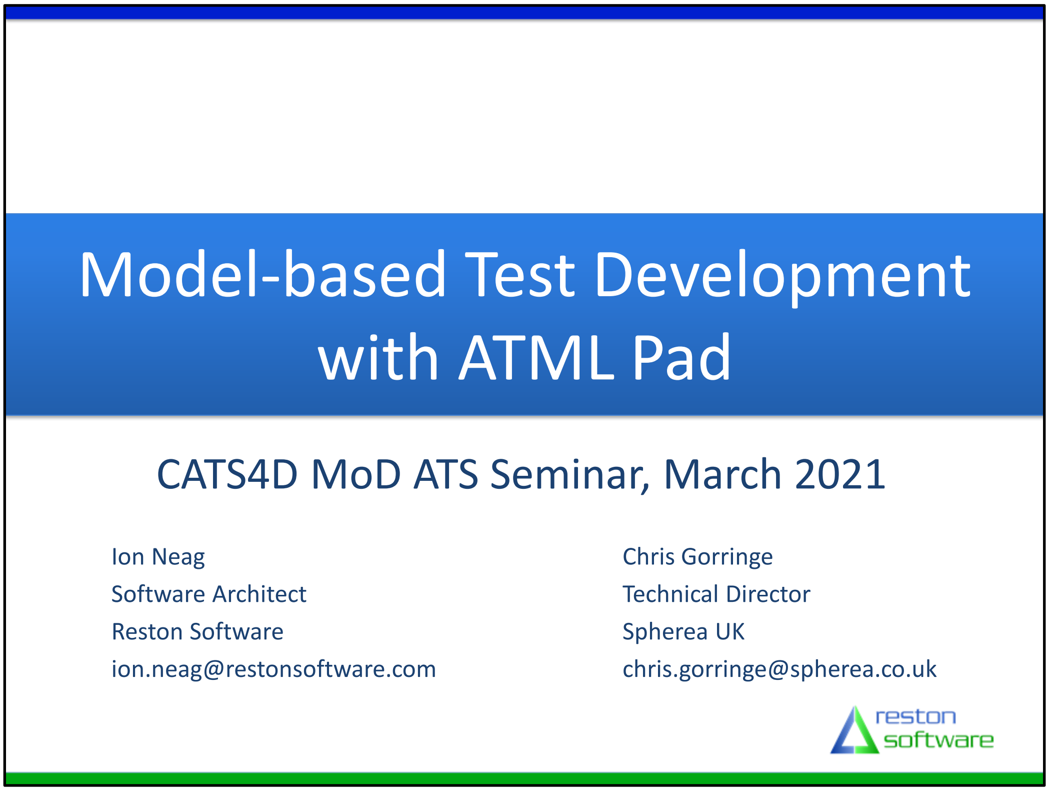


Model-based Test Development with ATML Pad

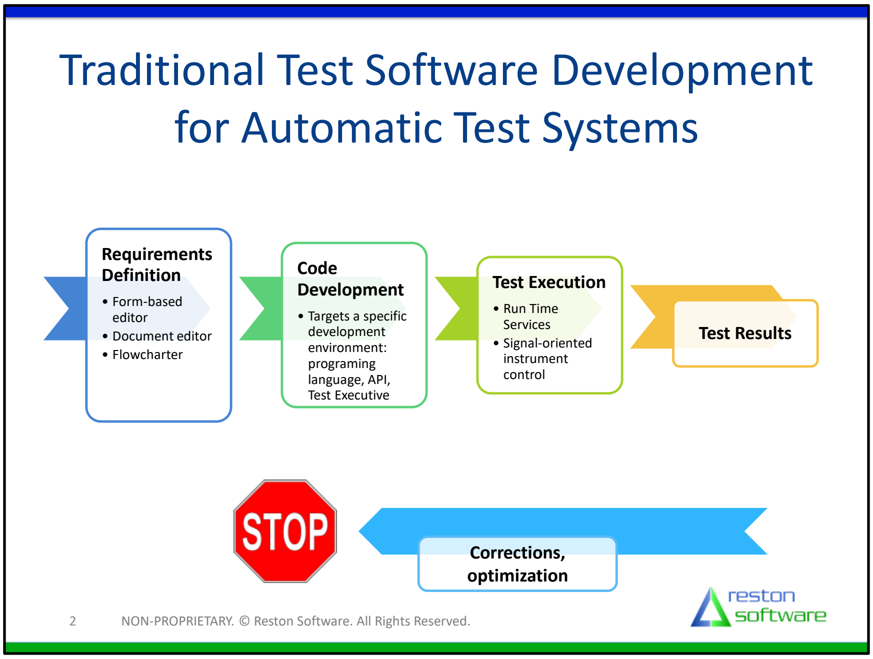
CATS4D MoD ATS Seminar, March 2021

Ion Neag
Software Architect
Reston Software
ion.neag@restonsoftware.com

Chris Gorringe
Technical Director
Spherea UK
chris.gorringe@spherea.co.uk

ATML Pad is a development environment for test descriptions, using the ATML Test Description standard format

This presentation discusses the use of standards and software tools in model-based development of test programs for Automatic Test Equipment (ATE)



Traditional Test Software Development for Automatic Test Systems

Requirements Definition
- Form-based editor
- Document editor
- Flowcharter

Code Development
- Targets a specific development environment: programing language, API, Test Executive

