12th IMEKO TC10 Workshop on Technical Diagnostics
New Perspectives in Measurements, Tools and Techniques for Industrial Applications
June 6-7, 2013, Florence, Italy

A Novel Statistical Approach for Testing Automotive Control Software

I. Arsie[1], G. Betta[2], D. Capriglione[2], A. Pietrosanto[1], P. Sommella[1]

[1] *DIIn, University of Salerno, Via P. Don Melillo 1, 84084 Fisciano (SA), ITALY,*
E-mail: {iarsie,apietrosanto, psommella}@unisa.it
[2] *DIEI, University of Cassino and Southern Lazio, Via G. Di Biasio 43, 03043 Cassino (FR), ITALY,*
E-mail: {betta,capriglione}@unicas.it

Abstract- The paper describes an effective testing method for control software. Starting from a previous release, the authors propose a proper update of their methodology to make it feasible for testing a new class of software which requires to take into account the time interdependence among the software inputs. The statistical approach has been applied to a typical automotive case study that concerns with the validation of monitoring and control software for the engine management system.

I. Introduction

In the last few years, the rapid growth of Information Technology and the proliferation of PCs have resulted in numerous software products in several fields of today's society. Very complex application software are currently employed to manage communication systems such as telephone switches or process worldwide business transactions, but other examples can easily be found in medical, transportation, and industrial applications. Automotive systems, for instance, is a typical context where software procedures (Anti-lock braking system, Electronic stability program, cruise control, to cite a few), executed by suitable micro-electronic circuitry, are able to grant passenger safety and comfort. In all these systems, software has become a critical part which can play a fundamental role in assuring the quality of the overall product at the same extent of hardware components. Thus, such typical constraints of manufacturing as efficient process operating, safety, reliability, and fault tolerance are now also concerned with software. Evidence of this new perspective is the growing interest in software engineering [1], i.e. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. As result, some standards and/or guidelines have been issued about software as a key component which contributes to system behavior and performance. Significant examples are the International Standard ISO/IEC 9126 [1], which defines a quality model for software product, or the IEEE Standard 1012 [2] for Software Verification and Validation, which attempts to establish a common framework for all the activities and tasks in support of software life cycle steps, including acquisition, supply, development, operation, and maintenance processes.

The same efforts aiming to achieve an improvement in matching with the user satisfaction can be found in the highly competitive market of instrumentation, where an approach to quality assurance for measurement software is also required. In fact, in almost all of the current generation of measurement systems the use of software has dramatically increased. Independently from whether software is embedded within the measurement system, or provided by the system supplier separated from the measurement system, or developed separately to work with one or many types of measurement system, the key issue is to make the devices easier to use and at the same time more reliable, repeatable, and accurate. However, the hidden complexity of software can be a potential source of undetected errors and have important risk implications, especially when the measurement systems are used as subsystems of safety-related systems. Thus, in order to preserve the integrity of the measurement process, the reliability and/or quality of such software have also to be quantified. To this aim, already consolidated and newly developed practices in software engineering are investigated about applications involving measurement systems. For example, in [3] a framework for approaching to quality assurance is defined upon a risk assessment conformed to IEC 61508 [4] and consequently some validation techniques are suggested to be applied.

The Validation process can consider different phases in software development. Anyway, it is usually mostly focused in Testing activity, i.e. in verifying that the results of Coding activity both satisfy the requirements specified during Design and the expected user needs. Testing process is a very complex, time, and cost consuming process, that generally involves the application of different techniques, including both static methods which depend upon the analysis of the design and, above all, dynamic methods which execute test cases. As to the latter methods, a huge variety of techniques [5] has been proposed in literature for general purpose software, depending from the level of complexity to be analyzed (unit, integration, or system testing) and all following either a structural approach (that is based on a detailed knowledge of the code) or a functional one (according to which such knowledge is unnecessary). Anyway, in the last years researches on new techniques able to give a high assurance of suitability of the software system under test have been mostly focused on statistical testing,

