

Requirements for Standard Compliant Electrical Test Authoring in Manufacturing Applications

Dipl. Ing. Alfons Schulte

A semi-transparent blue-tinted background image showing a car's engine compartment on the right and a factory interior with robotic arms on the left.

Diagnostics
Systems
Applications

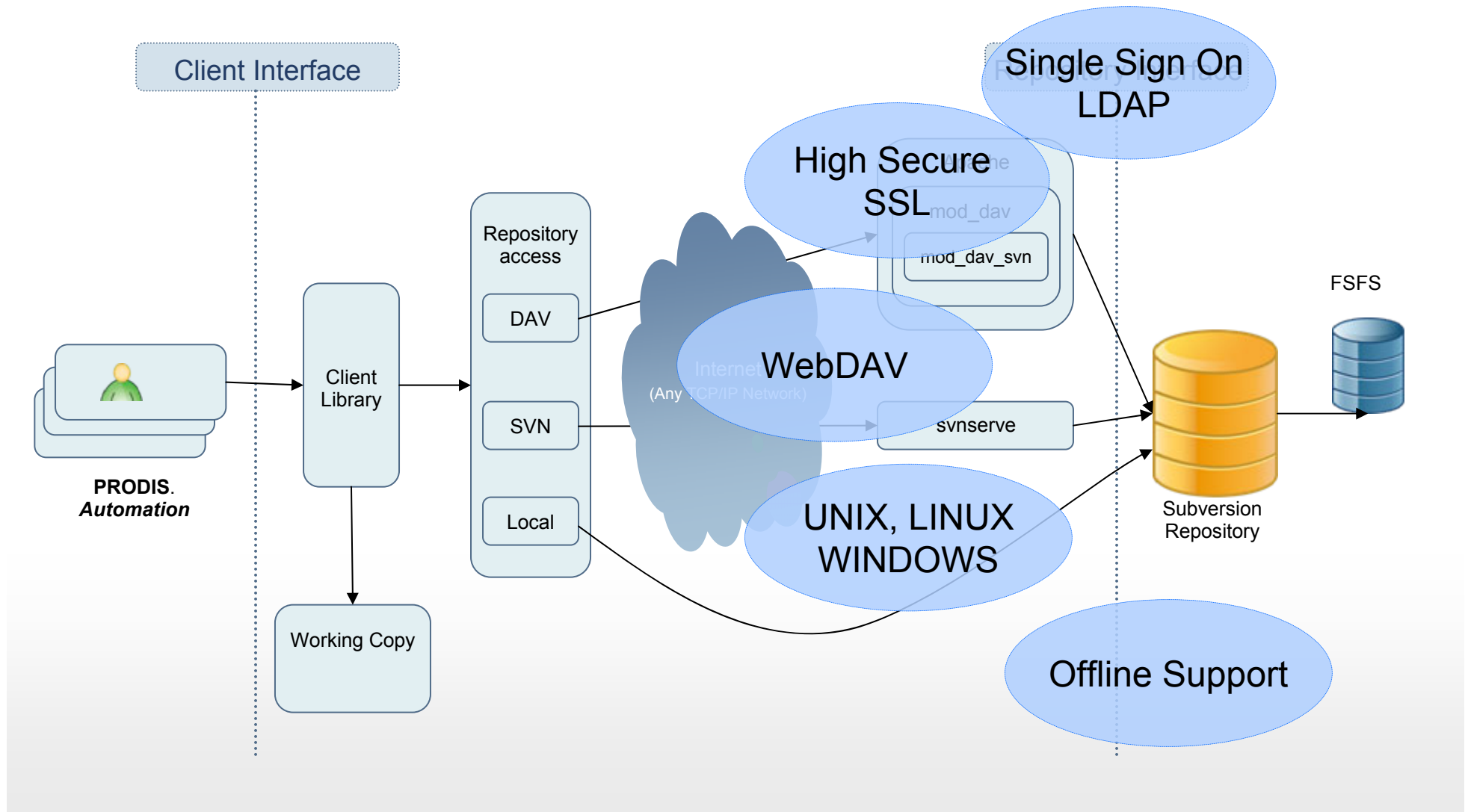
- **Authoring Requirements for Manufacturing E/E-Test**
- **Applicable Standards**
- **DSA's Approach: *PRODIS.Automation***
- **Customer Benefits**

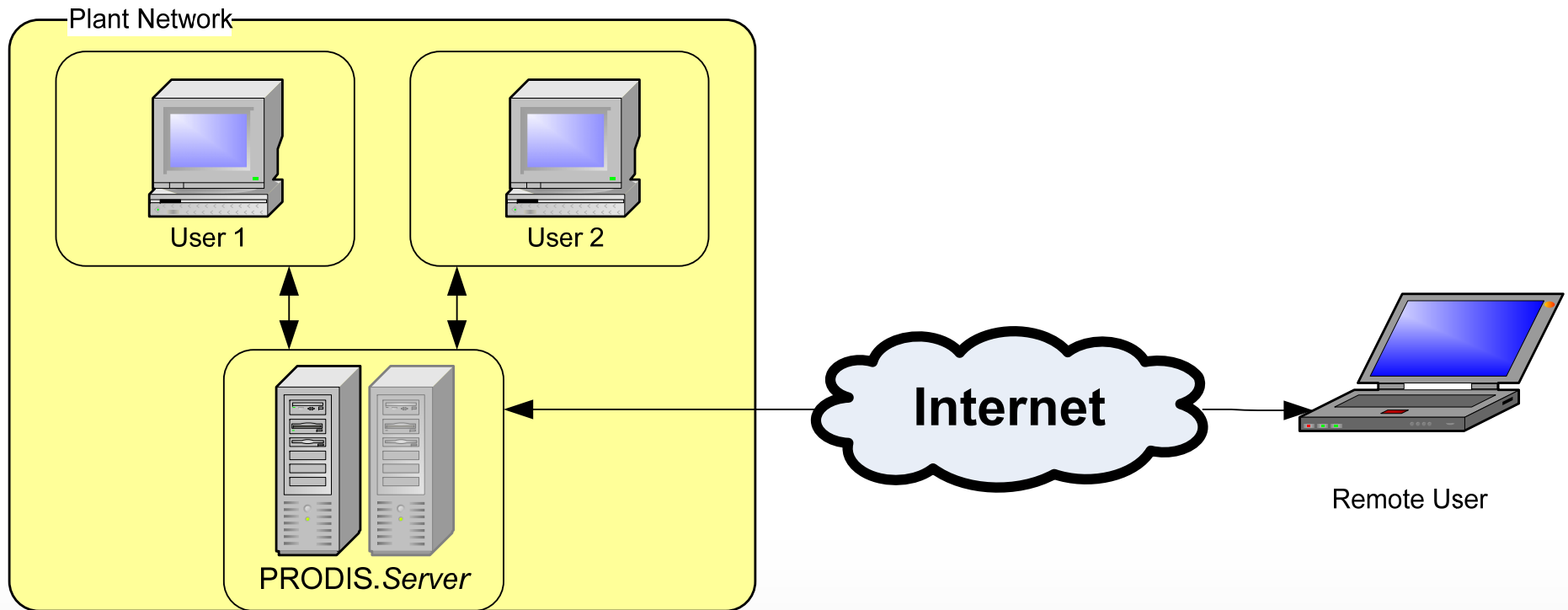
Authoring Requirements for Manufacturing E/E-Test