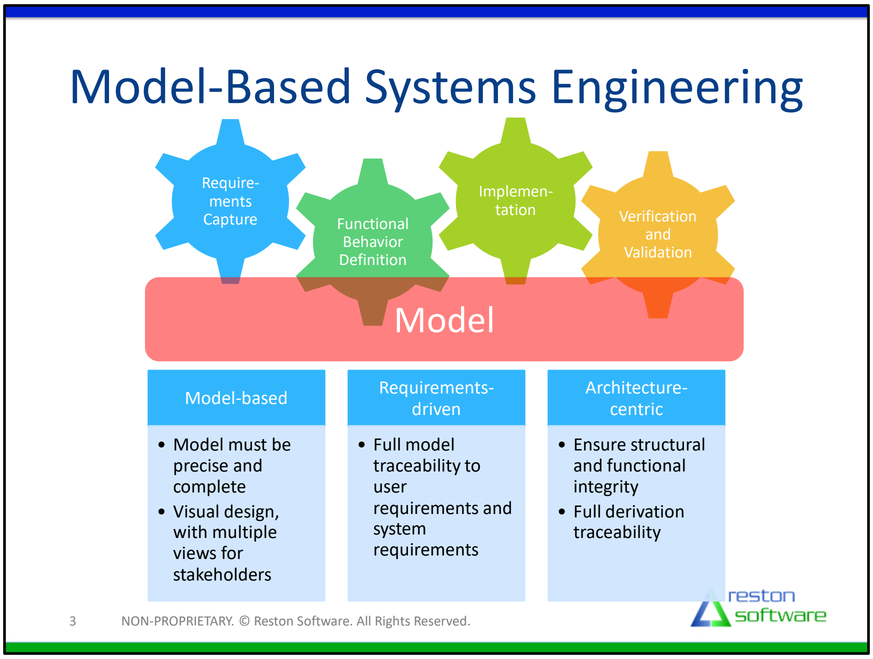
Test Execution
- Run Time Services
- Signal-oriented instrument control

Test Results

Corrections, optimization

Problem: corrections and optimizations are made in code and not in requirements.

- In time, the code diverges from requirements
- Original requirements become obsolete, can no longer be reused to support code changes during code maintenance, rehosting, or conversion



Model-Based Systems Engineering

Model-based

- Model must be precise and complete
- Visual design, with multiple views for stakeholders

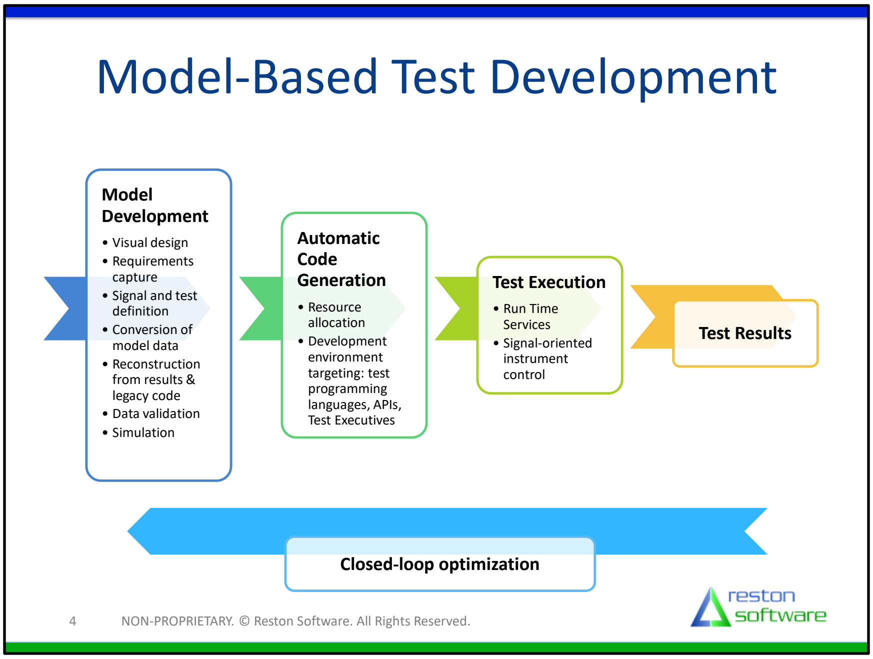
Requirements-driven

- Full model traceability to user requirements and system requirements

Architecture-centric

- Ensure structural and functional integrity
- Full derivation traceability

The discipline of Model-Based Systems Engineering advocates the use of a common digital model to support all phases of the System Engineering process



Model-Based Test Development

Model Development
- Visual design
- Requirements capture
- Signal and test definition
- Conversion of model data
- Reconstruction from results & legacy code
- Data validation
- Simulation

Automatic Code Generation
- Resource allocation
- Development environment targeting: test programming languages, APIs, Test Executives

Test Execution
- Run Time Services
- Signal-oriented instrument control

Test Results

Closed-loop optimization

Making use of model-based systems engineering principles, the test development process can be modified as shown.

