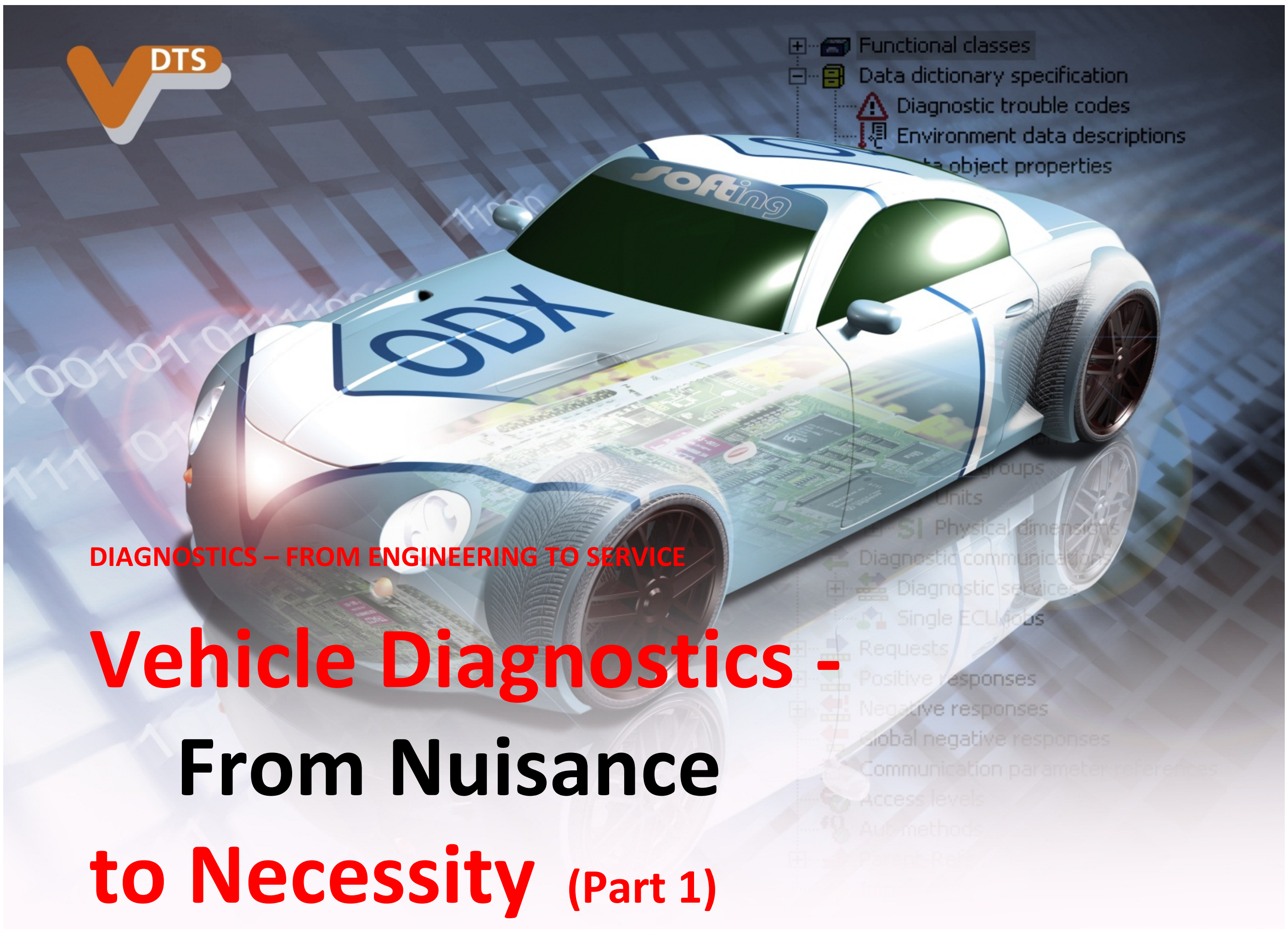
Originally published in the section Engineering Tools of the German technical magazine AUTOMOTIVE, issues 5-6 & 7-8 / 2011



DIAGNOSTICS – FROM ENGINEERING TO SERVICE

Vehicle Diagnostics - From Nuisance to Necessity (Part 1)

By Markus Steffelbauer, head of product management, Softing Automotive Electronics GmbH, Haar, Germany

For some years now, the vehicle diagnostics is undergoing a process of rapid change. The number of electronic control units in cars has been increasing steadily after a brief consolidation phase and is fast approaching 100 ECUs in a vehicle. Paralleling this, the complexity and the functions of an ECU and the vehicle network is also increasing significantly. Diagnostics thus constitutes an individual class of requirements for the vehicle per se; these must be implemented parallel to the intrinsic functions.

We can already find four bus systems in a medium class car and two to three more in the luxury class: all of these must be networked among each other. For diagnostics, there are many consequences: A mere limitation of diagnostics to the legally required range regarding exhaust standards has not been sufficient for a long time. A vehicle repair in a garage is no longer possible without diagnostics. The same goes for vehicle production. Diagnostics must be integrated early in the engineering process and must accompany it throughout to be able to control the complexity at the manufacturers but also at the involved ECU's suppliers. For the interaction of all partners to function, standards are indispensable.
They ensure that data and tools mesh at interfaces between ECU engineering, test facility, production and service organization and thus enable the concentration on intrinsic tasks. In the following, the underlying standards and the safeguarding of diagnostics in the engineering process will be comprehensively examined.
Today diagnostics plays a role throughout the entire value chain, from engineering through production to service. Decisive is that the diagnostic mechanisms - the communication of a tester with an ECU - are used today for a series of additional tasks (figure 1).