- **Graphical Editor for Scripts**
- **User / Roll Management**
- **Effective Integration into the OEM's Diagnostic Process**
- **Multi-User / Multi-Site Operation**
- **Multi-Language Operation**
- **Import & Export Functions**
- **Versioning & Release Management**
- **Simulation & Debugging**

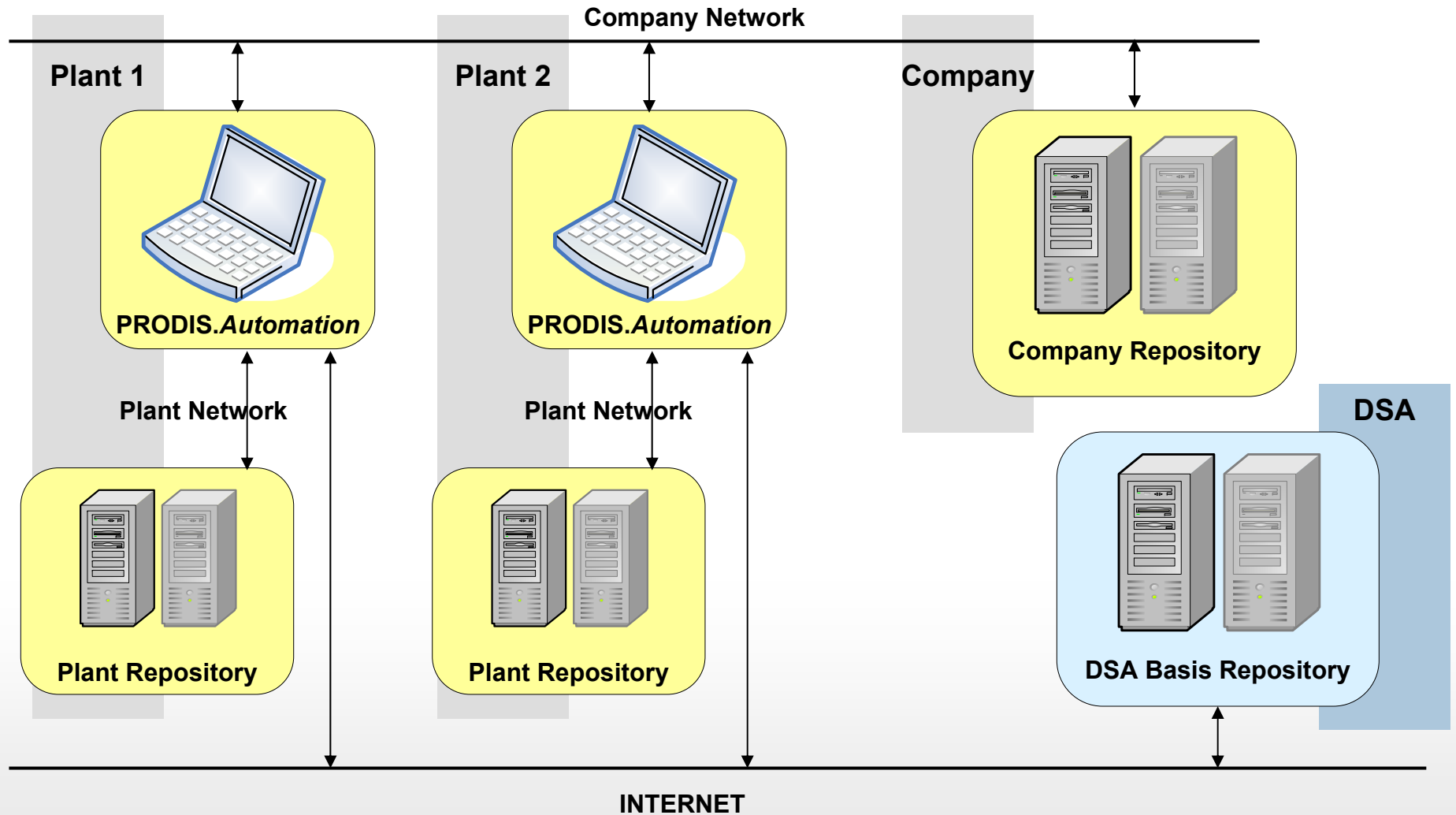
- Graphical Editor for Scripts
- User / Roll Management
- Effective Integration into the OEM's Diagnostic Process
- **Multi-User / Multi-Site Operation**
- Multi-Language Operation
- Import & Export Functions
- Versioning & Release Management
- Simulation & Debugging

