

# Automatic Evaluation of ECU Software Tests

**Mirko Conrad**

DaimlerChrysler AG, Research and Technology, Berlin, Germany

**Sadegh Sadeghipour, Hans-Werner Wiesbrock**

IT Power Consultants, Berlin, Germany

Copyright © 2005 Society of Automotive Engineers, Inc.

## ABSTRACT

When testing electronic control unit (ECU) software, test stimuli as well as test results are time-dependent signals. In order to effectively achieve high quality testing during development, the approved results of former tests serve as reference data for regression and back-to-back tests. The evaluation of those tests leads to a new task, the trustworthy comparison of time-dependent signals. To carry out this task we developed new concepts for signal comparisons and a tool, called *MEval*, for automating the test evaluation. Given a reference and a current result signal as inputs *MEval* evaluates their similarity. A new variant of a dynamic time warping algorithm, called *difference-matrix preprocessing*, allows an independent assessment of amplitude deviation and possible time shifts.

Using the automatic test evaluation we defined an integrated test process for the model-based development of ECU software. In this process, regression and back-to-back-tests are performed, automatically evaluated and released. We thus gain reproducible and trustworthy regression and back-to-back test evaluations leading to higher quality.

## INTRODUCTION

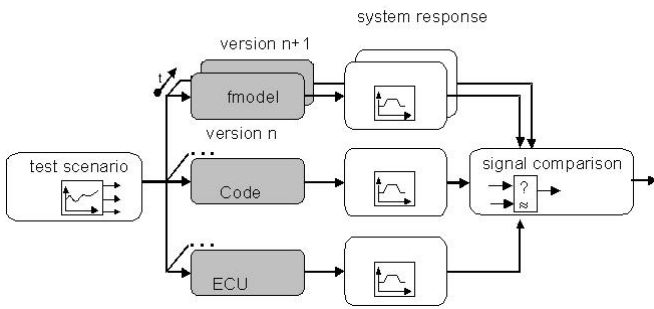
The number of embedded controls in the automotive domain is greatly increasing, causing a concomitant rise in the need for high quality software development. A main factor to improve quality is to achieve a superior standard of testing. Consequently, test effort nowadays often exceeds 30% of development expenses, and even more for critical systems. Automating some of the testing activities is a promising way of saving costs and gaining higher quality.

The evaluation of regression and back-to-back tests is a time-intensive activity. Back-to-back tests establish the equivalence of different representations of the test object, e.g. between the model and the program code generated. Regression tests ensure that modifications made to the test object respect the required functionalities.

When doing this, the approved results from former tests serve as reference data. The evaluation of those tests is based on a comparison of both outcomes. For ECU software, both test stimuli and test results are time-dependent signals. This leads to a new task, the trustworthy comparison of time-dependent (result) signals. Although efficient tools for the support and automation of testing activities are already on the market and have been employed successfully, there is still no automated test evaluation for control unit software, i.e. an automatic comparison of time-dependent signals. Testers still have to judge whether or not the test has been passed by 'looking closely' into signal waveforms. Moreover, the test evaluation is not always reproducible due to the lack of operability.

Since the mid 1990s, traditional, document-based software development in the automotive industry has increasingly been displaced by model-based development [HaRe01] [Bro03]. This opens up the possibility of the automated test evaluation of ECU software. Model-based development is characterized by the integrated deployment of executable models for specification, design and implementation, using commercial modeling and simulation environments such as MATLAB/Simlink/Stateflow [MATLAB] or ASCET-SD [ASCET-SD]. These tools use block diagrams and extended state machines as modeling notations.

Very early in the development procedure an executable model of the control software (functional model) is created, which can be simulated as well as tested. This executable model is used throughout the downstream development process and forms the 'blueprint' for the automatic or manual coding of the embedded software and its integration into the electronic control unit. Consequently, the test scenarios set up for testing the functional model can be deployed to test the later model versions, the program code generated from the model, or the ECU with the integrated software [CFS04]. Therefore, regression and back-to-back tests can and should be accomplished in different development phases (Figure 1).



**Figure 1: Different test phases use the same test scenarios during model-based development**

Much of the effort involved in these accompanying tests is taken up by test evaluation – an activity which grows rapidly in proportion to the number of test scenarios which have to be executed. However, when performing regression and back-to-back tests, in which the results of existing tests are used as reference data, test evaluation can be automated. By automating test evaluation, more extensive tests can be carried out, making it possible for the tester to concentrate on defining those test cases which are important and necessary. Defined comparison criteria lead to reproducible test evaluations and also have the advantage that the assessment no longer depends on the tester's subjective perception.

### SPECIFIC FEATURES OF THE TEST EVALUATION OF ECU SOFTWARE

Although automating the evaluation of regression and back-to-back tests of ECU software promises great advantages, the required technology does not yet exist. This is due to specific problems with the test evaluation for control unit software. In contrast to database applications and administrative software, control unit software characteristically interacts continually with the environment via sensors and actuators, processes time-dependent sensor signals and creates time-dependent outputs. Time-dependent test scenarios are derived from the software requirements or the system requirements and the test runs then deliver time-dependent signal courses. Consequently, time-dependent signal outputs have to be compared and evaluated during testing. The task of the test evaluation is, accordingly, signal comparisons between current output signals  $o'(t)$ , i.e. from the system reaction during the current test, and reference signals  $o(t)$ , i.e. from the system reaction of a different form of representation or an older version of the test object.

