is near enough to the goal profile. If A is measurable, we are in the Six Sigma domain (see "Digitalization" last year). The Eigenvector method solves such problems. The ***Convergence Gap*** decides whether the guess is valid or not.

If A is not directly measurable, or if Goals and Controls are unclear, then we are in the domain of ***Artificial Intelligence*** where we no longer have algorithms, and linear matrices, but learning.



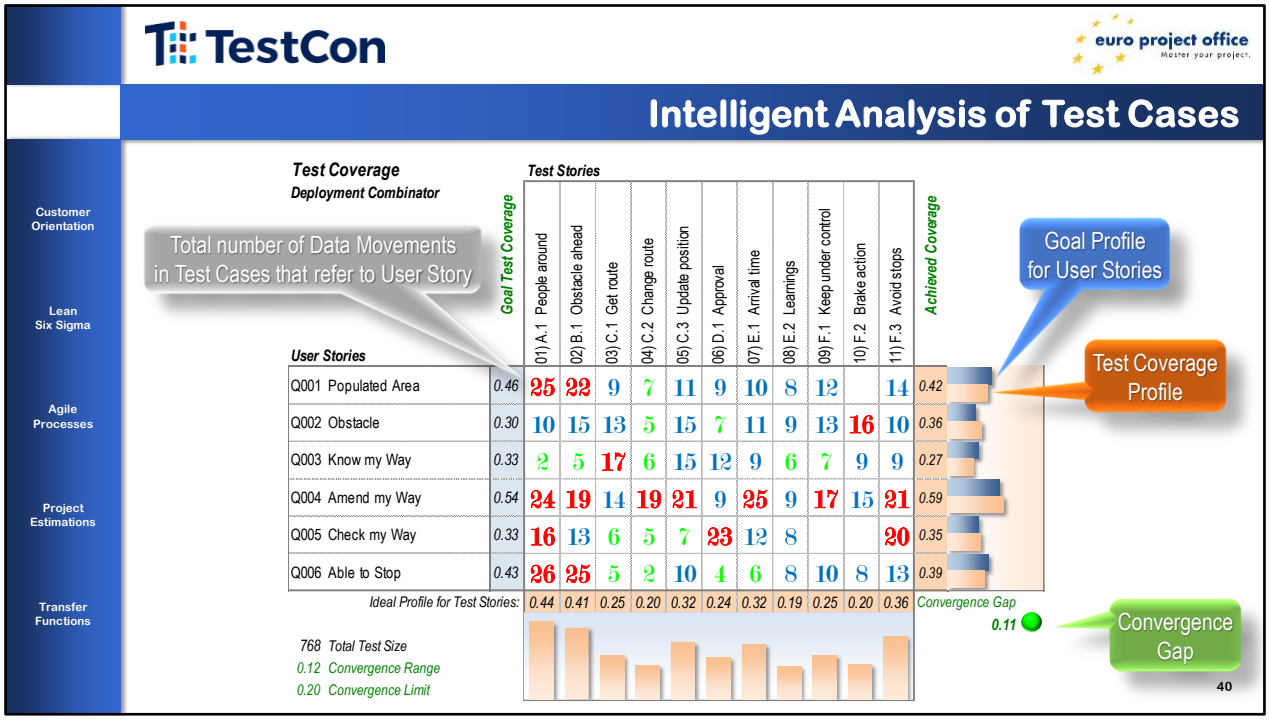
To solve the World Formula, we use Eigenvectors y_E of the symmetric matrix AA^T such that $y \cong y_E$. This yields a minimum set of test stories x Lean Testing means to use a minimum of test cases to meet customer needs and expectations, e.g., safety and privacy, and not waste testing efforts in testing features unimportant to the customer.

Six Sigma means that achievements are measured and compared with these expectations.

To do this, we need **Profiles**, i.e., vectors of Euclidian length one; thus, each vector y fulfilling

$$\sum_{i=1}^{n} y_i^2 = \|y\| = 1$$

where $y = \langle y_1, y_2, \ldots, y_n \rangle$ is the vector with its components and $\|\ldots\|$ the Euclidian norm.

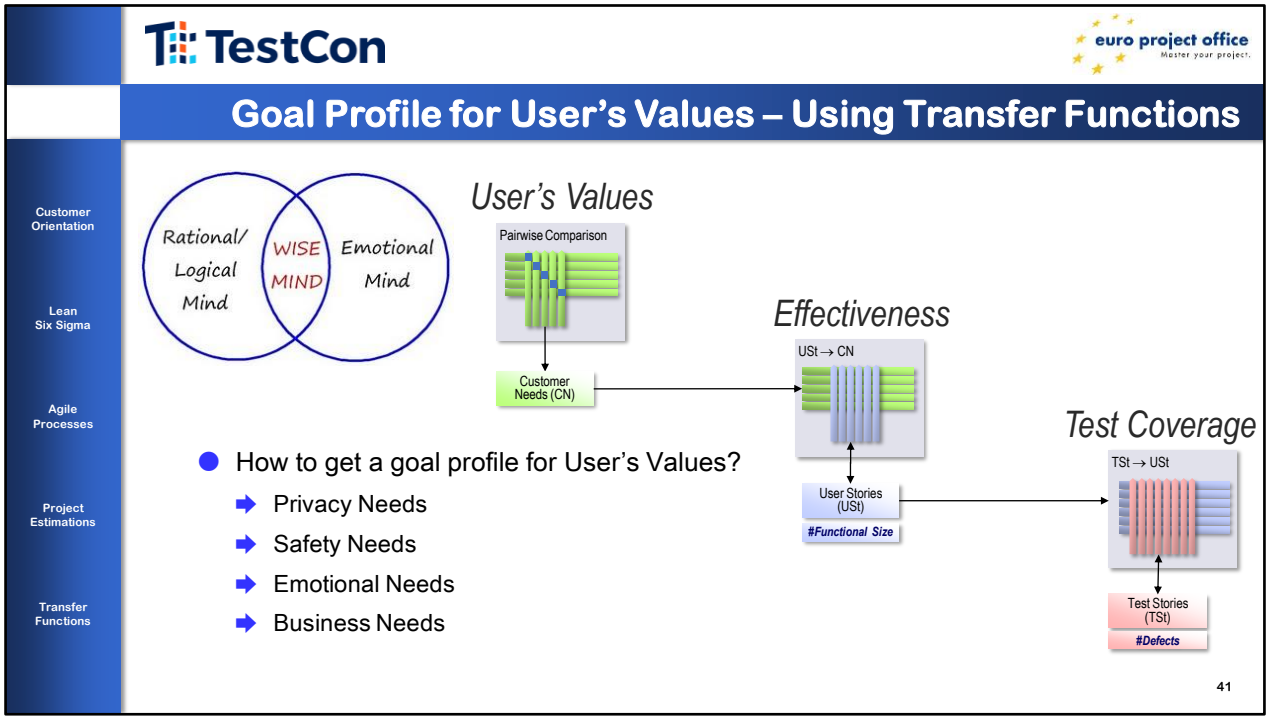


This is the test coverage matrix from test stories into user stories. The transfer function is defined by the number of data movements in each test story that pertains to some specific user story. Obviously, a data movement can occur in many test case, thus belong to many test stories, and pertain to more than one user story.

This is typical for the methods how big data is analyzed.

Its meaning is test coverage; it measures whether there are enough test stories to cover all user stories in full.

The convergence gap measures how well the tests fit the goal profile of the user stories, derived from the users' values. This important feature we'll need to explain further.



However, getting a valid goal profile is difficult, especially when no customer is available, e.g., in product development.

By focusing on privacy protection and safety matters, we are a little bit luckier, as the users' values are quite clear and obvious.

An excellent method to get a profile for users' values is the ***Analytic Hierarchy Process*** (AHP).

We use Six Sigma Transfer Functions to derive the customer value profile for user stories – however, agile teams have a variety of methods to do this. It is important to have profiles in the sense of unit vectors in a topics vector space – not a simple prioritization list without metrics that you can use in a sequence of transfer functions.

User Values first transform into functional effectiveness, yielding priority to user stories, and user story priority define relevance of tests in view of users' values.

TestCon

euro project office
Master your project.

How do We Know What is Relevant?

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

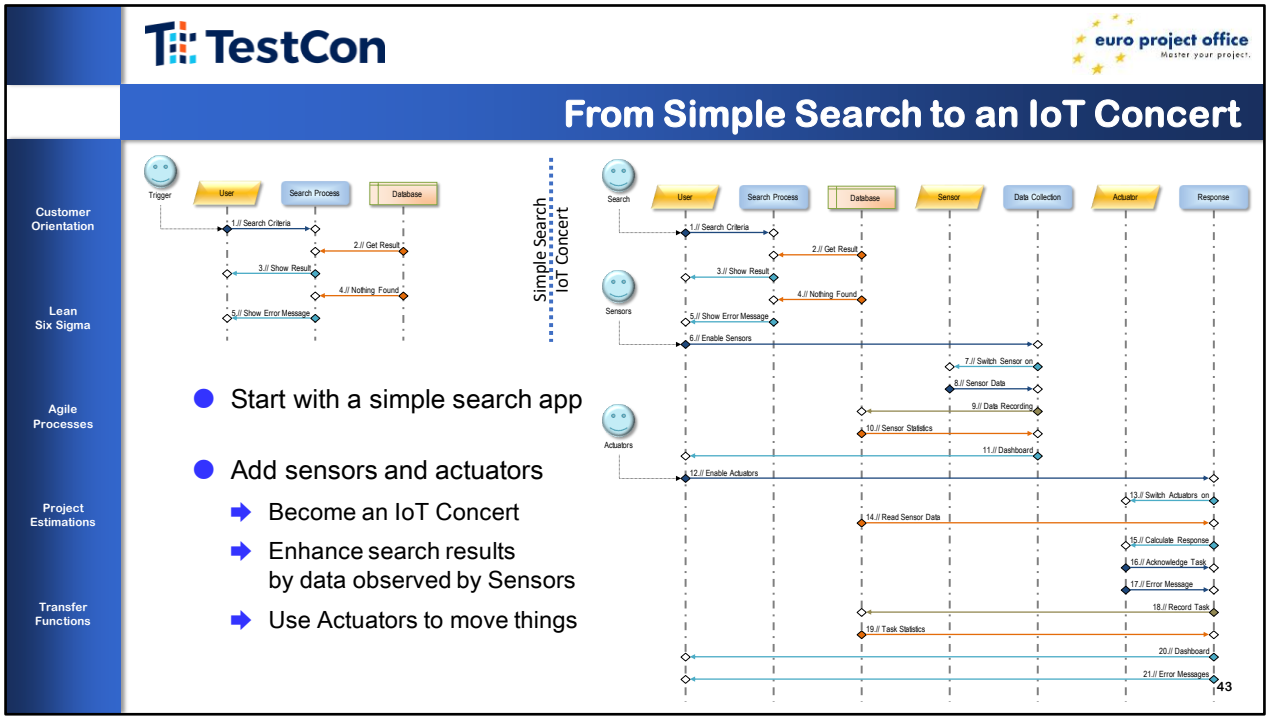
Convergence Gap = $\|y - Ax\|$

- We need a Goal Profile
  - Expressing the relative importance of User Stories
  - Agile Teams have such a profile
    - They need it for setting priorities in the Sprints!
- Compare what's being tested by the Test Stories with the User Story Profile
  - By counting frequency of data movements being executed
- If the Convergence Gap is close to zero
  - Test Stories test the responses that users expect
- If the Convergence Gap opens
  - Some Test Cases that matter for the users are missing, or superfluous

42

A goal profile for user stories is needed in Agile for prioritization. Traditional software development methods didn't need a goal profile; for them, it was enough to know which required FUR were compulsory, and which just nice to have. Obviously, in the beginning of a project, all FURs were compulsory. Only after some time, they detected that half of the initial requirements were obsolete, but many others, especially the important ones, were completely missing.

