

# XML-Format für den Software-in-the-Loop-Test

---

Thomas Liebezeit, Holger Schumann, Sven Rebeschief, Uzmee Bazarsuren

## Abstract

This paper presents a new XML test format developed for the SiLEST project. Automated regression tests at software-in-the-loop level were the main focus of interest for the development. The article uses a simplified example to describe the contents and interactions of the different files. It presents therefore an overview of the technology used in SiLEST and its possibilities.

## Zusammenfassung

Dieser Beitrag stellt das neu definierte XML-Format für Dateien der Testumgebung des SiLEST-Projekts vor. Das Format wurde für die automatisierte Durchführung von Software-in-the-Loop-Regressionstests entwickelt. Der Artikel stellt anhand eines vereinfachten Beispiels den Aufbau und das Zusammenwirken der einzelnen Bestandteile detailliert dar und gibt damit einen Überblick über die verwendeten Ansätze und ihre Möglichkeiten.

## 1. Einleitung

In dieser Arbeit wird ein neues XML-Format [1] vorgestellt, das im Rahmen des SiLEST-Projekts entwickelt wurde. Das SiLEST-Projekt<sup>1</sup> untersucht die Einsatzfähigkeit der Software-in-the-Loop- (SiL)-Technologie für den Test eingebetteter Software in der Automobil- und Raumfahrttechnik [2-4].

Das Ziel des Projekts besteht in der Entwicklung einer SiL-Testumgebung sowie im anschließenden Vergleich der damit realisierbaren Ergebnisse mit denen von Hardware-in-the-Loop-(HiL)- Tests. Für den Bereich der Automobilindustrie bilden automatisierte Regressionstests das Hauptinteresse. Diese sollen bei der modellbasierten Entwicklung von Motorsteuergerätefunktionen einen kostengünstigeren, umfassenden und früheren "in-the-Loop"-Test ermöglichen, der sich zusätzlich durch Wiederholbarkeit, Rückverfolgbarkeit und gute Dokumentation auszeichnet. Dabei sollen das statische und das dynamische Funktionsverhalten sowohl im Normalbetrieb, als auch bei Sensor- oder Aktuatorfehlern untersucht werden.

---

<sup>1</sup> Software in the Loop for Embedded Software Test , BMBF gefördert im Rahmen der Initiative "Software Engineering 2006" (Förderkennzeichen: 01ISC12A); Projektpartner sind neben IAV GmbH (Berlin) und TU Berlin / MDT auch DLR / SISTEC (Braunschweig), FhG FIRST (Berlin) und webdynamix GmbH (Cottbus). Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

## 2. SiL-Test

Ein SiL-Test ist dadurch charakterisiert, dass das gesamte Testsystem in einer Simulationsumgebung (z.B. Matlab/ Simulink) zur Verfügung steht. Ein Simulationsmodell für die Funktionsentwicklung eines Motorsteuergeräts untergliedert sich dabei beispielsweise wie folgt:

- *Steuergerätemodell*  
Zu testende Software-Funktion als Modell und/oder C-Code, Modell des Steuergeräte-Betriebssystems (Scheduler, restliche Steuergerätefunktion), Steuergeräte-I/O-Komponenten
- *Umgebungsmodell*  
Modelle für Motor und Getriebe, Fahrzeug-(Längs-)Dynamik, Aktuatorik und Sensorik (Nominal- und Fehlerverhalten), Fahrer und Umwelt

Charakteristischerweise handelt es sich bei der Softwarefunktion um einen Regler, der abgetastete analoge Signale regelt. Wichtig im Rahmen seiner Entwicklung ist der frühzeitige Funktionstest im geschlossenen Regelkreis.

## 3. SiL-Testumgebung

Für das SiLEST-Projekt wurde eine Testumgebung definiert. Diese besteht aus

- der *Testverwaltung* (Testmaster, TestDirector),
- der *Testautomatisierung* mit dem zu testenden System und
- der *Versionsverwaltung* (Subversion, StarTeam).