Model development

Models for UUT, Test Requirements, ITA, Test Station, Instruments, …
Graphical / visual design
Model data import from: EDA, systems simulation, diagnostic modeling, legacy systems
Model extraction / reconstruction from: test results, non-standard models (ex. TRD), unstructured legacy data (ex. ATLAS code)
Data validation
Simulation

Automatic code generation

Resource allocation – targeting implementation to a specific ATE
Targeting implementation to a specific test development environment: test programming languages, APIs, test executives

Test execution

Run-time signal services
Signal-oriented instrument control

Generation of test results

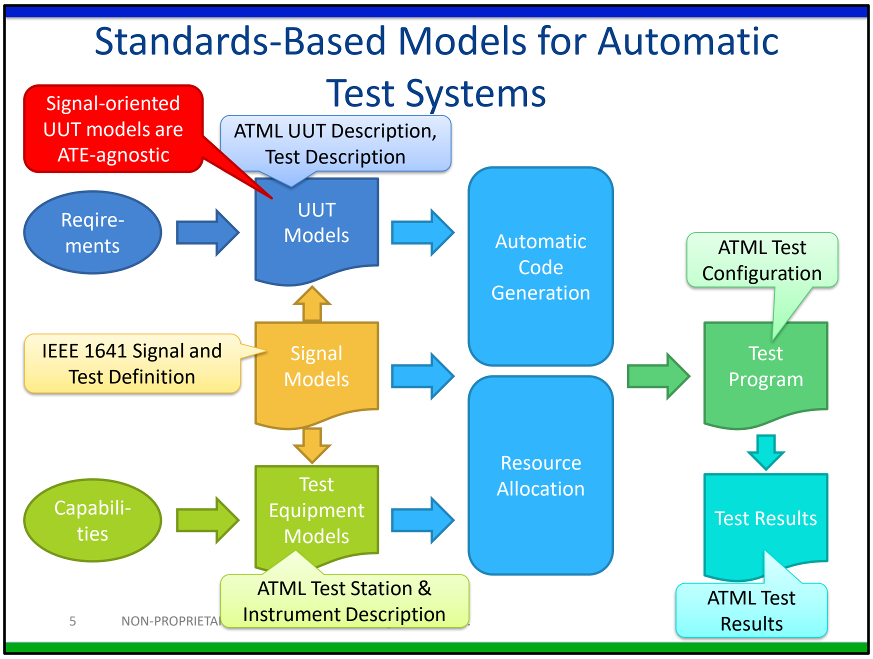
Closed-loop optimization of

Maintenance

Diagnostics

Test

Closed-loop optimization is performed on the Model, not on code. The code is regenerated automatically.



Standards-Based Models for Automatic Test Systems

Signal-oriented UUT models are ATE-agnostic

ATML UUT Description, Test Description

Reqire-ments

UUT Models

Automatic Code Generation

ATML Test Configuration

IEEE 1641 Signal and Test Definition

Signal Models

Test Program

Resource Allocation

Capabili-ties

Test Equipment Models

Test Results

ATML Test Station & Instrument Description

ATML Test Results

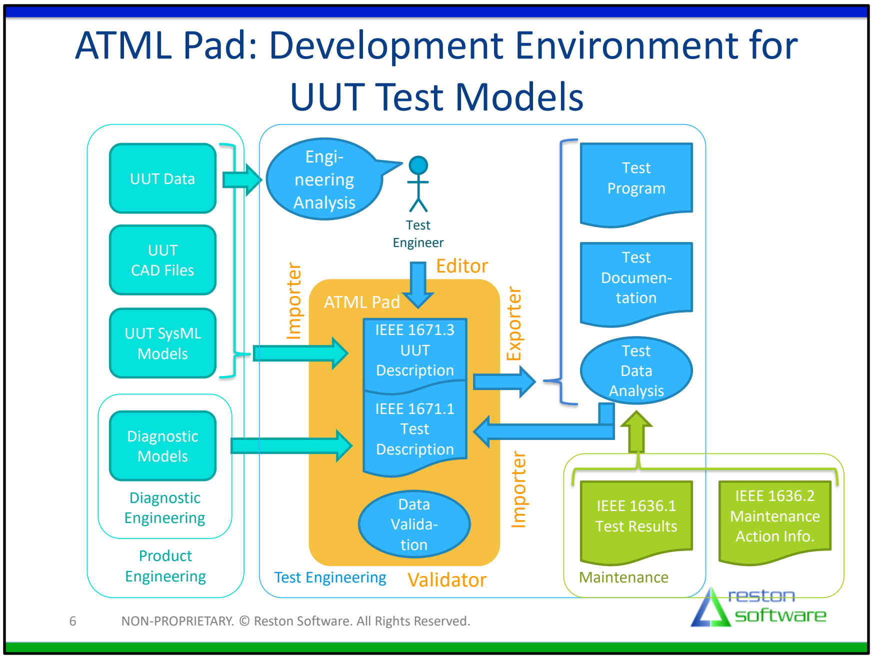
5 NON-PROPRIETA

What standard data formats are available for modeling Automatic Test Systems?

- IEEE 1671 ATML (Automatic Test Markup Language)
- IEEE 1641 Signal and Test Definition

Signal-based models tell you <u>what</u> to do, but not <u>how</u> to do it (in terms of instrument operations). This allows them to be implemented & re-implemented on a variety of ATE platforms.

Resource Allocation is the selection of an instrument or instrument subsystem for each signal operation.