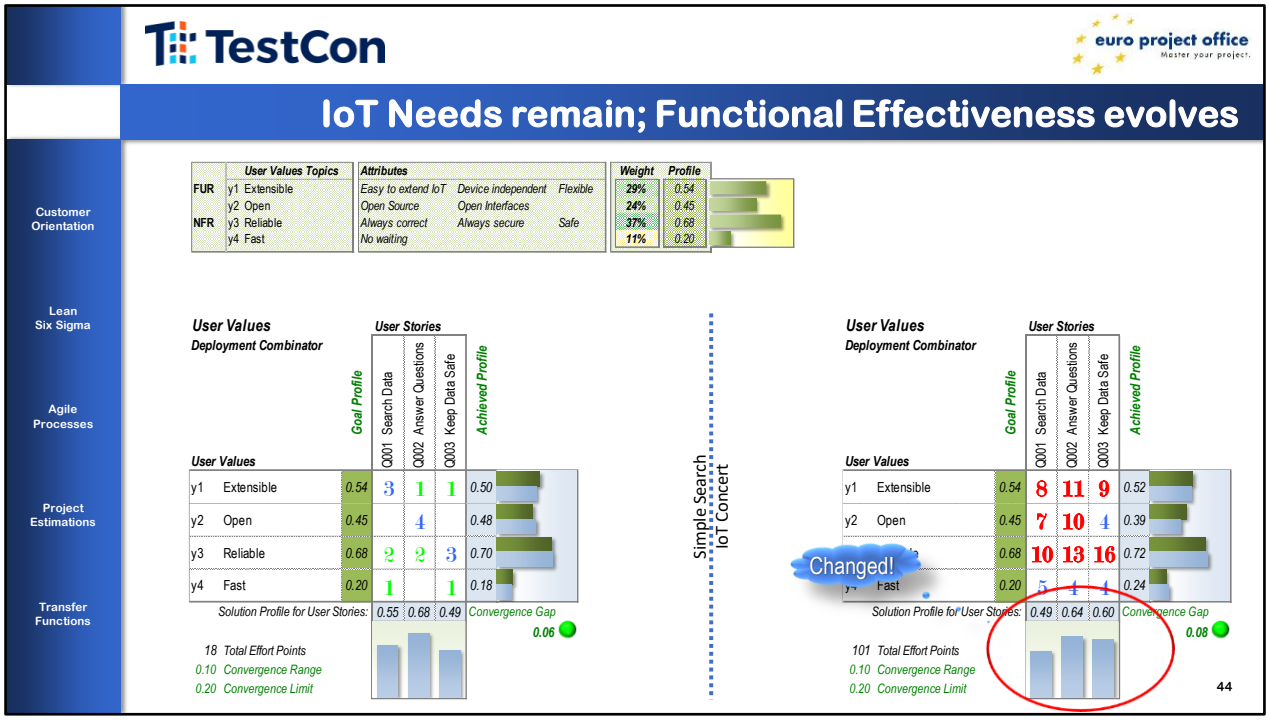
Therefore, agile development methods evolved, where finding the true user stories became the central part of the development work.



Assume our simple search program gets extended by connecting to some sensors and actuators. In case of an IoT concert, sensors can retrieve the searched item somewhere, physically. The actuator in turn could move it to some pick-up location.

Functionality extends considerably but the original tests covered the initial functionality only. In the Internet of Things, testers are no longer around when the system expands, for instance, by autoconfiguration with new components, or thanks to a little help by a gifted home programmer.

Similar things happen if an autonomous robot encounters a new environment that it hasn't be trained for.

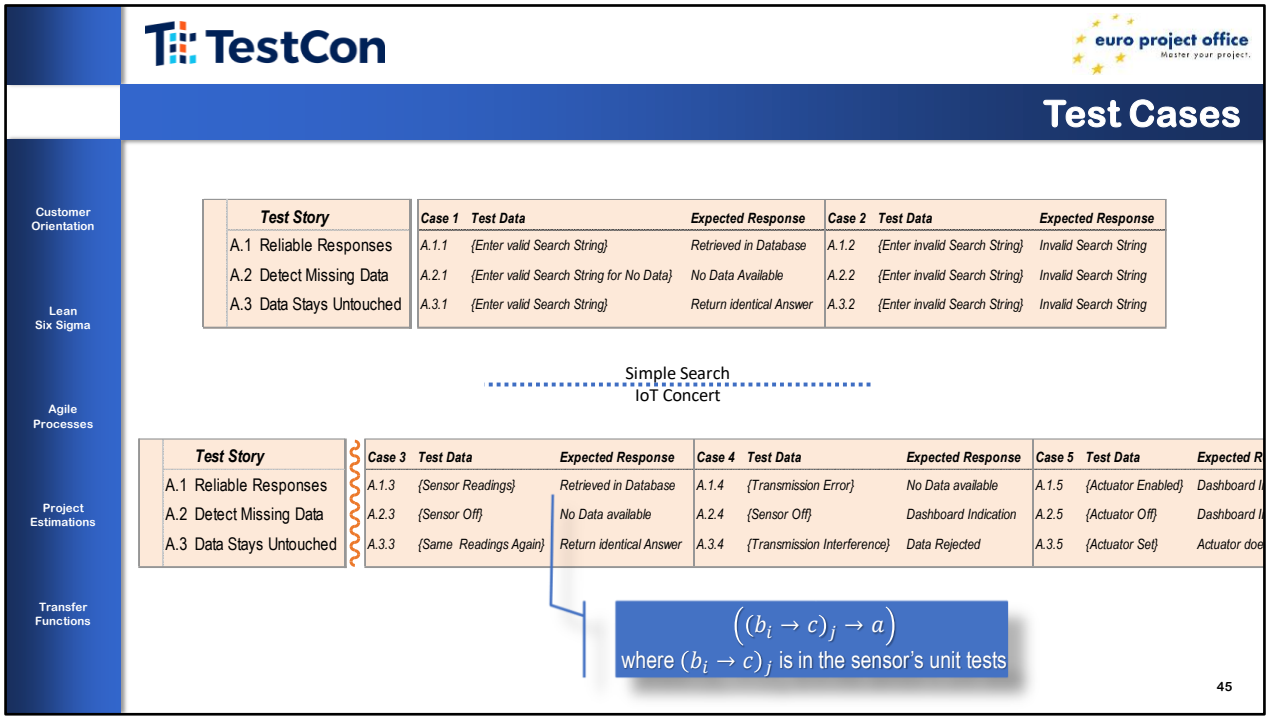


Assume four IoT Needs; two FUR and two NFR.

Functional Effectiveness remains maintained; the extended systems till responds to questions no longer limited to some data base.

However, the profile for the user stories changes, reflecting that communications with the new devices could affect reliability quite a bit.

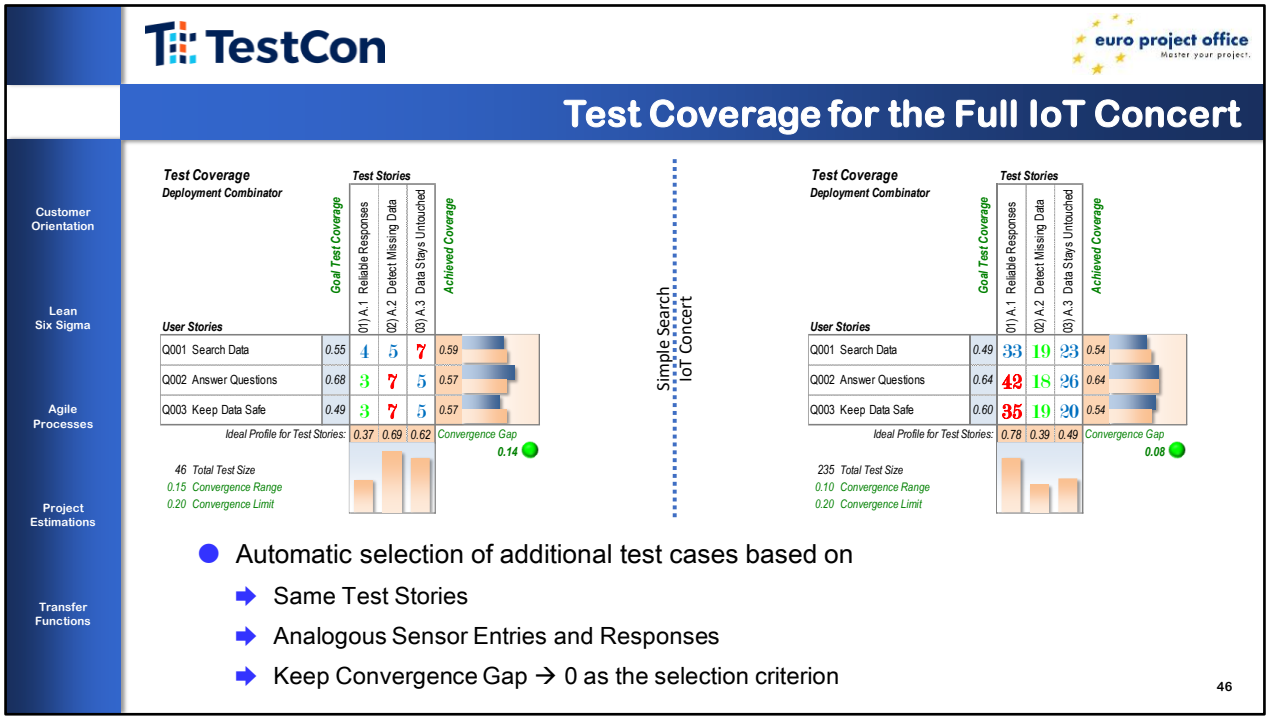
Functional Effectiveness can be assessed automatically because the new data movements have analogies with the initial five data movements for the simple search; sensors and actuators extend the reach of the previous simple search eXits and Entries. Assignment to the IoT Concert thus is by analogy.



Read the test cases with an arrow between Test Data and Expected Response.

For the IoT Concert, more test cases are needed covering sensors and actuators. These additional test cases still refer to the same test stories. The original test cases for Simple Search remain applicable; thus, we start with Test Case Nr. 3 on the second line.

This Test Case Nr. 3 is a combination of unit tests for the sensor with the test cases already in use for the Simple Search.



Also, the test coverage matrices look similar in structure. While the total test size increases considerably, the overall characteristics of the test coverage transfer function remains, but the test story profile adapts to the higher need for reliable and safe data keeping.

Selecting the additional test cases goes by the usual AI search methods using the convergence gap as its hash function.

TestCon

euro project office
Master your project.