Die Testverwaltung stellt den Zugang zu den einzelnen Tests und ihren Ergebnissen bereit. Sie verwaltet dazu alle zugehörigen Testfälle, -resultate und -reports in einer Datenbank. Damit lässt sich jederzeit nachvollziehen, welcher Anwender zu welchem Zeitpunkt was mit welchem Ergebnis getestet hat. Des Weiteren werden aus der Testverwaltung heraus neue Tests angewiesen.

Die Testablaufsteuerung ermöglicht die automatisierte Durchführung von Tests auf dem zu testenden System. Bei der Testablaufsteuerung handelt es sich um eine im Rahmen des SiLEST-Projekts neu entwickelte Java-Anwendung.

Die Versionsverwaltung übernimmt die Versionierung aller im Rahmen des Tests verwendeten Dateien, um eine umfassende Reproduzierbarkeit zu gewährleisten.

### 3.1 Dateien

Die vorgestellte Testumgebung arbeitet mit *Testfall*-Dateien, die unabhängig von einer Testmethode (z.B. SiL) für ein bestimmtes, zu entwickelndes System definiert sind. Damit wird eine Wiederverwendbarkeit der Testfälle für andere Testmethoden erreicht, was z.B. eine Vergleichbarkeit von SiL- und HiL-Testergebnissen ermöglicht. Um die Tests mit einer Realisierung des Systems (z.B. SiL-Variante) durchzuführen, wird diese in der *Testkonfiguration* in ihrem Aufbau beschrieben, und es wird

ein Mapping des Testfalls auf diese Beschreibung vorgegeben. Die dazu notwendige Abstraktion zwischen Testfall und Testsystem wird über die Verwendung von Labels erreicht. Die bei der Durchführung der Testfälle erzielten Resultate werden im *Testreport* für die Weiterverarbeitung gesichert.

Alle drei Datenformate sind jeweils über ein XML-Schema [1] definiert. Die Vorteile eines XML-Datenformats liegen vor allem in der klaren, offenen Strukturierung der Daten, was auf einfache Weise die Wiederverwendbarkeit über entsprechende Transformationen ermöglicht. Des Weiteren garantiert ein Schema eine einfache Validierung der Dateistruktur.