Subversion – Architecture Overview



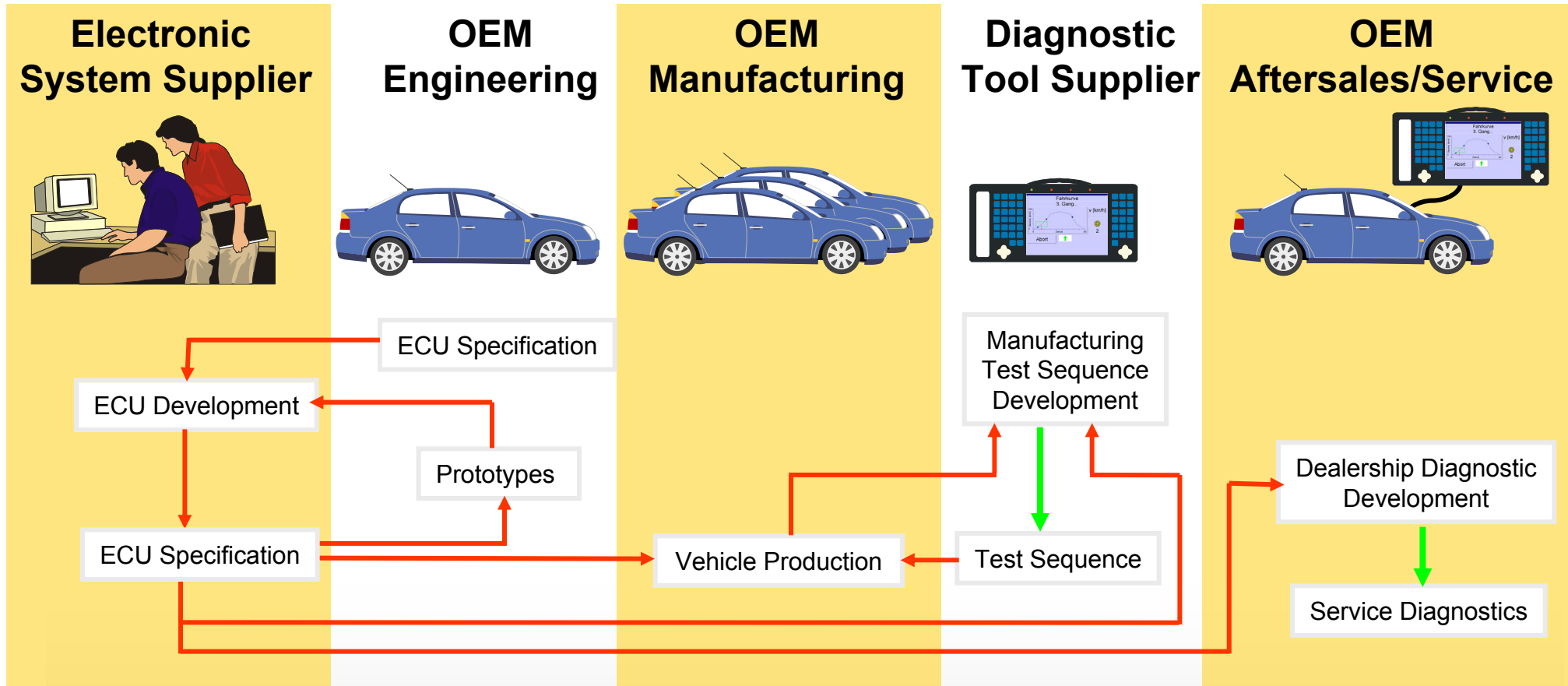


Multi-Site Access



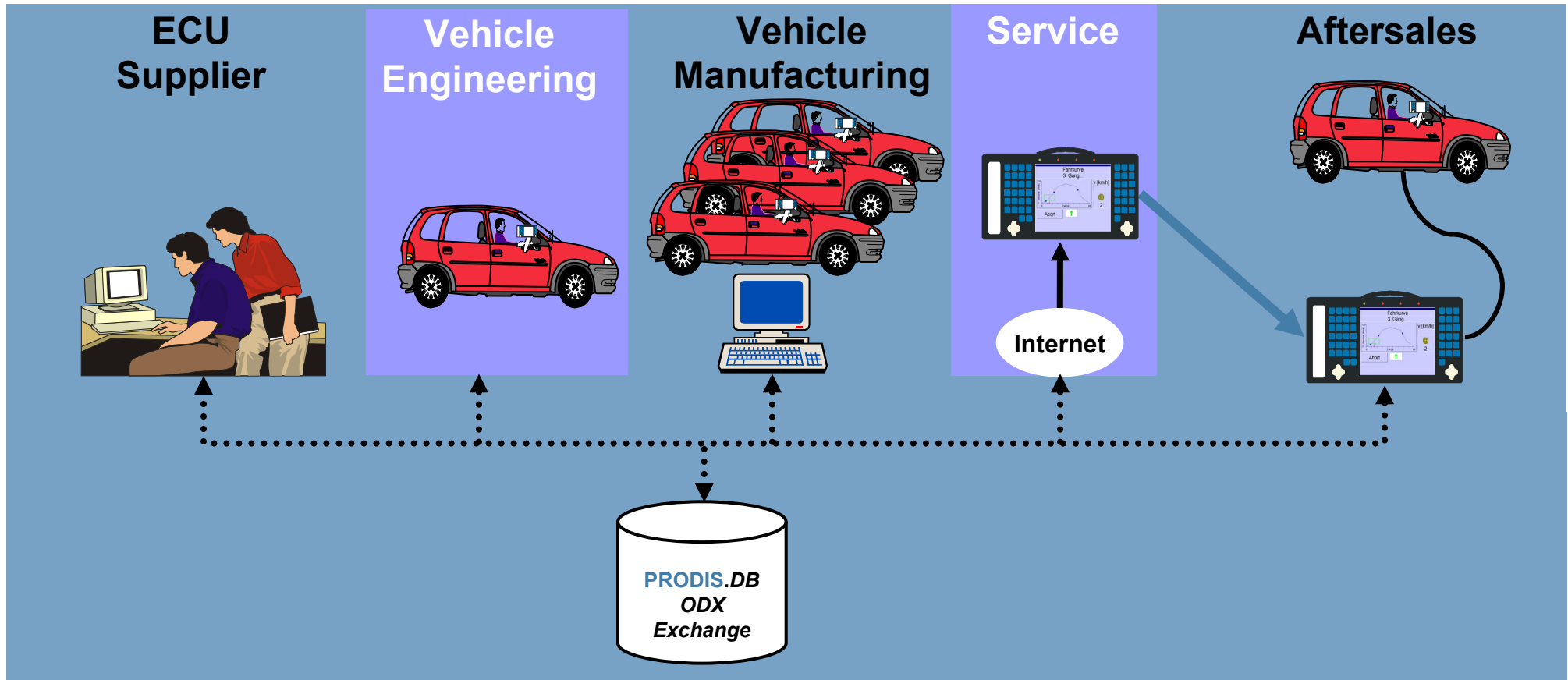
- Graphical Editor for Scripts
- User / Roll Management
- **Effective Integration into the OEM's Diagnostic Process**
- Multi-User / Multi-Site Operation
- Multi-Language Operation
- Import & Export Functions
- Versioning & Release Management
- Simulation & Debugging

Diagnostic-Process (yesterday)