Engineering Area

In ECU engineering, diagnostics is used in the following steps. Basis is the diagnostic communication, with new ECUs usually the standard UDS (Unified Diagnostic Services). The classical diagnostics as a function in the ECU serves to identify unexpected behavior within the ECU and its environment, to benchmark and to record these in the fault memory. From there, the entries can be selected and processed through an external test system. In addition to the classical fault memory functionalities - including determining conditions for the fault entry (environmental conditions) and erasing the fault memory - the same basic functionality is also used today for related applications.

Examples are:

- Flash programming: interchange of code or data in the ECU
- Variant coding: adaptation of an ECU's behavior in accordance with legal regulations (for instance, daytime running light in Scandinavia) or commercial conditions ("Software as product")
- Starting routines: ECU functions are externally initiated and then executed autonomously, for instance, a self-test
- Measuring: readout of values determined by the ECU with the help of sensors

Diagnostics communication will thus always be used when a precise data and release process is necessary (flash programming, variant coding) or when it allows a simple and inexpensive access to the black-box ECU.
In the engineering area, diagnostics is used in numerous applications, whether they be in the development of communications, of ECU functions, of flash and variant coding functions or in HiL, system test stations or in the integration of ECUs. Subsequently, diagnostics is used diversely in vehicle tests.

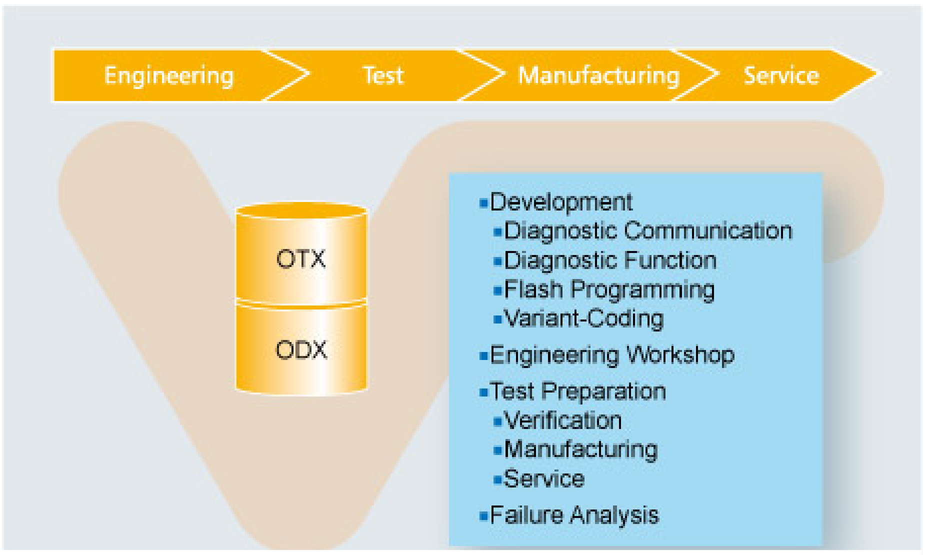


Figure 1: Application of diagnostics in the vehicle's lifecycle

Production

Precisely the simple access to the ECU allows a vast number of applications in production that would otherwise have been implemented only with more complex, cost-intensive solutions. The modern mechatronic systems can hardly undergo a test without access to the electronics. At the same time, they should be tested as early as possible. Also, a vehicle's variant coding, the key coding and initializing the vehicle immobilizer are carried out here. Ultimately, in the "end of line" test, a final control of vehicle functions and the deletion of the fault memory take place.
A special challenge in production is the differentiation between "real" and "systematic" faults. The latter arise from the fact that the ECUs keep numerous fault log entries that are produced because of the missing system environment at the time of installation. For example, faults are recorded when a signal is not available on the CAN bus because it should be sent from an ECU not yet constructed. The entry is, of course, correct. It does not refer, however, to a defective (sub)system.

After-Sales Service

The origins of diagnostics lie in after-sales service, more specifically in the rational insight that a vehicle currently based on 30 to 80 ECUs with which a vast number of vehicle functions are realized can no longer be repaired in a garage. In case the client reports unexpected symptoms, the garage technician must have the correct "wrench" at hand. The relevant tool is the service tester. It interconnects the reported symptoms with fault entries in the ECUs and measured values that can be determined parallel. Expert systems combined with the technician's expertise usually lead to a repair recommendation. Additionally, it is generally possible to program ECUs with new software version.
In all of the above-mentioned applications, ODX data are used. Except in test systems, they are also used for setting ECU parameters and for automatically generating test procedures. It is clear that the quality of the ODX data - alone but also in connection with the ECU - represents a decisive influencing variable for the quality at all. Apart from carrying out intensive reviews, this is achieved through complex data verification. The data verification is an important application of a modern engineering tester. Furthermore, it is used throughout the complete engineering process: as test environment for the important ECU functions of diagnostics communication, diagnostics function, flash programming, and variant coding. The engineering tester is also continually used as an inexpensive "debugger". In later phases, it plays a major role in vehicle inspections in testing centers, sometimes even in road tests of individual vehicle functions and in fault analysis, especially of communication problems. The engineering tester as a commissioning tool also plays a large role in test preparation. For the engineer, the combination of ODX data and ECU in conjunction with the D Server essentially constitutes a black box, which is difficult for him to put into operation. With the engineering tester, communication can be easily implemented and the correct behavior simulated.