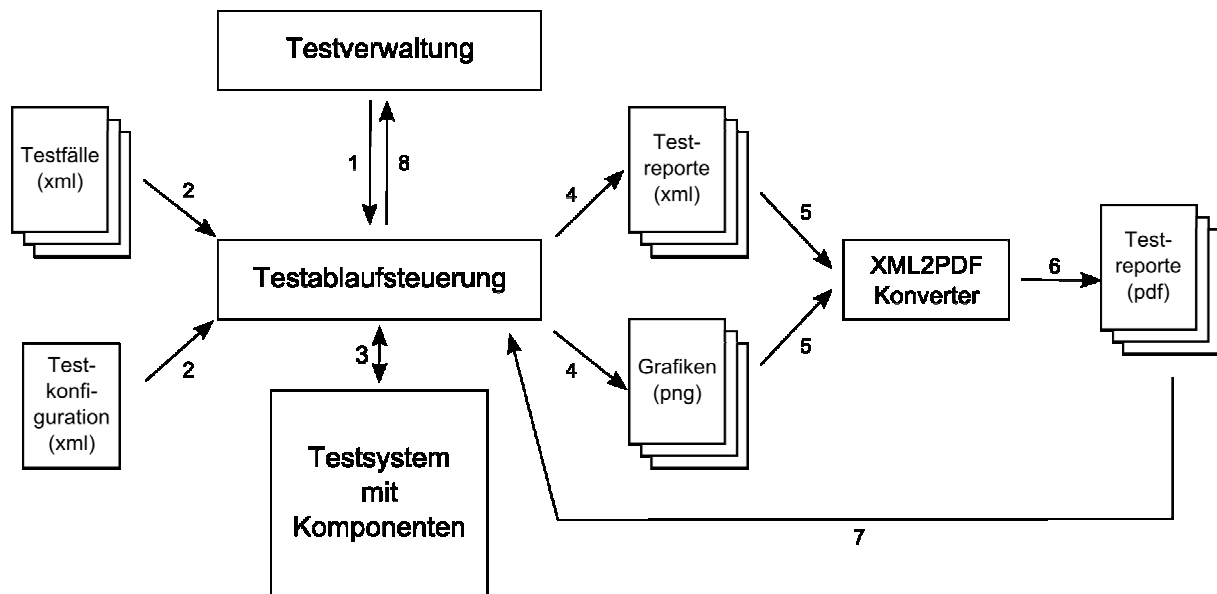


Abbildung 1: Dateien und Ablauf der Tests

### 3.2 Ablauf

Der prinzipielle Testablauf in Bezug auf die vorgestellten Dateien ist in Abbildung 1 dargestellt. Der Anstoß für die Abarbeitung von Testfällen erfolgt aus der Testverwaltung heraus (1). Die Testablaufsteuerung übernimmt die automatische Abarbeitung der übergebenen Testfälle auf dem Testsystem, wofür sie die Testkonfiguration nutzt (2, 3).

Im Ergebnis entstehen je Testfall ein XML-Testreport sowie Grafiken und das Testresultat (4). Über einen Konverter werden die Testreporte und Grafiken in PDF-Testreporte gewandelt (5, 6). Diese und das Testresultat werden von der Testablaufsteuerung an die Testverwaltung zurückgeliefert (7, 8), wo die Ergebnisse aller Tests übersichtlich verwaltet werden.

Der Benutzer der Testumgebung weist demnach in der Testverwaltung die automatisch durchzuführenden Tests an und erhält nach ihrer Durchführung an gleicher Stelle die Testresultate und -reporte angezeigt.

## 4. Beschreibung der XML-Formate

Die Beschreibung der XML-Formate soll exemplarisch an einem stark vereinfachten, theoretischen Beispielsystem erfolgen. Zudem soll dabei nicht auf Spezialfälle eingegangen werden. Für die Erstellung der Testfälle und der Testkonfiguration steht mit dem SiLEST-eigenen Testfalleditor ein Werkzeug bereit, für das der Testingenieur über keinerlei XML-spezifische Kenntnisse verfügen muss.

### 4.1 Beispielsystem

Als Beispiel soll die Entwicklung eines Leerlaufreglers dienen.

Da Testfälle im Rahmen von SiLEST unabhängig von der Testmethode (z.B. Software-in-the-Loop) sein sollen, wird für die Funktion und ihren Test ein Set von Labeln definiert, die das konkrete Testsystem abstrahieren. Für das Beispiel zeigt die folgende Tabelle eine mögliche Definition.

Label	Beschreibung
N_SOLL	Sollwertvorgabe für die Leerlaufdrehzahl
N_IST	Geregelte Drehzahl
K_PARAM	Reglerparameter K
SENSOR	Drehzahlsensor

Die Funktion gestaltet sich damit wie folgt: Für eine Sollwertvorgabe N\_SOLL erzeugt der Regler eine Antwort N\_IST. Sein Verhalten ist dabei abhängig vom eingestellten Parameter K\_PARAM und den vom SENSOR ermittelten Drehzahlwerten.

### 4.2 Testkonfiguration

Für das Beispiel existiert ein Simulink-Modell als Testsystem. Es untergliedert sich, wie in Abbildung 2 dargestellt, in zwei Subsysteme für den Regler und das Motormodell. Das Regler-Subsystem besitzt n\_Mot, die aktuelle Motordrehzahl als Eingang und fuel\_mass als Ausgang (Einspritzmenge). Des Weiteren beschreibt der Parameter K einen Kennwert des Regelalgorithmus. Das Motormodell erhält die Einspritzmenge als Eingang und gibt die aktuelle Drehzahl n\_ist aus. Diese wird vom Sensor S erfasst und an den Regler zurückgeliefert. Die Sollwertvorgabe kommt aus dem Matlab-Workspace, und die erzielte Drehzahl des Motors wird dorthin zurückgeschrieben. Das gesamte Modell ist zudem für die Tests vorkalibriert.