evaluating the possibility of applying methods derived, for example, from experimental design, Taguchi's Robust Method, orthogonal arrays, or probability theory [6]-[13].
Following this trend, in previous papers [14], [15], the authors introduced a method based on suitable statistical techniques, which can be used for validation of complex instrument software. In particular, the suggested method reveals an efficient strategy to overcome the limits that consolidated testing techniques often present when they are applied to measurement system. In fact, traditional techniques are mostly suitable to boolean or a few-level discrete variables systems and/or disregarding the correlations existing among inputs. On the other hand, measurement software is typically concerned with the processing of information representative of quantities, that are often, *i)* continuously time-varying in wide ranges and, *ii)* correlated one each other in a complex way. Thus, traditional methods could hardly allow efficient and realistic testing results, and new techniques have to be applied to assure the robustness of the whole measurement software product. The proposed method was validated by applying it to a reference measurement software employed in an automotive environment [15]. In more details, the test procedure has been applied to the instrument diagnostic software of a commercial car engine employed to locate sensor faults of the charge detection system.
The good performance obtained has led the authors to investigate the possibility to extend the suggested method to the validation of a wider and more general class of measurement software. To this aim, the method has been suitably updated and its application is shown with reference to an automotive software module providing the main actions for automotive engine control. After a brief recall of main and key features of the test method, a description of the peculiarities of the control software under test and first preliminary results are given.

II. Brief recall of the test methodology

The method proposed for testing measurement software follows a functional or black box approach, which requires to verify the system under test with a suitable studied set of externally controlled input-output combinations, by generating the so-called *test case*. Thus, the internal implementation of the software being executed is not needed to be known and only the outputs, given certain input values, are checked to be conformed to the functional requirements. The approach is consequently more suitable for the latest stages of software testing, such as integration and/or system test, where greater attention is focused on the user's perspective and the reliability of the whole product. In particular, the method suggested is considered as a statistical test technique, because the distributions of the input quantities are taken in account in generating the test cases. By carrying on this strategy, the test activity attempts to reproduce the behavior of the software system the user can actually experiment.
In the most common test applications, the number of real combinations can be quite unlimited, by making an exhaustive approach practically unfeasible. Thus, the challenge is to design a Test Set, which, although including just a finite number of effective conditions, can be anyway considered well-representative of the system operational profile, or at least, it can be achieved a quantitative estimation of the test goodness [16]. Moreover, a particular feature of measurement software is that they often process input quantities (representative of physical phenomena) between them correlated. As a consequence, these relationships have to be evaluated before the test case generating in order to exclude redundant and/or wrong input combination.
The main steps of the proposed testing methodology are schematized in Figure 1.
At first, a measurement campaign has to be performed for collecting a significant statistical sample of the software input quantities looking for covering the entire input domain, the output of this stage is a two-dimensional array, ***I***, containing a suitable number of samples for each one of the *N* input quantities. In the second step the estimation of the statistical distribution for each input quantity is performed. Such distributions are then sampled by applying the Latin Hypercube Sampling (LHS) with the aim of efficiently covering the

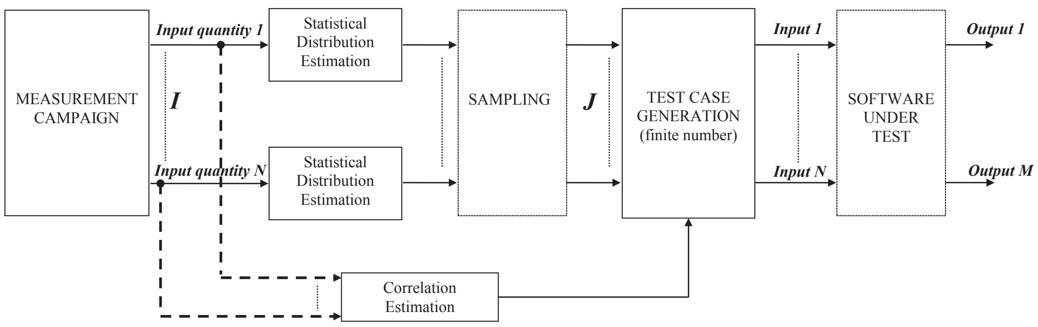

Figure 1. Block diagram of the test methodology.