Manual test evaluation can be performed by means of a visual assessment of the signal plot and, if necessary, by comparing the signal plots from the test and reference signals. Depending on the context, the check will not necessarily focus on absolute equality but rather on a certain similarity: signals are considered to be similar, if the signal plots are sufficiently close together. This kind of visual check is, however, not only highly subjective

but also, depending on the number and size of the signals and tests, highly prone to error and, at the least, is very time-consuming. It also requires experienced testers.

Automated test evaluation aims to automatically assess whether the test object produces the same reaction as former approved versions or other approved representations of the test object. Accordingly, the test evaluation judges whether or not a reference signal  $o(t)$  and a current output signal  $o'(t)$  are similar. If no similarity between the signals is automatically recognizable, error types and flaws which have occurred are detected and localized as far as possible. In this case, significant characteristic values or compact visual representations should support the human tester in his/her search for errors.

The domain of automotive control unit development under consideration has certain boundary conditions which make it possible to restrict the general problem of signal comparison. When comparing signal pairs consisting of a reference signal  $o(t)$  and a current output signal  $o'(t)$ , the existence of the following characteristics can be assumed:

- $o(t)$  and  $o'(t)$  are time-discrete.
- $o(t)$  and  $o'(t)$  are the same length.
- The data types (value ranges) of  $o(t)$  and  $o'(t)$  are the same. Both quasi-continuous and n-ary or binary signals may occur.
- The signal courses are usually a periodic.
- The sequence of certain features (spikes, zero crossings,...) in the signal courses is significant.
- Deviations in both time and amplitude are of interest; the relevance of different error types differs, however.

### EXISTING SIGNAL COMPARISON METHODS

Besides manual, optical comparison procedures ('looking closely'), a few automated comparison procedures are used in industrial practice to compare signals as part of test evaluation. The procedures are usually based on simple difference calculations or standard signal processing procedures [DSS+01] [RWS+01] [CS03]. These signal comparison procedures, which were predominantly developed in the area of communication technology, generally do not take the conditions listed in the last section sufficiently into account and therefore do not lead to satisfactory evaluation results.

### DIFFERENCE PROCEDURES

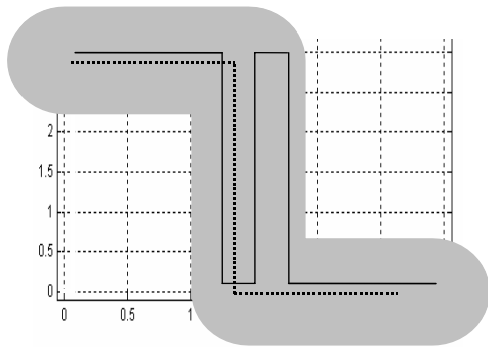
Difference procedures determine and, if necessary, suitably normalize the amplitude difference between the output signal and reference signal for each point in time. Absolute, relative and slope-dependent differences are often used. If the maximum of the respective difference vector is less than an allowable tolerance, the signals

are classified as similar. If the opposite is the case, the signals are considered to be dissimilar.

Methods based on difference calculations are amongst those comparison procedures which are easy to implement. They make it possible to deal with amplitude errors in a graduated way and can be adapted to special boundary conditions (e.g. signals with strong amplitude or slope differences) by using different weight functions adjusted by proper parameters. Applied alone, they are, however, not suitable for managing cases of local time displacement. In these procedures, 'quantization noise', which is a consequence of the transition from floating-point arithmetic to fixed-point arithmetic, leads to classification of the signals as dissimilar, despite the fact that corresponding signal pairs are seen as similar in the application context [CFP03].

### TUBULAR DIFFERENCE METHOD

A special window technique, termed tubular difference, can be regarded as an extension to the difference procedures with respect to better robustness when dealing with slight time deviations. An appropriate tube with a predetermined diameter is placed around the reference signal. If the current output signal is located inside this tube, both signals should be considered similar. This procedure is very widespread but is not always suitable for trustworthy test evaluation. Figure 2 shows an example which occurred during a pilot study concerning engine control software. The dotted line represents the reference signal and the solid line the current output signal. Due to a mistake, the output signal momentarily jumped back onto the flag set position. As the entire test scenario was of long duration, the 'looking closely' method overlooked this error. However, the tolerance tube procedure was also unable to detect this error.



**Figure 2: The tubular difference method for signal comparison**

### STATISTICAL METHODS

Discrete cross correlation and correlation coefficient methods are common, statistical procedures for judging the similarity of signals.

In communication technology, the cross correlation method is normally used to detect temporal shifts between (noisy) signals. It is suitable for identifying global temporal shifts between simple signals but is not able to handle more complex signals [RWS+01]. High, temporally tightly limited amplitude deviations can remain undetected when using the correlation coefficient procedure.

### ECU SIGNAL COMPARISON REQUIREMENTS

The fundamental differences between the objectives of message recognition and the objectives pursued when processing control unit signals, mean that other procedures are necessary for the assessment of ECU signal courses when automating test evaluation. The automatic test evaluation of control unit software has to perform the following tasks:

- Discretization and rounding errors must be recognized and tolerated in a controlled way. Possible errors must be detected.
- Systematic errors such as, for example, exchanged signal characteristics and deviations in amplitude which are too large, should be recognized.

The results should be then presented to the tester in a compact form in order to support him/her in his/her assessment. In particular, this means:

- the localization of flaws,
  - compact visual representations, and
  - significant characteristic values
- must be provided by the tool.