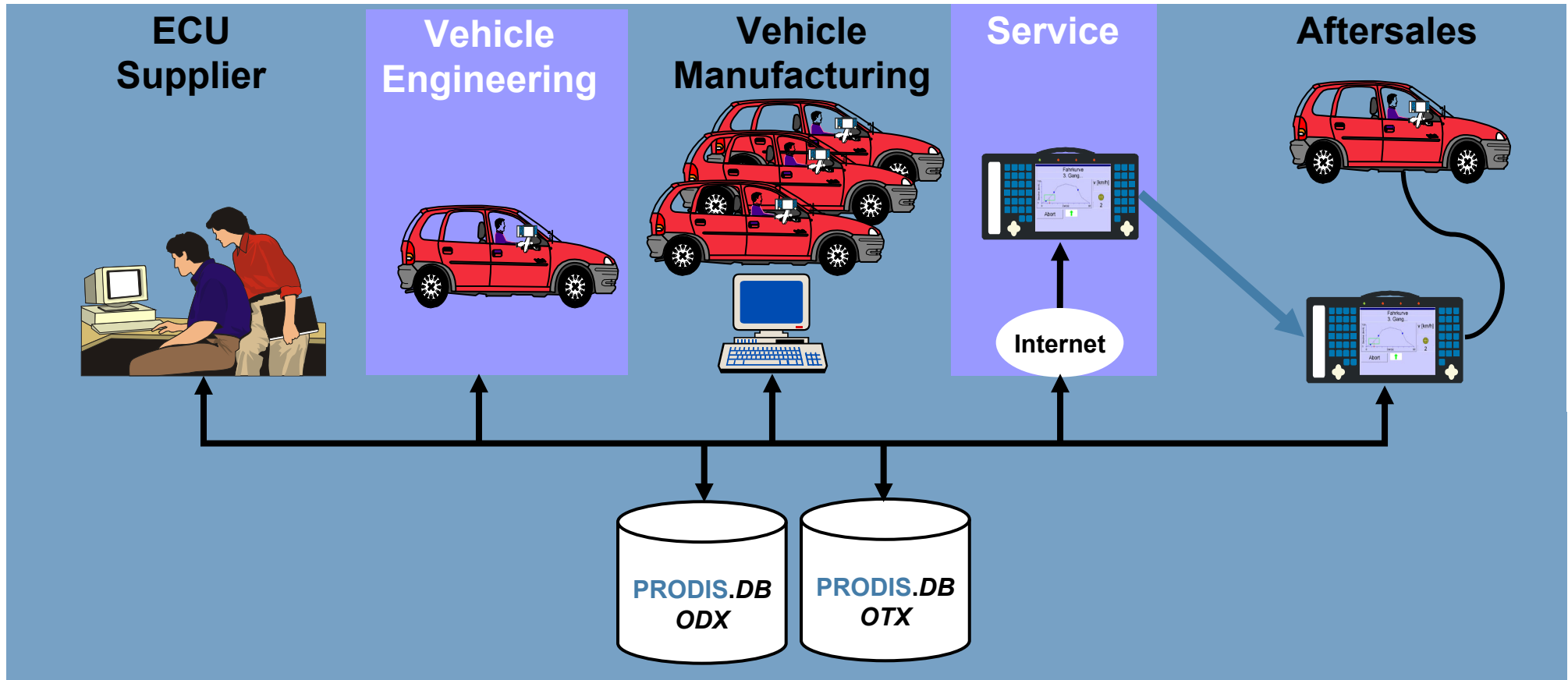
The current situation is characterized by media disruption and a inhomogeneous system components.

Diagnostic-Process (today)



The diagnostic process chain uses common data (ODX) but still a inhomogeneous test sequences due to incompatible scripting products.

Diagnostic-Process (tomorrow)



The diagnostic process uses data and sequence (script) integration without media or system (tool) disruption.

Applicable Standards

■ ODX Browsing

- ➔ supported versions:
 - ODX 2.0.1, 2.1.0
 - ODX 2.2 available 11/2010
- ➔ comfortable assignment of request and response parameters