N-dimensional input domain of the software system through a reduced number of samples as compared with other sampling methods [17]. Output of the LHS is a two-dimensional array, ***J***, containing new values of the software input quantities which could be submitted to the software under test. Nevertheless, as previous said, the correlations existing between the input quantities have to be estimated and taken into account in the test case generation. Such correlations are evaluated on the experimental data by means of the Spearman's rank coefficients [18]. Moreover, the bootstrap methods [19] are employed to overcome the limits deriving from the sparseness or partiality of the available data, thus allowing the variability associated to the particular measurement campaign through a confidence interval for each rank coefficient to be quantified. Then, these pieces of information are used in the optimal choice of the Test Set dimension and to provide plausible test cases to the software under test.
As it can be derived by this brief recall of the method, the suggested procedure results in a suitable strategy to explore the system domain for detecting those possible combinations of input values which can lead the system itself in unpredictable states, where the expected functionalities are not reachable. Such a feature is particularly attractive for validating software strategies of measurement-based systems. As an example, the closed-loop control software is often designed on expected values of system outputs, which can be highly different from the measured and/or actuated quantities, because some factors such as noise effects, electrical interference, quantization, measurement uncertainty. Although the simplification in the model design is often absolutely necessary for the complexity of the analyzed system, the next tuning of the measurement and control software could not be able to detect possible mismatching with the expected behavior of the implemented system.
In fact, the validation process is often limited to respect the conditions as defined in the applicable legal normative, and disregard a complete behavior observable by the users. In automotive context, the design and the control of car engines aims to respect the limits in pollution emissions which are related to standard operative cycles, which can be highly different from the cycles in the day-life use (it depends from the driving style, traffic and road conditions, to cite a few). On the other hand, the growing interest and efforts in assuring the quality of the product involves a continuous improvement also in the validation activity. Thus, the test method proposed by the authors could be an efficient technique to be applied (also in conjunction with other tools and/or by inserting suitable modification) for verifying the reliability of the complex control software based on measurement information.

III. The software under test

The new case study concerns with the Engine Control System, a field where innovative approaches and techniques have been proposed in the last years because of continuous government constraints on exhaust emissions and engine fuel economy.
In particular it is made reference to the Air-Fuel Ratio (AFR) control software in Spark Ignition (SI) engines described in [20]. The control system estimates the right amount of fuel to be injected to meet the target AFR value, thus obtaining the highest efficiency of the three-way catalytic converters. To these aims, it tries to perform an adaptive control more efficiently than classical techniques especially during the engine transients.
By considering as inputs the engine state variables (*i.e.* p, intake manifold pressure, and n, engine speed), the engine temperature T_{H2O} and the target stechiometric AFR (as depicted in Figure 2), the control software under test is able to predict the injection time, T_{inj}, and achieve a very narrow band for the AFR signal around the desired value.
To test the control strategy suitable transients have to be simulated imposing fast throttle opening-closing actions, which characterize standard manoeuvers such as standing start and up/down gear shift. The new control strategy has been developed with the help of a dynamic engine/vehicle simulation environment named ODECS [21]. Even though very useful in the designing phase, ODECS was not thought to adequately test the control strategy. Nevertheless, an optimum if not exhaustive exploration of all the plausible working conditions which the control system could be engaged with should be made to effectively validate the control strategy.
To these end, the test method previously proposed by the authors is not properly suitable in driving the validation process of such a kind of software. In fact, even if the previous method allows taking into account the correlations between the input quantities, unlike other applications, in order to obtain plausible test cases it

