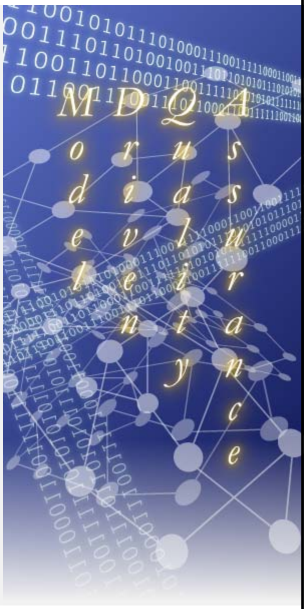
Model Based Generation of TTCN-3 Test Cases

Tutorial

Antti Huima
Managing Director
Conformiq Software Ltd



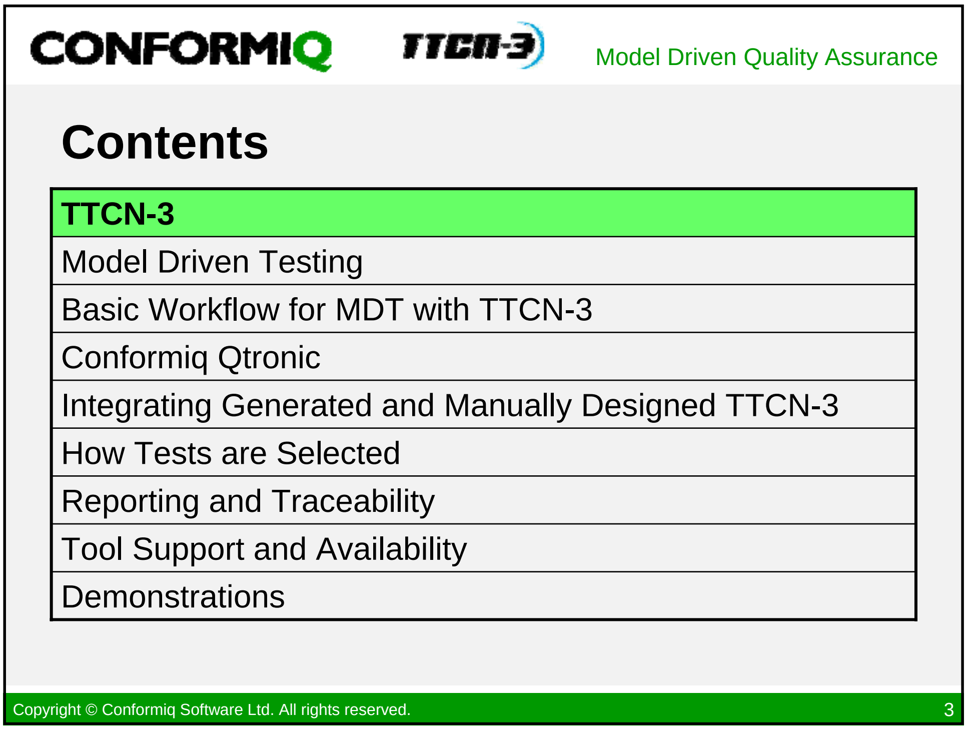
About the Presenter

Antti Huima (M.Sc. Tech.)

- Managing director of Conformiq Software Ltd. (formerly R&D director)

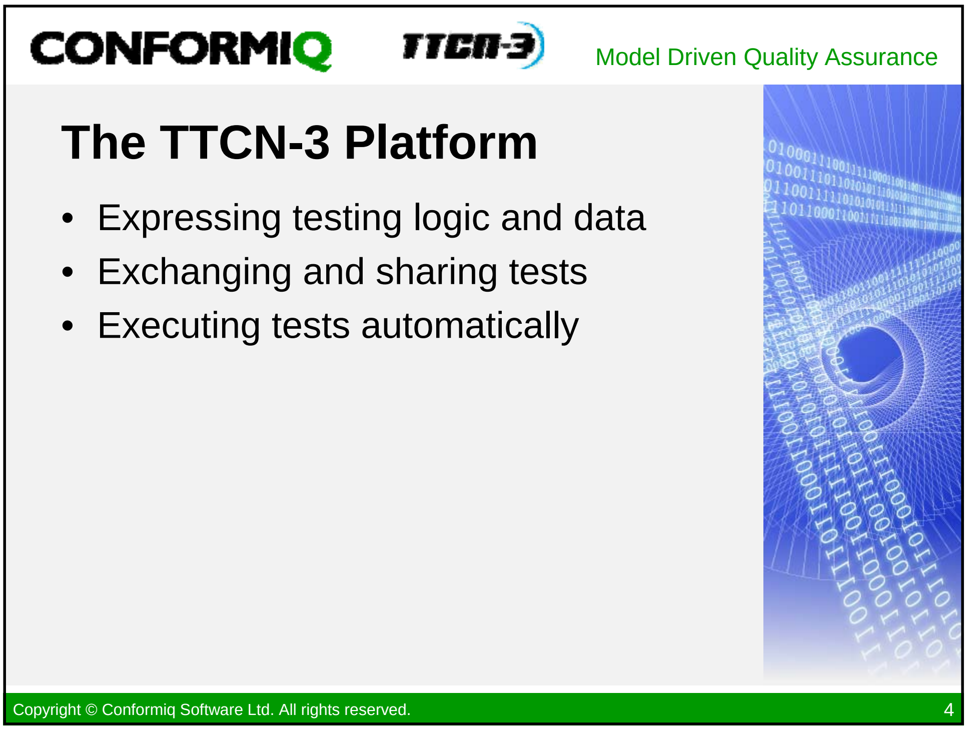
Conformiq Software Ltd.

- Est. 1998
- Model driven testing and quality assurance



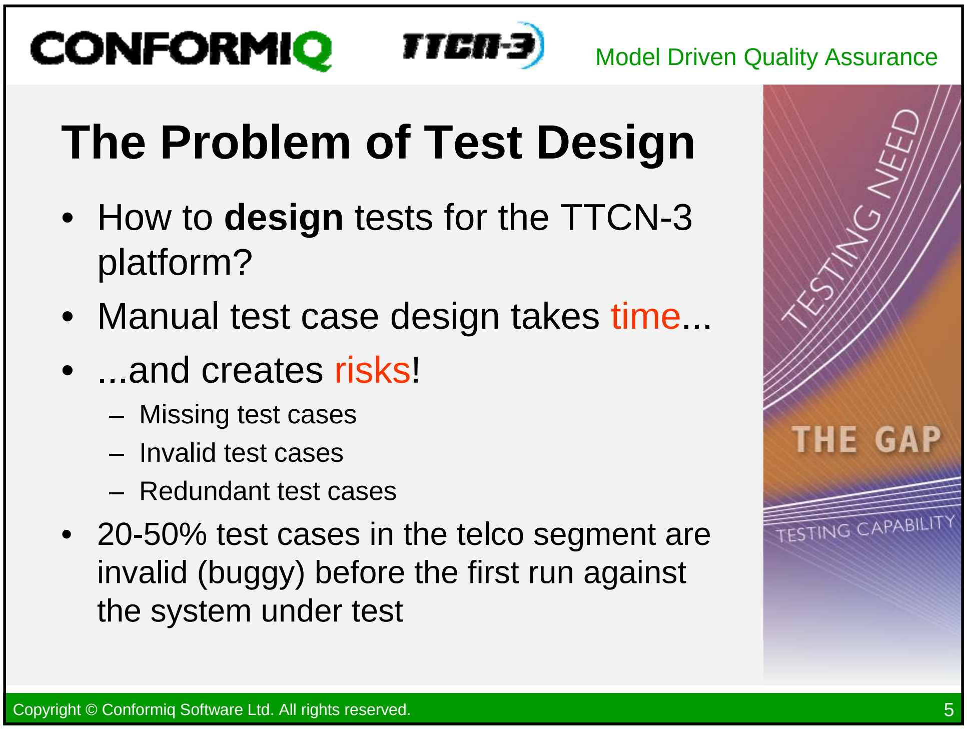
Contents

- **TTCN-3**
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



The TTCN-3 Platform

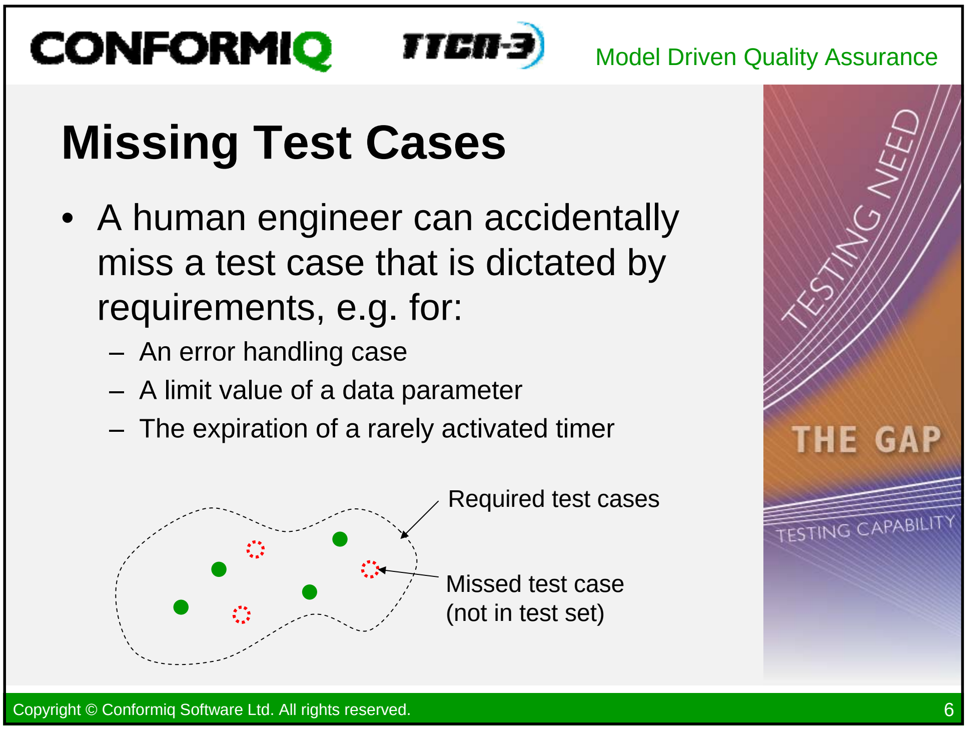
- Expressing testing logic and data
- Exchanging and sharing tests
- Executing tests automatically