On-Board Diagnostics (OBD)

On-Board Diagnostics is a special area of diagnostics since it is statutory. It requires of the vehicle manufacturer - originally only of passenger cars but now also of trucks - that all exhaust relevant systems continually undergo a self-test. Anything irregular must be immediately communicated to the driver through a warning light so that he/she can have it repaired. For California, this "optional" provision is in so far a minimization as the driver must pay high fines if he/she does not have the exhaust system repaired. This self-test is supported through the demand that the findings of the OBD can be read at any time with a simple diagnostics tester, the so-called scan tool. A functioning OBD as well as a functioning communication with the scan tools constitutes a market-access requirement for manufacturers. Correspondingly exact must the tests of these functionalities be. They are also not completed with the car registration: an accumulation of reports concerning vehicles already sold can quickly lead to recalls, which can be demanded by the executives.

Requirements for Diagnostics Systems

If a modern test system in the vehicle environment is examined, general blocks that always recur can be identified. Initially, it is decisive that usually engineering system and a runtime environment is present. The engineering system configures the runtime environment in which the test engineer configures the test run and the visualization for later usage. In many cases, test documentation has already been defined. The engineering system often works in an administrator mode. In the runtime environment, the tester's possibilities to intervene are frequently very restricted. The tester only has access to the application mode. The objective is for the tester to concentrate only on the test environment and that the tool works with few interactions.

The test systems have input data for the set-up and test implementation that parameterize the configurations and side conditions. Standardized input parameters are ODX data, required today, and increasingly OTX data (figure 2). Besides these, proprietary data are frequently used, for example logistical information such as release status and assembly instructions. These are very dependent upon processes, sometimes also upon infrastructure and cannot easily be standardized.

In addition, corresponding output data is produced. These can also be standardized or proprietary. In diagnostics today, contrary (as of now) to measurement technology, there is no standardized output format still. A typical output format is test documentation in which the successful and failed test cases are recorded. These can either be analyzed by specialists or played back into the system for later regression tests. Error statistics are also frequently produced with the test results.

Requirements for engineering testers

Especially for engineering testers, which are used throughout the complete process, a range of additional requirements must be stipulated. This is primarily due to the applications but also to the underlying processing layers. Particularly ODX data are processed through a so-called D Server. This puts the processed data at disposal. For the test system, though, it is hardly possible to determine what purpose the data have. That is, diagnostics services for communications controls mostly do not differ in practice at the interface from diagnostics services for measurements or for parameterizing an ECU. The diagnostics tester must support the user, though, through application-specific presentations that must be correspondingly configured. Only then will the presentation be connected with the appropriate diagnostics services.

Working with the fault memory poses perhaps the most important diagnostics function. It must be read but also cleared in order to be able to check the correct re-entry of a fault. Depending on the user, relevant are: the faults of the complete vehicle or of an ECU; the list of failures alone; and/or the failures with the saved environmental conditions and the status information. Precisely the status information is important for deeper analysis. It describes the errors more closely. As it is possible, for instance to ascertain whether a fault is permanently or only temporarily noted, whether it recurs sporadically or constantly. Further important information is the readiness flag. It shows whether the routine implemented to identify faults in the ECU has already run completely. For a whole series of faults, preconditions must be met, for example: the motor temperature must have reached its minimum value, given revolutions must have been exceeded, etc.

The demands on the visualization of flash programming are completely different. These are essentially limited to the selection of programming data (i.e. program code, family of characteristics etc.), a start button and ideally a progress indicator since appreciable programming time accrues with data of a number of megabytes. The actual challenge is in the sequence of the flash programming. Here an authentication is first implemented, switched to the programming session, the remaining bus participants told not to enter bus faults in the following and, finally, the bus communication between the ECUs ended in order to have enough bandwidth. Then the data transmission begins as high-performance as possible. At the conclusion, checksums are reviewed, the programmed ECU re-initialized and the bus communication re-activated. To finish, sometimes system adaptations must still be carried out.

By measurements, the main differentiations are between graphic representations of individual measurement values and textual representations. With graphic representations, the limitations are mainly set by usability; almost everything is possible from the kind of representation (pointer instrument, bar charts, plotter, …) through the colors used up to and including the display area (values range, scope with color change, …). Additionally, blocks of measurements frequently must be shown. These are selected from the ECU at once in order not to allow latencies in corresponding values to flow into the measuring via bus transmission.

When carrying out ECU routines, however, a very wide application spectrum is to be covered. One end is formed through ECU routines through which the feedback alone is given through actuators. Examples include the pointer test on the instrument cluster and the valves on the ECU for the anti-lock braking system (ABS), which opens and closes. For this kind of routine, the tester must actually only provide a start button and