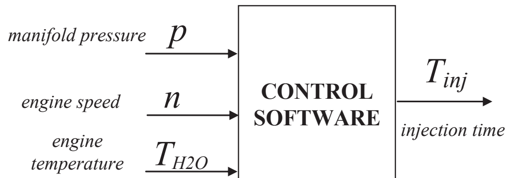

Figure 2. Schematic of the control software under test

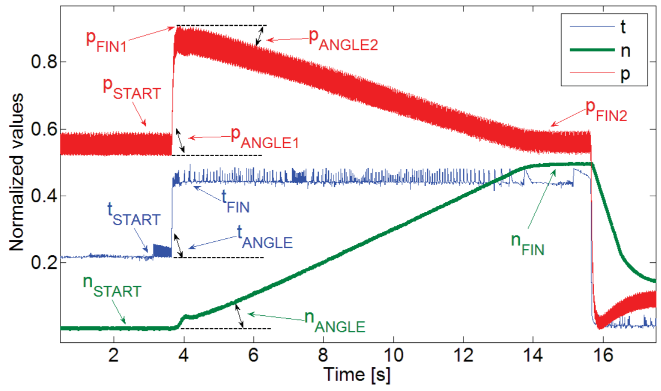

Figure 3. Time evolutions of *t*, *n*, and *p* acquired on a test bench (Values Normalized in the range [0-1]) due to the fast transition of the throttle (*t=throttle, n=engine speed, p=intake manifold pressure*).

would not be enough the application of the estimated correlations but also the system state and its dynamics (generally described by differential equations) should be respected among samples of the same quantity. As an example, Figure 3 shows the time evolution of *p* and *n*, acquired on a test bench, for given transitions of the throttle angle, *t*, and given load conditions. One can observe the strict dependence of the state variables (*p* and *n*) dynamics on *t*. Therefore, to effectively validate the control software, the sequence of test cases (couples of *p* and *n*) has to be generated by satisfying suitable dynamic profiles.

IV. The new proposal

Starting from the previously designed test methodology (see Figure 1), some changes are introduced to extend its application to the peculiarities of the new class of software under test:

(i) The *input quantity i* does no longer represent the *i-th* measured quantity but a suitable parameter analytically describing the transient evolution of an input quantity. As an example, if a transient evolution of *n* is linearly approximated (as suggested by Figure 3), the starting (n_{START}) and final (n_{FIN}) values and the slope (n_{ANGLE}) describing the considered linear regression curve have to be considered as input quantities. Analogous parameters can be adopted for describing the transient evolution for *t*, i.e. t_{START}, t_{FIN} and t_{ANGLE}. As for *p*, its observed time evolution suggests to adopt a step-linear approximation which involves a greater number of parameters describing the transient evolution, i.e. p_{START}, p_{ANGLE1}, p_{FIN1}, p_{ANGLE2}, and p_{FIN2}. Consequently, differently from the previous applications, the input vector of the scheme reported in figure 1 becomes $\boldsymbol{I}=(n_{START}, n_{FIN}, n_{ANGLE}, t_{START}, t_{FIN}, t_{ANGLE}, p_{START}, p_{ANGLE1}, p_{FIN1}, p_{ANGLE2}, p_{FIN2})$.

(ii) The measurement campaign has to provide a suitable set of transient conditions for each quantity of interest. In this case the data acquisition will be carried out through a test bench but same consideration can be made about the field data acquisition. For each observed transient evolution, the estimation of the ***I***-vector components has to follow (for example adopting the minimum squared error method). In this way, a number, *K*, of vectors, $\boldsymbol{I_i}$ (i=1,..*K*), is used as starting point for the next steps of the test methodology to estimate the statistical distribution of each parameter belonging to ***I*** as well as their mutual correlations (see Figure 1).