ATML Pad: Development Environment for UUT Test Models

UUT Data

UUT CAD Files

UUT SysML Models

Diagnostic Models

Diagnostic Engineering

Product Engineering

Engi-neering Analysis

Test Engineer

Editor

Importer

ATML Pad

IEEE 1671.3 UUT Description

IEEE 1671.1 Test Description

Data Valida-tion

Exporter

Importer

Test Engineering

Validator

Test Program

Test Documen-tation

Test Data Analysis

IEEE 1636.1 Test Results

IEEE 1636.2 Maintenance Action Info.

Maintenance

reston software

6 NON-PROPRIETARY. © Reston Software. All Rights Reserved.

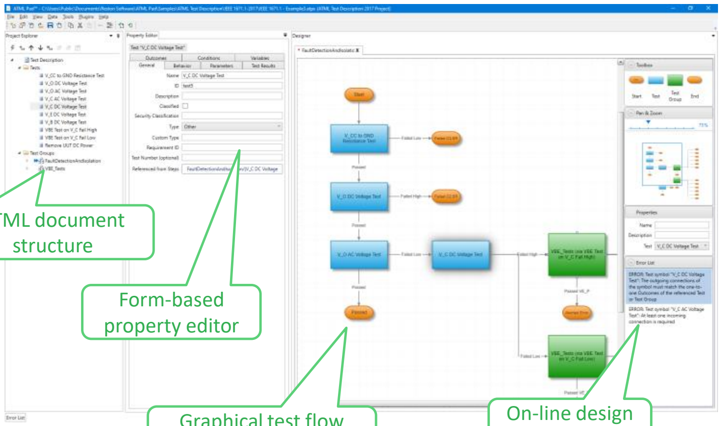
Use cases

- Create UUT and Test Description through test engineering analysis, from UUT data and
- Import UUT & Test data (limited)
- Validate UUT and Test Description
- Generate test program (automatic code generation, resource allocation, switch path calculation, …)
- Generate test program documentation
- Input to test data analysis (ex. for test & diagnostic improvements)

ATML Pad is:

- Editor
- Validator
- Data Converter (Import & Export)

ATML Pad: Development Environment for UUT Test Models



ERROR: Test symbol "V_C DC Voltage Test": The outgoing connections of the symbol must match the one-to-one Outcomes of the referenced Test or Test Group

ERROR: Test symbol "V_C AC Voltage Test": At least one incoming connection is required

ATML document structure

Form-based property editor

Graphical test flow designer

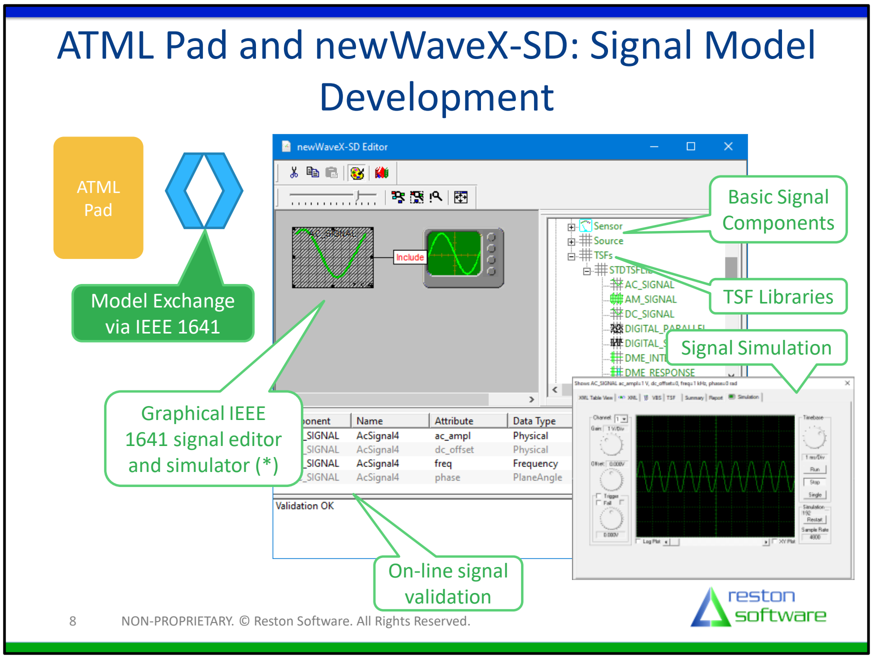
On-line design validation

7 NON-PROPRIETARY. © Reston Software. All Rights Reserved.

reston software

Visual editor features:

- Document structure (tests, test sequences, etc.)
- Item properties
- Graphical design of test flow
  - Drag & drop color-coded symbols from toolbox
  - Pan & zoom controls
- On-line design validation: missing items, missing connections, and other design problems are displayed dynamically in the error list. This feature guides beginner users through the design process, helping them understand and apply the design rules.