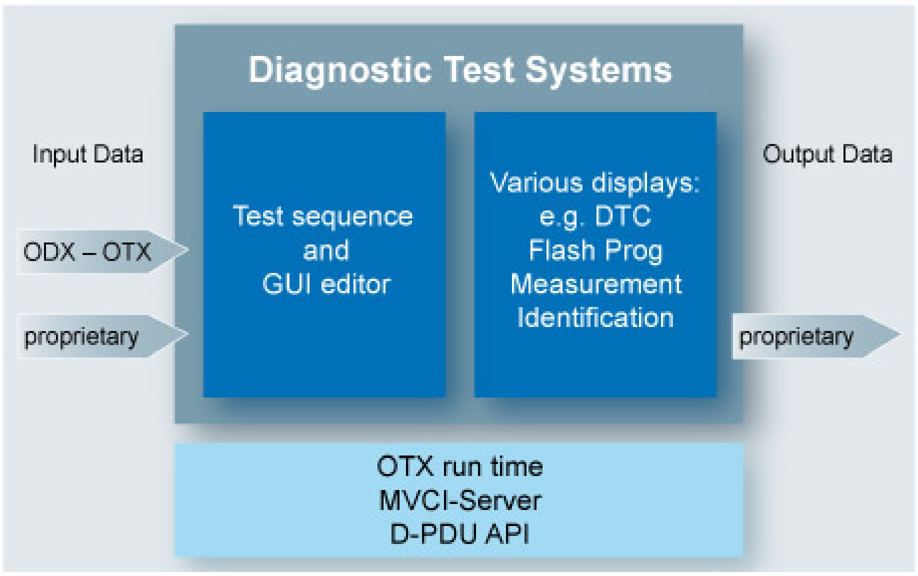

Diagnostic Test Systems
Input Data
ODX – OTX
proprietary
Test sequence and GUI editor
Various displays: e.g. DTC Flash Prog Measurement Identification
Output Data
proprietary
OTX run time MVCI-Server D-PDU API

Figure 2: General demands on diagnostic test systems

perhaps a description field for the test. The opposite end is represented by self-tests that, as a result, return a list of results via the bus. Here the tester must present the relevant parts of the ECU reply on the monitor. Similar is the important representation of the vehicle or ECU identification, which is made up from a list of information such as the VIN (vehicle identification number), hardware and software versions etc.

The use case data verification requires a much broader presentation scope. The access to the parameterization of diagnostics services and to the results must be so thorough that all methods used in automatic tests can be manually checked by the engineering tester. In the engineering tester, all values must be adjustable, even those that are actually not allowed, so that correct behavior on the periphery can be verified. With the inspection, it is especially important that not only values can be analyzed but also the structure of the results. This is a source of error for the access through a test automation that may not be underestimated. All tests, incidentally, are not automatable, for the plausibility can be investigated but not the correct conversion of hexadecimal (bus) data into symbolic data. This is a task for specialists that have a massive influence on the over-all quality of the system.

Variant Coding

With the help of variant coding, functions in the ECU are turned on and off. This has immense cost advantages because not every variant of the ECU software must be singly verified. The function "variant coding" must therefore be tested even more exactly. This is connected with a great deal of effort because the individual code bits are not independent. This applied to within an ECU and all the more for the total vehicle. This becomes apparent with the combination of motor and transmission. The engine performance is usually coded for an engine nowadays but must be combined with designated transmissions: on the one side, a transmission is not specified for a torque; on the other, frequently only an automatic transmission is offered for high output. The diagnostic tester must allow combining simply a variety of codes for the test. Simultaneously, setting faulty code combinations should be possible here in order to be able to check fault behavior.

Residual Bus Simulation

Residual bus simulation is actually not a subject for diagnostics. However, it has a great influence since the diagnostics does not function in many cases - especially when only a single ECU or a subnetwork should be implemented. Frequently, the diagnostics in an ECU is not active as long as certain signals are not transmitted to the bus. A typical example is the ignition detection, which is carried out from an ECU and then put to the disposal of all other ECUs as a signal on the CAN bus. The same holds for all signals that are necessary for a plausible diagnostics (for example, a speed signal). In any case, the engineering tester can simulate these signals by sending individual CAN messages on the bus cyclically. Logically, the individual signals should also be alterable since, for instance, the diagnostics behavior of low vehicle speeds to high can change.