(iii) The *test case generation* section processes the input vector ***J*** (same size of ***I*** and containing plausible profile parameters of *t*, *n* and *p*) to submit a sequence of transient evolutions of *t*, *n* and *p* to the software under test. To emulate the quantization noise introduced by the data acquisition hardware, all the software under test input signals will be obtained by adding a suitable random noise to the generated transient evolutions. Then, according to the previously proposed scheme, the control software is tested by using suitable transient evolutions satisfying real dynamics and physics input quantity links.

V. The application example

To show the practical application of the new proposal, the testing of the control software described in section III is here reported with reference to the main steps of Figure 1.

As far as the *measurement campaign* in concerned, a suitable advanced engine test bench has been adopted to collect data. It was composed by a Fiat Spark-Ignition Engine (4 cylinders, 1.2 L), an eddy current dynamometer by Borghi and Severi and the Puma Test Bench Automation System by AVL. Data acquisition has been achieved in correspondence with the engine state variables evolution determined by a well-planned series of transients for the independent variable (opening angle for throttle valve, *t*). In particular, fifteen profiles have been designed,

TABLE I. *RANK CORRELATION MATRIX FOR THE I-VECTOR COMPONENTS*

t_{angle}	p_{angle1}	p_{angle2}	n_{angle}	p_{START}	p_{FIN1}	p_{FIN2}	n_{START}	n_{FIN}	t_{START}	t_{FIN}	
1.00	0.50	0.13	-0.16	-0.18	0.13	-0.04	0.06	0.24	-0.29	0.00	t_{angle}
	1.00	0.05	-0.12	-0.37	0.14	0.01	0.16	0.37	-0.23	0.20	p_{angle1}
		1.00	-0.68	-0.54	-0.10	0.05	0.60	0.37	0.12	0.26	p_{angle2}
			1.00	0.38	0.27	0.17	-0.40	-0.20	-0.03	-0.03	n_{angle}
				1.00	0.27	0.17	-0.46	-0.32	0.37	-0.04	p_{START}
					1.00	0.86	0.17	0.44	0.25	0.76	p_{FIN1}
						1.00	0.43	0.54	0.43	0.84	p_{FIN2}
							1.00	0.82	0.45	0.64	n_{START}
								1.00	0.29	0.73	n_{FIN}
									1.00	0.50	t_{START}
										1.00	t_{FIN}

TABLE II. *STANDARD DEVIATION OF THE RANK CORRELATION COEFFICIENTS*

t_{angle}	p_{angle1}	p_{angle2}	n_{angle}	p_{START}	p_{FIN1}	p_{FIN2}	n_{START}	n_{FIN}	t_{START}	t_{FIN}	
0.00	0.10	0.11	0.10	0.11	0.11	0.11	0.11	0.11	0.12	0.11	t_{angle}
	0.00	0.11	0.11	0.10	0.10	0.10	0.10	0.09	0.12	0.10	p_{angle1}
		0.00	0.07	0.09	0.11	0.10	0.09	0.11	0.11	0.11	p_{angle2}
			0.00	0.11	0.10	0.10	0.09	0.10	0.12	0.10	n_{angle}
				0.00	0.09	0.11	0.08	0.10	0.10	0.11	p_{START}
					0.00	0.04	0.11	0.10	0.11	0.06	p_{FIN1}
						0.00	0.09	0.09	0.09	0.05	p_{FIN2}
							0.00	0.04	0.08	0.07	n_{START}
								0.00	0.11	0.07	n_{FIN}
									0.00	0.09	t_{START}
										0.00	t_{FIN}