■ GDI Browsing

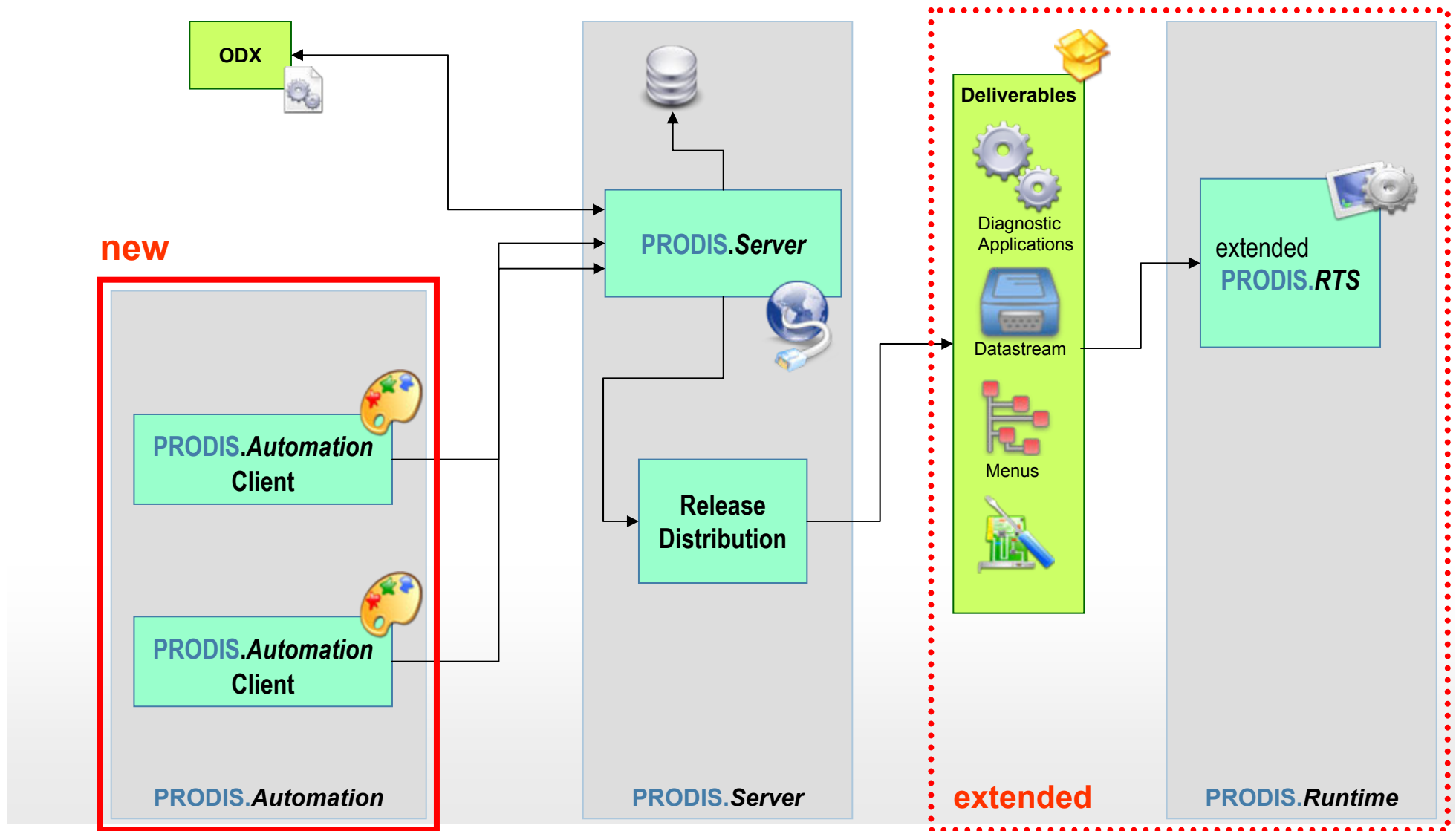
- ➔ Browsing of DCDs version 4.3.2

■ OTX Import & Export Functions

- ➔ OTX export and import available 03/2011

DSA's Approach: **PRODIS**.Automation

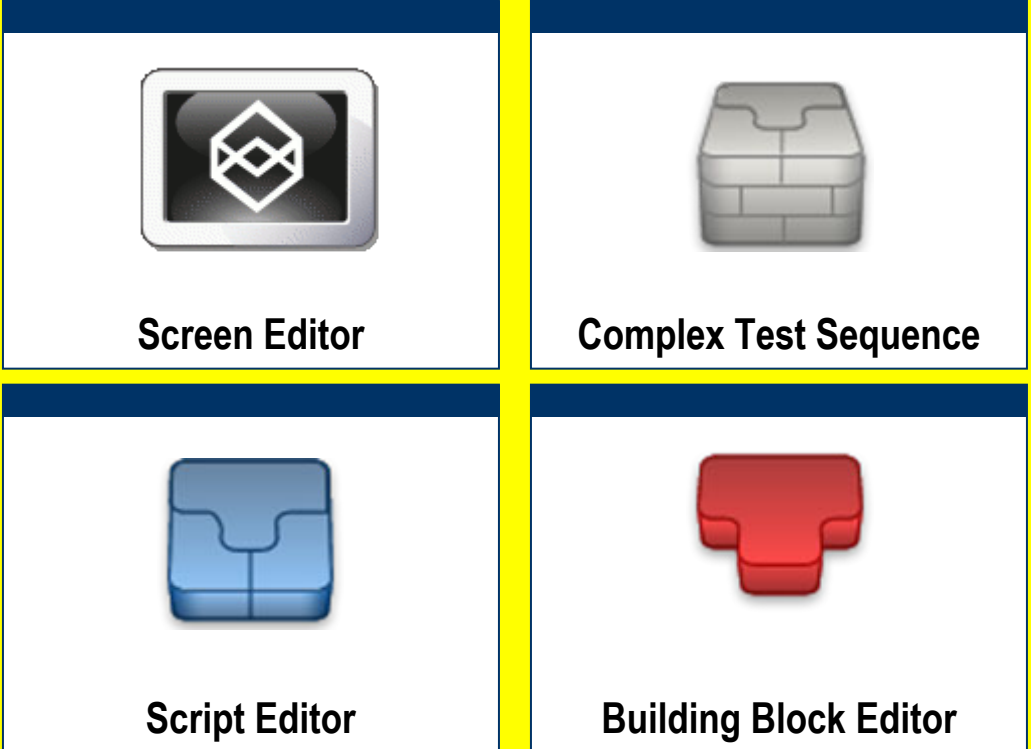
Overall System Architecture



Integration with DSA's **PRODIS** product family



PRODIS.*Authoring*
Datastream Editor



Screen Editor

Complex Test Sequence

Script Editor

Building Block Editor

PRODIS.*Automation Core System*

CTS



Main application flow

Mostly deals with handling the User input and the output to The screen

Screen



Graphic Display

Output to the user in easy to configure widgets

Scripts



Common Tasks

Routines that handle common diagnostic routines such as reading values

Building Blocks

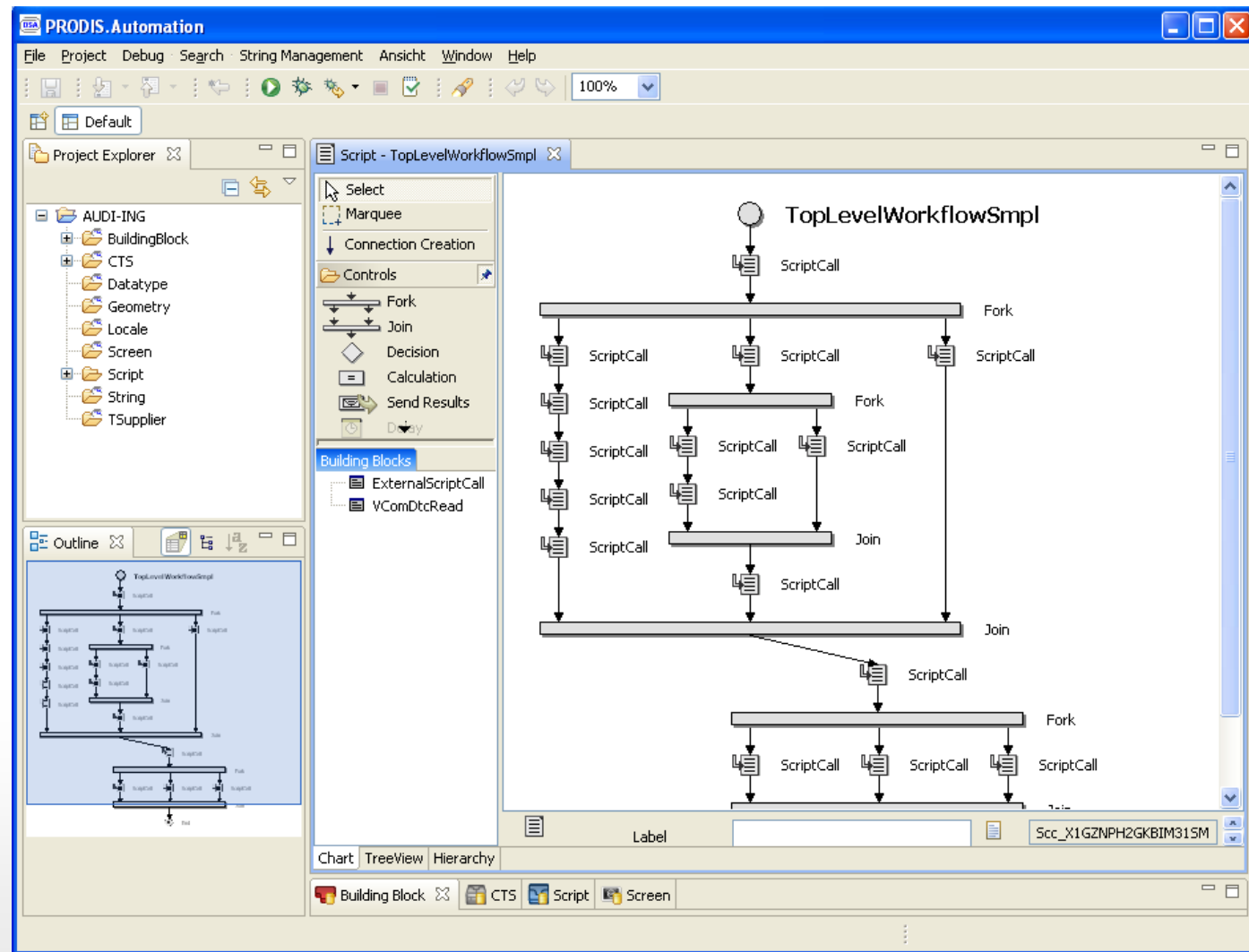


Low-level Access

Calls to the Vehicle Electronics, And to external interfaces.