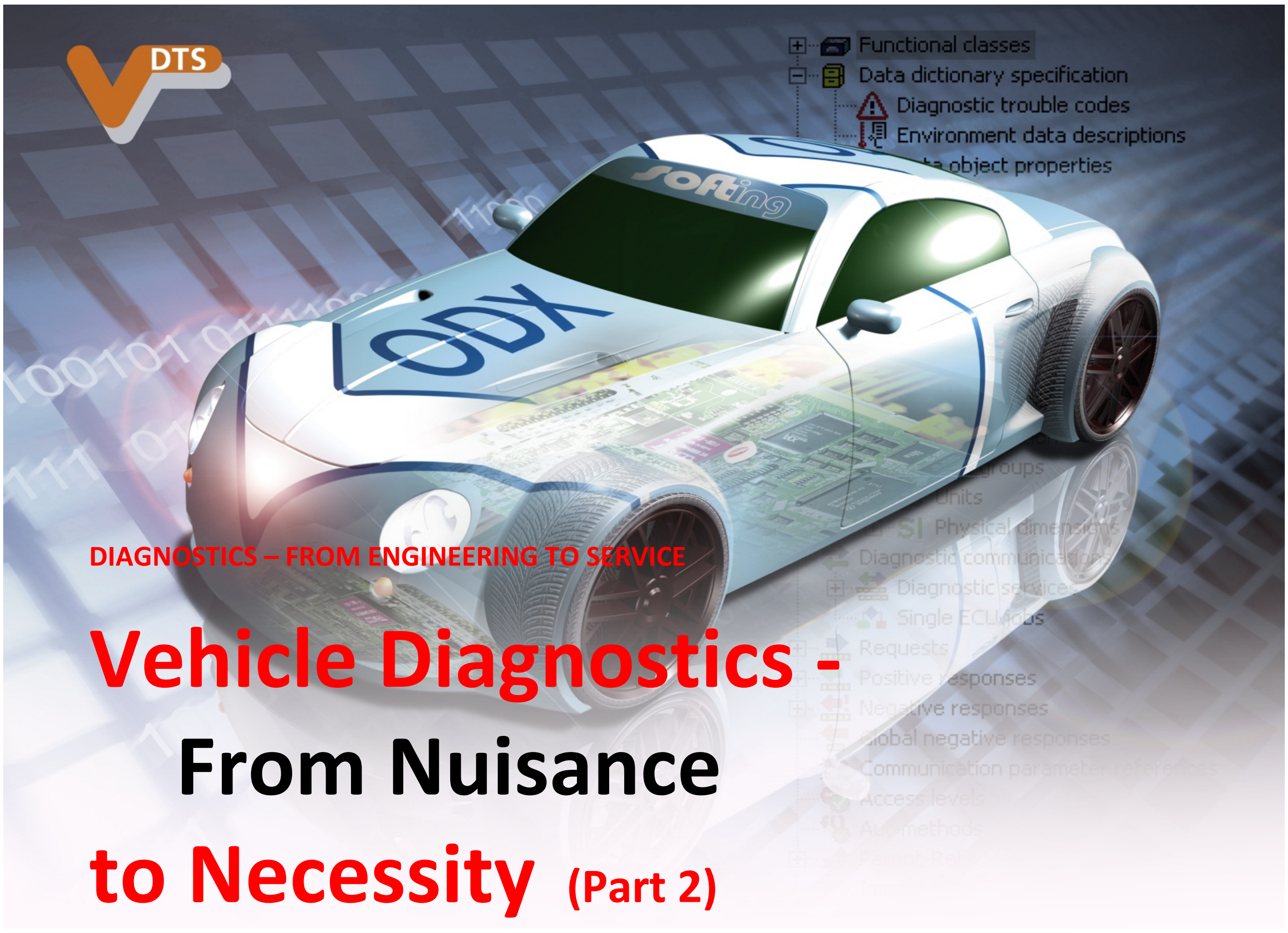
Also the use case analysis generates demands on the engineering tester. On the one hand, he contributes to data visualization. He must therefore prepare the data accordingly and represent the measurements on appropriate instruments, list the fault memory, and present the data for flash programming so that they can be programmed accurately into the ECU. On the other hand, he contributes to the commissioning and should therefore be able to enable an exact as possible access to ODX information and possible communication errors. If problems in the communication itself should arise, the bus communication too must be presented so that the particular problem can be efficiently located and documented.

For the test of OBD functions, the OBD modes - specialized diagnostic services - must always be presented in adequate forms. Roughly structured, they serve to read measurement values, to process the failure memory and to analyze test results. The addressing always takes place on the function OBD; therefore a number of ECUs can be concerned. The Vehicle Communication Interface (VCI) must control the functional addressing, and the diagnostics system must be able to handle a number of answers from different ECUs. The subfunction supported by the individual OBD modes will be reported by the ECUs in bit codes to the test system. It is especially important here that the tester allows requests that initially seem incorrect to be sent to the ECU since the scan tools can sometimes interpret specifications differently.

Which of the important standards must be observed?

Standards used today in diagnostics can be divided into three categories: protocols, data descriptions and application programming interfaces.

The oldest category is surely that of protocols. Originally introduced to comply with legislature's demands for a possibility to supervise exhaust emissions, these protocols fulfill innumerable other tasks today. The huge number of protocols specified for manufacturers has been be replaced today by standardized protocols; however, the importance of these earlier protocols for after-sales testers should not be underestimated: after all, the vehicles can be seen on the streets for many years.



DIAGNOSTICS – FROM ENGINEERING TO SERVICE

Vehicle Diagnostics - From Nuisance to Necessity (Part 2)

For some years now, the vehicle diagnostics is undergoing a process of rapid change. Diagnostics thus constitutes an individual class of requirements for the vehicle per se; these must be implemented parallel to the intrinsic functions. In the second part of this article the different protocols and programming interfaces will be shown. Finally, at the example of Softing's DTS-Monaco an universal engineering tester will be introduced which is based on D-PDU API, ODX, MCD-3D and OTX.

The communication protocol UDS

Today, the most common protocols in Europe are ISO 14765 (KWP2000 on CAN) and, based on the former, ISO 14229 (UDS - Unified Diagnostic Services). They share a common transportation protocol and describe in substance the same classes of diagnostic services. However, correlative to its name, the services for diverse applications, and correspondent to the earlier protocols used by the different manufacturers, were so universalized for UDS that a migration is relatively easy. In substance, the following service categories are described:

- Data reading
- Flash programming
- Failure memory
- ECU routines
- Input/Output control
- Control functions

Additionally, to enable savings in non-competitive areas, a kind of diagnostics operating system was engineered conjointly by OEMs and toolmakers a number of years ago within the framework of ASAM e.V. (Association for Standardization of Automation and Measuring Systems). It is a data-driven system; the data describe the abilities of the system in interaction with the corresponding ECU or vehicle. The necessity of such an action is easy to understand when we call to mind that a door ECU implements completely other functionalities and units than, for instance, an engine ECU.

