

Using Testbase for Embedded Systems Diagnostics

TYX TestBase [1] is a software product that enables the visual development and run-time execution of diagnostic strategies utilizing large bodies of test procedures. Its flexibility allows modification of parametric data without having to resort to modification of source code. TestBase supports test and diagnostics in production (factory) and maintenance environments.

This article describes the expansion of TestBase into a third environment - the *run-time execution of test and diagnostics on embedded systems*. With this new functionality, systems and system components can be constantly monitored and diagnosed in real-time, allowing mission commanders and operators an on-going view of the health of their systems. Used together, the existing TestBase and the newer Embedded TestBase provide *integrated test and diagnostics capabilities* for all three environments.

Functionality and Architecture

Figure 1 shows the architecture of an integrated test and diagnostics solution supported by TestBase, encompassing the three operational environments:

1. *The **Diagnostic Development Environment** contains the software applications that enable the development of diagnostic strategies to be executed in the Maintenance and Embedded Environments.*
2. *The **Maintenance Environment** contains test and diagnostics software, test instrumentation and data storage components that support the testing, diagnosis and maintenance of the vehicle off-line between missions.*
3. *The **Embedded Environment** contains the vehicle hardware and software, including the software components that implement test and diagnostics on-line and in real-time during a mission.*

Diagnostic Development Environment

The development of diagnostic strategies is performed using the **TestBase Integrated Development Environment (IDE)**, which enables the visual design of "fault tree" strategies. The diagnostic strategy information, including sequence information and parametric data, is stored in a TestBase Database. The TestBase IDE is capable of importing diagnostic strategy information from Third-Party Diagnostic Development Software, such as DSI eXpress [2] and Intusoft Test Designer [3]. These applications generate diagnostic strategies using vehicle design data in the form of CAD/CAEE files, simulation models and human expertise. The transfer of diagnostic strategy data is performed via an XML-based format called DiagML, which was jointly developed by TYX and DSI International and is offered as an open industry specification.