### A NEW APPROACH AND ALGORITHM

In order to meet the requirements of an automated test evaluation of ECU software, as described in the previous section, the authors developed an innovative multi-stage generic procedure for signal comparison as well as a new algorithm, called *difference-matrix preprocessing* [WCF+02], which is a new variant of a dynamic time warping algorithm [RJ93].

The generic, multi-stage signal comparison procedure first preprocesses the signals in an appropriate way. They are then compared to see whether or not they are similar according to elementary criteria. In this way, temporal shifts and deviations in amplitude are identified and judged separately. Both components can be instantiated and parameterized with different algorithms. In addition, standard components and default parameterizations were identified for an automatic signal comparison.

### PREPROCESSING USING A DIFFERENCE-MATRIX

When using this preprocessing method, the current output signal is adjusted to the reference signal by stretching or compressing it temporally. The extent to which the

signal has to be stretched or compressed indicates the temporal displacement.

The basic idea is as follows: suppose we have to investigate the similarity between two signals which had been shifted in time w.r.t. each other. We would shift one of the signals towards the other and would examine the remaining deviation. Moving one of the signals towards the other corresponds, mathematically formulated, to a special reparametrization (temporal reordering) of the signal. We must now look for a general reparametrization, so that the signal lies on top of the other signal as well as possible. Thus, possible local shifts and compressions are taken into account.

The task is therefore to find a suitable reparametrization (temporal reordering)  $\gamma: [1, t_{max}] \rightarrow [1, t_{max}]$  of the current output signal  $o'(t)$  so that it approximates the reference signal  $o(t)$  as well as possible (best match), i.e. so that  $o'(\gamma(t)) \approx o(t)$  is valid. When doing this, the temporal order of the sample points must be respected.

To get an idea of this kind of reparametrization think of the signal values as sitting on a spring, i.e. the time axis of the test signal is replaced by a spring which can be stressed or compressed.

The individual sample points are now moved along their time axis, so that the signal courses of the reference and the output signal are covered as well as possible (Figure 3). The resulting spring tensions then define a suitable reparametrization.

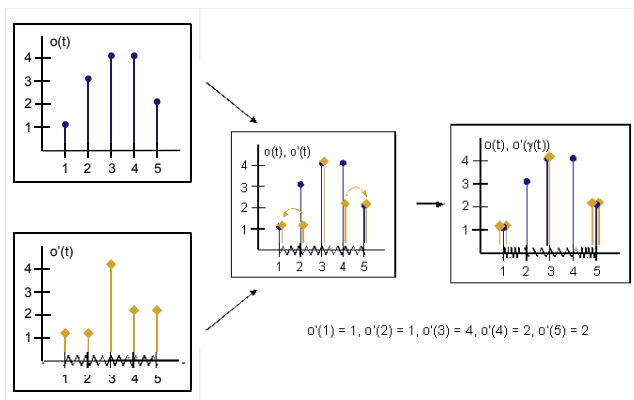


Figure 3: Reparameterization of test signal

The starting point for the calculation of this kind of reparametrization is the so-called difference-matrix  $DiffM(t_i, t_j) = |o(t_i) - o'(t_j)|$ , whose entries consist of the absolute values of the signal differences. Paths are now looked for within the difference-matrix, which go downwards from the upper-left corner, one row at a time, whereby they are able to move any number of steps to the right. Figure 4 shows a difference-matrix for the signal pair from Figure 3.

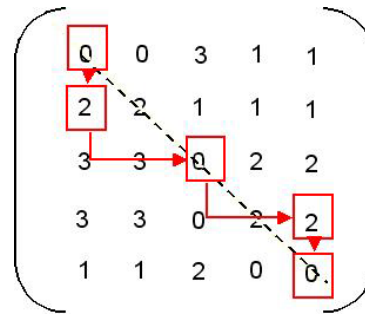


Figure 4: A path through the difference-matrix

The length of this kind of path is defined as the sum of all the matrix entries and terms which sanction deviation from the main diagonal. Looking at the vertical steps the path takes, we recognize the spring tension found for the best match. The shortest path defines a proper reparametrization of the current output signal. Using this reparametrization for the current output signal the deviation in amplitude will usually be smaller than before. The example shown in Figure 4 defines the following reparametrization.

$$\gamma: \{t_1 \mapsto t_1, t_2 \mapsto t_1, t_3 \mapsto t_3, t_4 \mapsto t_5, t_5 \mapsto t_5\}$$

The analysis of the computed reparametrization enables us to detect possible time shifts.

Looking at the example in Figure 3, the best match for mapping the current output signal onto the reference signal would lead to a situation where to all test signal values one finds a corresponding reference signal value of exact equal value. Therefore accepting the found reparametrization would imply classifying the test signal as equal to the reference signal. But, upon closer inspection, this would be an incorrect classification. The reference signal at time point 2 has value 3 which never could correspond to a test signal value of the same value. The remaining difference would always be at least 1. For a trustworthy comparison we have to symmetrize the preprocessing. In this variant, we apply difference-matrix preprocessing twice, exchanging the roles of the reference and output signals and taking the maximal deviation remaining in both applications. In this way, we can assess time shifts but also detect short-time deviations in amplitudes (peaks).

For an example of difference-matrix preprocessing consider the signals shown in Figure 5. At first sight one might classify the two signals as pretty different. Their amplitudes differ by more than 1. The maximal deviation appears at  $t \approx 1.7$  (Figure 6).