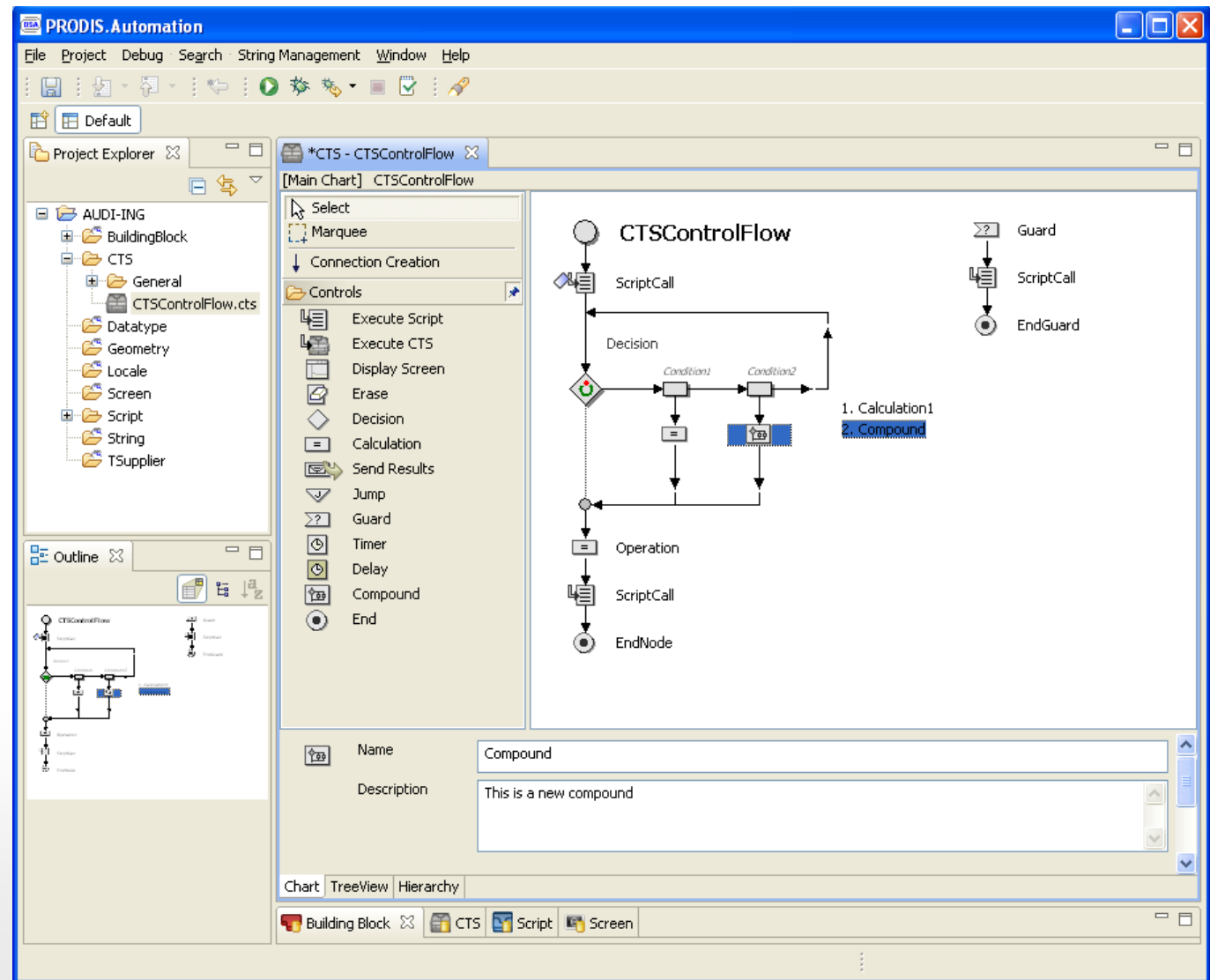
Level 1: Top Level Workflow Sequence

- Graphical representation of sequence and parallel tasks



Level 2: Diagnostic Sequence

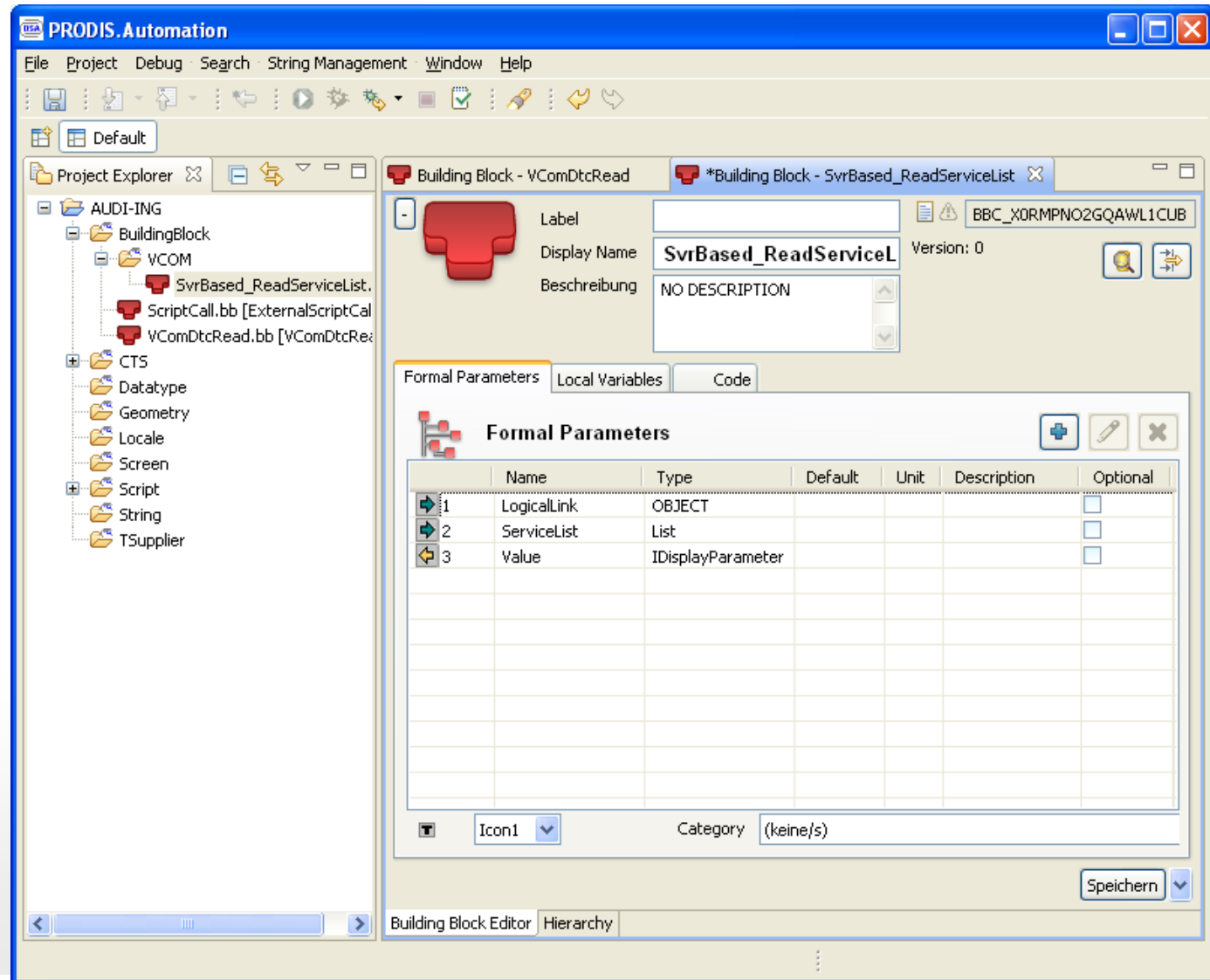
- Graphical representation of diagnostic progression
- Representation of parallel processing