An Advanced Driver Assistance System (ADAS)

Customer Orientation

Lean Six Sigma

Agile Processes

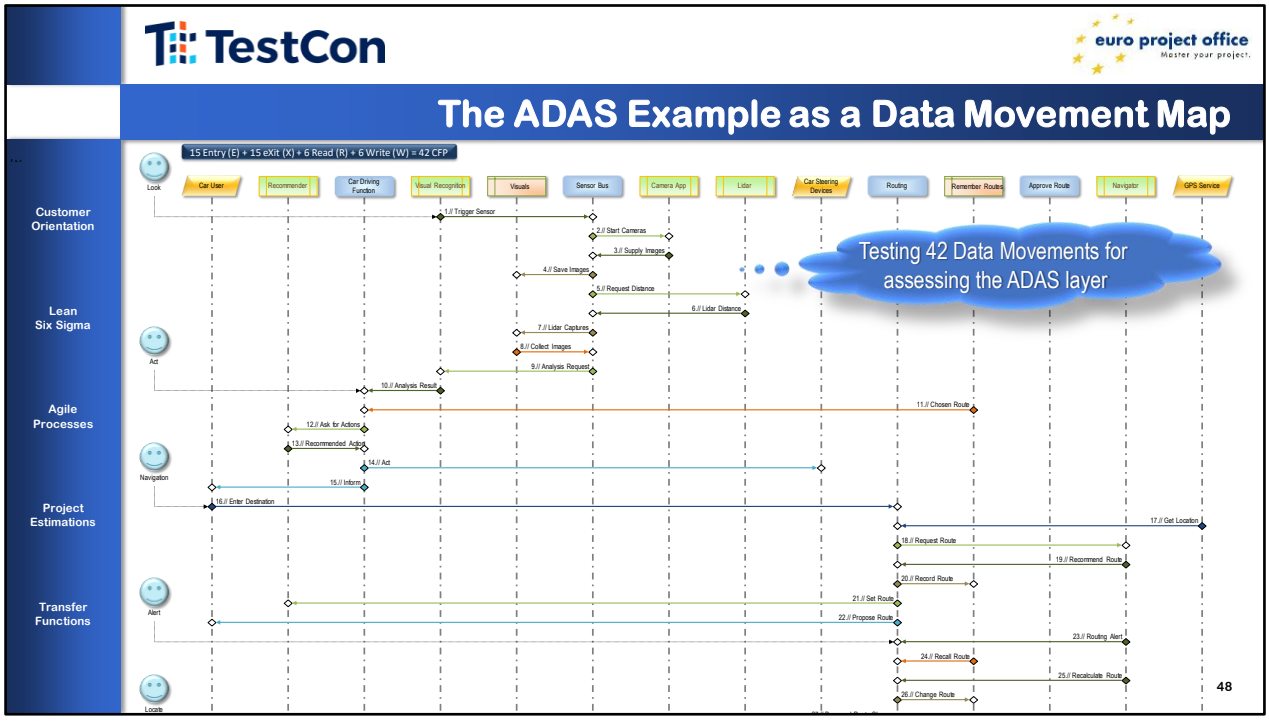
Project Estimations

Transfer Functions

- Advanced Driving Assistance Systems are commonplace in cars and trucks
  - We test a high-level overview, essentially the Kubernetes microservice architecture
- We need a model for the functionality of the software under test
  - When testing an ADAS, the software built into a permanent storage device, e.g., an SSD device, is not relevant, while all functional processes belonging to the car driving functions are
  - Camera, visual recognition engine and driving recommendation engine provide relevant functionality whose cooperation needs being tested
- The Data Movement Map identifies the components under test
  - Other application engines need separate module tests

47

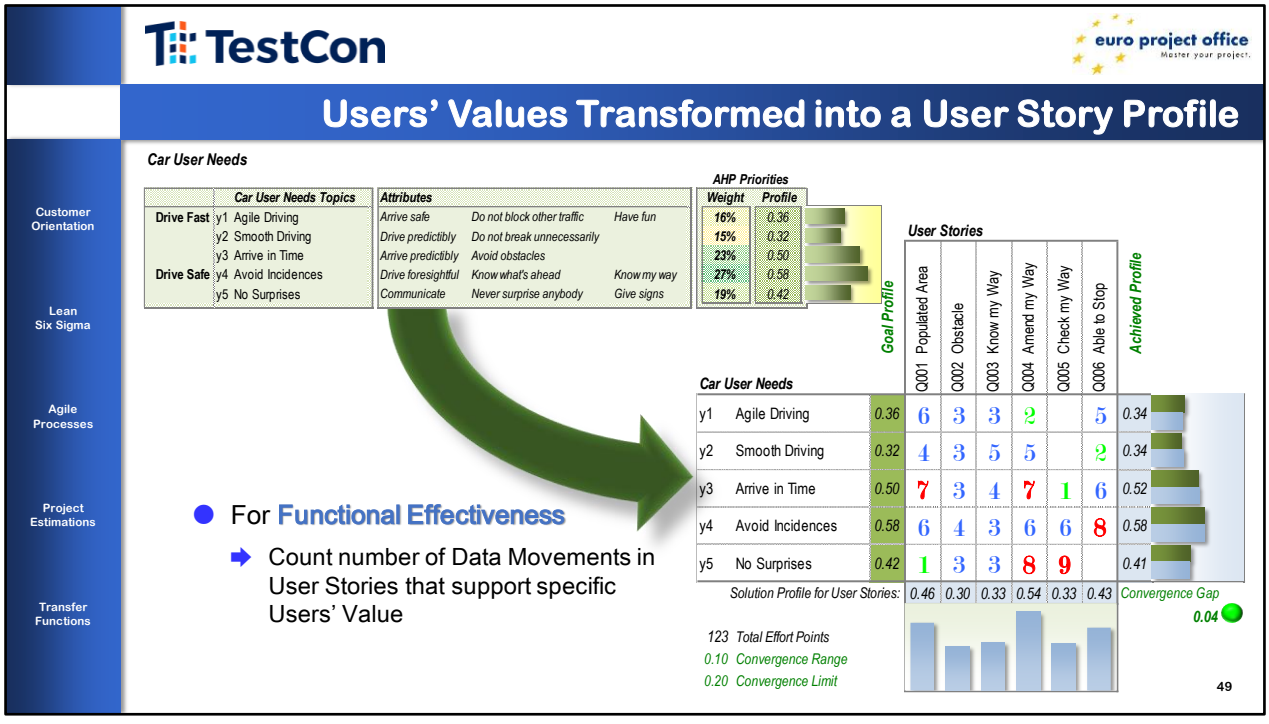
As an example, we test an **Advanced Driving Assistance System** (ADAS) on the top level – microservice architecture – where it might look relatively simply. It is assumed that components such as Camera App, Visual Recognition, the Lidar and the Navigator work as expected. The Recommender might be an AI Deep Learning System, or a more conventional rule-based engine. Its testing is not within the scope of our example.



The **Data Movement Map** shows the top-level functionality with the primary flow of information between the services involved.

Note another advantage of such model-based testing with data movement maps instead of code: you're not obliged to refer to the 10'000s of lines of code for testing, you can refer to the 42 data movements only that are relevant for ADAS tests.

Thus, we test by layers and modules.



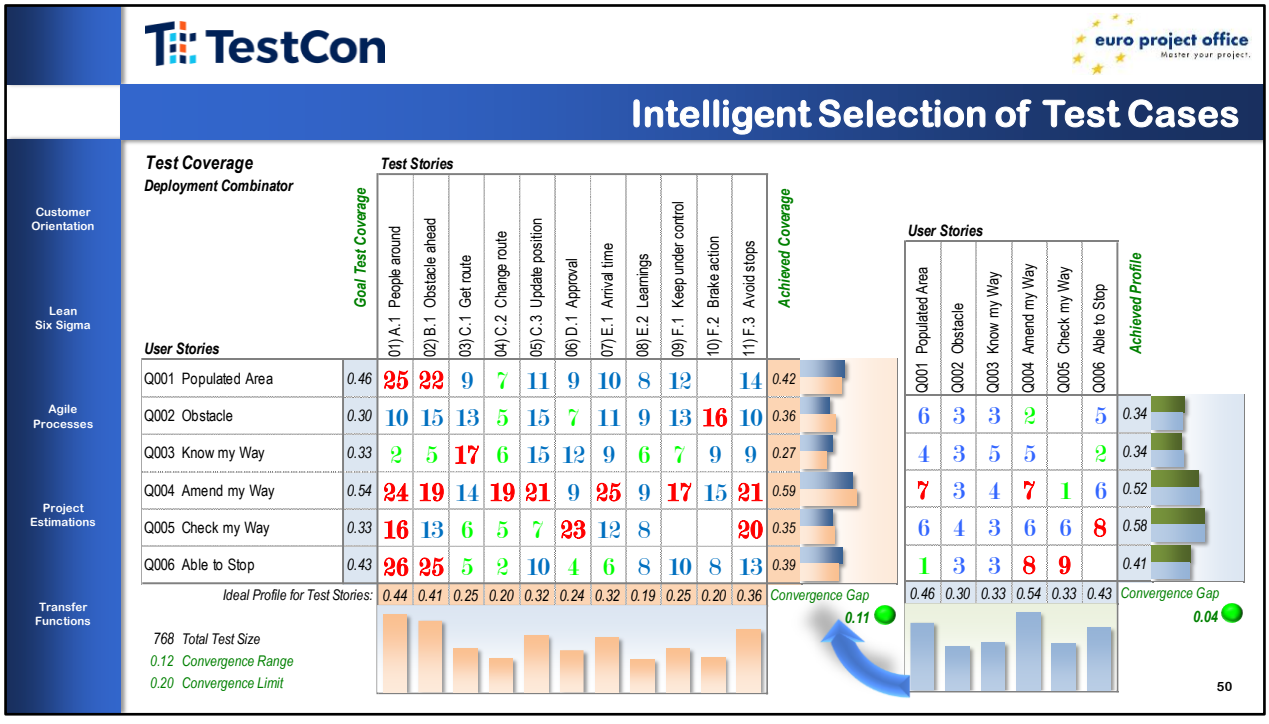
This is a frequency count like the techniques used in big data analytics.

It transforms the users' value profile into a goal profile for User Stories.

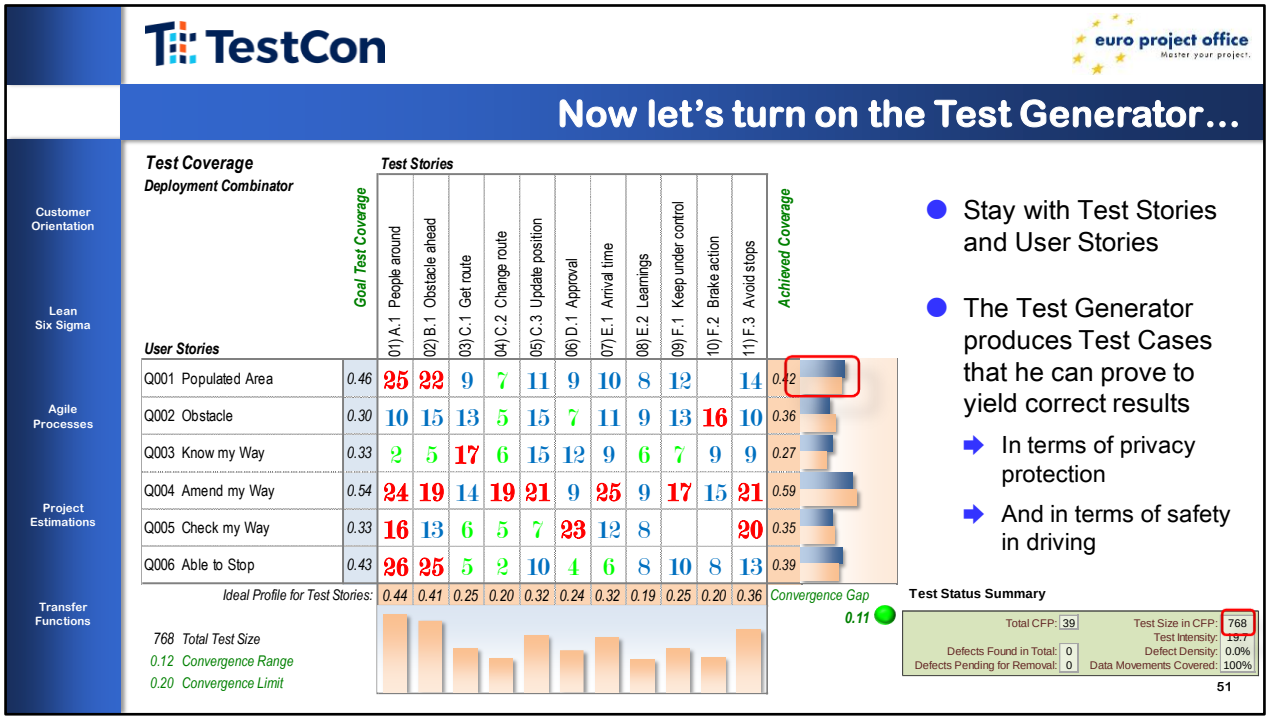
You can also ask users directly for the priorities; however, you will not get a link to the functionality. Here, you know that the functionality implemented meets the values – or the needs – of the users. The transfer function between User Stories and Users' Values is called ***Functional Efficiency***, indicating the degree of functional coverage.