The Problem of Test Design

- How to **design** tests for the TTCN-3 platform?
- Manual test case design takes time...
- ...and creates risks!
  - Missing test cases
  - Invalid test cases
  - Redundant test cases
- 20-50% test cases in the telco segment are invalid (buggy) before the first run against the system under test

TESTING NEED
THE GAP
TESTING CAPABILITY



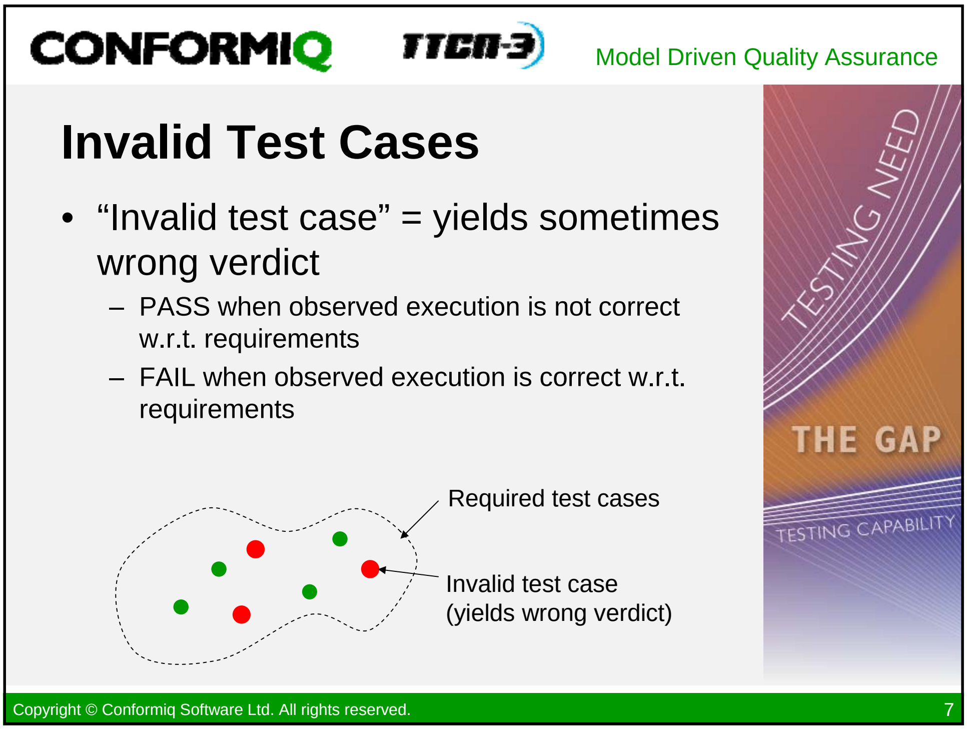
Missing Test Cases

- A human engineer can accidentally miss a test case that is dictated by requirements, e.g. for:
  - An error handling case
  - A limit value of a data parameter
  - The expiration of a rarely activated timer

Required test cases

Missed test case (not in test set)

TESTING NEED
THE GAP
TESTING CAPABILITY



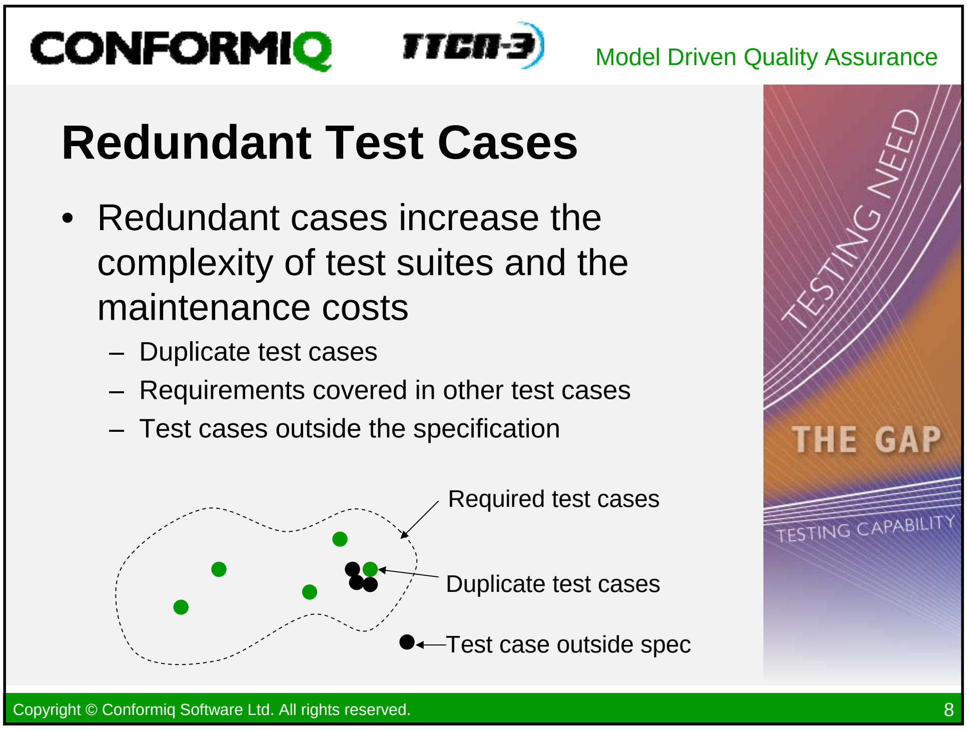
Invalid Test Cases

- "Invalid test case" = yields sometimes wrong verdict
  - PASS when observed execution is not correct w.r.t. requirements
  - FAIL when observed execution is correct w.r.t. requirements

Required test cases

Invalid test case (yields wrong verdict)

TESTING NEED
THE GAP
TESTING CAPABILITY



Redundant Test Cases

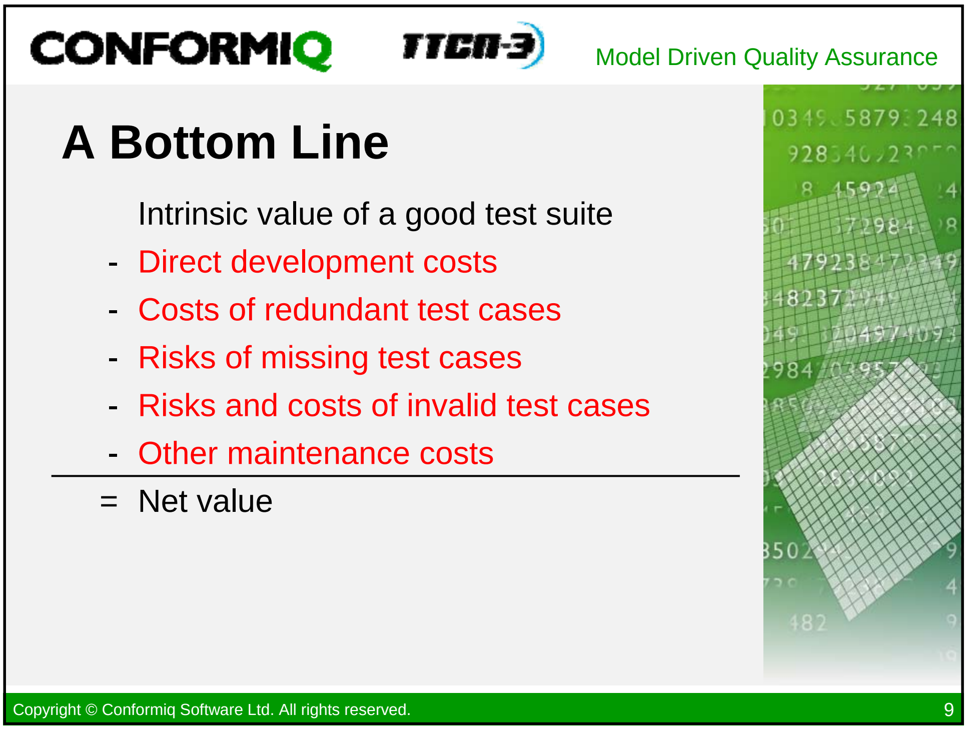
- Redundant cases increase the complexity of test suites and the maintenance costs
  - Duplicate test cases
  - Requirements covered in other test cases
  - Test cases outside the specification