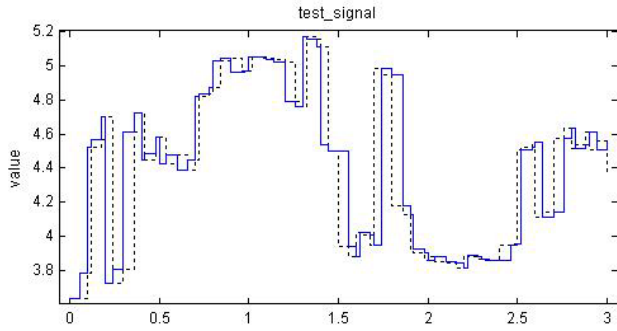


Figure 5: Time shifted reference and output signals

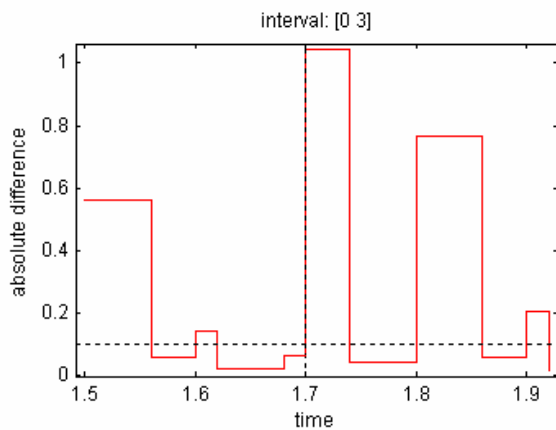


Figure 6: Comparison result without preprocessing

However, the application of difference-matrix preprocessing leads to the following result (Figure 7).

**Preprocessing:** difference matrix  
successfully passed  
(reparametrization deviation < 2)

**Comparison:** absolute difference

**Result:**  
successfully passed  
(difference < 0.1, max. deviation: 0.037236)

Figure 7: Comparison result after applying difference-matrix preprocessing

This can be explained when looking at the reparametrization calculated (Figure 8). Accepting the proposed reparametrization, the two signals can hardly be considered different. Figure 8 shows that in the time interval ]0, 1.5[ the output signal seems to be time shifted forward by one time step, i.e. it comes 1 time step earlier than the reference signal. In the time interval ]1.5, 3[ it seems to be shifted to one time step later than the respective reference signal.

This example shows that slight time shifts may lead to rather different looking signals but, due to system inertia, these may not lead to different system behaviour.

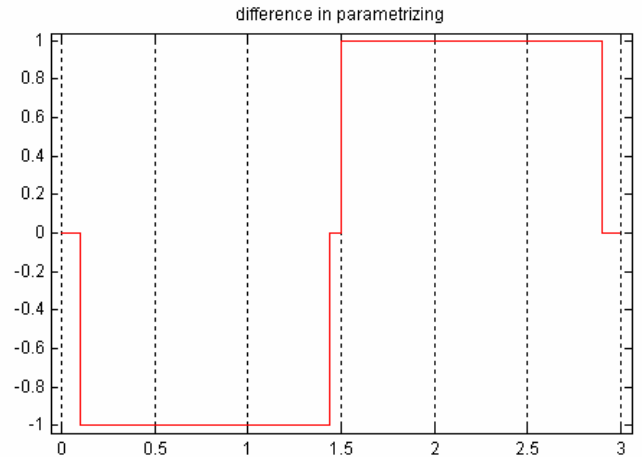


Figure 8: Reparametrization deviation of both signals

## COMPARISON OF SIGNALS

If the time deviations discovered by the preprocessing component do not cross a given threshold an assessment is then made as to the similarity amongst the reparameterized signals using difference procedures.

Among the most widely-used methods for signal comparison are relative and slope-dependent differences. With the aim of relativizing deviations in the case of large amplitudes or slopes, these methods lend disproportionate weight to areas with small amplitudes or slopes. In the field of zero points or flat regions this already leads to unacceptably high differences with the smallest of amplitude deviations. Therefore, the methods mentioned are modified by means of the following formulas:

- Absolute difference

$$absDiff(t_i) = |o(t_i) - o'(t_i)|$$

- Modified relative difference

$$\begin{aligned} modRelDiff(t_i) &= \min(relDiff(t_i), absDiff(t_i)) \\ &= \frac{|o(t_i) - o'(t_i)|}{\max(\sqrt{|o(t_i)|} \cdot |o'(t_i)|, 1)} \end{aligned}$$