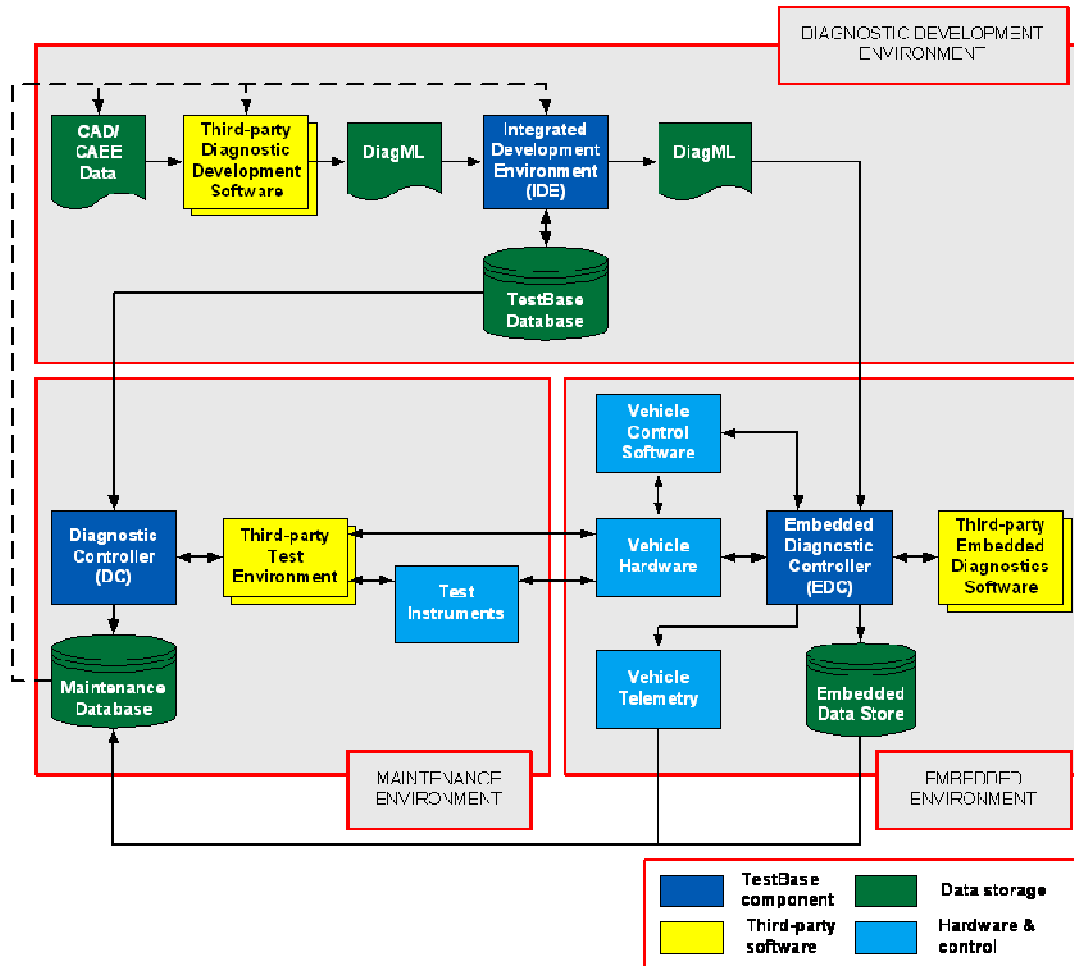


Figure 1

Maintenance Environment

The execution of diagnostic strategies in the Maintenance Environment is performed by the **TestBase Diagnostic Controller (DC)**, which retrieves sequence information and parametric data from the TestBase Database.

The DC triggers the execution of external software modules, called “test procedures”, according to the sequence information from the TestBase Database. Test procedures receive parametric data from the DC which has, in turn, retrieved them from the TestBase Database.

Test procedures return an outcome to the DC indicating the result of the test, and possibly other test results such as measurement values. The test procedures are developed and executed with various Third-party Test Environments.

TestBase supports specialized test applications such as VEE, LabView, LabWindows/CVI, PAWS ATLAS, and general-purpose programming languages including C, C++ and Visual Basic. The test procedures may use Test Instruments, as well as a direct communication channel to the Vehicle Hardware, used to transmit commands and receive sensor and status data.

At the end of a diagnostic strategy execution, the DC determines a diagnostic conclusion, representing a fault or a fault group. The DC stores test and diagnostics results (parametric data passed to test procedures, outcomes and other results returned by test procedures, diagnostic conclusions, etc.) in a Maintenance Database. TestBase currently supports relational databases and an XML-based file format. The Maintenance Database may store additional maintenance-related information input by maintenance personnel, such as the faults identified during repair, the duration and cost of repair operations, etc.

Historical data from the Maintenance Database may be used to improve the vehicle design, by redesigning the components that cause the most frequent failures or by adding redundancy. In addition, Maintenance Database data may be used to improve the diagnostics design, in order to maximize fault detection and isolation capabilities. Lastly, the statistical analysis of Maintenance Database information may support the optimization of maintenance operations.

Embedded Environment

The execution of test and diagnostics operations in the Embedded Environment is coordinated by the **TestBase Embedded Diagnostic Controller (EDC)**. While this component provides the same basic functionality as the DC used in the Maintenance Environment, its implementation is optimized for embedded execution. This optimization meets the following objectives:

- Portability of implementation to various embedded operating systems (OSs)
- Reduced resource requirements in terms of memory utilization and computational load
- Capability to interact with the vehicle control software
- Capability to interact with third-party embedded diagnostics software

To avoid the high resource consumption of a database engine, diagnostic strategy data is uploaded from the TestBase IDE to the Embedded Environment using DiagML or an application-specific data format generated from DiagML.

The EDC operates under the supervision of the **Vehicle Control Software**, which requests the execution of diagnostic strategies corresponding to various control modes. The EDC receives sensor data and state information and returns diagnostic conclusions. In addition, the EDC may send command requests, instructing the Vehicle Control Software to stimulate the Vehicle Hardware in order to facilitate diagnosis. The EDC is also capable of interacting directly with the **Vehicle Hardware**, via application-specific software routines (the embedded equivalent of the test procedures used in the Maintenance Environment).