Required test cases

Duplicate test cases

Test case outside spec

TESTING NEED
THE GAP
TESTING CAPABILITY

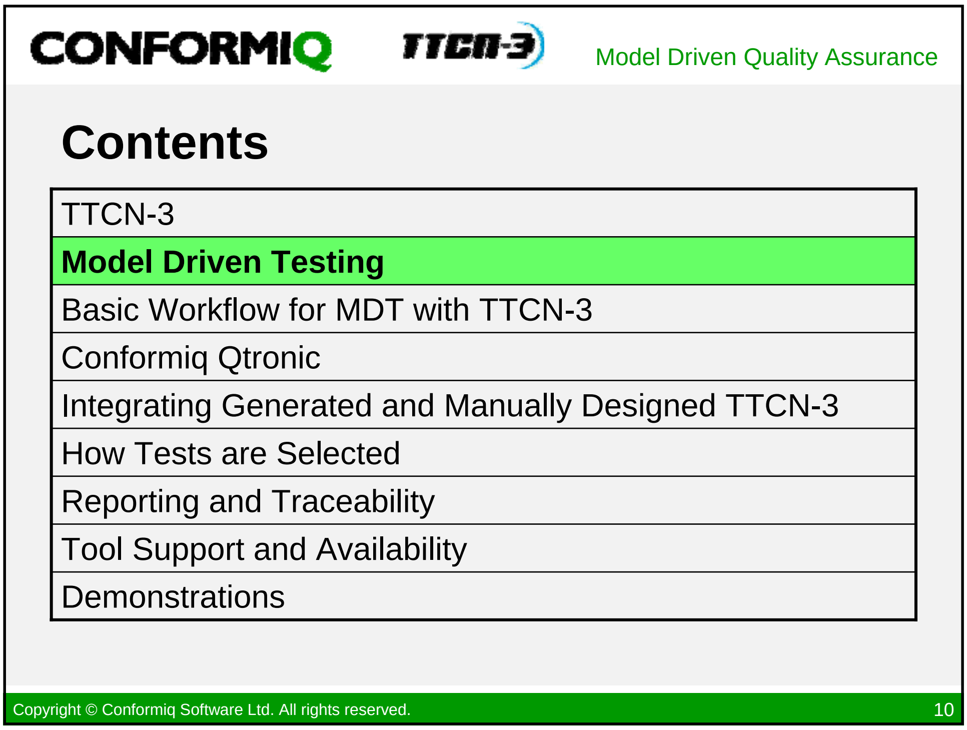


A Bottom Line

Intrinsic value of a good test suite

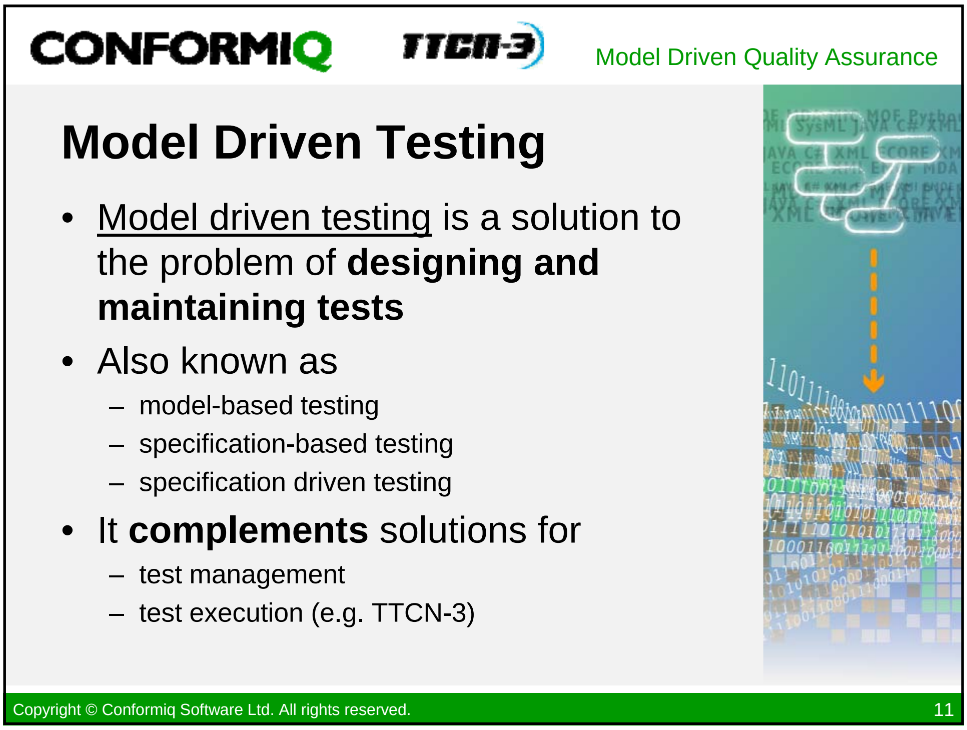
- Direct development costs
- Costs of redundant test cases
- Risks of missing test cases
- Risks and costs of invalid test cases
- Other maintenance costs

= Net value



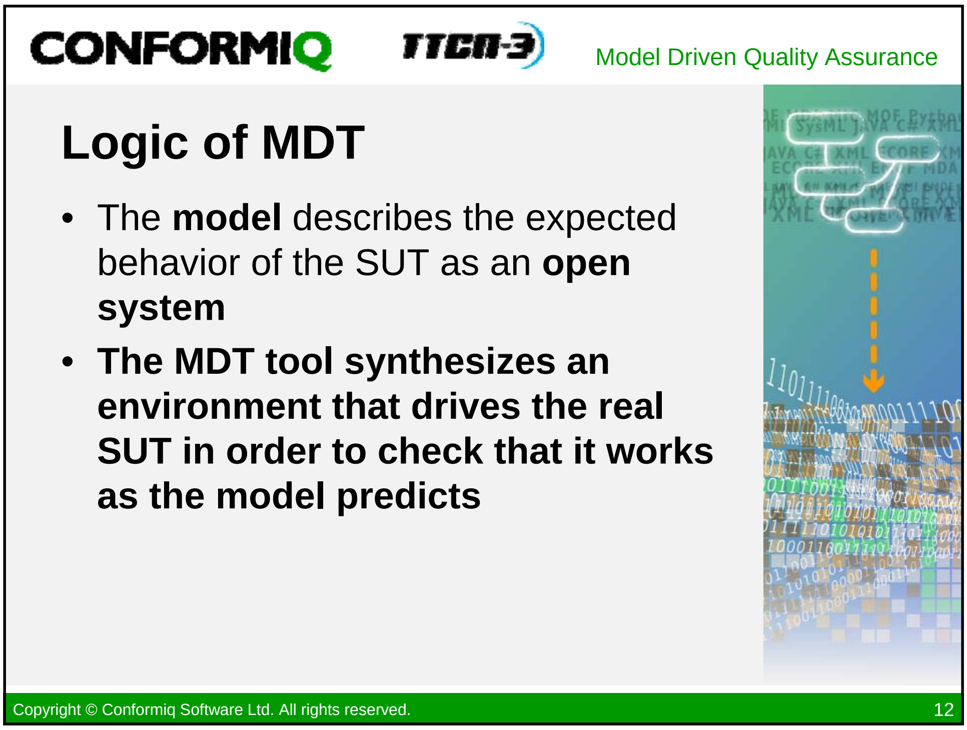
Contents

- TTCN-3
- **Model Driven Testing**
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



Model Driven Testing

- Model driven testing is a solution to the problem of **designing and maintaining tests**
- Also known as
  - model-based testing
  - specification-based testing
  - specification driven testing
- It **complements** solutions for
  - test management
  - test execution (e.g. TTCN-3)



Logic of MDT

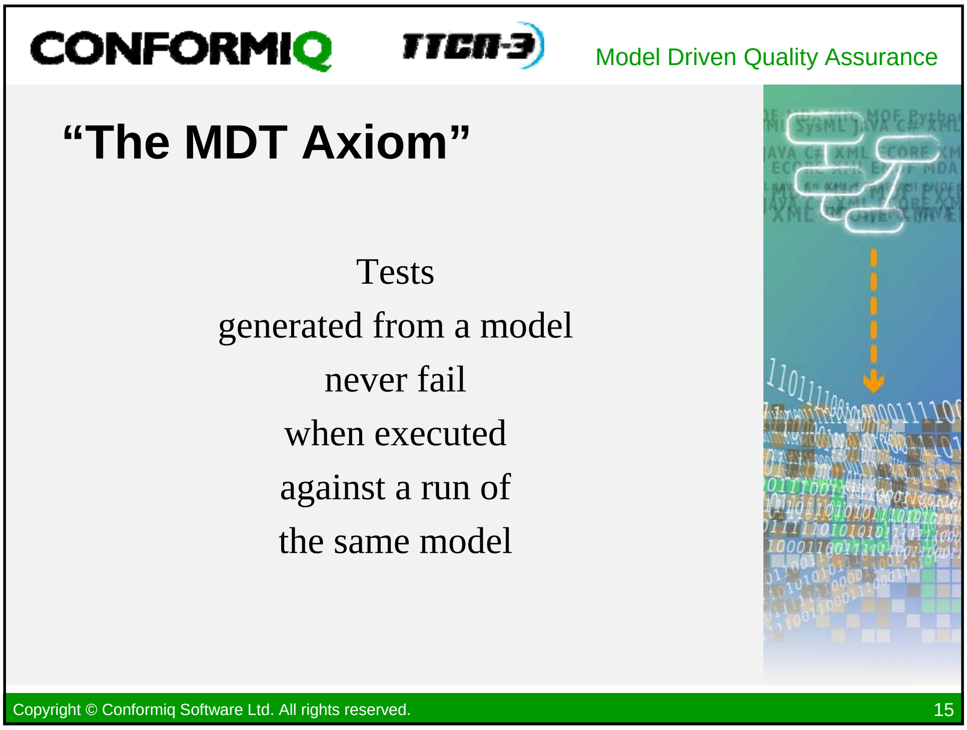
- The **model** describes the expected behavior of the SUT as an **open system**
- **The MDT tool synthesizes an environment that drives the real SUT in order to check that it works as the model predicts**

Heuristics

- Model driven testing heuristics are used to make sure that all important functionality of the model is exercised, for example:
  - statechart state and transition coverage
  - date definition / use coverage
  - boundary value analysis
  - condition and atomic condition coverage
  - requirements coverage