- Modified slope-dependent difference

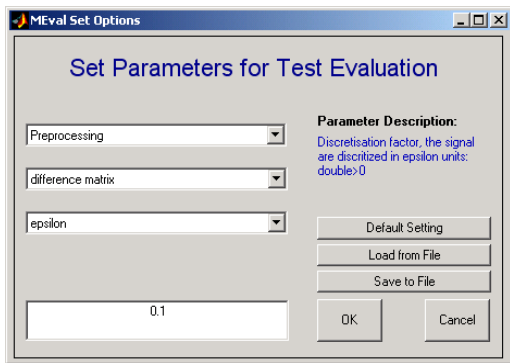
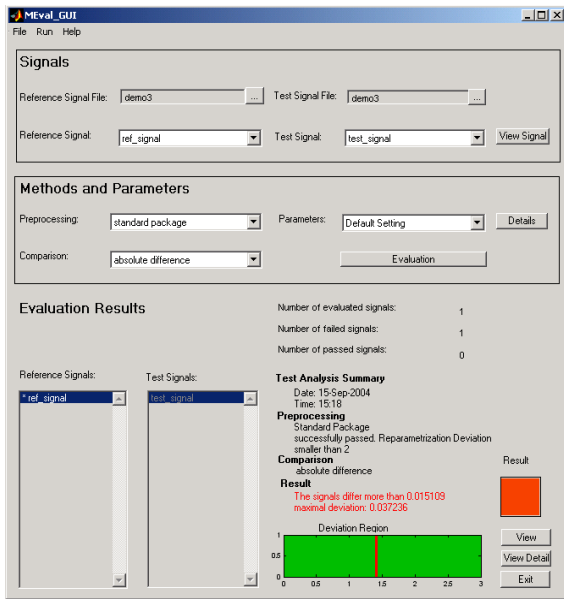
$$\begin{aligned} modSlopeDiff(t_i) &= \min(slopeDiff(t_i), absDiff(t_i)) \\ &= \frac{|o(t_i) - o'(t_i)|}{\max\left(\sqrt{\left|\frac{1}{2\Delta t}(o(t_{i-1}) - o(t_{i+1}))\right|} \cdot \sqrt{\left|\frac{1}{2\Delta t}(o'(t_{i-1}) - o'(t_{i+1}))\right|}, 1\right)} \end{aligned}$$

The formulas *absDiff*, *relDiff*, *slopeDiff*, *modRelDiff* and *modSlopeDiff* denote absolute difference, relative difference, slope-dependent difference, modified relative difference, and modified slope-dependent difference. The

modifications represented by these formulas lend weight to deviations at large amplitudes or slopes without over-estimating the parts of small amplitude values or slopes.

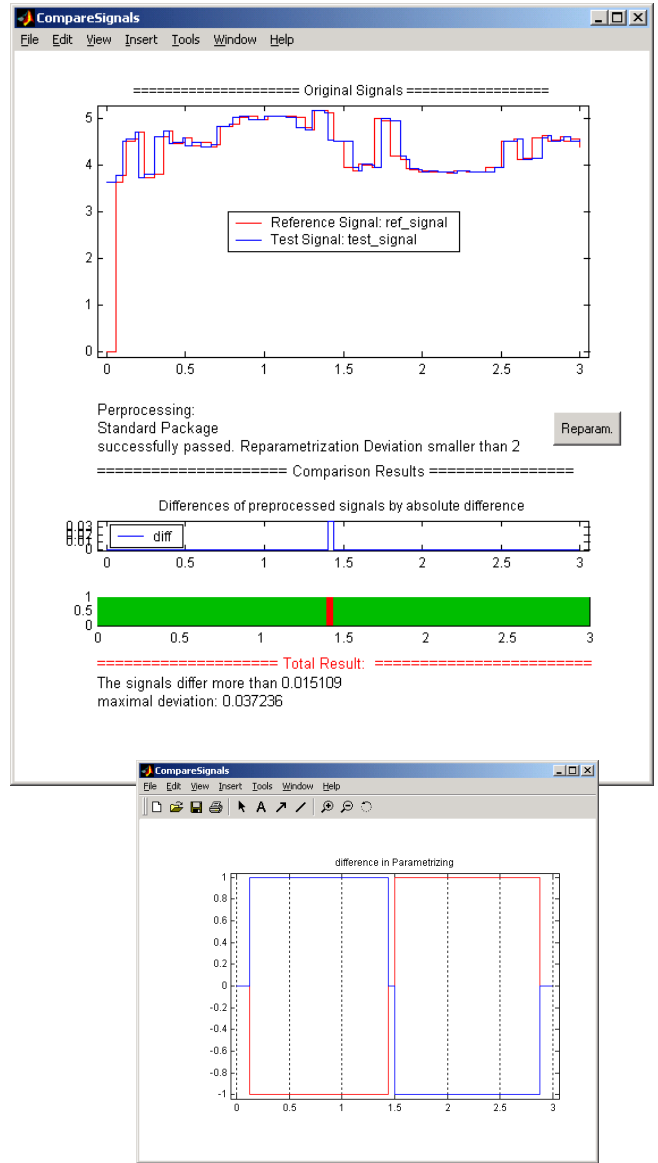
### AUTOMATED TEST EVALUATION WITH MEVAL

The multi-stage signal comparison procedure as well as the difference-matrix preprocessing method, described in the previous section, has been implemented in a tool called *MEval* [MEval], which is based on MATLAB/Simulink.



**Figure 9: MEval interactive mode and parameter setting GUIs**

*MEval* runs in both interactive and batch modes. The interactive mode can be used to search for optimal methods and parameter values for typical signals by observing the evaluation results for individual settings. Figure 9 shows the interactive mode GUI (Figure 9, above) as well as the method and parameter setting GUI (Figure 9, below). The evaluation results consist of signal difference curves before and after preprocessing (Figure 10, above), and a graphical representation of the reparametrization carried out during signal preprocessing (Figure 10, below).



**Figure 10: Test evaluation results in MEval**

### PREPROCESSING METHODS

In addition to the difference-matrix and symmetric difference-matrix preprocessing, described in the previous section, *MEval* offers the following conventional preprocessing methods to the user:

- *Partition by large deviation:* This preprocessing method filters those signal intervals with large deviations. In order to define a 'large' deviation, the user has to choose a comparison method and its correspondent  $\epsilon$ -tolerance.
- *Partition synchronization on trigger:* This method synchronizes two signals with regard to appropriate characteristics. *MEval* offers synchronization on zero points and on maxima.