- Guard: Control flow depending on specific event



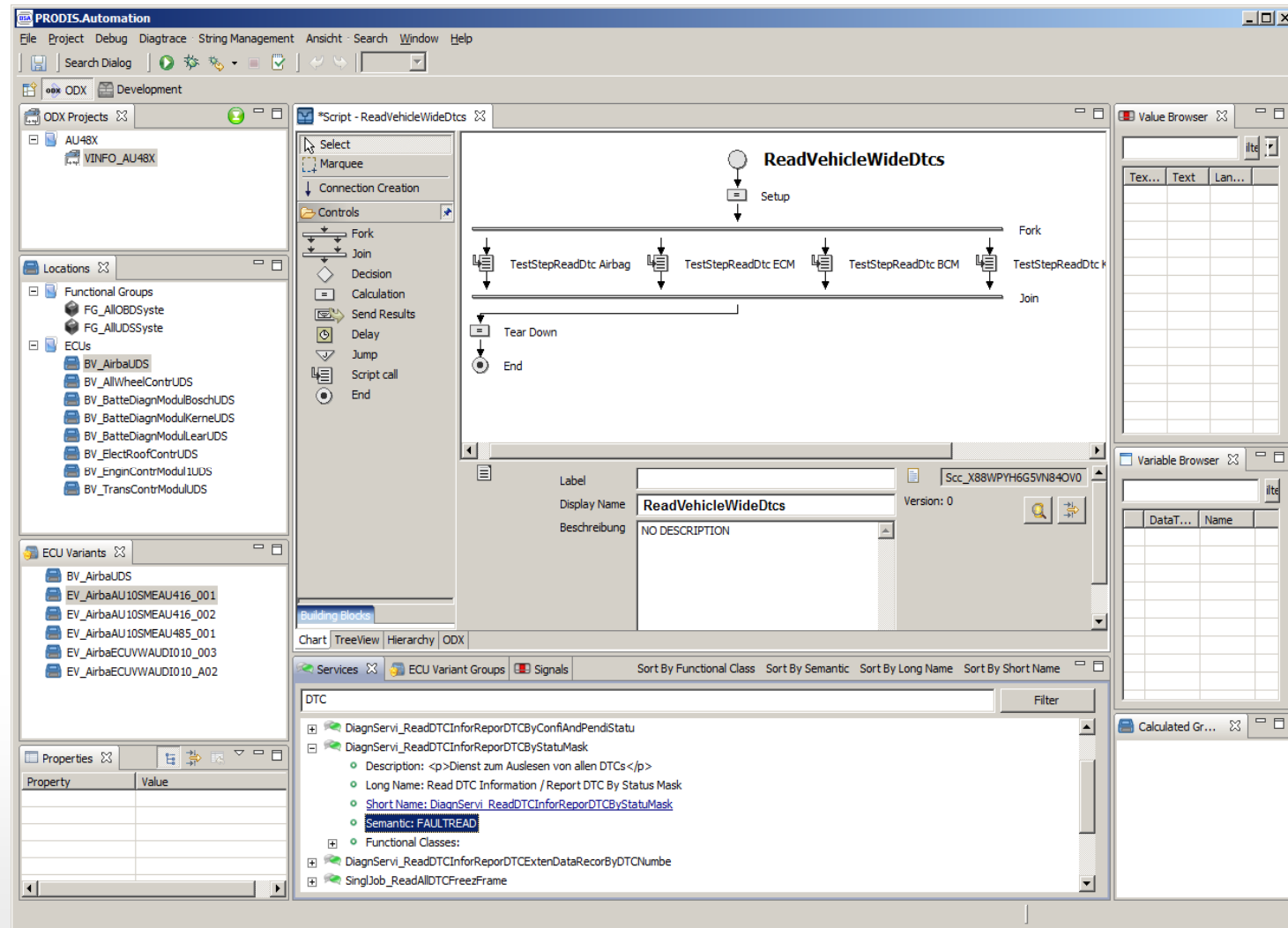
Level 3: Library Elements



- Generation of new library elements
- Usage of JAVA as programming language
- Sequence highlighting
- automatic code completion
- Integrated source code debugger