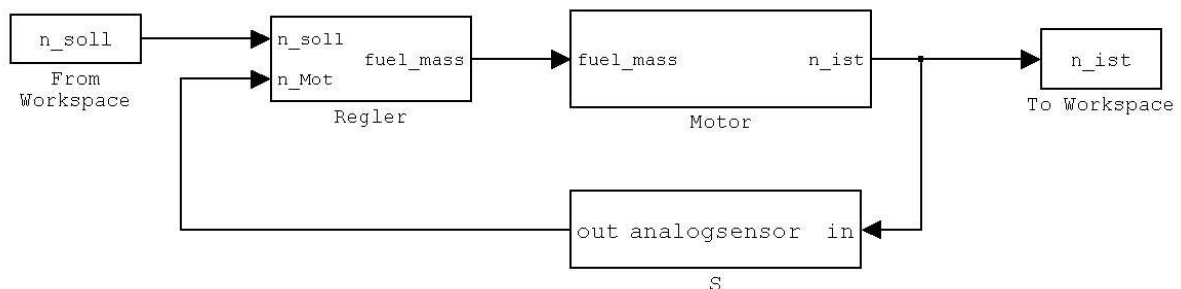


Abbildung 2: Beispiel Simulink-Modell

Die Testkonfiguration besteht aus einer abstrakten Beschreibung des Testsystems und der Zuordnung der Ein- und Ausgänge sowie der Parameter zu den definierten allgemeinen Testfalllabeln.

```
1 <CONFIGURATION xsi:schemaLocation="http://www.silest.de/ configuration.xsd"
  createdby="tvI" created="2006-04-10" info_link="www.silest.de" version="0.1" ...
  schema_version="0.4" project="SiLEST" type="SIL" xmlns="http://www.silest.de/">
2   <DESCRIPTION/>
3   <SETUP>
4     <MODEL name="bsp" module="matlabR14" model_file="bsp_model.mdl"
  workspace_file="workspace.mat" step_width="0.1">
5       <INPUT_LIST>
6         <INPUT port="n_soll" comment="Solldrehzahl"/>
7       </INPUT_LIST>
8       <OUTPUT_LIST>
9         <OUTPUT port="n_ist" comment="Istdrehzahl"/>
10      </OUTPUT_LIST>
11      <PARAMETER_LIST>
12        <PARAMETER name="K" comment="Kennwert" type="value"/>
13        <PARAMETER name="S" comment="Sensor" type="sensor"/>
14      </PARAMETER_LIST>
15    </MODEL>
16  </SETUP>
17  <MAPPING>
18    <CONNECT>
19      <SETUP_PORT port="n_soll" component="bsp"/>
20      <TESTCASE_PORT name="N_SOLL" namespace="silest"/>
21    </CONNECT>
  ...
30    <CONNECT>
31      <SETUP_PARAMETER name="S" component="bsp"/>
32      <TESTCASE_PARAMETER name="SENSOR" namespace="silest"/>
33    </CONNECT>
34  </MAPPING>
35 </CONFIGURATION>
```

Abbildung 3: Testkonfiguration (Beispiel)

Für das vorliegende Beispiel zeigt Abbildung 3 die resultierende Datei (gekürzt um das Mapping von N\_IST und K\_PARAM). Das Testsystem (das Modell) wird mit seinen aus Testsicht vorhandenen Ein- und Ausgängen sowie Parametern beschrieben (Zeilen 3-16, <SETUP>).

Darauf folgt die Zuordnung von Testfallein- und -ausgängen bzw. Parametern zum Testsystem (Zeilen 17-34, <MAPPING>). Dabei werden die Label aus dem Testfall den Testsystemkomponenten zugeordnet. Für die Zuordnung wird der im Testfall definierte Namensraum genutzt.