In order to increase the performance of difference-matrix algorithms, a preprocessing *standard package* has been defined in *MEval*. The standard package is accomplished in two steps: starting from a tester specified comparison method, e.g. modified relative difference, the signals are first partitioned into intervals with significant deviation, using the *partition by large deviation* preprocessing method. Secondly, the signals are preprocessed using the symmetric difference-matrix method, which is applied to these intervals.

## PARAMETER DEFAULT VALUES

For each selected preprocessing and signal comparison method a set of parameters and thresholds has to be set by the user. In order to increase the automation grade of the test evaluation and to reduce the user interactions with *MEval*, default tolerance thresholds and further default parameter values are derived from the characteristic features of the reference signal based on statistical calculations.

The default tolerance thresholds for the signal comparison methods are computed according to the following formulae ( $tol_{absDiff}$ ,  $tol_{relDiff}$  and  $tol_{slopeDiff}$  are the default tolerance thresholds of absolute difference, modified relative difference and modified slope-dependent difference):

$$tol_{absDiff} \sim \max(o(t)) - E(o(t)) + \sigma(o(t))$$

$$tol_{relDiff} \sim tol_{absDiff} / \max(E(|o(t)|), 1)$$

$$tol_{slopeDiff} \sim tol_{absDiff} / \max(E(|o(t)'|), 1)$$

with

$$o(t)' = \frac{o(t_{i+1}) - o(t_{i-1})}{2 \cdot |t_{i+1} - t_{i-1}|},$$

where  $o()$  denotes the reference signal,  $E()$  the mean value and  $\sigma()$  the standard deviation.

## DEPLOYMENT SCENARIOS

The multi-stage signal comparison procedure and the difference-matrix preprocessing method introduced in this paper can be applied to any signal pair, independent of their origin. Accordingly, *MEval* can be deployed as a stand-alone tool, with which test outputs generated in different test environments, e.g. model-in-the-loop (MiL), software-in-the-loop (SiL) or hardware-in-the-loop (HiL), are compared.

### Regression tests

Within the scope of model-based development, consecutive model versions have to be checked with respect to their consistency and correct further development. In addition, the functional correctness of the model

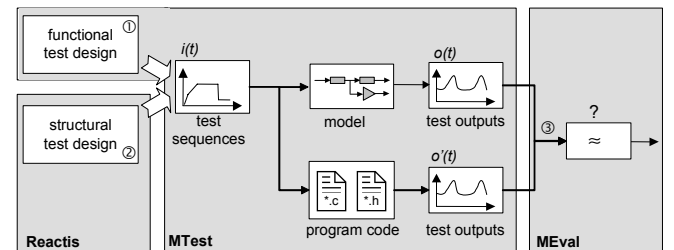
after error removal and model optimization needs to be checked. These checks are carried out by regression tests. *MEval* can be used for automatic, efficient and trustworthy regression tests. The effectiveness of *MEval* is increased if it is integrated into model-based testing environments such as MTest [LBE+04] [MTest]. MTest is a tool for testing MATLAB/Simulink models which supports a systematic definition of test scenarios as well as the automatic test data generation and test execution. By integrating *MEval* into MTest an automatic evaluation of regression tests within the same tool environment would be possible as well. By reusing the same test scenarios the new model versions can be tested against the earlier approved versions and the test results can be determined with a high degree of efficiency and reliability. A prototypical integration of *MEval* into MTest has already been implemented by the authors.

### Back-to-back tests

With regard to back-to-back tests, *MEval* may be deployed in combination with an automatic test data generator. The objective of a back-to-back test is to show the equivalence between the test object and its reference. This is the case, for example, when crosschecking the program code generated, manually or automatically, from a model against the original model. In order to be able to show such equivalence, a huge set of test data, regardless of its functional meaning or quality, is needed. The test data used should be capable of reaching a high degree of structural coverage of the model and code. An analysis of the relationship between the emerging concept of model coverage and the well-established method of code coverage can be found in [BCS+03] and [Ald02].

Reactis [Reactis] is a powerful tool for automatic structural test data generation. Given a Simulink model, Reactis automatically generates time-dependent test data (test sequences), to cover structural model elements, e.g. switch blocks or Stateflow states.

The deployment scenario for *MEval* together with Reactis and MTest for back-to-back tests of program code generated from a Simulink model against the original model would be as follows (Figure 11):



**Figure 11: Deployment of *MEval*, *Reactis*, and *MTest* for back-to-back tests**

1. Functional test design is carried out within MTest. The functional test sequences are used to validate the Simulink model.
2. The Simulink model is loaded into Reactis and structural test sequences are generated. In order to achieve a good result this action may need to be repeated a few times while adjusting the generation parameters.  
The union of all test sequences is used to test the model as well as the generated program code.
3. The test outputs of both tests are loaded into *MEval*, where the corresponding signals are preprocessed and compared.  
In the case of a mismatch between the courses of values of corresponding signals identified by *MEval*, the model and the code need to be investigated in order to find the origin of the problem.

The scenario described above can also be used for validating code generators [SC03]. In this case, where the program code is automatically generated from the model, the result of back-to-back tests leads to increased or decreased confidence in the code generator. For TargetLink [TL], a generator for C code from Simulink models, the execution and evaluation of back-to-back tests can be fully supported by the integrated testing environment including MTest and *MEval*.