A note of caution: customers do not only ask for functionality – quality aspects such as appearance, ease-of-use, and redundancy also matter. For simplicity, our users care for driving matters only.

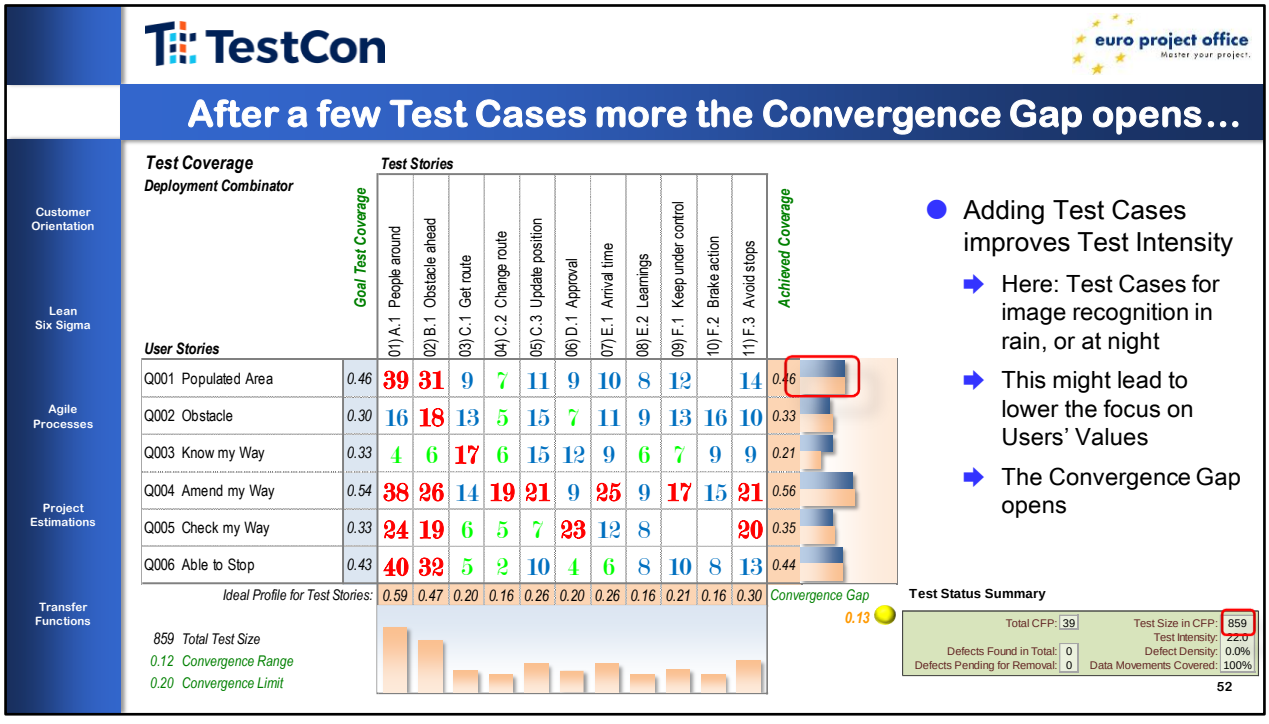
Counting data movements that support some users' value is unfortunately not something that can be easily automated. It requires an understanding of the domain and the semantics behind the software.



From the functional efficiency matrix, we get a profile for the user stories that we can use as goal profile for the test coverage matrix.

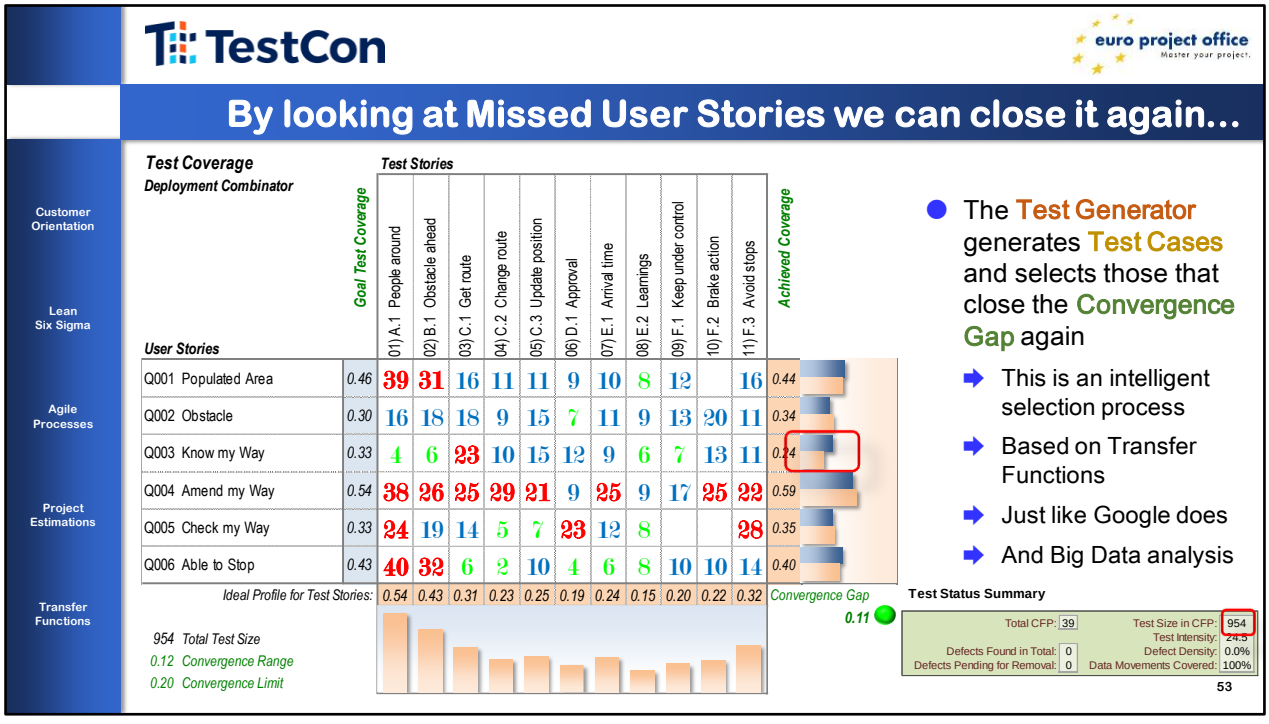


We now turn on the test generator, starting with a convergence gap close to the convergence range, i.e., within 10% of the goal profile.



Now, we add more images to the car driving system "Look&Act".

They focus on recognizing humans, or obstacles, that are typical for populated areas; consequently, the weight of these tests grow in relation to other areas, such as knowing the characteristics of the route taken.



This is just a little bit of Big Data.

The test generator finds that after adding more test cases regarding bad weather, the weather forecast capability of the map service must also be tested and whether such forecast is taken into due consideration when choosing a route.

This sounds very reasonable, and it is. However, we have a hash function that actually allows measuring such “reasonable thinking” and making it independent from peoples’ capabilities.

Naturally, if we have good testers in the team, we do not need artificial intelligence. We better rely on humans. However, often enough these people are in heavy demand and might not find the time to constantly adapt test suites just because a superb new User Story hit the stages of the Backlog.

TestCon

euro project office
Master your project.

Another Example – The D²TLDMUTS

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

- The Double-Decker Tilting Long-Distance Multiple Unit Trainset (D²TLDMUTS) serves as an example to explain the new concepts. D²TLDMUTS is pronounced “Double-Tiddlemutzz”, with a sharp “zz” hiss at the end. It refers to a large Intercity multiple unit trainset, able to run on international railway traffic as a double-decker with restaurant, with children’s corner, offering space for people with disabilities, featuring roll compensation for driving faster around a curve, comfortable enough for three to six hours of daytime train riding
- It has been ordered by a European railway operator, originally targeted for 2013 but in summer 2020, commissioning continues
  - The many different systems of the D²TLDMUTS were only "tested" during the commissioning phase
  - We lack an understanding **how to test complex systems**

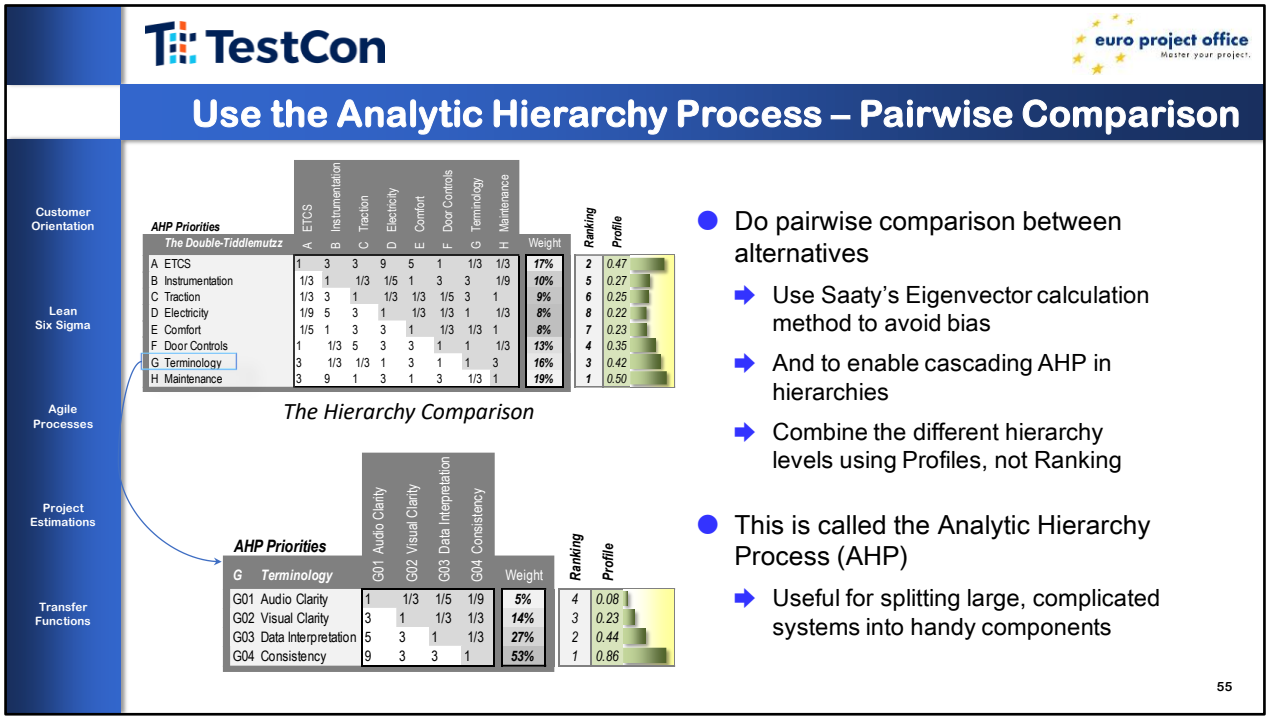
54

The problems encountered with the D2TLDMUTS are basic: it is virtually impossible for humans to create complete test suites for such a complex, software-intense system. Consequently, commissioning such a train set takes very long, much longer than ever planned. Defects touching across the various systems are detected in this trial period only. This is very late, because every modification of train software requires an extra re-certification and a new admission procedure.

Key of testing complex systems is understanding the needs (or values) of the train operator, in our case, or the needs of the customer, in general. The needs of the train operator are the key means for distinguishing relevant test cases from unnecessary tests, allowing test case automation and finally *Autonomous Real-time Testing* (ART).