Besteht das Testsystem aus zwei oder mehr Komponenten, können im Mappingteil auch deren Ein- und Ausgänge miteinander verbunden werden. In diesem Fall übernimmt der Kopplungsmanager der Testablaufsteuerung in der Testdurchführung den Datenaustausch zwischen den Komponenten.

### 4.3 Testfall

Ein möglicher Testfall für das Beispielsystem besteht darin, für ein gegebenes Kaltstartprofil eine entsprechend folgende Motordrehzahl einzuregeln. Erschwerend kommt hinzu, dass der Drehzahlsensor ab einem gewissen Zeitpunkt verstärkt ein Rauschen aufweist und der Parameter K des Leerlaufreglers ungenau appliziert wurde (Werte siehe Testfalldefinition). Das vorgegebene Drehzahlprofil ist in Abbildung 4 dargestellt.

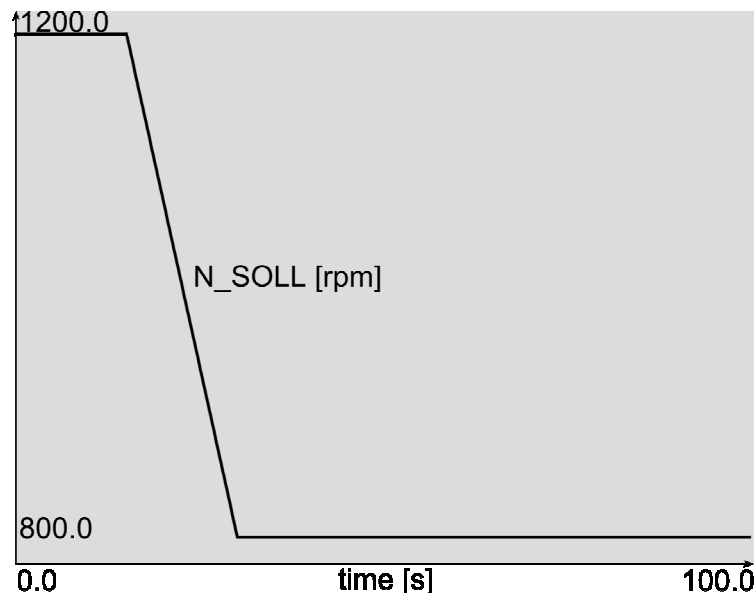


Abbildung 4: Vorgegebenes Kaltstart-Drehzahlprofil

Dieser Testfall wird in der in Abbildung 5 dargestellten Testfalldatei beschrieben. Der gegenüber der Standard-Kalibrierung veränderte Parameter K\_PARAM wird im Kalibrationsteil festgelegt (<PARAMETER>, Zeile 4). In den Zeilen 7 bis 12 findet die Definition des Drehzahl-Sollverlaufes (<NORMAL\_INPUT>, Label N\_SOLL) statt. Das Fehlverhalten des Sensors wird in Zeile 13 bis 17 beschrieben (<FAULT\_TRIGGER>). Es besteht aus dem zeitweiligen additiven Aufschalten eines Rauschanteils auf den Sensorausgang.