## CAPABILITIES AND LIMITS

Regression and back-to-back tests are time-critical for software development. Too much effort invested in these kinds of tests will either lead to delays in the completion of the ECU software or to a reduction in the test's depth and/or test coverage because of tests not carried out.

An automation of the test evaluation leads to a reduction in the time needed for each test cycle. The time saved can be used either to reduce testing and thus the development time or to execute supplementary tests within the same overall test time.

Automated signal comparison also has the advantage that it relieves the tester of undemanding routine tasks and reduces the risk of human error.

However, there are also limits to the deployment of automatic test evaluation. For example, the tester has the possibility of classifying one and the same signal pair in different contexts either as similar (e.g. if it is an uninteresting output signal or generated by an uncritical module) or as dissimilar (e.g. if it is an output signal important for the function or a safety-relevant module). A completely automated signal comparison would classify the signal pair either as similar in both cases or as dissimilar in both cases. Here, we have a basic problem which cannot be solved by other or improved algorithms either.

The possibility of defining classes of preprocessing and comparison algorithms and their tolerance thresholds and parameters depending on the criticality or impor-

tance of the individual signals, represents a move towards human behavior. However, the assignment of a signal pair to one of these classes would still have to be carried out by a human tester. Thus, the procedures and parameters of an automated test evaluation, as a part of test criteria, need to be defined by experts in a project-wide and custom-designed way, e.g. by application and test engineers. In any case, after automated signal comparison the tester is still responsible for checking the automatically determined comparison results and should only use them as an aid to or basis for his/her own assessments.

## CONCLUSION

In this paper an innovative procedure for the automated test evaluation of control unit software has been introduced and discussed.

The main idea is to base test evaluation on a signal comparison and to automate this comparison. This kind of automatic test evaluation can mainly be applied to regression tests and back-to-back tests, which are of particular importance for model-based development.

As traditional procedures for signal comparison are only insufficiently applicable to the test evaluation of control unit software, a generic multistage procedure was developed and implemented in a tool: *MEval*. The multistage procedure for signal comparison can be understood as a construction kit, in which appropriate, custom-designed procedure components are put together and parameterized. The starting point for these specific adjustments is a standard combination of procedure components, which is called the standard package, in combination with a suitable default parameterization.

The difference-matrix preprocessing method is a core component of the multi-stage procedure for signal comparison. It establishes analysis possibilities for local and global temporal extensions, compressions and shifts which go far beyond traditional comparison methods. In particular, a separate analysis of temporal deviations and amplitude deviations is now made possible. Peaks or exchanged characteristics sequences can clearly be identified, small temporal deviations, on the other hand, can be tolerated. A modified version of the difference-matrix procedure is employed by the test evaluation component of the STEP-X test environment [HSM+04].

The performance of the difference-matrix procedure is linear with regard to signal length and depends on the search depth desired in finding the shortest way through the modified difference-matrix. It is comparable to the performance of the tolerance tube procedure. With the usual signal lengths of  $\leq 2000$  sample points it is possible to carry out the procedure online. Because of the prefixed partitioning in intervals of significant deviations it is also possible to analyze signals with more than 20,000 sampling points and few time-restricted deviations within a short period of time.

The approach to the derivation of default tolerance threshold values and other procedure parameters from a statistical analysis of the reference signals has been confirmed on the basis of numerous signal pairs used in pilot studies. For this purpose, model test data from different projects (engine control, driving functions, etc.) were compared to the test output of the generated code. The automatic evaluation was then contrasted with manual assessments.

Pilot studies have shown that an automatic test evaluation saves time and reduces costs. The quality of the analyses was also impressive. Every error which a manual review discovered was found and even several more, which had mostly been overlooked. These results confirm the conservative approach of the procedure to rather point out more errors than to prematurely mark as similar diverging signals which would have been rejected during a manual check.

### Future Work

The signal comparison procedure presented in this paper analyzes individual signal pairs isolated from other input and output signals of the test object. By considering temporal signal dependencies and characteristics it is possible to achieve a higher rate of test evaluation automation. Prerequisites for this task are the identification and classification of relevant signal characteristics, e.g. (local) minima and maxima, zero crossings and jumping points, as well as the description of temporal-logical dependencies between signals. These are issues which the authors are currently researching.

The custom-designed determination of the preprocessing and comparison algorithms and their tolerance thresholds and parameters can only be carried out on a practical basis. Here, the use of the *MEval* tool offers the possibility of recording and analyzing the corresponding data in different projects and application areas. On the basis of these analyses, custom-designed recommendations can then be developed for the selection of procedures and parameter values.

### **ACKNOWLEDGMENTS**

The work described was partially performed as part of the IMMOS project funded by the German Federal Ministry of Education and Research (project ref. 01ISC31D), <http://www.immos-project.de>

### **REFERENCES**

[Ald02] Aldrich, W. J.: Using Model Coverage Analysis to Improve the Controls Development Process. AIAA Modeling and Simulation Technologies Conference and Exhibition, Monterey, US, 2002.

[ASCET-SD] ASCET-SD (product information). ETAS GmbH, [en.etasgroup.com/products/ascet\\_sd/](http://en.etasgroup.com/products/ascet_sd/)

[BCS+03] Baresel, A., Conrad, M., Sadeghipour, S., Wegener, J.: The Interplay between Model Coverage and Code Coverage, 11. Europ. Int. Conf. on Software Testing, Analysis and Review (EuroSTAR 03), Amsterdam, NL, 2003.