Within ASAM e.V., an interface (ASAM MCD-3D) was defined to enable symbolic access to ECU and vehicle information and the data description (ASAM MCD-2D ODX – Open Diagnostic data eXchange) defined as exchange format between those involved in the diagnostics process. Both were also taken over as ISO standards. Within the ISO, two further standards were completed: D-PDU API as low-level application programming interface for a simple exchange of diagnostics interfaces and OTX (Open Test sequence eXchange format). The latter facilitates the exchange of diagnostics sequences, for instance between engineering and production.

D-PDU API – the VCI integration layer

The standard describes an API on hexadecimal level. The transportation protocol is treated completely transparently, i.e., for the higher application it is unimportant which protocol was used in D-PDU API. The data in the form of a byte stream are overhanded to the interface together with the addressing information and parameterized protocol details (e.g. timings). The implementation correspondent to the set protocol follows automatically. ECU replies will then be likewise reported back as byte stream with address information of the answering ECU. More far-reaching capabilities of the interface - such as inputs and outputs - that could be addressed can also be operated with D-PDU API. The description of interface capabilities, which deviates from product to product, is provided as an XML file. A change of the interface is thus very easy. For instance a test program in the engineering department can be run with a USB interface from Manufacturer A, and later in a test stand with a Wi-Fi interface from Manufacturer B. In practice, adaptations - albeit few - are necessary.

ODX – the exchange format for diagnostics data

The standard describes the data contents of diagnostics communications for ECUs built into vehicles in an XML file format. It thus places at disposal the documentation and the parameterization of the test system in a source. The file can thus be utilized equally from the specification phase to the use in production and after-sales services. Through the file, a conversion from hexadecimal to symbolic values takes place. For instance for an engine ECU there is the description of a diagnostics service Reading Revolutions. The corresponding hexadecimal values to send by way of D-PDU API can be determined from the ODX data file. From the answer, the hexadecimal value can be determined, which will be converted to "1900 rpm". Besides this communications information, in ODX the protocol-parameterization, irregularity list of a vehicle in a universalized form, flash programming and variant coding data are also described.

One of the main goals of standardization was the creation of a standard that is machine-readable and has long-term stability (use of XML), that supports the engineering process and is redundant-free as much as possible (derivation concept).

The derivation concept is based on the idea that a large amount of information for a designated ECU can be given that remains constant over the entire life cycle. This information is described in the so-called Base Variant. The individual variants of an ECU, described by the identification string or a software-version, for example, differ only slightly. The main part of the information can thus be inherited from the basis variant for a variant; in the ECU variant. Only the Delta then needs to be described. This can be added, or superfluous information can be omitted. Moreover, information defined differently for ECU variants than for Base Variant can be overwritten, similar to an object-oriented programming language. In addition to these two levels, Base Variant and ECU Variant, there are two further levels: the protocol describes services and parameterizations that are given through the diagnostics protocol and functions thus as a "blueprint" for the ECUs derived from it. The Functional Group level, finally, enables the description of the functional addressing through which several ECUs of a logical group can be addressed over a common address. The best-known example is OBD functionality, which summarizes all emissions-relevant ECUs and queries them together.

With OEM, data for a production series are usually summarized in an ODX data file. The cumulative amount of data can be relatively large because the database includes following information:

- The protocols used in the vehicle
- The information on the ECUs accessible through functional addressing
- All ECUs used in the vehicle, thus the optional (gearbox control device for automatic transmission) as well as the alternative ECUs (engine control devices for 4 and 6 cylinder diesel and for gasoline engines)
- The variants throughout the vehicle's production period

Concrete ODX databases currently reach the 100 megabyte limit. To simplify data exchange, the PDX format (packed ODX) was standardized in addition to ODX. This involves an indexed ZIP data file.

ASAM MCD-3D – the diagnostics operating system

The standard describes an API for access to ECUs with help of ODX data. The relevant runtime system is usually called a D Server or a MCVI Server. In the diagnostics process, it can be used in many different applications - engineering, production, after-sales services - and thus guarantees a consistent usage of the ODX data. This is especially possible because reference implementations for use with various programming languages have been provided for COM/DCOM, JAVA and C++ and are available commercially.

Access occurs on a symbolic level, i.e., an ECU under its name "engine ECU" is selected and subsequently diagnostics services can be carried out for this ECU. An example is the above described service Reading Revolutions that is processed by the runtime engine in the D Server and is sent over D-PDU API to the ECU. The answer is also processed over the ODX data and the result allocated to the API as "1900 rpm".

To improve performance especially in production, in principle as many queries as wished can be processes parallel to one another. The limitations mainly result from the bus load of the (normally) utilized CAN bus and the capability of the gateway ECU to distribute queries sensibly among the different busses.

Currently, ODX data are processed in real D Servers with the help of a runtime format. This has, on the one hand, performance reasons - in a binary format, the data can be packed and optimized for access - and, on the other hand, security reasons. Especially in after-sales services, the data are basically unprotected on the service testers and can be changed and taken by those interested. Binary data are easy to encode and to secure against access so that increased security has clearly been achieved.

OTX – the interchange format for diagnostic sequences

The standard describes an interchange format for diagnostic test sequences. Here, too - similar to ODX - the machine-readability and long-term availability were the main requirements. The idea behind the standard is to create a possibility to give a formal description of basic sequences between test system and ECU as early as during the specification phase. These sequences can be further used and specialized during the process: it is no longer necessary to start from scratch and to copy a basic sequence from a written specification. Through machine-readability, a graphical representation of sequences will also be supported, for instance as flowchart, which offers engineers in engineering, verification, production and after-sales services a universal basis for their discussions.