each one constituted by ten step-like functions different in the starting value (t_{START}), final value (t_{FIN}), and slope (t_{ANGLE}) in order to study the behaviour corresponding to both fast and slow manoeuvring of accelerating and braking (throttle opening and closing). Each throttle profile so obtained has been adopted for driving the measurement campaign on the test bench by the automation system. As a consequence a number of 150 different transient evolutions were available for the next steps of the testing procedure. Moreover, in order to estimate the system repeatability, multiple acquisitions has been repeated for each profile. The obtained results proved a good repeatability (lower than 1%) of all parameters describing the time evolution of the quantities of interest (t, p, n).
As for the *correlation estimation*, starting from the data collected by the measurement campaign, the target rank correlation matrix has been evaluated by adopting the procedure described in [14], [15]. Table I reports the obtained rank coefficients for the quantities of interest calculated with reference to the rising transient evolutions (given the symmetry, for a better readability only the upper triangular matrix has been reported). It shows that some quantities exhibits a not negligible correlation, as confirmed by the corresponding standard deviations (see Table II) estimated by considering the normal bootstrap technique with 500 resamples.
As an example, Table III reports a set of six vectors $\boldsymbol{J_M}$ (M=1,..,6) achieved with the proposed procedure, which, suitably combined by the *test case generation* section, provide a plausible sequence of t, n, and p rising transient evolutions to use for the software testing. Figure 4 reports the transient evolution generated by considering the vector $\boldsymbol{J_1}$. One can observe the good agreement with the acquired transient evolutions shown in Figure 3. Analogues considerations can be made for the generation of falling transient evolutions.

VI. Conclusions

A methodology to test a monitoring and control software for the engine management system has been presented in this paper. Starting from a previous release, the testing methodology has been suitably updated to take into account also the time interdependence among the software inputs. The main novelties introduced concern with sections devoted to the estimation of the input correlations and the test case generation.
In particular, the new release of the testing methodology identifies the correlations among suitable parameters analytically describing the transient evolutions of the input quantities.
As for the test case generation section, it has been updated to provide the expected input quantities to the software under test.
Consequently, the new proposal allows the software under test to be validated by using suitable transient evolutions satisfying real dynamics and physics input quantity links.
The technique has been successfully applied to a typical automotive case study that concerns with the validation of monitoring and control software for the engine management system.

TABLE III. J_I VECTOR COMPONENTS (I=1,..,6) FOR RISING TRANSIENT EVOLUTIONS

	J_1	J_2	J_3	J_4	J_5	J_6
n_{START}	0.22	0.28	0.37	0.16	0.66	0.42
n_{FIN}	0.52	0.33	0.48	0.52	0.87	0.64
n_{ANGLE}	0.031	0.027	0.022	0.031	0.014	0.036
t_{START}	0.17	0.10	0.17	0.13	0.27	0.39
t_{FIN}	0.40	0.17	0.32	0.38	0.74	0.49
t_{ANGLE}	1.3	0.21	0.75	0.93	0.97	0.5
p_{START}	0.40	0.23	0.28	0.35	0.29	0.66
p_{ANGLE1}	1.3	0.14	0.86	1.2	0.11	0.15
p_{FIN1}	0.80	030	0.54	0.78	0.86	0.72
p_{ANGLE2}	-0.021	-0.003	-0.014	-0.015	-0.002	-0.015
p_{FIN2}	0.57	0.29	0.46	0.52	0.84	0.65

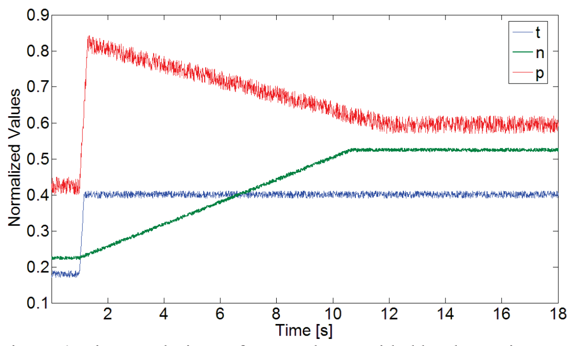

Figure 4. Time evolutions of t, n, and p provided by the testing procedure for the vector $\boldsymbol{J_1}$ (Values Normalized in the range [0-1]).

References

[1] "Standard ISO/IEC 9126-1: Software engineering -Product quality - Part 1: Quality model", *International Organization for Standardization*, 2001.