[Bro03] Broy, M.: Automotive Software Engineering. 25th Intl. Conference on Software Engineering (ICSE 03), Portland, USA, pp. 719-720, 2003.

[CFP03] Conrad, M., Fey, I., Pohlheim, H.: Automatisierung der Testauswertung für Steuergerätesoftware. VDI-Berichte, Vol. 1789, VDI Verlag, pp. 299-315, 2003.

[CFS04] Conrad, M., Fey, I., Sadeghipour S.: Systematic Model-Based Testing of Embedded Automotive Software - The MB3T Approach. ICSE 2004 workshop on Software Engineering for Automotive Systems (SEAS '04), Edinburgh, UK, 2004.

[CS03] Conrad, M., Sax, E.: Mixed Signals. In Broekman, B., Notenboom, E.: Testing Embedded Software, Addison Wesley, 2003.

[DSS+01] Dornseiff, M., Stahl, M., Sieger, M., Sax, E.: Durchgängige Testmethoden für komplexe Steuerungssysteme – Optimierung der Prüftiefe durch effiziente Testprozesse. 10. International Congress Elektronik im Kraftfahrzeug, Baden-Baden, Germany, 2001.

[HaRe01] Model-Based Tools Update. The Hansen Report on Automotive Electronic, Vol. 14, No. 5, [www.hansenreport.com](http://www.hansenreport.com), June 2001.

[HSM+04] Horstmann, M., Schnieder, E., Mäder, P., Nienaber, S., Schulz H.-M.: A framework for interlacing Test and/or Design, ICSE 2004 workshop on Software Engineering for Automotive Systems (SEAS '04), Edinburgh, UK, 2004.

[LBE+04] Lamberg, K., Beine, M., Eschmann, M., Otterbach, R., Conrad, M., Fey, I.: Model-based testing of embedded automotive software using MTest. SAE World Congress, Detroit, USA, 2004.

[MATLAB] MATLAB/Simulink/Stateflow (product information). The MathWorks Inc., [www.mathworks.com/products](http://www.mathworks.com/products).

[MEval] MEval (product information). IT Power Consultants, [www.itpower.de/meval\\_e.html](http://www.itpower.de/meval_e.html)

[MTest] MTest (product information). dSPACE GmbH, [www.dspace.de/www/en/pub/products/sw/expsoft/mtest.htm](http://www.dspace.de/www/en/pub/products/sw/expsoft/mtest.htm)

[Reactis] Reactis Tester (product information), Reactive Systems Inc., [www.reactive-systems.com](http://www.reactive-systems.com).

[RJ93] Rabiner, L., Juang, B.H.. Fundamentals of speech recognition. Prentice Hall Signal Processing Series, 1993.

[RWS+01] Ritter, C., Willibald, J., Sax, E., Müller-Glaser, K. D: Entwurfsbegleitender Test für die modellbasierte Entwicklung eingebetteter Systeme. 13th Workshop Testmethods on Reliability of Circuits and Systems, Mi-esbach, Germany, 2001.

[SC03] Stürmer, I., Conrad, M.: Test Suite Design for Code Generation Tools. 18. IEEE Int. Conf. on Automated Software Engineering (ASE '03), Montreal, Canada, Oct. 2003

[TL] TargetLink (product information). dSPACE GmbH, [www.dspace.de/ww/en/pub/products/sw/targetli.htm](http://www.dspace.de/ww/en/pub/products/sw/targetli.htm)

[WCF+02] Wiesbrock, H.-W., Conrad, M., Fey, I., Pohlheim, H.: Ein neues automatisiertes Auswerteverfahren für Regressions- und Back-to-Back-Tests eingebetteter Regelsysteme. In Softwaretechnik-Trends, 22 (2002) 3, 2002.

## CONTACT

In 1995 **Mirko Conrad** earned a Diploma degree in Computer Science from the Technical University of Berlin (Germany). In 2004 he received his PhD from the TU Berlin for his work on model-based testing of embedded automotive software. Since 1995 he is project manager and research scientist at the Software Technology Lab of DaimlerChrysler Research & Technology. He is member of the Special Interest Group for Testing, Analysis and Verification of Software in the German Computer Society (GI TAV) and the MathWorks Automotive Advisory Board (MAAB).

E-Mail: [Mirko.Conrad@DaimlerChrysler.com](mailto:Mirko.Conrad@DaimlerChrysler.com)

**Sadegh Sadeghipour** received his PhD in Computer Science from the Technical University of Berlin in 1998. After a research period on testing methods of embedded systems at the Software Technology Lab of Daimler-Chrysler Research & Technology he co-founded IT Power Consultants in 2000. The company provides the German automotive industry with consulting and tools in the field of ECU software.

E-Mail: [Sadegh.Sadeghipour@itpower.de](mailto:Sadegh.Sadeghipour@itpower.de)

**Hans-Werner Wiesbrock** received his PhD in Mathematics from the University Bonn, 1983, and Habilitation in Theoretical Physics from the FU Berlin, 1993. Since 1999 he works as a computer scientist in the Software Industries. His main interests concern analytical and constructive methods for quality improvement of embedded systems, especially their mathematical backgrounds.

E-Mail: [Hans-Werner.Wiesbrock@itpower.de](mailto:Hans-Werner.Wiesbrock@itpower.de)