The standard OTX is subdivided into a number of areas: The core comprises a programming language with the typical elements: variables, solutions/ instructions and operations (grinding, bifurcation, benchmarking, …). It is not related to diagnostics and can basically be used for a variety of tasks. Standardized libraries are available for special tasks. In addition to diagnostics communication (connection to the D Server), especially string operation, size handling, presentation functions (HMI) and internationalization in order to be able to offer versions of sequences in different languages for the different markets. The standardized libraries use a general add-on mechanism. This can also be used for non-standardized add-ons, whereby any test system in the sequence is integrable. Typical examples are HiL systems, simulations and measuring.

OTX supports the engineering process through various mechanisms. First, it allows a mere specification view for early process phases. Here, the idea of the sequence is described, without furnishing a concrete implementation of the sequence. Thus, it can, for example, already sketch the communication with an ECU without real ODX-files in which real diagnostics services are described. A loadable OTX-sequence is not thereby created. This will be made up for later in the implementation of sequences. To support the variant diversity similar to ODX without having to release a sequence completely every time, special mechanisms were also realized.

In contrast to ODX, for OTX there is no standardized API for a state machine. As interchange format, the tool production can process the format in an interpreter, compile it in a machine format or to import it into an already existing test system and process it there further.

Engineering tester—an example

DTS-Monaco is an example for an engineering tester. It was based on the standards D-PDU API, ODX, MCD-3D and OTX and thus has a command of all relevant standards (figure 3). DTS-Monaco consists of only a few main components and can be expanded for specific uses: the framework, the HMI controls and the runtimes for the standards OTX and ODX. ODX is processed through the D Server DTS-COS. It places at disposal the standardized API complying with ASAM MCD-3D, offers in addition, though, several other extensions that simplify the use in test automations and enable functions in the engineering tester that go beyond the standard. For OTX a runtime interpreter was integrated. The framework provides the basis functionalities that are necessary for diagnostics. Besides the link to the runtime systems, this is especially the tool configuration, which can be saved and reloaded. The role administration is also integrated in the framework. Monaco supports the roles “administrator” and “user”. The administrator can author configurations, save and then provide them to the user. This pertains to the surface configurations as well as to OTX sequences. These can be provided by the administrator in the tool and adapted. The user can only load these pre-configurations in the tool and use them. Changes are only partially possible. This ensures that every user utilizes only those functions on the vehicle for which he/she has been trained.

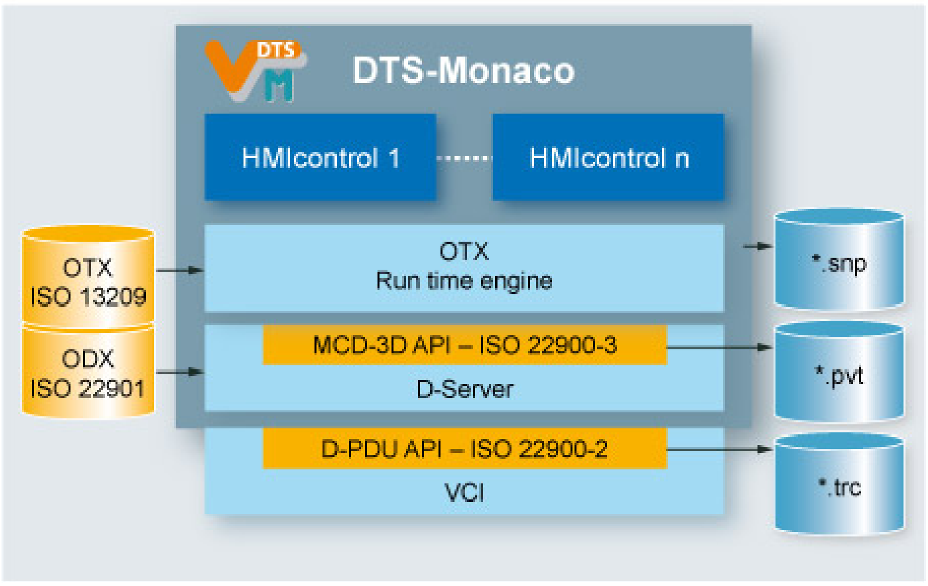


Figure 3: Softing’s DTS-Monaco is an example of an engineering tester. Any protocol can be used, if there are a D-PDU API interface and suitable ODX data.

HMI controls run within the framework as components. HMI controls translate the communications analysis, provided through the D Server and the OTX runtime, into the user view, expected by the user for his/her specific tasks. This can be a fault memory list or a measuring instrument. Every HMI control brings its own configuration. There are both common and specific properties. With the failure memory, this would be whether status information should be shown, for example, and with the measuring instrument if it should be shown as pointer instrument or as bar graph.

Implementation as components is a simple possibility to discuss additional uses with a client, to implement and to install later or in a specific setup. The advantage is that this component can be independently tested and the system integrity is not influenced at all. These add-ons are especially important when a link to a logistics system is required. This is, of course, user-specific, thus can be easily implemented in a product but should not be visible to every user and thus must be installed independently.

Data supply

ODX and OTX runtime environments are the basis for DTS-Monaco. The most important data in the system are thus OTX and ODX data. Access to ODX data takes place - whether through HMI controls or OTX - via names (Reading Revolutions). These ODX names must be administered with the configurations and contiguously with the OTX sequences. This occurs through projects.