[2] "Software Verification and Validation"*, IEEE Standard 1012-2004*.

[3] B. Wichmann, G. Parkin, R. Barker, "Validation of Software in Measurement Systems – Software Support for Metrology Best Practice Guide No. 1" *National Physical Laboratory, UK*, January 2007.

[4] "Standard IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements", *International Electrotechnical Commission*, 2010.

[5] H. Freeman, "Software Testing", *IEEE Instrumentation and Measurement Magazine*, Vol. 5, Issue 3, September 2002, pp. 48-50.

[6] A. Causevic, D. Sundmark, S. Punnekkat, "An Industrial Survey on Contemporary Aspects of Software Testing," *Proceedings of 2010 Third International Conference on Software Testing, Verification and Validation*, April 2010, Paris (France), pp.393-401.

[7] M.S. Phadke, "Quality Engineering Using Robust Design", Prentice Hall, Englewood Cliffs, N.J., 1989.

[8] S. Stoica, "Robust test methods applied to functional design verification" *Proceedings of Test Conference*, pp. 848-857, September 1999.

[9] D.M. Cohen "The AETG System: An Approach to Testing Based on Combinatorial Design", *IEEE Transactions On Software Engineering*, July 1997.

[10] M.A. Bailey, T.E. Moyers and S. Ntafos, "An application of random software testing", *Military Communications Conference, MILCOM'95*, Conference Record, IEEE, Vol. 3, November 1995.

[11] S.C. Ntafos, "On comparisons of random, partition, and proportional partition testing", *IEEE Transactions on Software Engineering*, Vol. 27, Issue 10, October 2001.

[12] M. Catelani, L. Ciani, V. L. Scarano, A. Bacioccola, "A Novel Approach To Automated Testing To Increase Software Reliability", I2MTC 2008 - IEEE International Instrumentation and Measurement Technology Conference Victoria, Vancouver Island, Canada, May 12-15, 2008.

[13] J. Musa, "Operational Profiles in Software Reliability Engineering", *IEEE Software* , Mar. 1993 , pp. 14-32

[14] G. Betta, D. Capriglione, A. Pietrosanto, P. Sommella, "A Statistical Approach for Improving the Performance of a Testing Methodology for Measurement Software," *IEEE Transactions on Instrumentation and Measurement*, Vol.57, Issue 6, June 2008, pp. 1118-1126.

[15] G. Betta, D. Capriglione, A. Pietrosanto, P. Sommella, "A methodology to Test Instrument Software: An Automotive Diagnostic System Application," *IEEE Transactions on Instrumentation and Measurement*, Vol.57, Issue 12, December 2008, pp. 2733-2741.

[16] M. Catelani, L. Ciani, V. L. Scarano, A. Bacioccola, "Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use", *Computer Standards and Interfaces*, Volume 33, Issue 2, February 2011, pp. 152-158.

[17] J.C. Helton, F.J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems", *Elsevier, Reliability Engineering and System Safety 81* , 2003, pp. 23-69.

[18] D. Bonnet, T. Wright, "Sample size requirements for estimating Pearson, Kendall and Spearman correlations", *Psykometrika* Vol. 65 n. 1, March 2000, pp. 23-28.

[19] S. Ley, M. R. Smith, "Evaluation of several nonparametric bootstrap methods to estimate confidence intervals for software metrics", *Software Engineering, IEEE Transactions on*, Vol. 29, Issue 11, November 2003, pp. 996-1003.

[20] I. Arsie, C. Pianese, M. Sorrentino, "Nonlinear Recurrent Neural Networks for Air Fuel Ratio Control in SI Engines", *Proceedings of SAE 2004 World Congress & Exhibition,* March 2004.

[21] I. Arsie, R. Flora, C. Pianese, G. Rizzo, G. Serra, "A Computer Code for S.I. Engine Control and Powertrain Simulation", *SAE 2000 Transactions – Journal of Engines,* Vol. 109-3, pp.935-949.