Reference Implementation

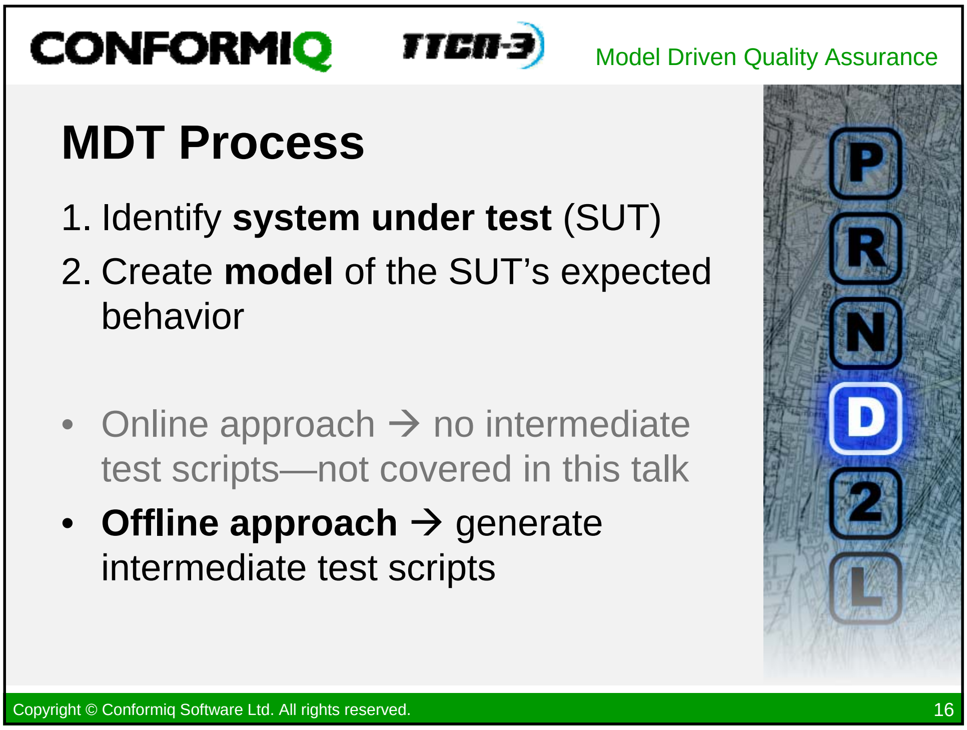
- The model is basically an **abstract reference implementation** of the SUT, because it
  - is executable
  - describes the behavior of the SUT (albeit generally on a higher level of abstraction)
- As a matter of fact, the SUT model can be often **simulated**



Model Driven Quality Assurance

“The MDT Axiom”

Tests
generated from a model
never fail
when executed
against a run of
the same model



Model Driven Quality Assurance

MDT Process

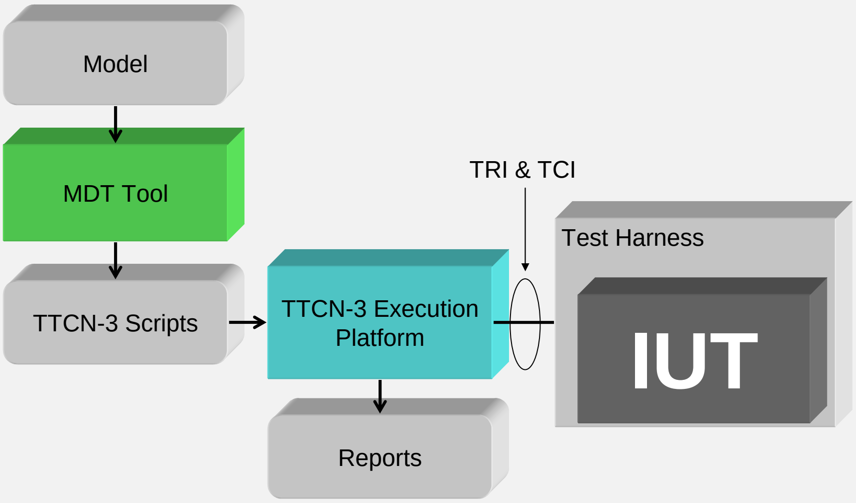
1. Identify **system under test** (SUT)
2. Create **model** of the SUT’s expected behavior

- Online approach → no intermediate test scripts—not covered in this talk
- **Offline approach →** generate intermediate test scripts

Offline TTCN-3 Approach

3. Command the **MDT tool** to generate test cases in TTCN-3
4. Store the generated test cases in version control / configuration management system (optional step)
5. Execute the generated tests on a separate **TTCN-3 platform** (e.g. company internal tools, or from Telelogic, OpenTTCN, Danet, Elvior, TestingTech...)

Offline TTCN-3 Approach

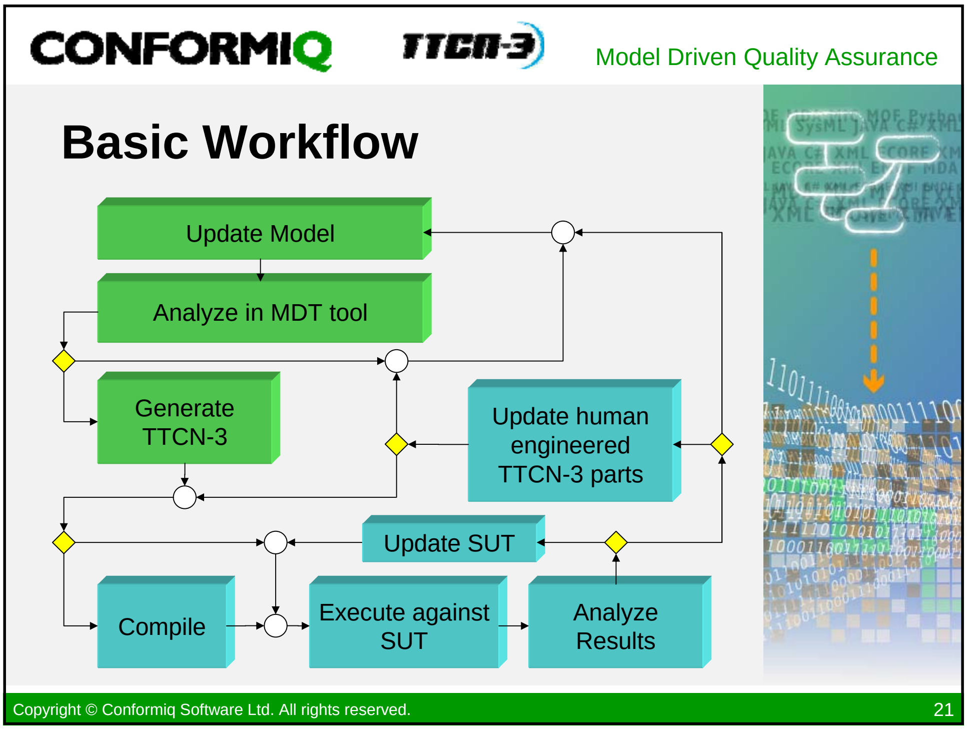


The State-of-the-Art Solution

- Derive test cases automatically from **functional models**...
- generating also test **data**, **time**, and **expected results** (**test oracles**)...
- using well-established heuristics like model-level **branch coverage** or **boundary value analysis**...
- and algorithms including **symbolic state space analysis**, **constraint solving** and **combinatorial optimization**

Contents

- TTCN-3
- Model Driven Testing
- **Basic Workflow for MDT with TTCN-3**
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