Commissioning such a software-intense system takes an unpredictable amount of time.

It is unknown how big the software is; probably, even the supplier does not know. Today, publishing software size seems nothing aimed at the public, and train manufacturers still do not behave as a software house, although they are. It is impossible to let testers set up enough test cases, manually. Too many systems interact



You must use quotients, not subtraction. Then you get a matrix that has Principal Eigenvector. This corresponds to a priority valuation that smoothens the individual variations in perception.

Standard values are $9,7,5,3,1, {}^{1}/_{3}, {}^{1}/_{5}, {}^{1}/_{7}, {}^{1}/_{9}$. Reciprocal values are mirrored at the matrix diagonal. If "Visual Clarity" is 3-times more important than "Audio Clarity", "Audio Clarity" in turn is ${}^{1}/_{3}$ of "Visual Clarity".

TestCon

euro project office
Master your project.

The D²TLDMUTS – Testing Large & Complex Systems

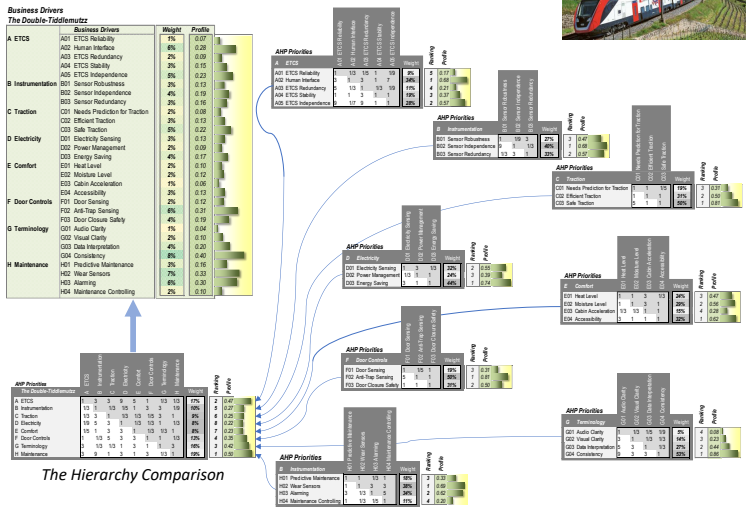
Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions



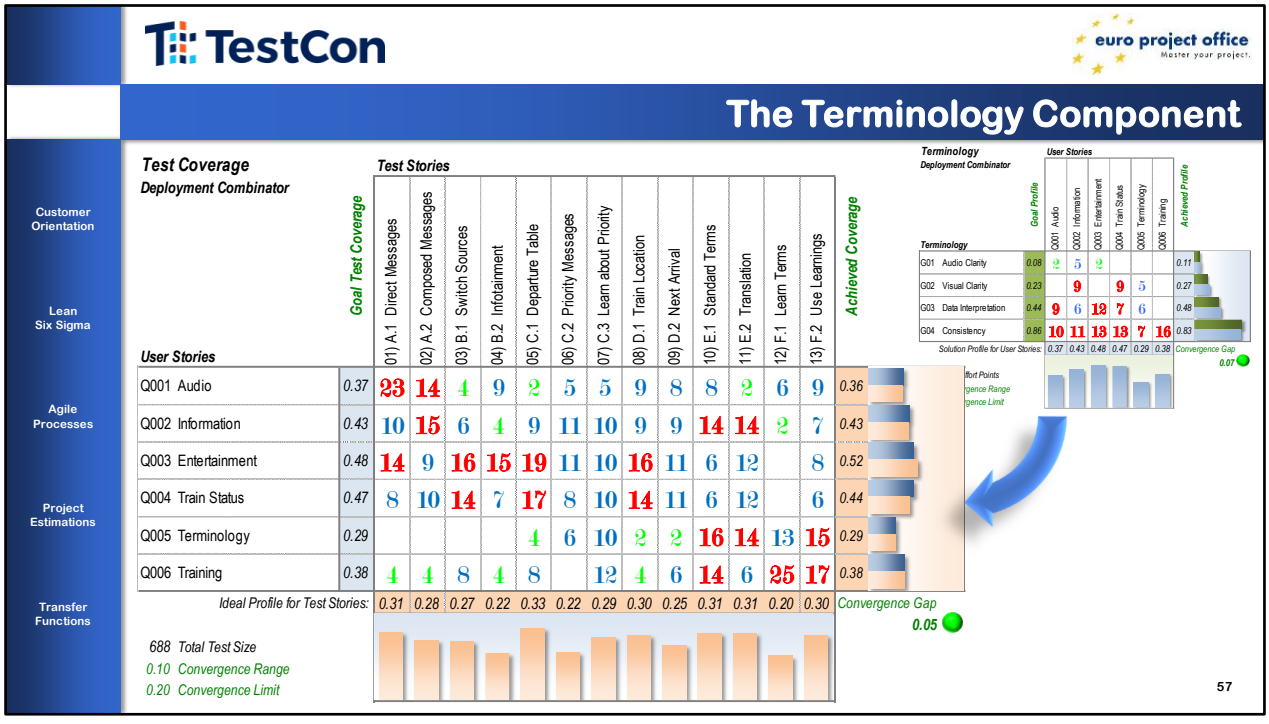
The Hierarchy Comparison

- The method of choice to find priorities is the Analytic Hierarchy Process (AHP)
- Surprise: The need for unambiguous communication G04: Consistency wins, overall
- This is an eye-opener!
  - ➡ We probably need a Terminology Expert

56

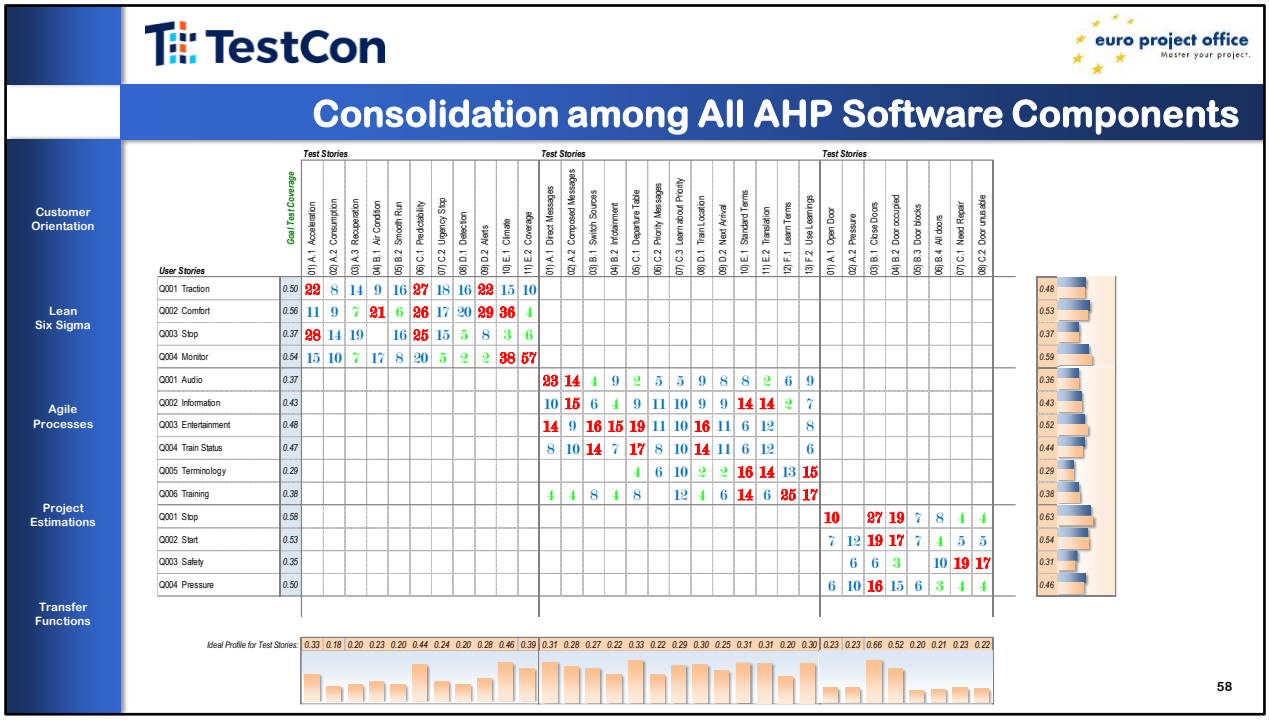
For a train set that assembles components of various suppliers with software developed during different ages, consistent communication is not something for free, but something that requires decent consideration and dedicated work. The components of the D2TLDMUTS originate from different ages and suppliers; regulations have changed over time and with regulation terminology, the meaning of terms.

The hierarchy and combination of priority profile works only because profiles are unit vectors, and because vector spaces can be combined.



The Terminology component translates notions from the various subsystems of the D²TLDMUTS. Without terminology management, talking between the various components of the D²TLDMUTS becomes an issue for testing, but also for running the D²TLDMUTS.

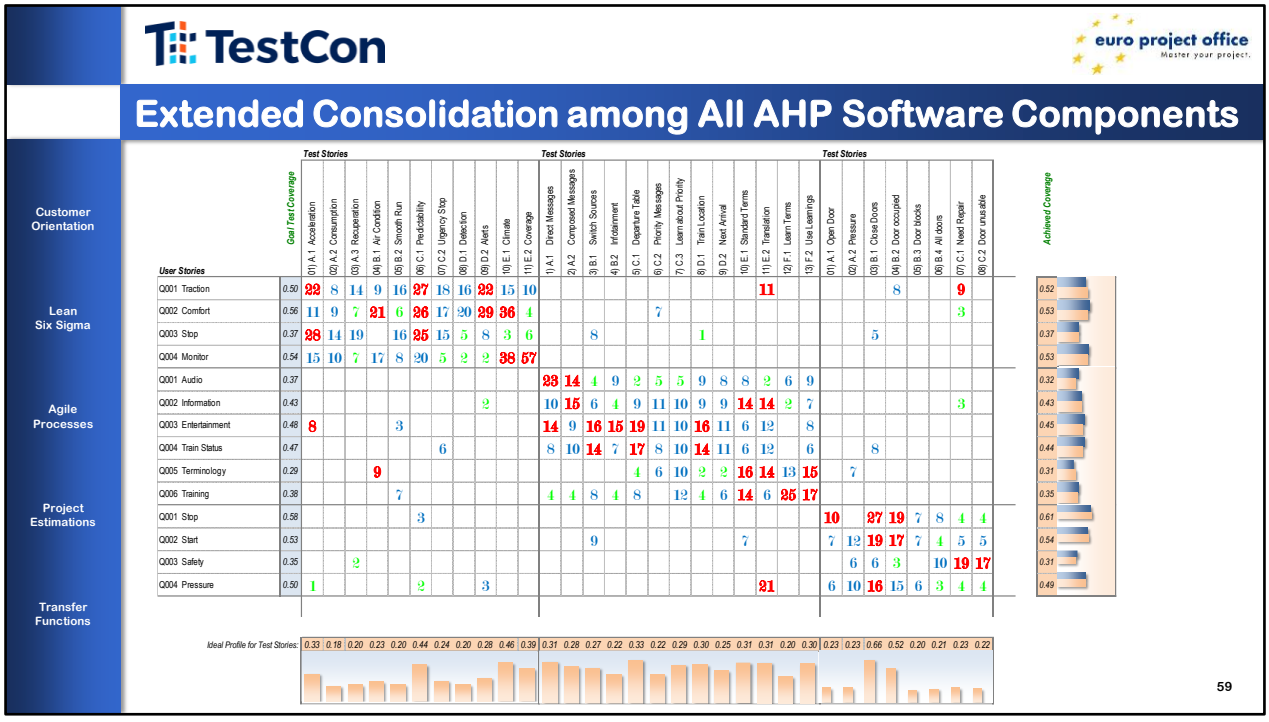
Moreover, without terminology management, testing such large systems is useless.