ATML Pad and newWaveX-SD: Signal Model Development

Model Exchange via IEEE 1641

Graphical IEEE 1641 signal editor and simulator (*)

Basic Signal Components

TSF Libraries

Signal Simulation

On-line signal validation

ponent	Name	Attribute	Data Type
_SIGNAL	AcSignal4	ac_ampl	Physical
_SIGNAL	AcSignal4	dc_offset	Physical
_SIGNAL	AcSignal4	freq	Frequency
_SIGNAL	AcSignal4	phase	PlaneAngle

Validation OK

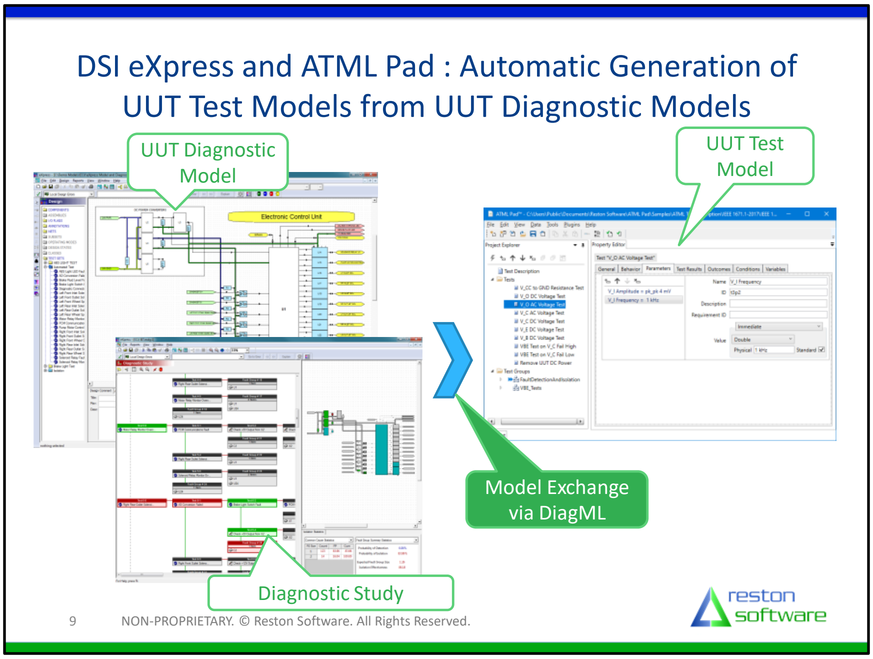
Shows AC_SIGNAL ac_ampl=1 V, dc_offset=0, freq=1 kHz, phase=0 rad

8 NON-PROPRIETARY. © Reston Software. All Rights Reserved.

newWaveX-SD (Signal Development) is a graphical design environment for signal-based test & measurement developed by Spherea Technology. It provides the facilities to design, build and simulate test signals prior to their inclusion in a test program.

newWaveX-SD is integrated natively within **ATML Pad** through the standard format.

- Signal definitions can be viewed and edited in a pop-up windows that displays the **newWaveX-SD** signal editor.
- The editor is used to configure signals, create new signal definitions, and simulate signals.



DSI eXpress and ATML Pad : Automatic Generation of UUT Test Models from UUT Diagnostic Models

UUT Diagnostic Model

UUT Test Model

Model Exchange via DiagML

Diagnostic Study

9 NON-PROPRIETARY. © Reston Software. All Rights Reserved.

eXpress is a model-based *diagnostics engineering* application developed by DSI International. **eXpress** supports the design, capture, integration, evaluation and optimization of system diagnostics, prognostics health management (PHM), systems testability engineering, failure mode and effects analysis and system safety analysis.

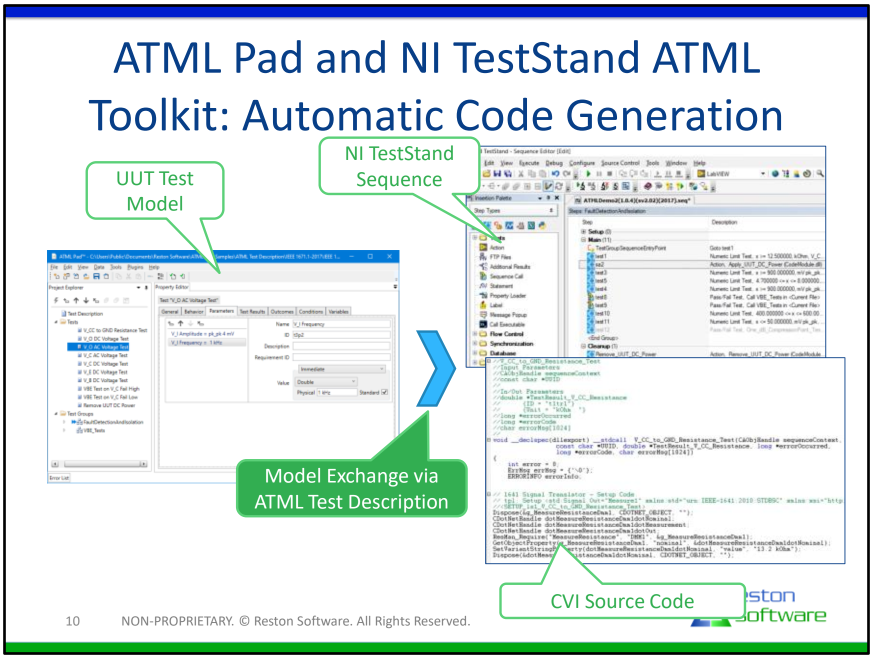
eXpress is integrated with **ATML Pad** through ***DiagML***, an open XML-based format used to represent UUT, test, and diagnostic data. ***DiagML*** is a precursor of ATML Test Description.

