The Parametric Method for Functional Testing of Virtual Instruments

Marek Florczyk[1]

[1]*University of Zielona Góra, Institute of Electrical Metrology, ul. Podgórna 50, 65-246 Zielona Góra, Poland, phone +48(68)3282658, fax+48(68)3254615, m.florczyk@ime.uz.zgora.pl*

Abstract- The article presents the parametric method for functional testing of virtual instruments. Presented method allows to define the quality of project implementation of virtual instruments. Owning to the definition of weight matrices of user selections, for particular elements of the user interface in virtual instruments, already on the stage of their design, it is possible to estimate the risk related to their implementation. The presented parametric method for functional testing allows to define mutations, which result from incorrect design implementation of virtual instruments. Tests by parametric method for functional testing can be carried out both manually and automatically.

I. Introduction

Owning to the dynamic development of computer science technology, virtual instruments constitute significant part of presented measuring devices. Large functional possibilities and possibility to adjust virtual instrument user interface to users' needs, require designers and programmers to apply complex testing procedures during all stages of works over a virtual instrument. The application of classical approach to testing of virtual instruments, based on, which has taken place e.g. in case of the voltmeter or oscilloscope, tests only within the measurement scenario, usually results in the occurrence of errors - unidentified during the testing process virtual instrument – during the user operation. The measurement scenario is an algorithm of user's procedures during the operation of the measuring device and is usually included in the instrument instruction manual. Tests which are carried out within the measurement scenario (algorithm) consist in the realization of defined tasks (so-called test cases) [1].
A traditional measuring device differs fundamentally from a virtual instrument in the manner of its application. Extensive functional possibilities, related to making adjustment owing to the user interface of virtual instruments, involve the risk that adjustments will be made at variance with the measurement scenario assumed by its designer. In case of any traditional measuring device, it is simply impossible to select two measurement functions at the same time, e.g. measuring DC and AC voltage. This results from fact that defined functions are usually assigned to defined keys (buttons), and their selection cancels previously selected function. This is possible in case of virtual instruments, when the user interface has not been designed properly (Fig. 1.).

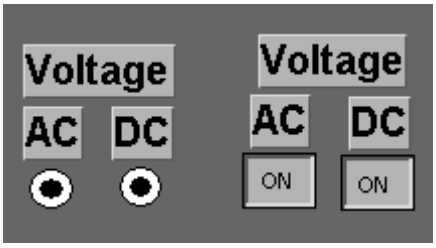

Figure 1. The picture of a improperly designed virtual instrument panel (LabView)

The user, who proceeds according to the measurement scenario (the user manual), will not cause device errors, because he will operate within the measurement scenario tested by virtual instrument designers. However, when the user accidentally executes an operation, which has not been included the in measurement scenario, e.g. he will change a selection order of an individually functions, a measuring device may begin to work unsteady or may even totally terminate. In the virtual instrument, its source code of the user interface constitutes nearly 50% of the whole application source code. The tests used most often – verifying the correctness of the source code – will not detect errors related to the user interface used by a user. The virtual instrument can not work properly despite the fact that its source code is correct (Fig. 1.). Thus, it is necessary to search for methods, which make it possible to verify not only the correctness of the virtual instrument source code, but also to verify the correctness of the

user interface and software connections with the device part of the virtual instrument (e.g. the acquisition card). The entire range of accessible testing methods is used to perform the tests. The most often used testing techniques are [2]:

- *Random testing techniques* – in which data generated to carry out tests have random character.
- *Functional testing techniques* – in which data generated to carry out tests are based on the specification of a testing system and tests are carried out usually by black-box method.
- *Control flow testing techniques* – in which information flow in the system is tested, and test are carried out by white-box method.
- *Data flow testing techniques* – in which values of individual data are analyzed within individual states of the system.
- *Mutation testing techniques* – based on the modeling of typical errors made by users and observing their results.
- *Regression testing techniques* – which are based on the closed loop, where tests are carried out on previously gathered data form tests.
- *Improvement testing techniques* – in which, performed tests aim at detecting possible errors as early as possible.

Black-Box and White-Box methods are frequently used for software testing. Even methods related to automatic testing (among others, neural networks and genetic or evolutionary algorithms are used) are often used [3,4,5,6,7,8]. A large number of source code testing methods (which can be used to carry out tests of the virtual instrument source code), does not make up for a small number of functional testing methods (e.g. [9]), which can be used for virtual instrument testing. The parametric method for functional testing of virtual instruments is one of the few functional testing methods.