To build the full test coverage matrix $\boldsymbol{F}$, it is not good enough to add the sequence of test coverage matrices $\boldsymbol{F}_i$, because the parts are of unequal importance for the customer. However, when multiplying each of the matrices $\boldsymbol{E}_i$ and $\boldsymbol{F}_i$ by the respective component of the solution profile $\bar{y}_i$ for the hierarchy comparison, the profiles remain the same and adding these matrices together yields a transfer function from all test stories into all user stories, thus the full coverage matrix. Additionally, its convergence gap remains small because the convergence gaps of the part matrices were already small.

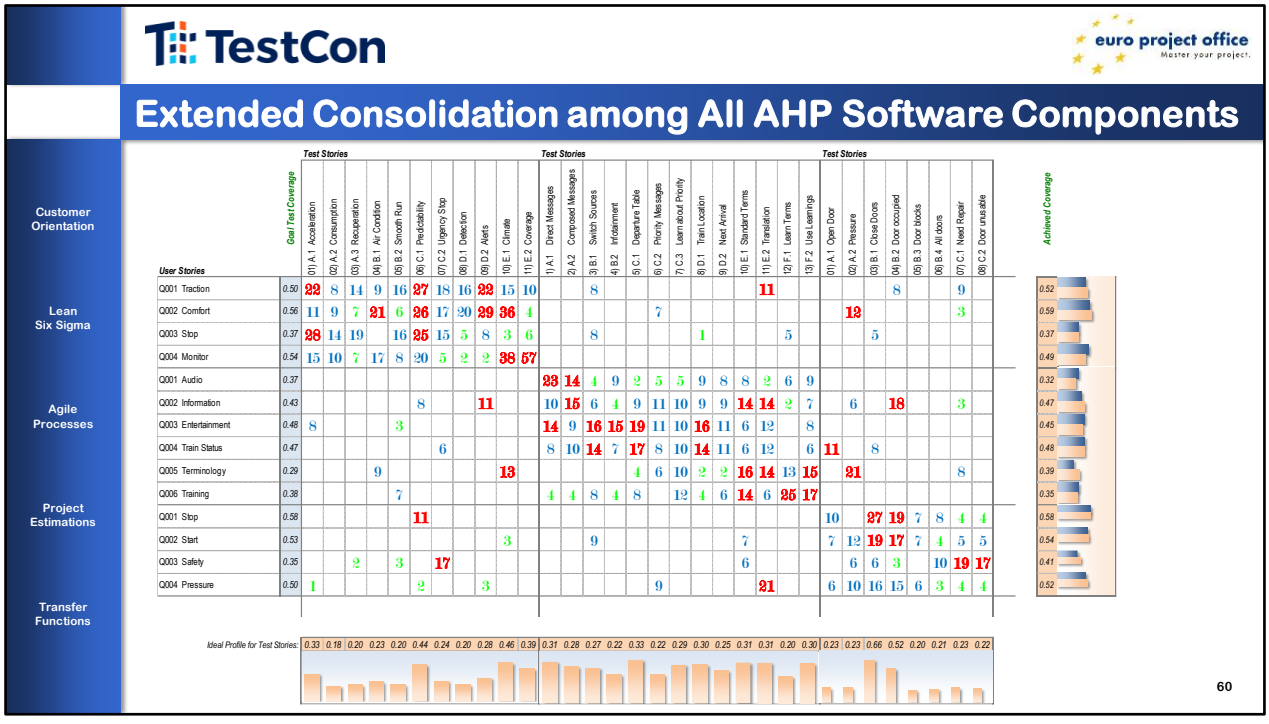
$$\boldsymbol{F} = \sum_{i=1}^{k} \bar{y}_i \boldsymbol{F}_i\,,\ i = 1, \ldots, k; k \in \mathrm{N}; k > 0$$

The matrix $\boldsymbol{F}$ is sparsely filled: no test cases exist outside of the diagonal part matrices $\boldsymbol{F}_i$. This means that no test cases cover the interactions between different part applications required by the $\boldsymbol{A}_1$. However, such interactions exist and are essential for proper functioning of the whole complex system. Also, the initial set of test cases contains enough test stories that suggest test cases linking different part applications. In the D2TLDMUTS case, suitable test cases use the Combination application and eventually the Terminology application to move data across the other part applications.

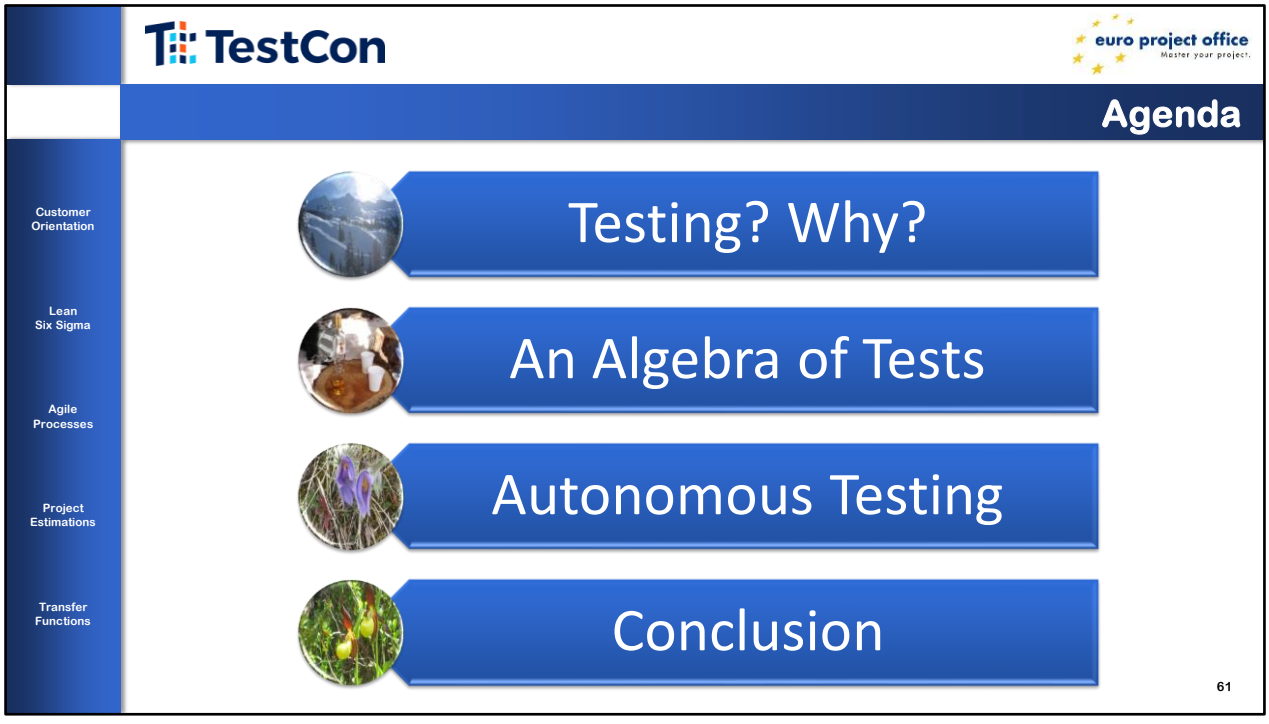


Now let the ART algorithms on and see how the empty space begins to fill up, with test cases combining various applications.

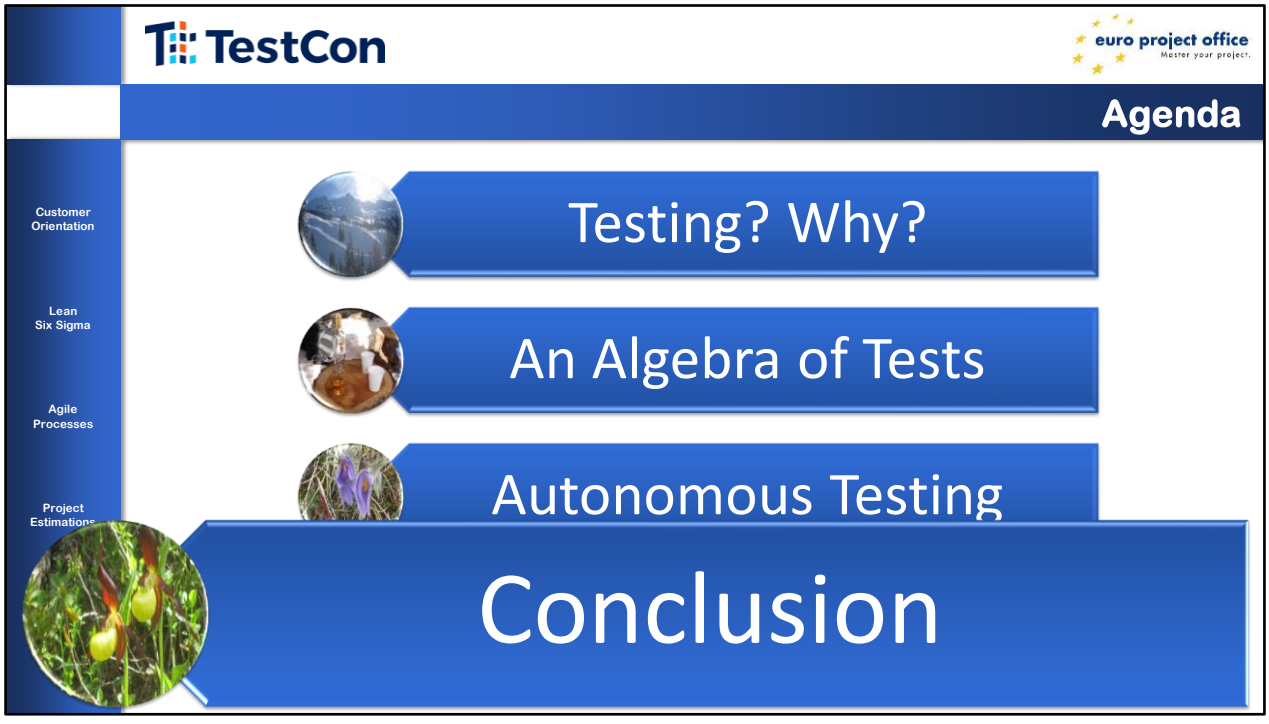
The Terminology application is paramount for defining data movement paths between different applications.