Design-to-test development flow using eXpress, DiagML, and ATML Pad:

1. Import UUT design data from CAD, SysML, or spreadsheets to eXpress (optional)
2. Develop diagnostic model in eXpress, adding functional dependencies, failure information, …
3. Develop diagnostic study in eXpress, generating fault trees from the diagnostic model
4. Export fault tree data to DiagML
5. Import DiagML into ATML Pad. The ATMl document will contain “sequence” test groups, tests, and test points. The test behavior will be undefined.
6. Use ATML Pad to add detailed test information: test operations, signals,

measurements, limits, …

7. Export ATML Test Description from ATML Pad
8. Use ATML Test Description to generate test programs



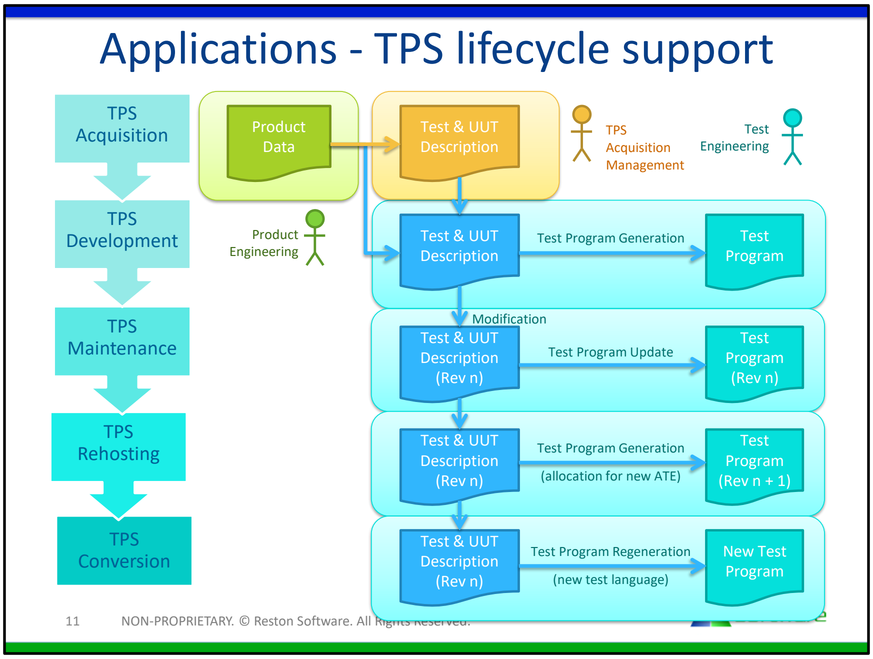
ATML Pad and NI TestStand ATML Toolkit: Automatic Code Generation

```
//V_CC_to_GND_Resistance_Test
//Input Parameters
//CAObjHandle sequenceContext
//const char *UUID
//
//In/Out Parameters
//double *TestResult_V_CC_Resistance
//        {ID = "t1tr1"}
//        {Unit = "kOhm  "}
//long *errorOccurred
//long *errorCode
//char errorMsg[1024]
//
void __declspec(dllexport) __stdcall  V_CC_to_GND_Resistance_Test(CAObjHandle sequenceContext,
                    const char *UUID, double *TestResult_V_CC_Resistence, long *errorOccurred,
                    long *errorCode, char errorMsg[1024])
{
    int error = 0;
    ErrMsg errMsg = {'\0'};
    ERRORINFO errorInfo;

// 1641 Signal Translator - Setup Code
// tpl: Setup <std:Signal Out="Measure1" xmlns:std="urn:IEEE-1641:2010:STDBSC" xmlns:xsi="http
//<SETUP_1s1_V_CC_to_GND_Resistance_Test>
Dispose(&g_MeasureResistanceDmal, CDOTNET_OBJECT, "");
CDotNetHandle dotMeasureResistanceDmaldotNominal;
CDotNetHandle dotMeasureResistanceDmaldotMeasurement;
CDotNetHandle dotMeasureResistanceDmaldotOut;
ResMan_Require("MeasureResistance", "DMM1", &g_MeasureResistanceDmal);
GetObjectProperty(g_MeasureResistanceDmal, "nominal", &dotMeasureResistanceDmaldotNominal);
SetVariantStringProperty(dotMeasureResistanceDmaldotNominal, "value", "13.2 kOhm");
Dispose(&dotMeasureResistanceDmaldotNominal, CDOTNET_OBJECT, "");
```

The **NI TestStand ATML Toolkit** is an add-on component of National Instruments (NI) TestStand. It allows TestStand to translate ATML Test Description documents into TestStand sequences and code modules written in LabVIEW or LabWindows™/CVI.

The **NI TestStand ATML Toolkit** is integrated with **ATML Pad** through a plug-in, using the standard ***ATML Test Description*** format. ATML Pad invokes the translator on the model that is currently loaded. The translator generates the test program and opens the generated sequence file in TestStand.



Applications - TPS lifecycle support

TPS Acquisition

TPS Development

TPS Maintenance

TPS Rehosting

TPS Conversion

Product Data

Test & UUT Description

TPS Acquisition Management

Test Engineering

Product Engineering

Test & UUT Description → Test Program Generation → Test Program

Modification

Test & UUT Description (Rev n) → Test Program Update → Test Program (Rev n)

Test & UUT Description (Rev n) → Test Program Generation (allocation for new ATE) → Test Program (Rev n + 1)

Test & UUT Description (Rev n) → Test Program Regeneration (new test language) → New Test Program