Basic Workflow

Update Model

Analyze in MDT tool

Generate TTCN-3

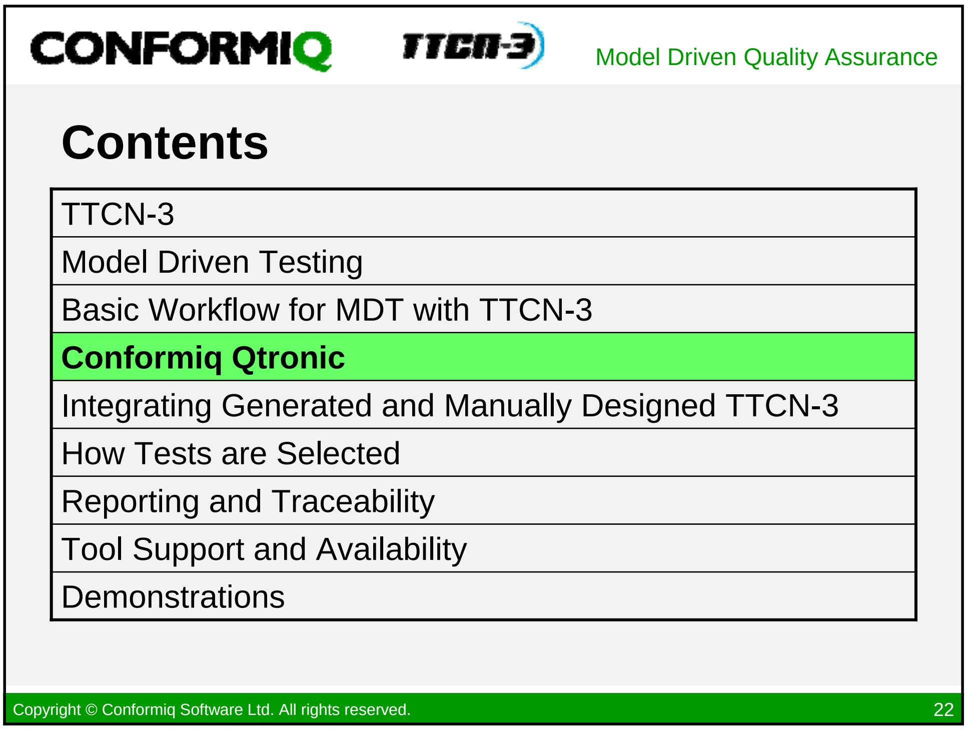
Update human engineered TTCN-3 parts

Update SUT

Compile

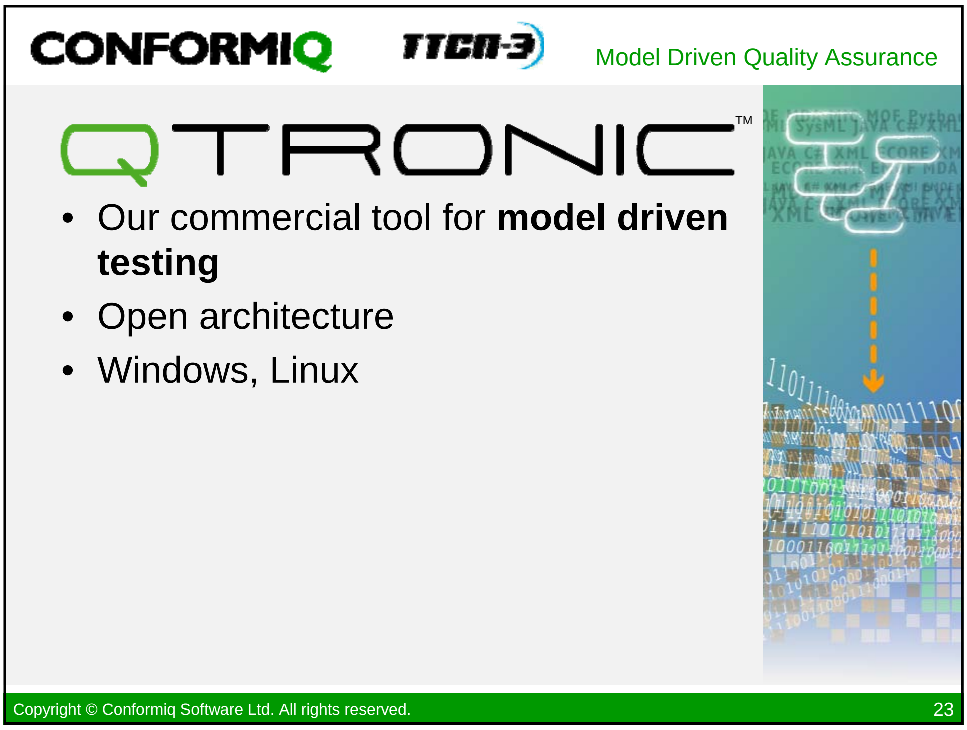
Execute against SUT

Analyze Results



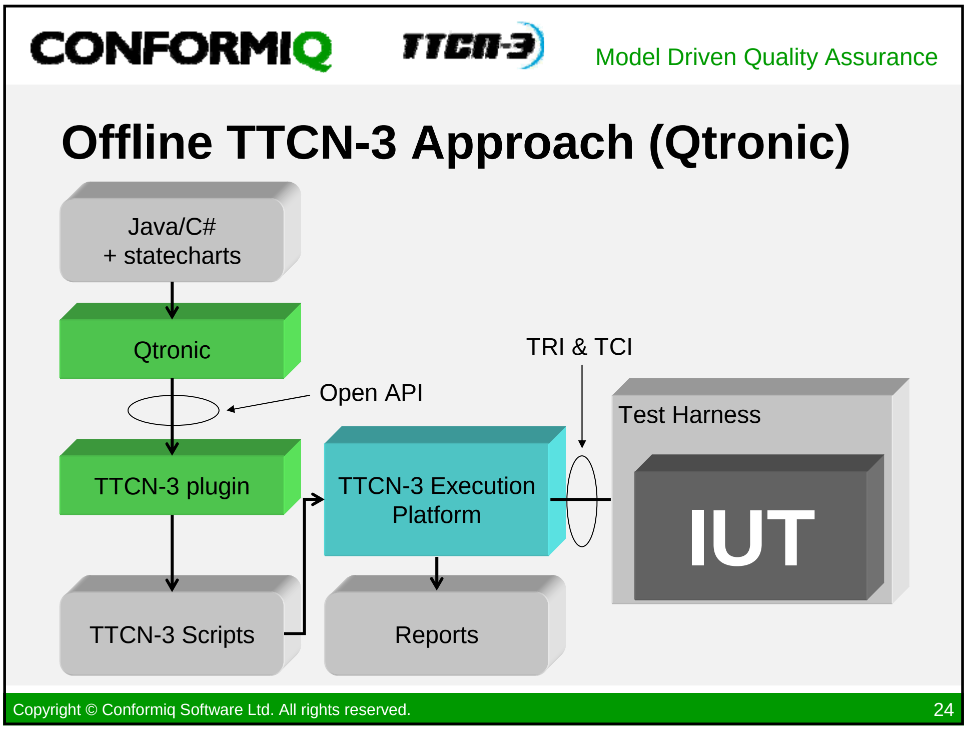
Contents

- TTCN-3
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- **Conformiq Qtronic**
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



QTRONIC™

- Our commercial tool for **model driven testing**
- Open architecture
- Windows, Linux



Offline TTCN-3 Approach (Qtronic)

Models

- Extended Java
- *Optional* UML state charts

- Created in
  - Any text editor (Java parts)
  - Qtronic Modeler (UML state charts)
  - Third party UML tools (e.g. IBM Rational, Mentor Graphics tools, Enterprise Architect)

Supported Constructs

- Full **data** (strings, numbers, records, classes, arrays…)
- Full **time** (timeouts, dynamic timeouts…)
- Full **control structures** (methods, dynamic polymorphism…)
- Full **concurrency** (multiple Java threads in model, ITC primitives…)
- Full **Java** + templates, macros, record values, type inference…

Simple Model Anatomy

- **system** block declares **ports** of the SUT
  - These map to the ports of the MTC
- **record** declarations describe **message types**
  - These map to **type record**s in TTCN-3
- **class** declarations describe **active objects**
  - Behavior described in state charts and/or Java/C# syntax
  - These are not mapped to TTCN-3 because they are used to explain behavior
- **void main()** serves as the entry point

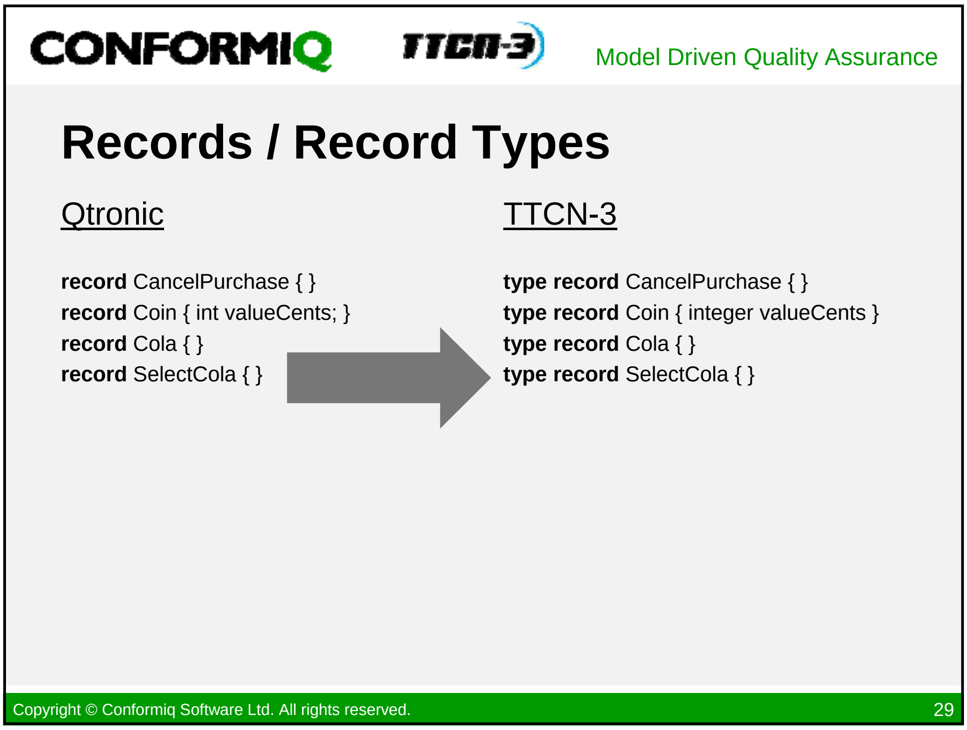
System Block / MTC Component

Qtronic

```
system {
   Inbound moneyIn : Coin;
   Inbound keyPress : SelectCola,
     CancelPurchase;
   Outbound moneyOut : Coin;
   Outbound product : Cola;
}
```

TTCN-3

```
type port keyPressPort message {
      out SelectCola;
      out CancelPurchase;        }
type port moneyInPort message {
      out Coin;          }
type port moneyOutPort message {
      in Coin;            }
type port productPort message {
      in Cola;            }
type component SystemType {
        port keyPressPort keyPress;
        port moneyInPort moneyIn;
        port moneyOutPort moneyOut;
        port productPort product; ... }
```



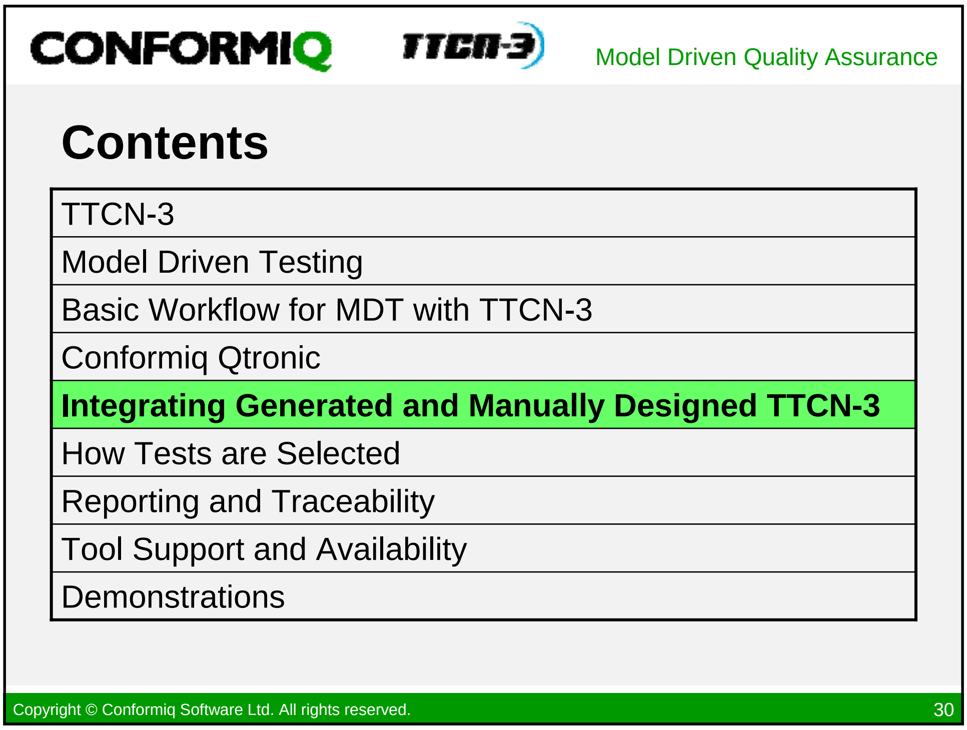
Records / Record Types

Qtronic

record CancelPurchase { }
record Coin { int valueCents; }
record Cola { }
record SelectCola { }