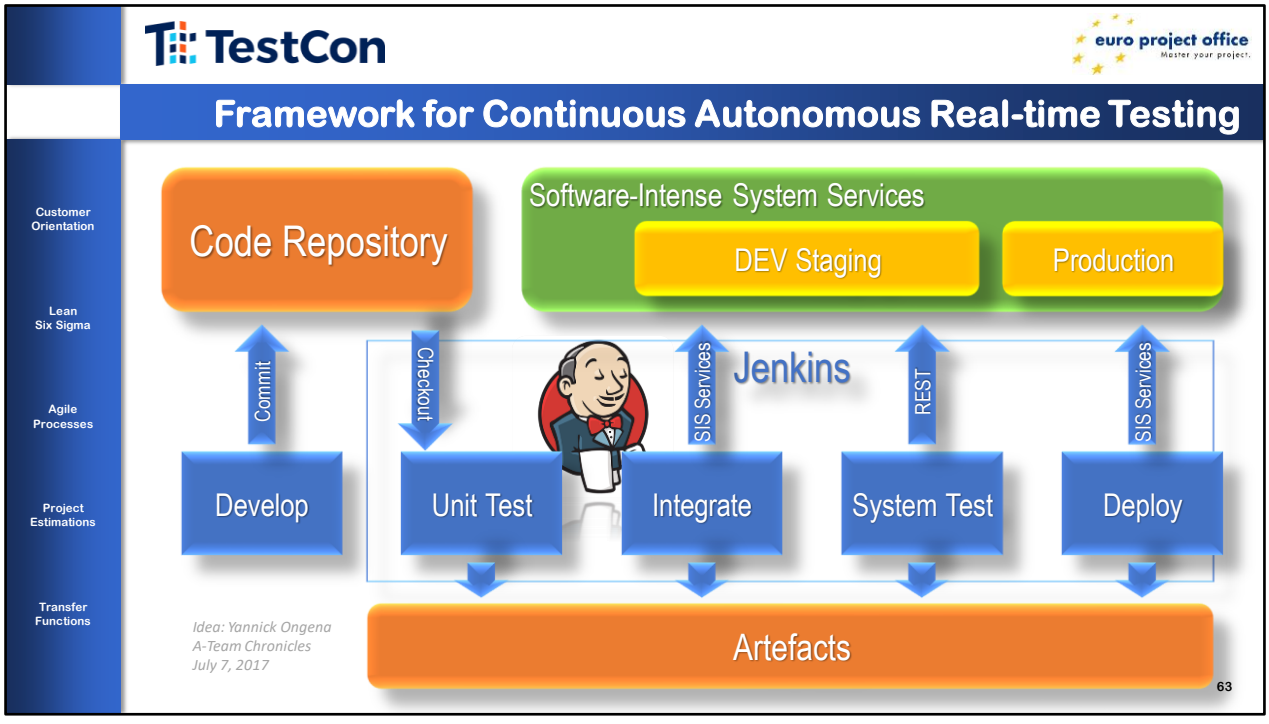
The ART algorithms can fill as many empty cells as needed to achieve satisfactory test intensity. Test coverage might remain with 100%, but test coverage alone is probably not enough for safety-critical systems.



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion



- Testing? Why?
- An Algebra of Tests
- Autonomous Testing
- Conclusion

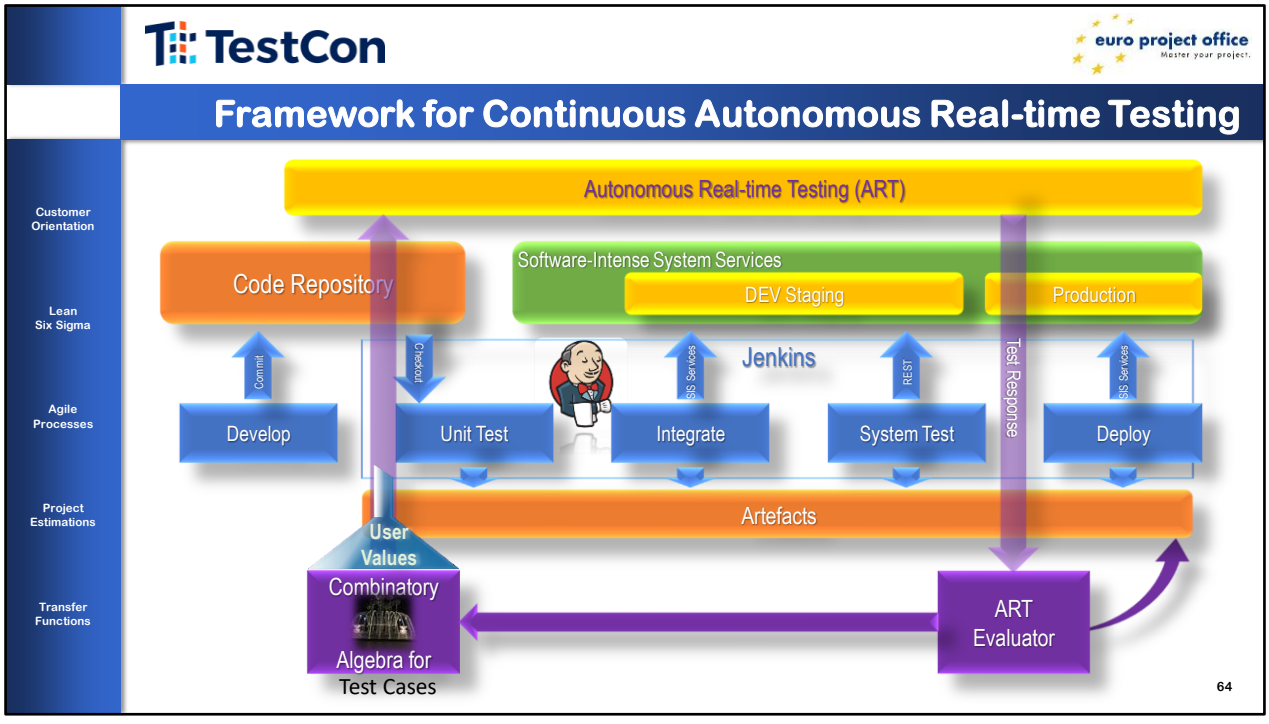


Tools for Continuous Integration / Continuous Delivery

One of the standard tools used in software development is the open-source software Jenkins. The default interaction model with Jenkins, historically, has been web UI driven, requiring users to manually create jobs, then manually fill in the details through a web browser. This requires effort to create and manage jobs to test and build multiple projects, it also keeps the configuration of a job to build/test/deploy separate from the actual code being built/tested/deployed. This prevents users from applying their existing CI/CD best practices to the job configurations themselves.

The new approach is creating pipelines. Users can implement a project's entire build/test/deploy pipeline in a "Jenkinsfile" and store that alongside their code, treating their pipeline as another piece of code checked into source control.

Similar approaches are advertised by Atlassian, the provider of Jira, the software development tool of choice used by agile teams.



The framework for CI/CD is extended by an overlaying activity: Generating test cases that continuously test the Software-Intense System (SIS), 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 activity but extends to product lifetime. Test results are kept for future learning and improvement.

TestCon

euro project office
Master your project.

Conclusions

Customer Orientation

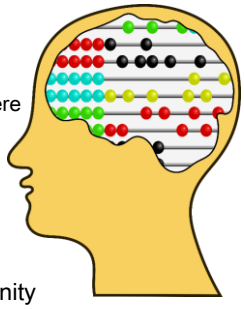
Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- You do not have to know implementation or code of the software under test
  - Analyze user needs instead – use the Analytic Hierarchy Process (AHP)
  - Draw a Data Movement Map that meets their expectations
  - You can test functionality of Neural Networks and Support Vector Machines as if they were ordinary software
  - But you must do it continuously
    - Because learning systems keep changing
    - You must test AI under operations, real-time conditions, everywhere
- Artificial Intelligence is not intelligent
  - It's big data, stupid!
  - AI deals with variations
  - AI must be constantly tested before you allow it impacting humanity



65

AI embraces variation; contrary to Six Sigma.

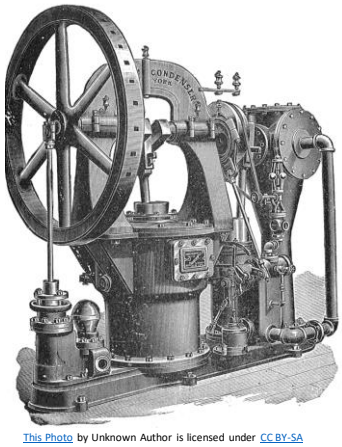
You can draw a Data Movement Map and use it to derive as many test cases you need. You can extend your tests using the Combinatory Logic approach.

TestCon

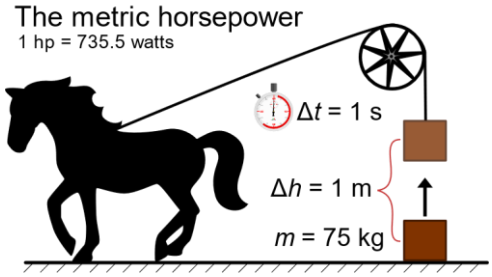
euro project office
Master your project.

The Horsepower Metric made the Difference

- Customer Orientation
- Lean Six Sigma
- Agile Processes
- Project Estimations
- Transfer Functions



This Photo by Unknown Author is licensed under CC BY-SA



By Own work, original version in German by Sgbeer - This file was derived from: Pferdestaerke.svg:, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=35217113

How good is James Watt's steam engine compared to the horses that did the work before?

66

The idea used by James Watt was to help market his improved steam engine. ... "Watt found by experiment in 1782 that a 'brewery horse' could produce 32,400 foot-pounds [43,929 N·m] per minute." James Watt and Matthew Boulton standardized that figure at 33,000 foot-pounds (44,742 N·m) per minute the next year.

All the owner of a draft horse needed to know was that Watt's steam engine could do 5 times more work than his single draft horse was doing.

TestCon

euro project office
Master your project.

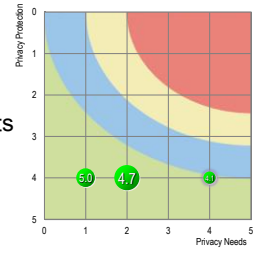
Software Metrics for the Automotive Industry

Customer Orientation
Lean Six Sigma
Agile Processes
Project Estimations
Transfer Functions

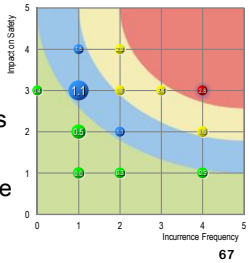
- Functional Size
  - COSMIC
    - E.g., 23'456 FP

- Test Metrics
  - Test Size
    - E.g., 723'456 FP
  - Test Intensity
    - E.g., $\frac{723`456}{23`456} = 30.8$
  - Acceptable Defect Density
    - E.g., 0 Defects

- Privacy Index & Minimum Privacy
  - For Data Movements
  - Featuring overall Privacy Protection



- Security Index & Maximum Risk
  - For Data Movements
  - Featuring overall Safety Risk Exposure



67

Besides Functional Size and Test Metrics, the customer needs to know how well Privacy and Safety are assured by tests. Graphical indications that are easy to understand are needed. The proposal here is oriented towards FMEA and probably not exactly what the average customer understands.

The recommended way is to give software customers confidence on both privacy protection and safety risk hazards; whatever is relevant.

Since the COSMIC model is easy to build automatically from code, test size should also be measured, and thresholds can be agreed in contracts.

TestCon

euro project office
Master your project.

Software Metrics are Something for the Public

Customer Orientation

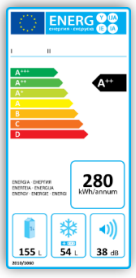
Lean Six Sigma

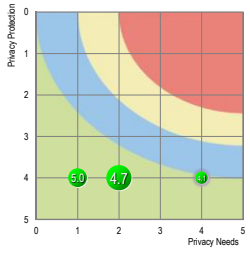
Agile Processes

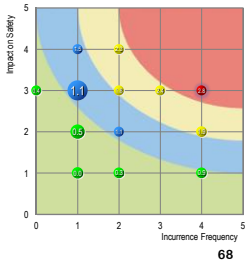
Project Estimations

Transfer Functions

- We trust software more than humans
  - Human can be distracted, angry, sick, exhibit bad humor
  - Software is more predictable, repeatable and is testable
  - Sometimes, we believe software was tested before released
- However, how should consumers know?
  - We need metrics that consumers can understand
  - For instance, Privacy Protection and Safety Risk exposure
  - With graphical indicators similar to EC power consumption labels







68

It is important that software which is autonomously real-time tested, gets a label that consumers understand. Otherwise, there is little chance that somebody is ready to pay for the additional cost of testing.

The graphics to the right summarize privacy exposure, respectively safety risks, based on our data movement map for the Autonomous Driving model used in this talk. This presentation is a proposal; in order to make it a standard; automotive industry is challenged. Or the EC, or consumer organizations, could raise the necessary demand to make it a standard.

However, in automotive, a fast move is improbable. The mindset is still that of car-makers, not of programmers. Digitalization still creates **Angst**.