II. The Parametric Method For Functional Testing Of Virtual Instruments

The appearance of the user interface and the hardware platform are defined on the stage of making assumptions for the virtual instrument and data flow. Besides, the connection between the software and hardware part is also defined. Thus, the virtual instrument can be shown as the following three groups of parameters:

- The set F (describing the program part) – includes the program parameters constituting the program part and the user interface.
- The set P (describing the hardware part) – includes parameters related to references to the hardware part.
- The set PF (describing associations) – includes parameters describing relations between the program and hardware parts.

The wording "parameter" means that a given element of the set can assume a defined value range e.g. in case of the set F, parameters can be all elements of the user interface, with which any adjustments are made (e.g. switches, buttons etc.).

Considering that the virtual instrument can be described by the three sets of parameters, taking into account that each parameter can be given weight, it is possible to define the following dependency:

$$\mathrm{VIQM} = \mathrm{VIQM}_{\mathrm{Prog}} + \mathrm{VIQM}_{\mathrm{Dev}} + \mathrm{VIQM}_{\mathrm{Conn}} \tag{1}$$

The Matrix VIQM defines the quality of the virtual instrument when:

$$\mathrm{VIQM}_{\mathrm{Prog}} = \sum_{i,j}\left(\mathrm{f}_{(i,j)}\mathrm{w}_{\mathrm{f}(i,j)}\right) \quad \mathrm{VIQM}_{\mathrm{Dev}} = \sum_{i,j}\left(\mathrm{p}_{(i,j)}\mathrm{w}_{\mathrm{p}(i,j)}\right) \quad \mathrm{VIQM}_{\mathrm{Conn}} = \sum_{i,j}\left(\mathrm{pf}_{(i,j)}\mathrm{w}_{\mathrm{pf}(i,j)}\right) \tag{2}$$

where, f – is the matrix of parameters related to the program part, p – is the matrix of parameters related to the equipment part, pf – is the matrix of parameters describing relations between the program and equipment parts, w_f – constitutes the weight matrix of the f parameter, w_p – constitutes the weight matrix of p parameters, w_{pf} – constitutes the weight matrix of pf parameters.

Every coefficient of the matrix f, p, pf describes the correctness of a defined function, or combination of functions (3).

$$f=\begin{bmatrix} f_{(1,1)} & f_{(1,2)} & \cdots & f_{(1,j)} \\ f_{(2,1)} & f_{(2,2)} & \cdots & f_{(2,j)} \\ \vdots & & & \vdots \\ f_{(i,1)} & & & f_{(i,j)} \end{bmatrix} p=\begin{bmatrix} p_{(1,1)} & p_{(1,2)} & \cdots & p_{(1,j)} \\ p_{(2,1)} & p_{(2,2)} & \cdots & p_{(2,j)} \\ \vdots & & & \vdots \\ p_{(i,1)} & & & p_{(i,j)} \end{bmatrix} pf=\begin{bmatrix} pf_{(1,1)} & pf_{(1,2)} & \cdots & pf_{(1,j)} \\ pf_{(2,1)} & pf_{(2,2)} & \cdots & pf_{(2,j)} \\ \vdots & & & \vdots \\ pf_{(i,1)} & & & pf_{(i,j)} \end{bmatrix} \quad (3)$$

The parameter value is determined as a result of tests based on the following formula:

$$f_{i,j} = N_{b(i,j)} / N_{t(i,j)}, \qquad N_{t(i,j)} > 0 \quad (4)$$

$$f_{i,j} = 0, \qquad N_{t(i,j)} = 0 \quad (5)$$

where: $N_{b(i,j)}$ – the number of incorrect behaviors (inconsistent with expectations of the designer or user) of the virtual instrument, $N_{t(I,j)}$ – the number of tests which have been carried out.
The testing procedure assumes that according to (3) a defined measurement function or a combination of functions is chosen (understood as the selection of the first and then another function) – for example, the selection of voltage and then the selection of measurement range. Coefficients $f_{i,j}$ reach values from the range (0,1). The lower the value of the coefficient $f_{i,j}$ is, the higher is the quality of implementation of a defined function (or combination of functions).
Tests by the parametric method for functional testing can be carried out both manually and automatically. Tests by Black-Box method are used most frequently. When the test are carried out automatically, it is necessary to define a set of test values (a set of input functions), and a script, which will realize the testing algorithm according to the formula (2). The number of performed tests depends on required precision for tests (for example, a range of values (0, 10) can be tested with increments "1" or e.g. with increments "0,01", and the precision will be different).
In case when only the software part is tested, the following formula should be used:

$$VIQM_{Prog} = \sum_{i,j} \left(f_{(i,j)} w_{f(i,j)} \right) \quad (6)$$

Thus, all elements of the user interface are analyzed. Individual coefficients of the matrix $VIQM_{Prog}$ reach values from range <0,1>. Coefficients of the matrix $VIQM_{Prog}$ define the probability that the user will select defined elements of the user interface (or a combination of elements) and his selection will cause errors in the virtual instrument. The boundary value analysis [12] and logical tests [9] (e.g. in the LabView for switches and text rings) are used to perform tests on the user interface. The construction of the matrix (3) allows to divide test works between test teams in case of more complex projects with a large number of tests.
Matrices of weights are created on the basis of the measurement scenario or the structure of dialog windows of the virtual instrument. The weights can already be prepared on the stage of user interface design, which allows estimate the risk related to the implementation of defined functionality. Individual weights in the matrixes (8, 9, 10), correspond to individual parameters (form the matrix (3)) or their combination. The value of weight is the probability defined as:

$$w_{i,j} = P(A \cap B) = P(A)\,P(B/A) \quad (7)$$

Matrixes of weights are as follows:

$$w_f = \begin{bmatrix} w_{f(1,1)} & w_{f(1,2)} & \cdots & w_{f(1,j)} \\ w_{f(2,1)} & w_{f(2,2)} & \cdots & w_{f(2,j)} \\ \vdots & & & \vdots \\ w_{f(i,1)} & & & w_{f(i,j)} \end{bmatrix} \quad (8)$$

$$w_p = \begin{bmatrix} w_{p(1,1)} & w_{p(1,2)} & \cdots & w_{p(1,j)} \\ w_{p(2,1)} & w_{p(2,2)} & \cdots & w_{p(2,j)} \\ \vdots & & & \vdots \\ w_{p(i,1)} & & & w_{p(i,j)} \end{bmatrix} \quad (9)$$

$$w_{pf} = \begin{bmatrix} w_{pf(1,1)} & w_{pf(1,2)} & \cdots & w_{pf(1,j)} \\ w_{pf(2,1)} & w_{pf(1,2)} & \cdots & w_{pf(2,j)} \\ \vdots & & & \vdots \\ w_{pf(i,1)} & & & w_{pf(i,j)} \end{bmatrix} \tag{10}$$

The determination of coefficients of the weight matrix by means of the formula (6), means that the user makes selection of individual elements in a specific order – the most often, from the most important to the least important elements (form the point of view of the performed measurement). This means for the panel of the virtual instrument, that functions in the front panel are selected first, (the higher probability of selection), and then, function from the other panels of the virtual instrument (the lower probability, the selection is dependant on the earlier selection of functions from the front panel). In a real virtual instrument, access to individual functions can be realized through opening next application widows. It often happens that all functions are accessible from a single panel of a virtual instrument (in case of simple virtual instruments). When measurements take place according to the accepted measurement scenario, it is possible to calculate weights by means of the formula (6) on the basis of the measurement algorithm. An example of weight calculation has been shown in the figure 2.