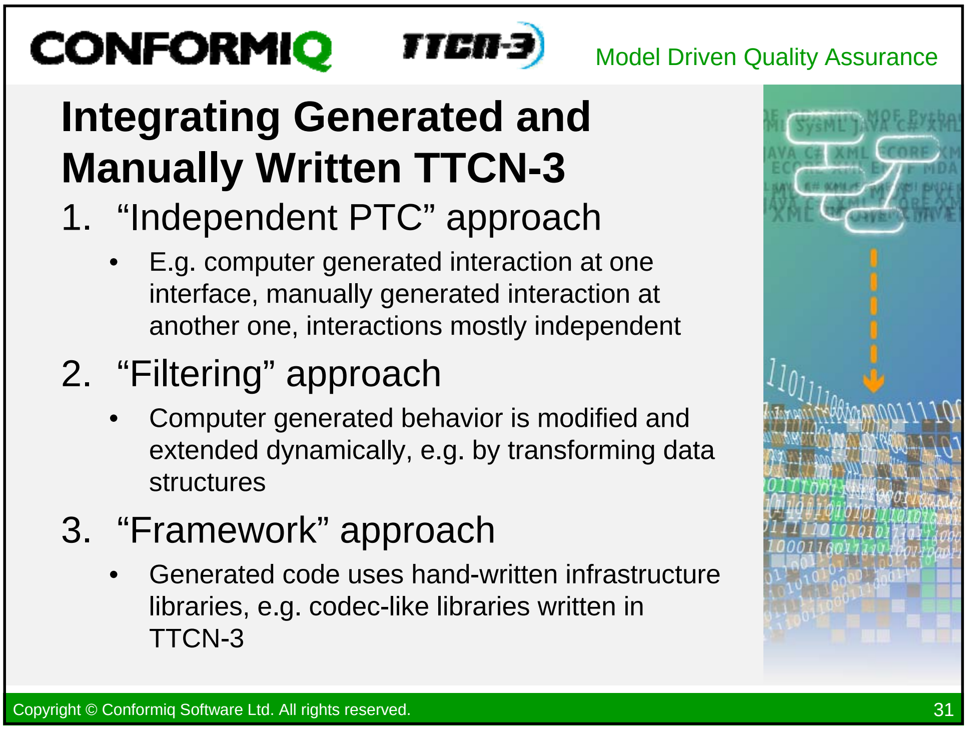
TTCN-3

type record CancelPurchase { }
type record Coin { integer valueCents }
type record Cola { }
type record SelectCola { }



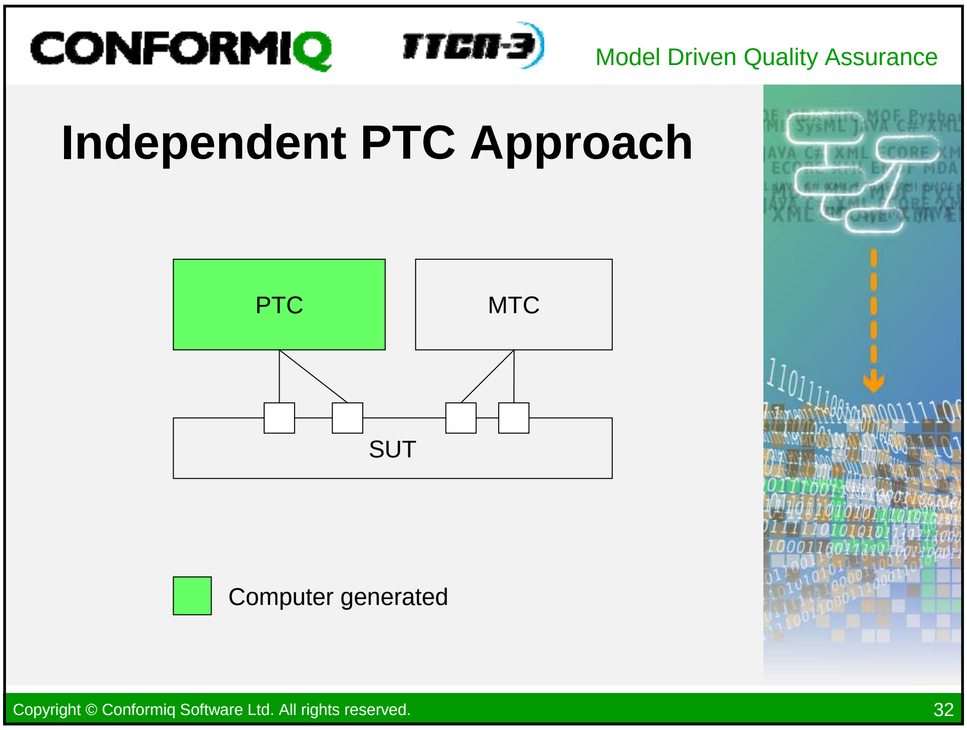
Contents

- TTCN-3
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- **Integrating Generated and Manually Designed TTCN-3**
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



Integrating Generated and Manually Written TTCN-3

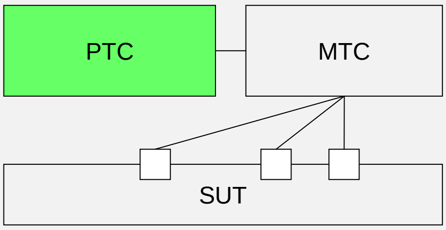
1. “Independent PTC” approach
   - E.g. computer generated interaction at one interface, manually generated interaction at another one, interactions mostly independent
2. “Filtering” approach
   - Computer generated behavior is modified and extended dynamically, e.g. by transforming data structures
3. “Framework” approach
   - Generated code uses hand-written infrastructure libraries, e.g. codec-like libraries written in TTCN-3