11 NON-PROPRIETARY. © Reston Software. All Rights Reserved.

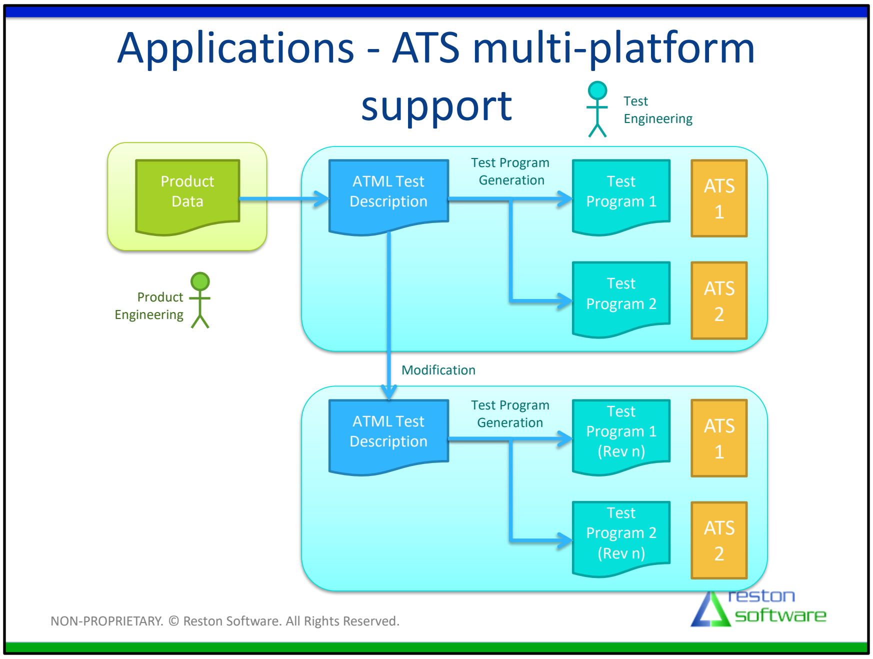
TPS Acquisition: An initial, high-level Test Description is created. This includes, for example, power requirements, signal requirements, and a test list. The high-level description allows the estimation of development time, cost, risk, and reuse potential through comparison with past developments.

TPS Development: Details are added to the Test Description. This includes test parameters, test results, test sequences, and detailed test operations. The description has a sufficient level of detail to enable automatic test program generation.

TPS Maintenance: Changes are made to the Test Description; the auto-generated test program is updated automatically. This ensures consistency between the Test Description (and thus test program documentation) and the test program.

TPS Rehosting: The test program requirements are re-allocated to the capabilities of a new ATE. A new revision of the test program is generated.

TPS Conversion: New code is generated in a different test language. A new test program is generated.



Applications - ATS multi-platform support

Test Engineering

Product Data

Product Engineering

ATML Test Description

Test Program Generation

Test Program 1

ATS 1

Test Program 2

ATS 2

Modification

ATML Test Description

Test Program Generation

Test Program 1 (Rev n)

ATS 1

Test Program 2 (Rev n)

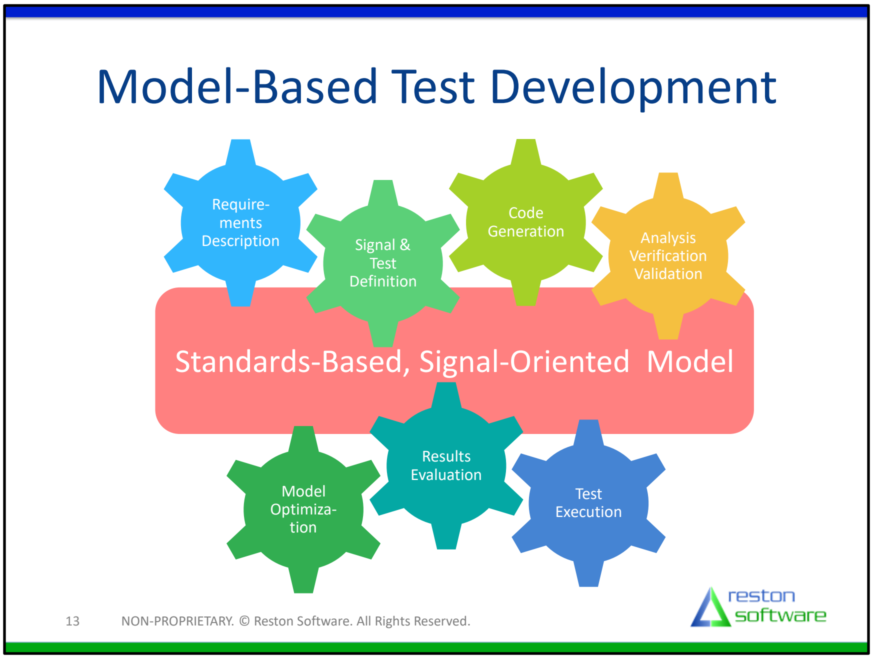
ATS 2

NON-PROPRIETARY. © Reston Software. All Rights Reserved.

reston software

1. From a common set of requirements, two test programs are generated, targeting two different hardware platforms (ATS 1 and ATS 1)
2. If a modification is needed (ex. correction or optimization), the modification is performed on the Test Description model. New revisions of the Test Programs are auto-generated.

This approach eliminates the need to modify two different test programs and preserves their consistency automatically.