Für das erwartete Ergebnis wird ein Bandtest der Motordrehzahl (Label N\_IST) festgelegt (<BELT> mit <BASE\_FUNCTION> und <EPSILON>, Zeilen 21-31). Hierzu wird, ebenso wie bei den Eingangsverläufen und z.T. auch bei der Beschreibung des Fehlverhaltens, auf die SiLEST-Signaldefinition zurückgegriffen. Diese beschreibt beliebige analoge Signale auf Basis eines Sets von Basissignalen (z.B. Linie, Wert, Funktion oder Sequenz), die intervallweise hintereinander gefügt werden. Der Testfall umfasst damit sowohl die Definition der abweichenden Kalibrierung (<CALIBRATION>), der Eingangsvorgaben und Fehlerfälle (<TESTCASE\_DEFINITION>, Zeilen 6-18), als auch Informationen zur Auswertung der Ergebnisse (<ANALYSIS>, Zeilen 19-33). Zusätzlich enthält er Verwaltungsinformationen und (hier für eine bessere Übersichtlichkeit weggelassen) detaillierte Beschreibungen, die für die Erstellung des PDF-Testreports verwendet werden. Abhängigkeiten zwischen einzelnen Testfällen lassen sich mit dem SiLEST-Testfallformat ebenfalls definieren.

```

1 <TESTCASE xsi:schemaLocation="http://www.silest.de/ testcase.xsd" name-
space="silest" info_link="www.silest.de" created="2006-04-10" createdby="tvI" ver-
sion="0.1" schema_version="0.6" start_time="0.0" end_time="100.0" time_unit="s"
... xmlns="http://www.silest.de/">
2 <DESCRIPTION>Leerlaufregler, Kaltstart</DESCRIPTION>
3 <CALIBRATION>
4 <PARAMETER type="normal" name="K_PARAM" value_unit="-
">10</PARAMETER>
5 </CALIBRATION>
6 <TESTCASE_DEFINITION>
7 <NORMAL_INPUT comment="Solldrehzahl" port="N_SOLL"
value_unit="rpm" initial_value="1200">
8 <INTERVAL>
9 <VALUE start_time="0" end_time="15" value="1200" hold="hold"/>
10 <LINE start_time="15" start_value="1200" end_time="30"
end_value="800" hold="hold"/>
11 </INTERVAL>
12 </NORMAL_INPUT>
13 <FAULT_TRIGGER comment="Drehzahl" port="SENSOR" va-
lue_unit="rpm">
14 <ANALOG_SIGNAL><MODIFY><ADD>
15 <NOISE start_time="50" end_time="-1" hold="zero"
min_amplitude="-10" max_amplitude="10"/>
16 </ADD></MODIFY></ANALOG_SIGNAL>
17 </FAULT_TRIGGER>
18 </TESTCASE_DEFINITION>
19 <ANALYSIS name="Test1" create_documentation="ever">
20 <OUTPUT caption="Istdrehzahl" comment="Geregelte Istdrehzahl"
port="N_IST" value_unit="rpm">
21 <BELT>
22 <BASE_FUNCTION>
23 <INTERVAL>
24 <VALUE start_time="0" end_time="15" value="1200" hold="hold"
/>
25 <LINE start_time="15" start_value="1200" end_time="30"
end_value="800" hold="hold"/>
26 </INTERVAL>
27 </BASE_FUNCTION>
28 <EPSILON>
29 <VALUE value="10" hold="hold"/>
30 </EPSILON>
31 </BELT>
32 </OUTPUT>
33 </ANALYSIS>
34 </TESTCASE>

```

Abbildung 5: Testfall (Beispiel)

## 4.4 Testreport

Ein Testreport enthält jeweils das Resultat der Durchführung eines Testfalls. Der Testreport vermerkt allgemeine Informationen zum Test, wie beispielsweise in welcher Version der Testfall getestet wurde und wann dies geschah (siehe Abbildung 6, Zeile 1, Darstellung gekürzt).

Des Weiteren wird für das Testsystem mit seinen Komponenten vermerkt, welche Versionen zum Einsatz kamen. Dazu liefern die Komponenten eine definierte Selbstauskunft (Zeilen 2-11, <TESTSYSTEM>).

Das Auswertesystem der Testablaufsteuerung nimmt nicht nur die Analyse der erzielten Signalverläufe und deren Dokumentation vor (Zeilen 16-18, <RESULT>), sondern erzeugt auch für jedes vorgegebene Signal eine Abbildung mit dem im Test verwendeten Signalverlauf (Zeile 12-14, <INPUTS>), die in die PDF-Version des Testreports als Grafik eingebaut wird.