Independent PTC Approach

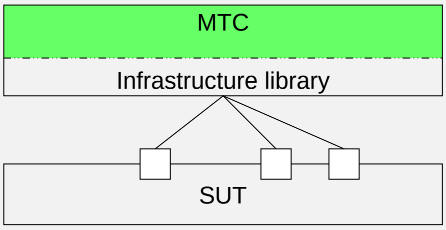
🟩 Computer generated

Filtering Approach

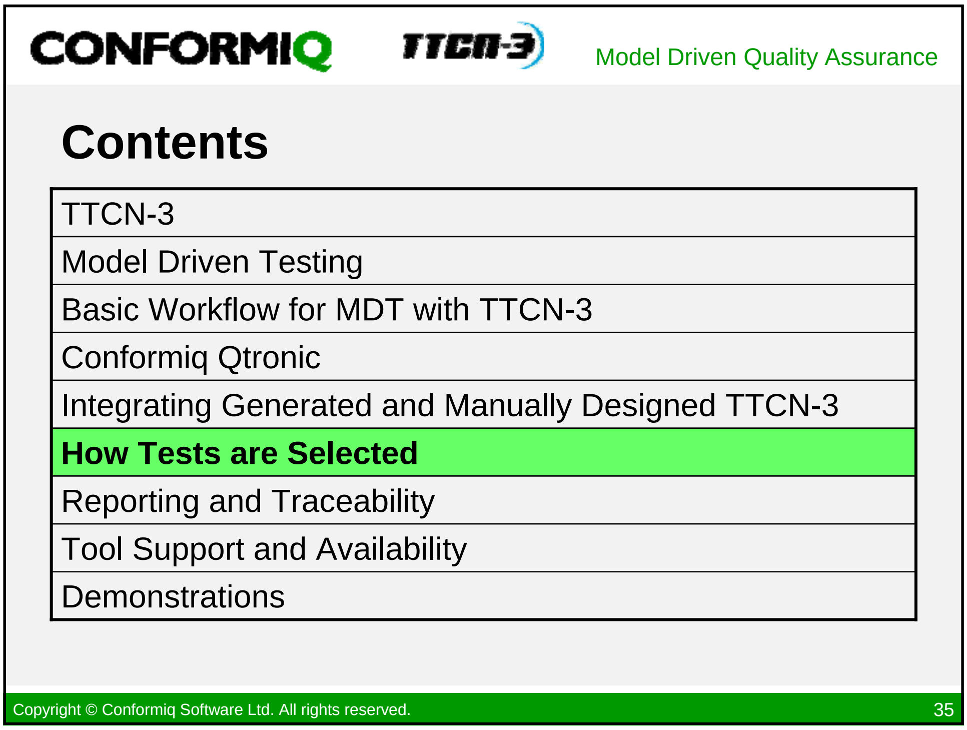


🟩 Computer generated

Framework Approach

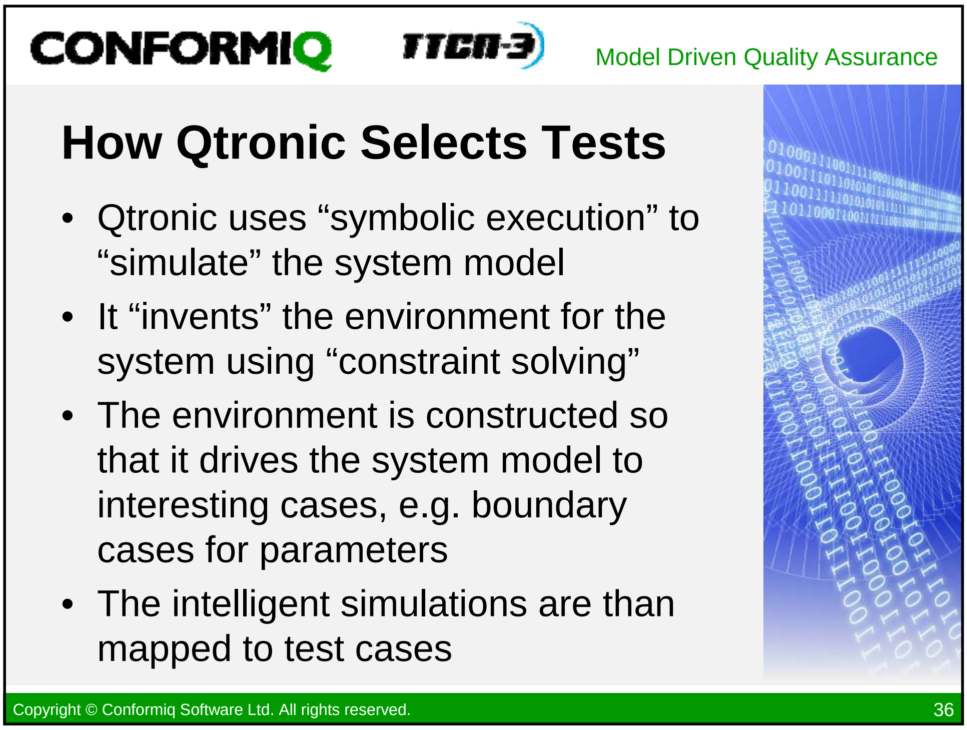


🟩 Computer generated



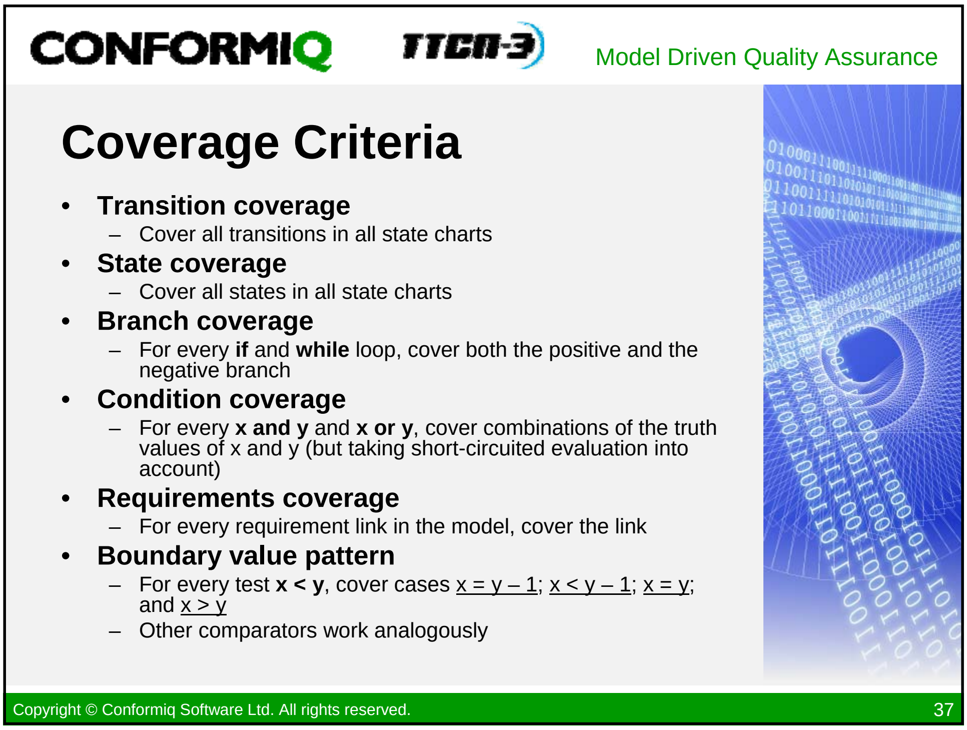
Contents

- TTCN-3
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- **How Tests are Selected**
- Reporting and Traceability
- Tool Support and Availability
- Demonstrations



How Qtronic Selects Tests

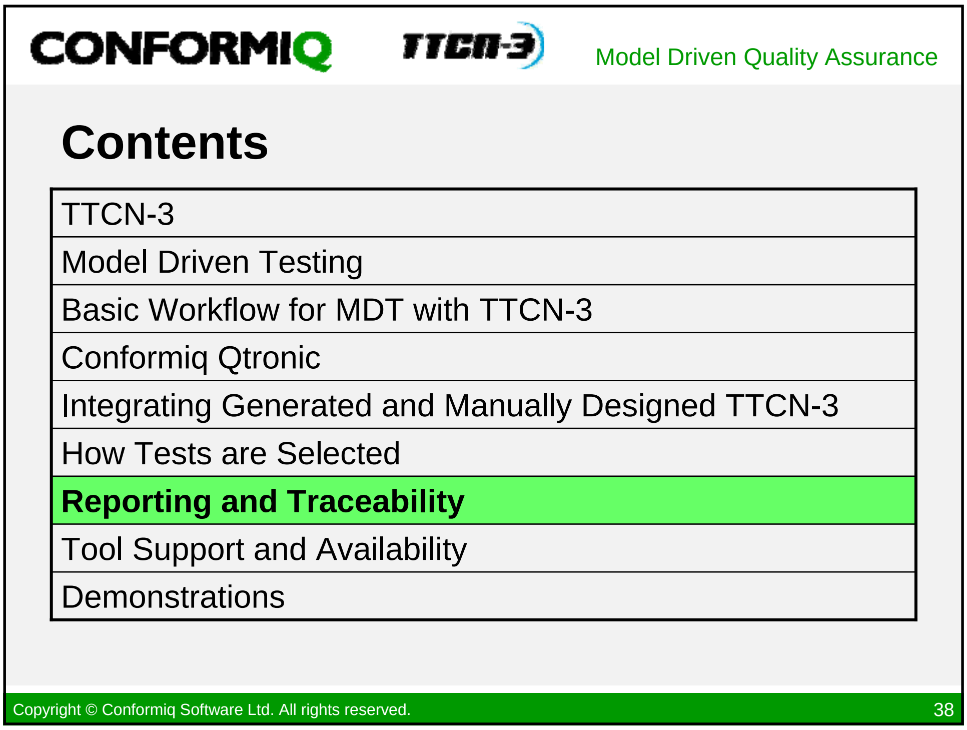
- Qtronic uses “symbolic execution” to “simulate” the system model
- It “invents” the environment for the system using “constraint solving”
- The environment is constructed so that it drives the system model to interesting cases, e.g. boundary cases for parameters
- The intelligent simulations are than mapped to test cases



Model Driven Quality Assurance

Coverage Criteria

- **Transition coverage**
  - Cover all transitions in all state charts
- **State coverage**
  - Cover all states in all state charts
- **Branch coverage**
  - For every **if** and **while** loop, cover both the positive and the negative branch
- **Condition coverage**
  - For every **x and y** and **x or y**, cover combinations of the truth values of x and y (but taking short-circuited evaluation into account)
- **Requirements coverage**
  - For every requirement link in the model, cover the link
- **Boundary value pattern**
  - For every test **x < y**, cover cases x = y – 1; x < y – 1; x = y; and x > y
  - Other comparators work analogously



Model Driven Quality Assurance

Contents

TTCN-3

Model Driven Testing

Basic Workflow for MDT with TTCN-3

Conformiq Qtronic

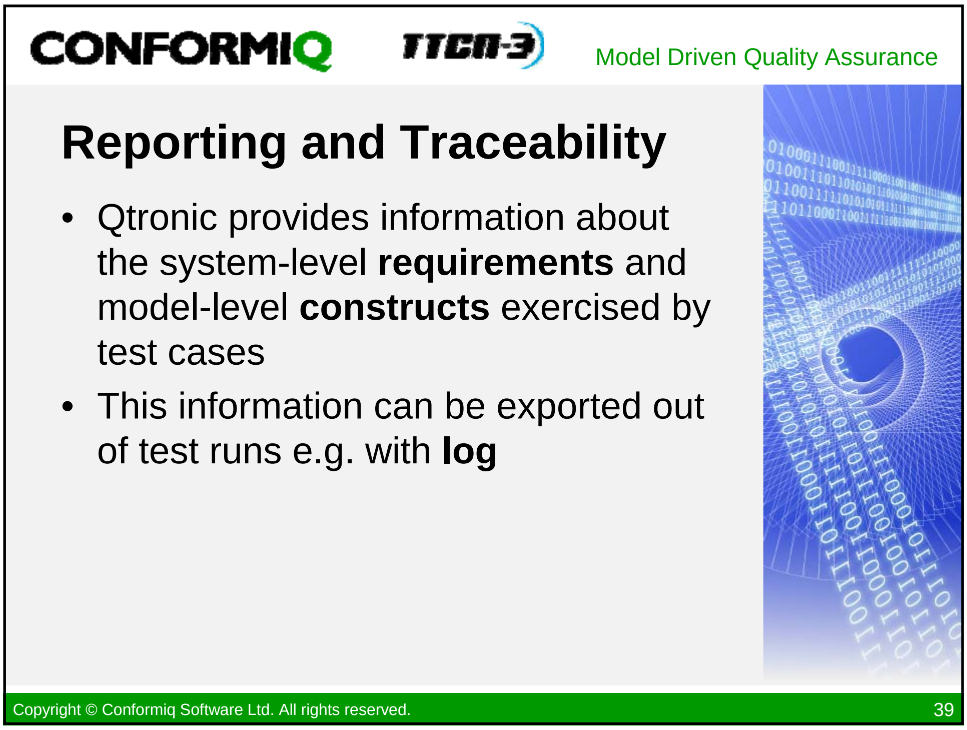
Integrating Generated and Manually Designed TTCN-3