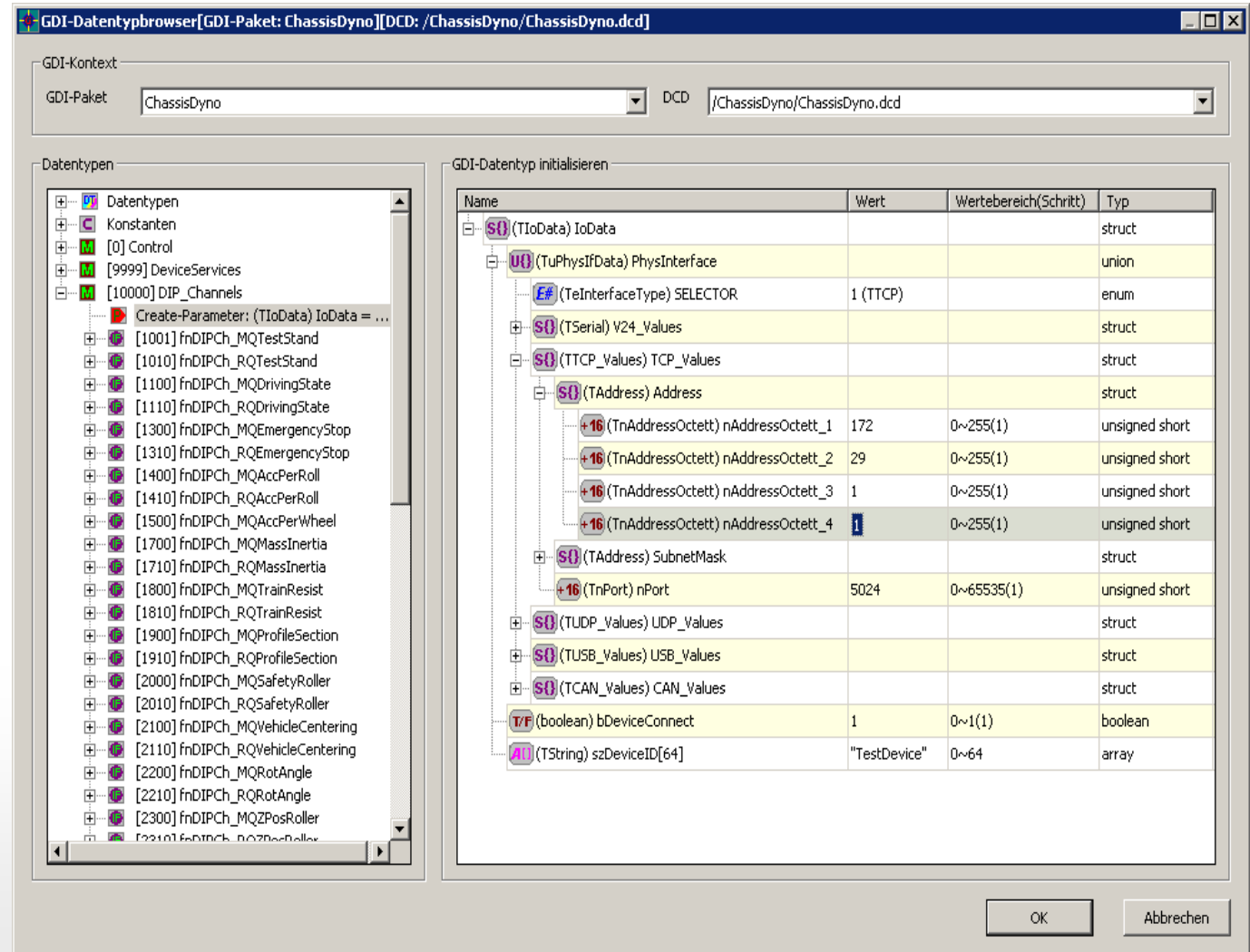


- Comfortable selection of ODX service
- Assignment of request and response parameters
- Sophisticated ECU Variant handling by defining ECU groups



The screenshot displays the PRODIS.Automation software interface. The main window shows a test script titled '*Script - ReadVehicleWideDtc'. The script flow starts with a 'Setup' block, followed by a 'Fork' block that branches into four parallel test steps: 'TestStepReadDtc Airbag', 'TestStepReadDtc ECM', 'TestStepReadDtc BCM', and 'TestStepReadDtc'. These steps are joined back together, followed by a 'Tear Down' block and an 'End' block. The interface includes several panels: 'ODX Projects' on the left showing 'AU48X' and 'VININFO_AU48X'; 'Locations' showing functional groups and ECUs; 'ECU Variants' listing various ECU models; 'Properties' at the bottom left; and 'Value Browser', 'Variable Browser', and 'Calculated Gr...' on the right. A 'Building Blocks' panel at the bottom center shows a list of DTC services, including 'DiagnSeri_ReadDTCInforReporDTCByConfAndPendiStatu' and 'DiagnSeri_ReadDTCInforReporDTCByStatuMask'.

- Reuse of existing *PRODIS.Office* GDI building blocks and sequences
- Integrated browsers for the selection and parameterization of GDI data structures
- Integrated browsers for the selection and parameterization of GDI communication objects and operations



Name	Wert	Wertebereich(Schritt)	Typ
(S0) (TioData) IoData			struct
(U0) (TuPhysIfData) PhysInterface			union
(E#) (TeInterfaceType) SELECTOR	1 (TTCP)		enum
(S0) (TSerial) V24_Values			struct
(S0) (TTCP_Values) TCP_Values			struct
(S0) (TAddress) Address			struct
(+16) (TnAddressOctett) nAddressOctett_1	172	0~255(1)	unsigned short
(+16) (TnAddressOctett) nAddressOctett_2	29	0~255(1)	unsigned short
(+16) (TnAddressOctett) nAddressOctett_3	1	0~255(1)	unsigned short
(+16) (TnAddressOctett) nAddressOctett_4	1	0~255(1)	unsigned short
(S0) (TAddress) SubnetMask			struct
(+16) (TnPort) nPort	5024	0~65535(1)	unsigned short
(S0) (TUDP_Values) UDP_Values			struct
(S0) (TUSB_Values) USB_Values			struct
(S0) (TCAN_Values) CAN_Values			struct
(TF) (boolean) bDeviceConnect	1	0~1(1)	boolean
(A0) (TString) szDeviceID[64]	"TestDevice"	0~64	array

Customer Benefits

- **fully supports all ASAM/ISO ODX standards**
- **allows comfortable control of ASAM GDI devices**
- **will be OTX ready by 03/2011**
- **enables multi-user access**
- **enables multi-site access**

- **effectively includes version and change management**

- **allows**

- ⇒ Setting **Tags** and
- ⇒ Opening **Branches**

thus allowing the user to perform easy rollback or concurrent development

- **enables soft migration from **PRODIS.Office****

- ⇒ by reuse of existing library elements and sequences and
- ⇒ integrated MSL interpreter in runtime environment

- **smoothly integrates into DSA's Manufacturing Test Suite**

Thank You