Der SiLEST-Testprozess kennt u.a. die Resultate "passed" (erfolgreich), "failed" (fehlgeschlagen) und "inconclusive" (keine Aussage möglich). Das Gesamtergebnis kann sich dabei aus mehreren Einzeltests zusammensetzen, die über die logischen Operatoren AND, OR und NOT miteinander verknüpft sind. Für das hier vorgestellte Beispiel ergibt sich das Testresultat "passed" (Zeile 16).

```
1 <TESTREPORT xmlns="http://www.silest.de/" ... testcase="Test1.xml" versi-
  on="0.1" created="2006-04-11T8:50:00" createdby="tv1" project="SiLEST" sche-
  ma_version="0.2">
2   <TESTSYSTEM configuration="bsp_configuration.xml" config_version="0.1"
  test_start="2006-04-11T9:01:00" test_end="2006-04-11T9:00:00">
3     <COMPONENT name="bsp" ressource_xpath="MODEL[name=bsp]">
      ...
10    </COMPONENT>
11  </TESTSYSTEM>
12  <INPUTS>
13    <FIGURE uri="bsp_n_soll.svg" port="n_soll" component="bsp"
  xpath="NORMAL_INPUT[port=N_SOLL]"/>
14  </INPUTS>
15  ...
16  <RESULT result="passed">
17    <PARTIAL_RESULT port="n_soll" component="bsp" result="passed"
  xpath="OUTPUT[port=N_IST]" figure="0_bsp_n_soll.svg"/>
18  </RESULT>
19 </TESTREPORT>
```

Abbildung 6: Testreport (Beispiel)

## 5. Fazit

Anhand eines stark vereinfachten, theoretischen Beispielsystems wurde das für das SiLEST-Projekt definierte XML-Format für Tests vorgestellt. Dieses besteht aus einer XML-Beschreibung der Testfälle, der Testkonfiguration (zu testendes Systems und Mapping des Testfalls auf dieses) und den Testreporten. Es dient als Datenformat im SiLEST-Testprozess. Die Arbeit mit den Testfällen und der Testkonfiguration

erfolgt über einen Testfalleeditor, so dass der Testingenieur nicht direkt mit dem XML-Format arbeitet. Der Testreport wird automatisch von der Testablaufsteuerung erzeugt. Die vorgestellten Dateien ermöglichen im Rahmen der erläuterten Testumgebung die Durchführung automatisierter Regressionstests während der Entwicklung eingebetteter Softwarefunktionen.

## Literatur

- [1] [www.w3c.org/XML](http://www.w3c.org/XML), Extensible Markup Language (XML)
- [2] [www.silest.de](http://www.silest.de), SiLEST-Projektseite
- [3] S. Rebeschies, Th. Liebezeit, U. Bazarsuren. „Automatisierter Closed-Loop-Softwaretest eingebetteter Motorsteuerfunktionen“ 26<sup>th</sup> Automotive Electronics Conference „Elektronik im Kraftfahrzeug“, 27.6.-28.6.2006
- [4] O. Maibaum, S. Rebeschies. „Test von adaptiven Softwaremechanismen zur Fehlerkompensation“, 2. Tagung „Simulation und Test in der Funktions- und Software-Entwicklung für die Automobilelektronik“, 14.3.-15.3.2005.

## **Autoren**

Dr.-Ing. Thomas Liebezeit, TU Berlin / MDT, Einsteinufer 17, 10587 Berlin,  
Tel.: 030/ 314-23950, Fax: 030/ 314-22120, Email: thomas.liebezeit@tu-berlin.de

Dipl.-Ing. (FH) Holger Schumann, DLR / SISTEC, Lilienthalplatz 7, 38108 Braun-  
schweig, Tel. 0531/ 295-2776, Fax: 0531/ 295-2767, Email: hol-  
ger.schumann@dlr.de

Dipl.-Ing. Sven Rebeschies, IAV GmbH / Abt. MS-T43, Carnotstr.1, 10587 Berlin,  
Tel.: 030/ 39978-9550, Fax: 030/ 39978-9299, Email: sven.rebeschies@iav.de

Dipl.-Ing. Uzme Bazarsuren, TU Berlin / MDT, Einsteinufer 17, 10587 Berlin,  
Tel.: 030/ 314-25612, Fax: 030/ 314-22120, Email: uzmee.bazarsuren@tu-berlin.de