How Tests are Selected

Reporting and Traceability

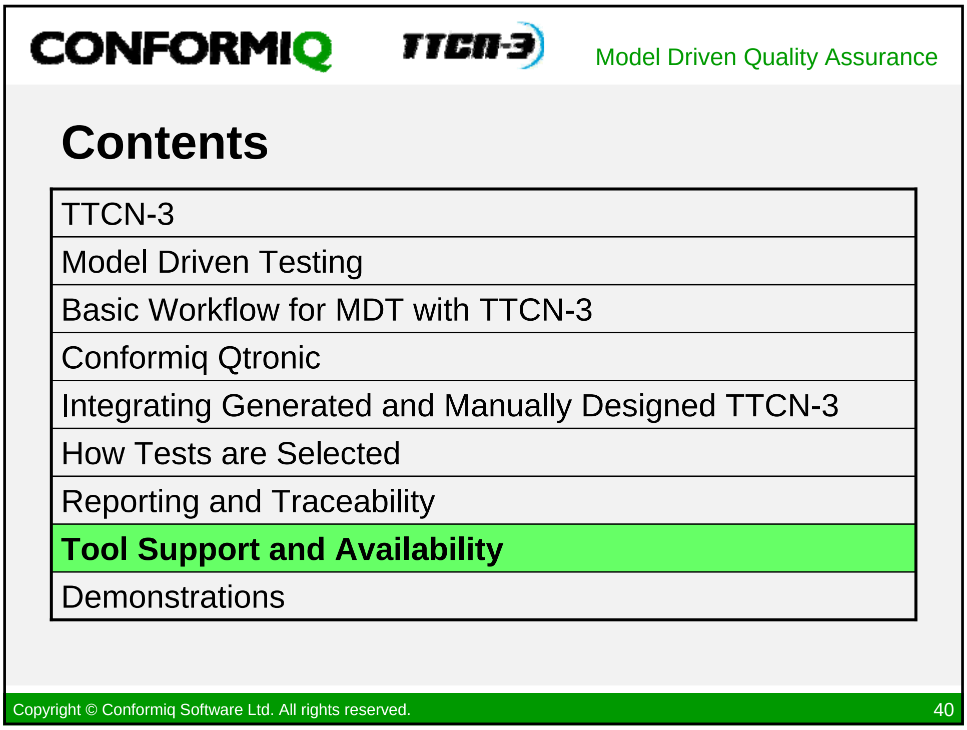
Tool Support and Availability

Demonstrations



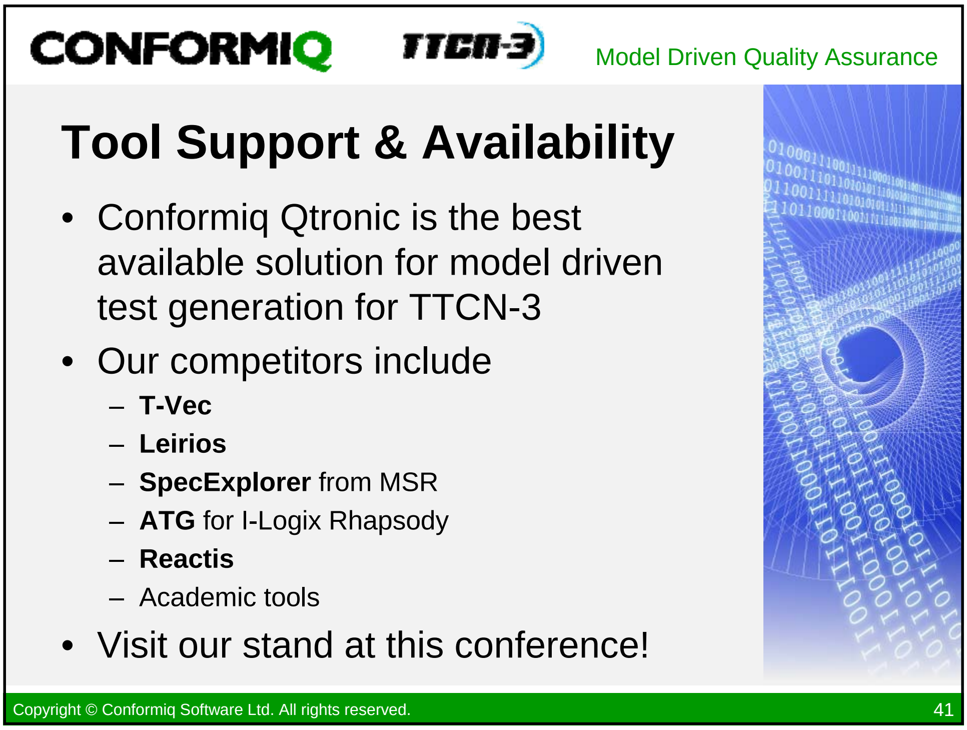
Reporting and Traceability

- Qtronic provides information about the system-level **requirements** and model-level **constructs** exercised by test cases
- This information can be exported out of test runs e.g. with **log**



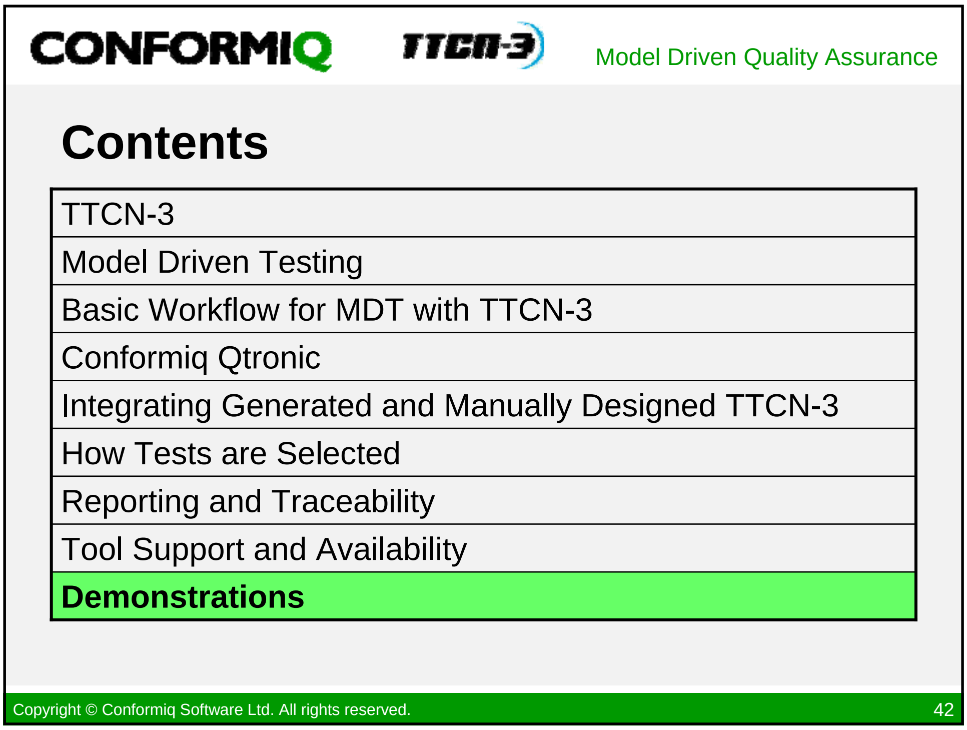
Contents

- TTCN-3
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- **Tool Support and Availability**
- Demonstrations



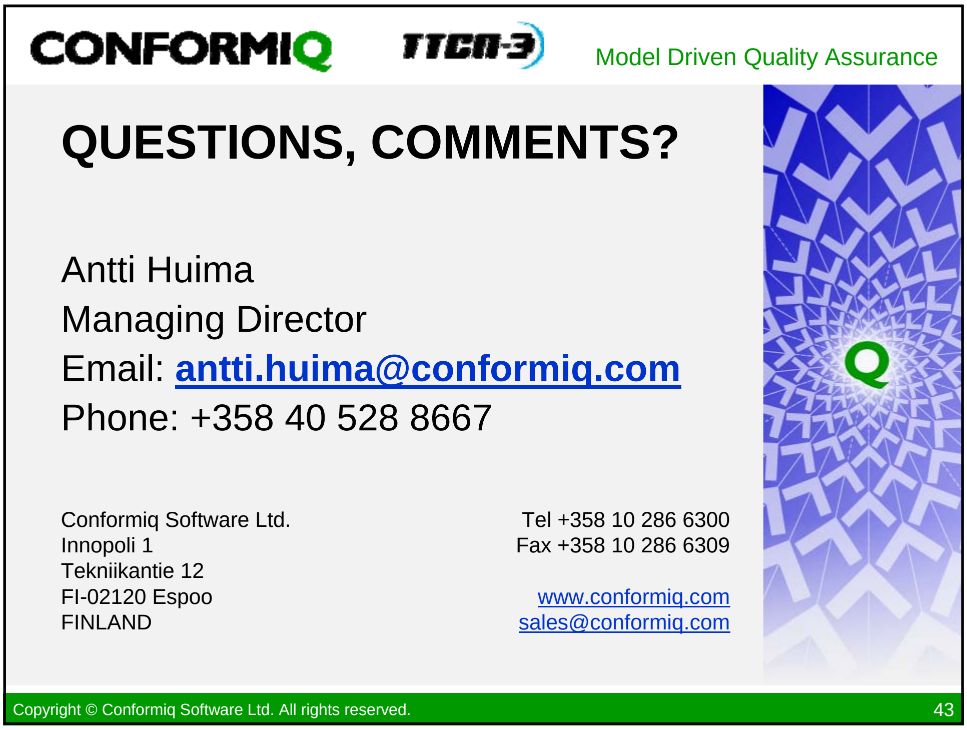
Tool Support & Availability

- Conformiq Qtronic is the best available solution for model driven test generation for TTCN-3
- Our competitors include
  - **T-Vec**
  - **Leirios**
  - **SpecExplorer** from MSR
  - **ATG** for I-Logix Rhapsody
  - **Reactis**
  - Academic tools
- Visit our stand at this conference!



Contents

- TTCN-3
- Model Driven Testing
- Basic Workflow for MDT with TTCN-3
- Conformiq Qtronic
- Integrating Generated and Manually Designed TTCN-3
- How Tests are Selected
- Reporting and Traceability
- Tool Support and Availability
- **Demonstrations**



CONFORMIQ
TTCN-3

Model Driven Quality Assurance

QUESTIONS, COMMENTS?

Antti Huima
Managing Director
Email: antti.huima@conformiq.com
Phone: +358 40 528 8667

Conformiq Software Ltd.
Innopoli 1
Tekniikantie 12
FI-02120 Espoo
FINLAND

Tel +358 10 286 6300
Fax +358 10 286 6309

www.conformiq.com
sales@conformiq.com