The sensor data, test results and diagnostic conclusions are stored by the EDC in an **Embedded Data Store** (database, file or memory), for download at the end of the mission. Alternatively, this information may be transmitted during the mission, via the **Vehicle Telemetry** subsystem. In both cases, the data is ultimately stored in the **Maintenance**

Database. Diagnostics information received from the Embedded Environment via telemetry may be used for advance preparation of maintenance operations to be performed at the end of the mission. In addition, it may be used by the TestBase DC as a starting point for maintenance diagnosis. Besides a reduction of diagnostics time, this approach has the potential for providing better diagnostic performance by using sensor data captured during the actual occurrence of faults.

Embedded Diagnostic Controller

TestBase’s Integrated Development Environment enables the export of diagnostic information generated in XML for use in third party Custom Fault Managers. This functionality allows the control and execution of external “test procedures” according to sequence information. The Embedded Diagnostic controller is optimized for embedded operations supporting various embedded Operating Systems such as Linux, Unix, and Windows. By reducing the resource requirements in the embedded system memory utilization and computational loads are greatly improved. This capability allows for the interaction with the vehicle control software and third party embedded diagnostic software.

The Embedded Diagnostic Controller consists of common modules written in ANSII C++ which implement most of the functionality and is highly portable. Custom modules can be written for specific operating system or re-hosted and integrated into specific embedded applications. The Source Code is available to modify for custom deployment or TYX can modify the software based on the customer’s specific requirements.

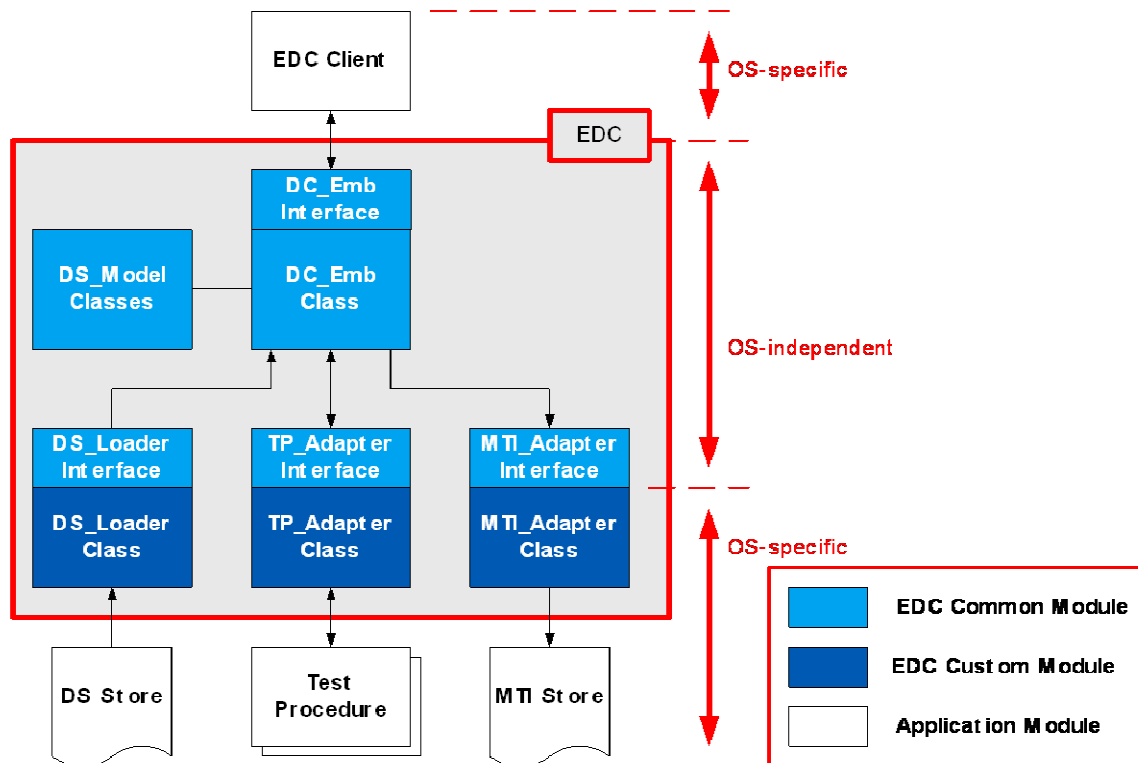


Figure 2 – General Architecture