TYX TestBase

Development of Diagnostics with DSI eXpress and TYX TestBase

For eXpress versions 5.10.x.

© Copyright TYX Corporation 2006

DSI eXpress User Group Meeting
Sept 22, 2006
Presented by: Brian Lennox
Western Regional Sales Manager
TYX Corporation
T: 661-296-1451
E-Mail: Brian.Lennox@TYX.com
DSI eXpress

- Model-Based Diagnostics Engineering and System Governing tool
  - Provides an object-oriented approach to full-system design
  - Supports analysis and optimization throughout all phases of development

- Functionality
  - Development of dependency models
  - Modeling of system test strategies
  - Diagnostic analysis (fault detection and fault isolation)
  - Failure Mode Effects and Criticality Assessment (FMECA)
Test Executive

- Open architecture enables integration between
  - Diagnostic development tools
  - Test languages and environments
  - User interface modules
  - Storage of test results

Functionality

- Visual development of test strategies
- Import of test strategies from third-party tools
- Execution of test strategies using third-party test environments
- Collection of test results
- Statistical analysis of test results
Integrated “Design-to-Test” Process

- System Designers
- Diagnostic Developers
- Test Developers
- Operator / User

- System Design
- Diagnostic Model Development
- Test Strategy Development
- Test Strategy Execution
- System Under Test

- Diagnostic Analysis, FMECA
- Test Results Analysis
- Test Results Collection

- eXpress
- TestBase
- Third-party software
Integration Architecture
DiagML:

- “Diagnostic Modeling Language”
- Based on XML
- Developed by a consortium of companies as an open specification
- After a trial period, open to membership by other companies
- Benefits
  - Explicit extensibility
  - Parsability
  - Transformability
  - Wide industry acceptance
  - Human readable
- Details at [http://www.diag-ml.com](http://www.diag-ml.com)
1. Build Diagnostic Model
2. Build Test Set
2. Build Test Set (cont’d)

- Attribute Definitions
  - Attributes: Address_DMM, Address_SigGen, Address_Switch, Cost, ExecutionEnvironment, FailRate, HighLimit, Locator, LowLimit, Time

- Default Values (Roll-Up Indenture shown):
  - Design
    - Objects
    - Points
    - Functions
    - Failure Modes
  - Nets
  - Test Sets
  - Tests
  - Failure Effects

- Attribute Value
  - GPIBO::22.INSTR

- Publishing Templates:
  - Include this Attribute when Publishing

- Details Attributes Test Interpretation Test PreRequisites Test Outcomes
  - Name: Address_DMM, GPIBO::22.INSTR, Address_SigGen, GPIBO::10::INSTR, Address_Switch, GPIBO::9::INSTR, ExecutionEnvironment, TCA_DLL, TCA_DLL, HighLimit, 0.1814, Locator, Demo_CVI.dll, OperatingPoint_Hi
  - Source: User-entry, User-entry, User-entry, User-entry, User-entry, User-entry, User-entry, User-entry, User-entry, User-entry
3. Generate Diagnostic Strategy

Select Document Type

- eXpress Design
- Diagnostic Study
- FMECA Study

Create  Cancel

Link New Diagnostic Study?

Do you want the new diagnostic study to be linked to the active document?

Yes  No  Cancel
3. Generate Diagnostic Strategy (cont’d)
3. Generate Diagnostic Strategy (cont’d)
Integrated Diagnostic Development...