The OTX data are processed directly in XML format. This simplifies handling because here data either have been taken from a specification phase and adapted or have been directly authored for the applications of the engineering tester. Here, data conversion would only be cumbersome in operations. For ODX data, this does not follow. These are authored in a small area of applications; in the majority of applications, though, they are subject to version administration and should only be altered under regulated guidelines. Here, therefore, data protection is advisable. In DTS-Monaco, a conversion takes place in a format that is encoded as well as password-protected. It cannot be opened even with a database editor if the password is not known. The data format supports various administrator processes: not only can data be summarized in a data file in an ODX database - typically binary data for a production series - but also each ECU with its variants can be saved in its own data file. Both procedures have advantages and disadvantages. Apart from these data, further data types can be administered in a project, for example simulations datasets, the CAN matrix or filter datasets.

Applications with HMI controls

One of the central HMI controls is the Diagnostic Services Control. It covers a large extent of the applications: data verification and analysis. This occurs through a three-part window: in the first part, the diagnostics services of the database can be "browsed" in a tree, sorted according to ECUs and to functional classes. The selected diagnostics service can be parameterized in the second part. This can be done symbolically by setting the request parameter "variable" to "revolutions" or directly by changing the hexadecimal values in the byte stream. This way, "not OK" values, which are not attainable symbolically, can be set for test purposes. In the third part of the window, results are presented, whereby it is configurable whether the results only are presented or also the structure of the results. For quick access, buttons can additional be configured so that access to the diagnostics services does not need a lengthy search with the browser. Two different methods are available to represent measured values: blocks of measurements can be textually represented in a list, which allows a very compact overview of the values. The repetition rate is adjustable, and access only with manual demand is also possible in order to keep the bus load low. Instruments can also be configured; the variability of these representation possibilities is considerable. Incidentally, such instruments can also be used with knobs or switches to alter values in the ECU. Representing a number of measurement values over time is additionally possible.

For flash programming, an individual HMI control is provided. This covers two applications: it can be used for engineering the function flash programming as well as for simple executions, for instance to update program versions or to swap datasets. In the former case, it is possible to put individual partial functions in the ECU into operation using a five-step process: initialization, compatibility check, validation and finalization. Thereby the single test runs are distinctly faster. In the latter case, swapping datasets, the data package to be programmed is only chosen and then the ECU is programmed at the touch of a button. In both cases, the programming progress is shown with a progress indicator.

Handling failure memory and identification is summarized in an HMI control called Quicktest. The Quicktest is frequently used especially in the prototype workshop. The ECU identifications are read and then the failure memories of all recognized ECUs. Users themselves can configure whether the surrounding conditions are also retrieved. The challenge is that the pilot plant frequently programs unreleased software versions in the ECUs so that variants cannot be recognized. Just as frequently, the valid ODX data are not yet available; in this case, too, no variants can be recognized and the diagnostics must take place using the basic variants. To be able to execute the common application Quick test with high performance, all ECUs possible should be addressed in parallel. This enables performance gains of factor 4 in practice. However, all ECUs cannot be addressed in parallel. First, the alternative ECUs must be recognized since only the ECU for the 4-cylinder diesel motor or for the 6-cylinder gasoline engine, for example, could be present. It is just as important to bear in mind, however, that - corresponding to the bus architecture - ECUs behind gateways often can be addressed only if the gateway itself has been addressed already. ECUs which need to be addressed sequentially have to be configured by a specialist.

Furthermore, numerous other views of diagnostics information are available that will not be discussed here.

Test Documentation

With DTS-Monaco, there are various available possibilities to document the tests performed. The symbolic trace records all send and receive information that is exchanged between tester and ECU(s). Thereby, values interpreted via ODX data are saved on the one hand; on the other, though, also the data sent and received on the hexadecimal level. Thus, a complete test is also subsequently representable and can be documented. The file is delivered as an XML data set so that conversions into other formats or filtering to necessary representations for particular customer processes are unproblematic.

Parallel to this, on the bus-communication level, a trace can be written down. This trace, usually a CAN trace, also allows abnormalities in the protocol to be recognized and be documented.

Conclusion

Standards have not only gained acceptance in the area of diagnostics; with increasing depths of application, there are even broader areas that are being standardized. Diagnostics protocols were the beginning from which producers soon no longer wanted to pay for implementing their special solutions with the increasing number of ECUs. Data formats and APIs to integrate diagnostic solutions in the most different test systems were a logical consequence. Independent of the fact that single-source is basically suitable to reduce costs; quality is also increased through the wide use of relevant standards. That is to say, as soon as diagnostics is easily integrated, nothing is to be said against also introducing its powerful methods in many areas and thus to reach a much larger test width and depth.

Quality, however, is the central argument for customer retention. Seeing how nervous producers react to recalls, it is clear that the message has been understood. Seeing how meticulously customers study emergency road service (breakdown) statistics and enlist them in their purchase decisions, it is clear why. When a car has a defect in early stages despite all efforts, though, a customer must be able to drive away from the garage after a stay as short as possible, satisfied. Diagnostics is the key to this.

The savings is very apparent if the costs for special testing mechanisms in engineering and production would be calculated. Naturally, special sensors and actuators for testing purposes can be built to carry out tests. Naturally, additional tools can be held ready for these tests. Faster and cheaper - and frequently sufficiently good - it is possible with the diagnostics testers available anyway over the installed mechanisms in the vehicle. Consequently, it is done exactly that way.

Markus Steffelbauer is head of product management at Softing Automotive Electronics. He takes responsibility for development and marketing of all hardware and software products and is member of several international committees.