TestCon

euro project office
Master your project.

Testing without Measurement is Waste

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- Software Testing is the Horsepower of Digital Economy
  - You need to know how much testing was done
  - You do not need the technical details of testing
  - Testing creates value
- Testing without measuring density and relevance for customers is waste
  - You need COSMIC for sizing tests
  - You need Voice of the Customer techniques to measure relevance for customers and users

69

Software Testing is Big Business, but without telling anybody how much testing was done, it's waste.

TestCon

euro project office
Master your project.

Constraints

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions

- Tools are not ready yet
  - The tools shown here are Excel prototypes not fit for production
  - The mathematics behind transfer functions and combinatory logics are not commonly known
  - What I've shown here is an outlook into what's coming upon us with Digitalization
- However, this future is just around the corner
  - While enthusiasm for autonomous cars has faded, ADAS driving assistance systems are abundant and ready to become daily gadgets for consumers
  - Every smartphone today needs SW metrics and autonomous real-time testing after downloading new Apps, or interconnecting with new services

70

While setting up a test generator based on the AI tools offered for instance by Microsoft, is certainly possible, it's not for free and requires substantial effort. Tools such as Jira are open for extensions; such extensions exist not yet.

However, the need for automated real-time testing is already here. Whenever you connect your smartphone to some cool new App, you should have the right to know what this means for your privacy protection. I'll appreciate running a test case that shows me who now has unlimited access to all my shared data…

TestCon

euro project office
Master your project.

My Tools

Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

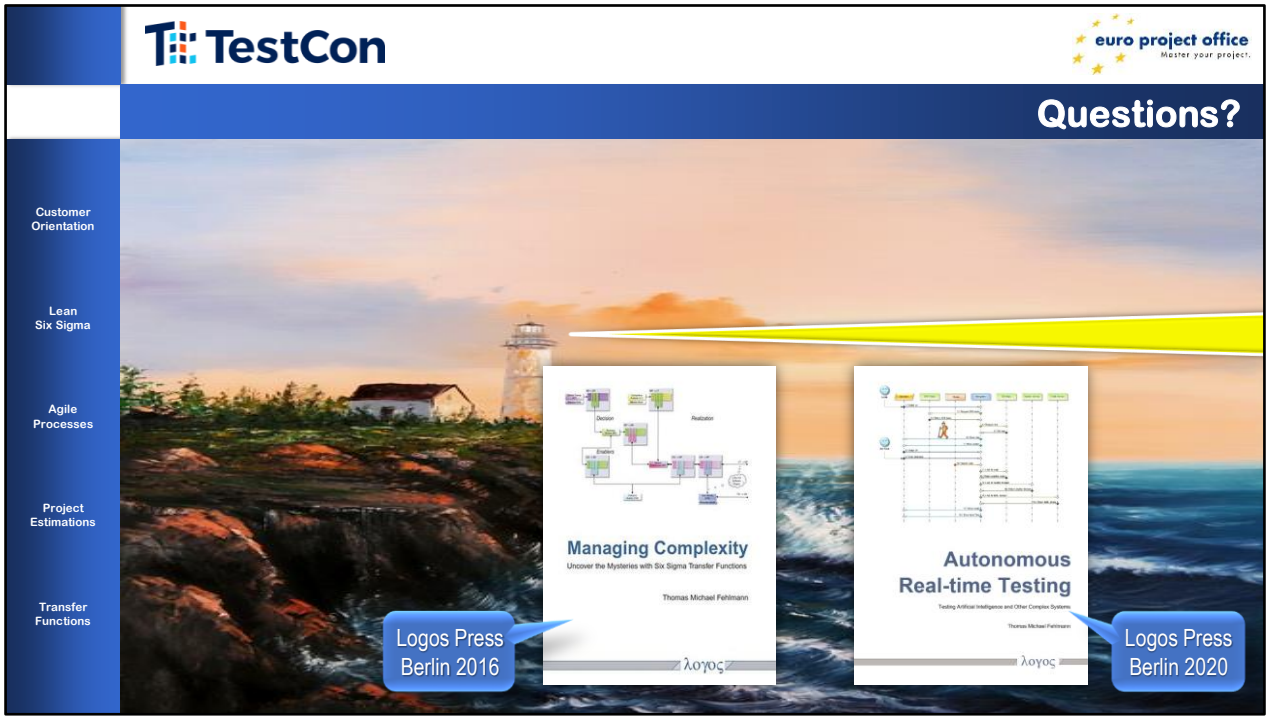
Transfer Functions

- If you dare to play with the tools that I used for this presentation
  - Here's the link: https://web.tresorit.com/l#tiKdO7HSgiZVOLcaN4-eZA
  - This is a two-way encrypted read-only tresor, not easily accessible for anyone who tries to modify my VBA code ☺
  - You'll find
    - The Excel templates
    - Many example talks
- Recommended before trying:
  - Connect with me on LinkedIn
  - Send e-Mail to thomas.fehlmann@e-p-o.com
  - Or leave me your business card

71

The samples presented in this talk are generated with tools programmed by the author, as Excel-based prototypes in VBA. Because of limited performance, their applicability is for small projects only, and requires quite a bit of manual, not automated, work.

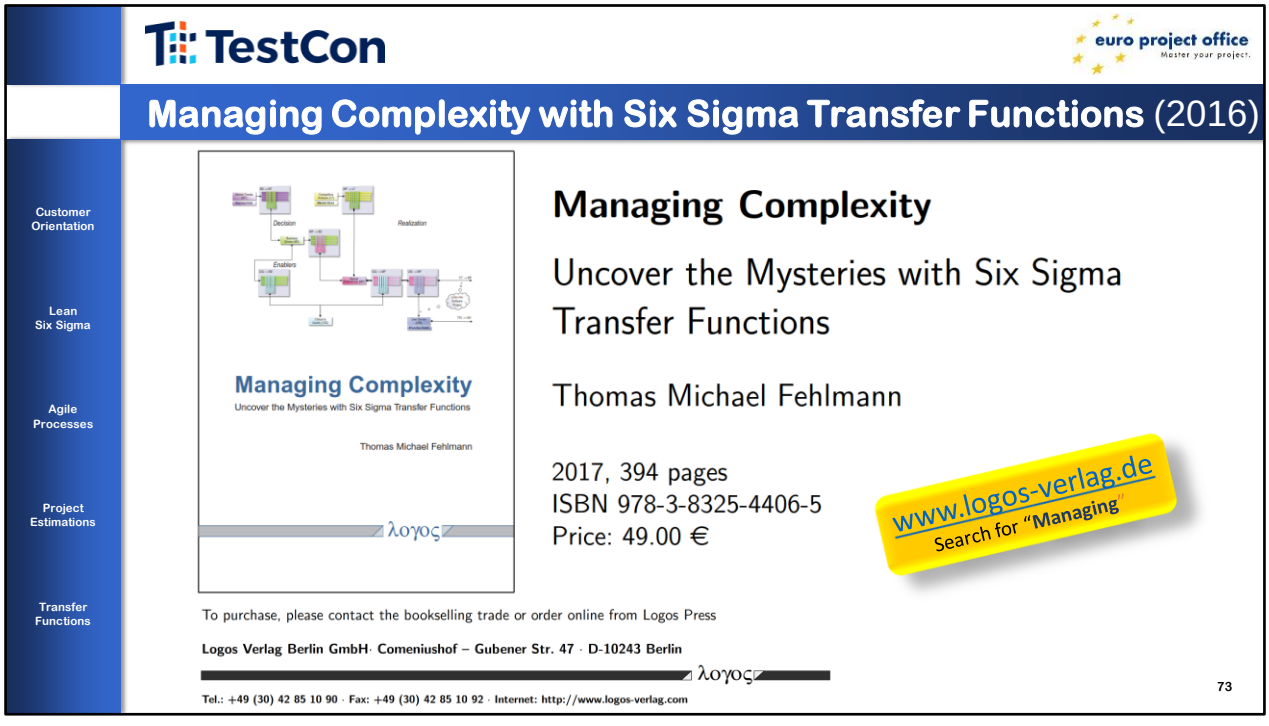
The tools used are publicly available, although not for sale, under a GNU Free Documentation License. However, I recommend to contact me before you start using them because they are not at all state of the art, or even – sad to say – well tested.



The speaker has published quite a bit on the subject together with Eberhard Kranich in Duisburg – e.g., in QFD symposia, at SW metrics conferences like IWSM / Mensura; in quality management and testing conferences, also Lean Six Sigma Conference in Glasgow, Strathclyde and Zurich and in the ATINER series of scientific publications.

Managing Complexity appeared 2016 in Logos Press, Berlin: https://www.logos-verlag.com/cgi-bin/buch?isbn=4406

Autonomous Real-time Testing is available since January 2020 with the same publisher: https://www.logos-verlag.com/cgi-bin/engbuchmid?isbn=5038



The rise of Information and Communication Technology (ICT) in the second half of the 20th century became the dominant force in economics. Its rise accelerates in the first 15 years of this century at an astonishing speed. The world of ICT right now is in the process of cosmic inflation.

In the early universe, quantum fluctuations in a microscopic inflationary agile region became the seed for growing structures in the universe of galactic nebula, galaxies and stars, making the universe transparent. This phenomenon, familiar to physicist and cosmologists, happens right now to ICT. The current observation is that "things" of the physical world become intelligent, receive IP addresses and connect to the Internet. The possibilities to create new ICT-based products seem unlimited; however, sponsors must fuel the inflation.

Complexity was already an issue when developing software in the early days of ICT. Software development is often done in projects that turn out to be exploratory in the sense that they aim at translating human voices, uttering requirements, into a machine-readable language. Requirements for the software to be build are usually not known at the beginning; the project must uncover them. Developing software without knowing the outcome in advance is a complex undertaking. Predicting the outcome of software projects by proven methods of civil engineering did not work out well.

Now, new levels of complexity arise with ICT. Agile approaches are appropriate for software development; however, predicting the outcome of projects still is difficult. New techniques must manage the growing levels of complexity within ICT. Fortunately, mathematics has provided these new techniques. They rely on transfer functions and Eigenwert theory. Its usefulness already has been proven in major search engines of this century. However, this is not the end of the story.

This book makes the mathematics of Lean Six Sigma transfer functions available to ICT practitioners. It provides the basic theory, explained with many examples, and even more suggestions, how Six Sigma Transfer Functions help with complex problems.

TestCon

euro project office
Master your project.

Testing Artificial Intelligence & Other Complex Systems (2020)

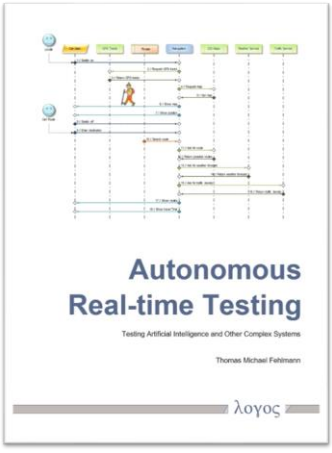
Customer Orientation

Lean Six Sigma

Agile Processes

Project Estimations

Transfer Functions



Autonomous Real-time Testing

Testing Artificial Intelligence and other Complex Systems

Available since January 2020

www.logos-verlag.de
Search for "Testing"

To purchase, please contact the bookselling trade or order online from Logos Press

Logos Verlag Berlin GmbH · Comeniushof – Gubener Str. 47 · D-10243 Berlin

λογος

Tel.: +49 (30) 42 85 10 90 · Fax: +49 (30) 42 85 10 92 · Internet: http://www.logos-verlag.com

74

Autonomous Real-time Testing is the sequel to Managing Complexity; it shows how the same tools used before also help mastering Artificial Intelligence, and complex systems that otherwise are hard, if not impossible, to test.

Appeared early in 2020.