4. Import Diagnostic Strategy in TestBase

- Import Diagnostic Strategy in TestBase
5. Develop TestBase Test Procedures

- One TestBase test procedure for each Test defined in eXpress
  - Implementation must be consistent with the characterization of the test procedure, in the Test Database where DiagML was imported.
    - Property “Adapter ProgID” indicates the test language/environment to be used
    - Property “Locator” indicates the location of the test procedure code (ex. DLL name and function name)
    - Input parameter “TestPoint” indicates the location of the measurement; use for switching (if applicable)
    - All other input parameters originate from eXpress Test attributes; implement as designed
  - Recommended: for test procedures that have identical functionality but different parameter values and/or test point, delegate to a unique underlying function

- The special test procedure “DisplayMaintenanceAction”
  - Displays or implements the required maintenance action
6. Execute Test Strategy

- Move TestBase databases to production or embedded environment
- Configure run-time options
  - Assign MTI database, for collection of test results
- Execute test strategy
  - Execution reports/remediates “diagnosed faults”; to enable statistical assessment of diagnostic performance, enter the “actual faults” in the MTI Database (ex. via the MTI Database GUI)
- Evaluate diagnostic performance
  - Performed off-line, after a sufficient amount of test results was accumulated
  - Generate statistic reports from MTI Database GUI (new feature in TestBase 2.6)
  - Use third-party software to retrieve and process test results from the MTI database
## Mapping of Design Entities

<table>
<thead>
<tr>
<th><strong>eXpress</strong></th>
<th><strong>TestBase</strong></th>
</tr>
</thead>
</table>
| Diagnostic strategy | ➢ Set of test procedures, in a Test Database  
➢ Test strategy with one/more diagnostic procedures, in a Diagnostic Database |
| Test node | ➢ Test procedure  
➢ “Test” block in the diagnostic procedure |
| Test Location | ➢ Test procedure input parameter “TestPoint”  
➢ Test input parameter value |
| Test attribute “ExecutionEnvironment” | ➢ Test procedure property “Execution Environment” (i.e., Adapter ProgID) |
| Test attribute “Locator” | ➢ Test procedure property “Locator” (ex. DLL name, function name) |
| Other test attributes | ➢ Test procedure input parameter  
➢ Test input parameter value |
| Fault Group node | ➢ Test procedure “DisplayMaintenanceAction”  
➢ “Test” block  
➢ “End” block |
| Fault Group objects | ➢ Value of input parameter “MaintenanceAction” of “Test” block  
➢ Diagnostic procedure outcome assigned to “End” block |
eXpress Design Rules

- Tests shall have only one Location
- Each test shall have the following attributes:
  - “ExecutionEnvironment” – indicates the test language/environment to be used for execution
  - “Locator” – indicates the location of the test procedure code
- Tests shall not have attributes named “TestPoint”
TestBase Design Rules

- Test procedures implementing eXpress Tests
  - All test procedures shall support the input parameter “TestPoint” and use it to determine the location of the measurement
  - All test procedures shall support input parameters corresponding to the Test attributes defined in eXpress (excepting attributes “ExecutionEnvironment” and “Locator”)
  - All test procedures shall support the Outcome values “PASS” and “FAIL”

- Special test procedure “DisplayMaintenanceAction”
  - Shall support the input parameter “MaintenanceAction”, of type string
  - May display the string to the user, or may implement a remediation action (if applicable)
  - Is not required to return an Outcome value
  - A default implementation is available in <TestBase installation directory>\Samples\TPs\CVI\Demo_CVI\Demo_CVI.prj
Example

Fault Isolation

- eXpress model: <TestBase installation directory>\Samples\Importers\DiagML\Demo UUT.exd
- Test strategy in DiagML format: <TestBase installation directory>\Samples\Importers\DiagML\Demo UUT.xml
- Test strategy imported in TestBase:
  - Test Database: <TestBase installation directory>\Samples\Projects\DemoTPs.ttd
  - Diagnostic Database: <TestBase installation directory>\Samples\Projects\Demo.tdd
    - UUT Model: “UUT”
    - Test Strategy: “DiagML import”
- Test procedures (LabWindows/CVI): <TestBase installation directory>\Samples\TPs\CVI\Demo_CVI\Demo_CVI.prj
Future enhancements

- Optimization of Export and Import, to Reduce:
  - The number of test procedures
  - The size of test strategies
  - The duration of import