Model-Based Test Development

Requirements Description

Signal & Test Definition

Code Generation

Analysis Verification Validation

Standards-Based, Signal-Oriented Model

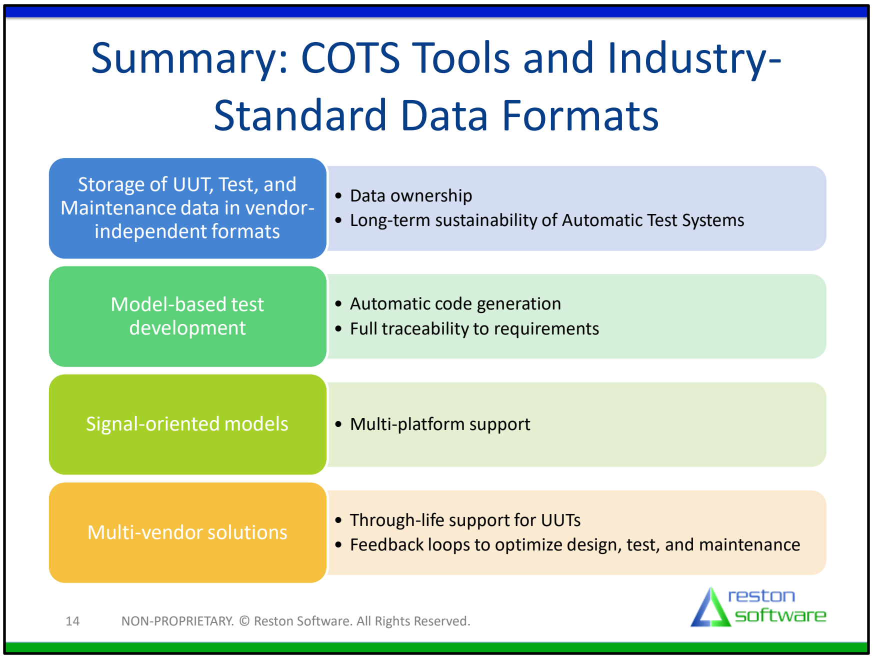
Model Optimization

Results Evaluation

Test Execution

reston software

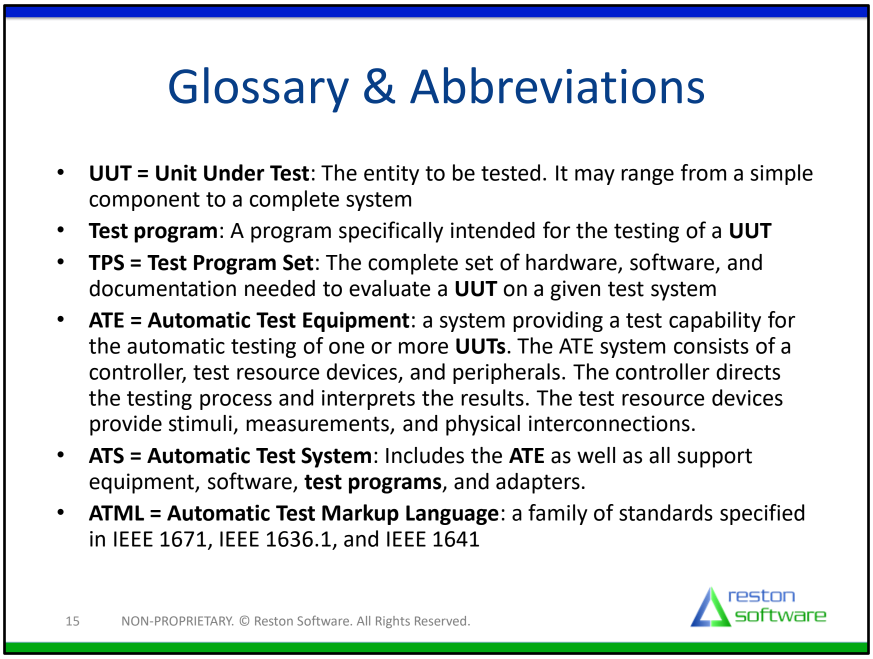
All stages of test program development (requirements analysis, design, implementation, use, maintenance, and optimization) are based on a *common model*, using *industry-standard formats* and *signal abstractions*.



Summary: COTS Tools and Industry-Standard Data Formats

Storage of UUT, Test, and Maintenance data in vendor-independent formats	• Data ownership • Long-term sustainability of Automatic Test Systems
Model-based test development	• Automatic code generation • Full traceability to requirements
Signal-oriented models	• Multi-platform support
Multi-vendor solutions	• Through-life support for UUTs • Feedback loops to optimize design, test, and maintenance

reston software



Glossary & Abbreviations

- **UUT = Unit Under Test**: The entity to be tested. It may range from a simple component to a complete system
- **Test program**: A program specifically intended for the testing of a **UUT**
- **TPS = Test Program Set**: The complete set of hardware, software, and documentation needed to evaluate a **UUT** on a given test system
- **ATE = Automatic Test Equipment**: a system providing a test capability for the automatic testing of one or more **UUTs**. The ATE system consists of a controller, test resource devices, and peripherals. The controller directs the testing process and interprets the results. The test resource devices provide stimuli, measurements, and physical interconnections.
- **ATS = Automatic Test System**: Includes the **ATE** as well as all support equipment, software, **test programs**, and adapters.
- **ATML = Automatic Test Markup Language**: a family of standards specified in IEEE 1671, IEEE 1636.1, and IEEE 1641

Thank you!

This Photo is licensed under CC BY-NC

16

reston software