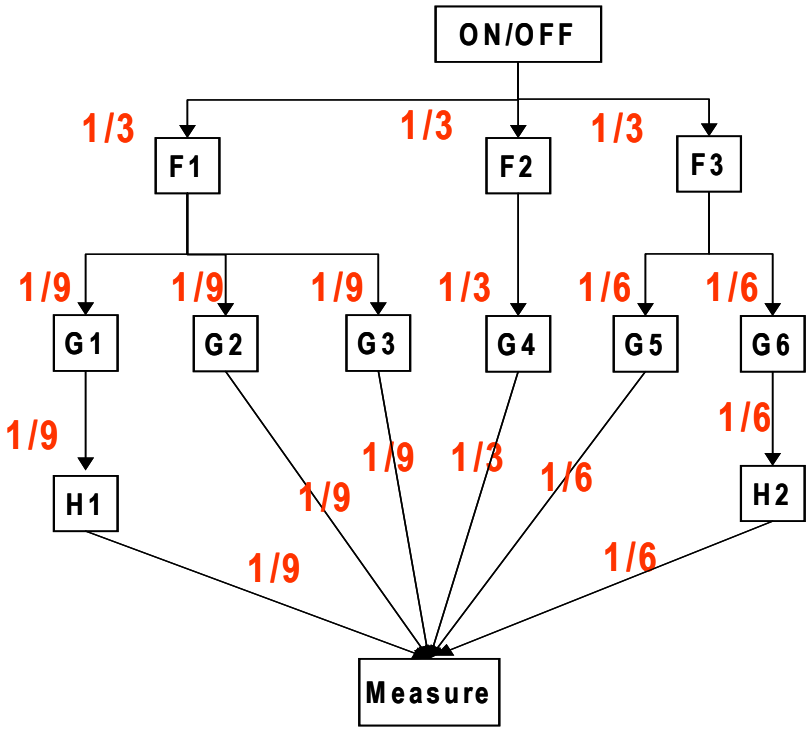

Figure 2. The diagram illustrates the method of weight calculation

The functions F1, F2, F3 are located on the front panel of the virtual instrument, while G1-G3, G4, G5-G6 functions are located on panels which are called from the earlier selection of the F function, etc. The weight matrix for Fig. 2. is as follows:

$$w_f = \begin{bmatrix} w_{ON-ON} & w_{ON-F1} & w_{ON-F2} & \cdots & \cdots & \cdots & w_{ON-Me} \\ w_{F1-ON} & w_{F1-F1} & w_{F1-F2} & & & & w_{F1-Me} \\ w_{F2-ON} & w_{F2-F1} & w_{F2-F2} & & & & w_{F2-Me} \\ w_{F3-ON} & w_{F3-F1} & w_{F3-F2} & \cdots & \cdots & \cdots & w_{F3-Me} \\ w_{G1-ON} & w_{G1-F1} & w_{G1-F2} & & & & w_{G1-Me} \\ w_{G2-ON} & w_{G2-F1} & w_{G2-F2} & & & & w_{G2-Me} \\ w_{G3-ON} & w_{G3-F1} & w_{G3-F2} & \cdots & \cdots & \cdots & w_{G3-Me} \\ w_{G4-ON} & w_{G4-F1} & w_{G4-F2} & & & & w_{G4-Me} \\ w_{G5-ON} & w_{G5-F1} & w_{G5-F2} & & & & w_{G5-Me} \\ w_{G6-ON} & w_{G6-F1} & w_{G6-F2} & & & & w_{G6-Me} \\ w_{H1-ON} & w_{H1-F1} & w_{H1-F2} & & & & w_{H1-Me} \\ w_{H2-ON} & w_{H2-F1} & w_{H2-F2} & & & & w_{H2-Me} \\ w_{Me-ON} & w_{Me-F1} & w_{Me-F2} & \cdots & \cdots & \cdots & w_{Me-Me} \end{bmatrix} \quad (11)$$

The value of weight, for which the defined combination of parameters does not exists, will have a value of 0. The determination of probability of selection of a defined function (or combination of functions) has a special meaning in case of large projects, when definite funds are earmarked for error removal. It is possible to select which errors and in which order should be removed before the virtual instrument is delivered to its users. Often, despite the precise determination of the measurement scenario, its inaccurate implementation by the programmer causes the so-called “mutations”, that is the possibility of selection of functions or combination of function incompatible with the measurement scenario. Despite the previous determination in the weight matrix that the probability of selection combinations of parameters is 0, tests for this combination parameters should be carried out. Due to the fact that combinations of parameters are tested, the parametric method for functional testing of virtual instruments allows to state the existence of mutations and to estimate the risk related to their implementation. The matrix f for the measurement scenario form the Fig. 2. is given:

$$f = \begin{bmatrix} f_{ON-ON} & f_{ON-F1} & f_{ON-F2} & \cdots & \cdots & \cdots & f_{ON-Me} \\ f_{F1-ON} & f_{F1-F1} & f_{F1-F2} & & & & f_{F1-Me} \\ f_{F2-ON} & f_{F2-F1} & f_{F2-F2} & & & & f_{F2-Me} \\ f_{F3-ON} & f_{F3-F1} & f_{F3-F2} & \cdots & \cdots & \cdots & f_{F3-Me} \\ f_{G1-ON} & f_{G1-F1} & f_{G1-F2} & & & & f_{G1-Me} \\ f_{G2-ON} & f_{G2-F1} & f_{G2-F2} & & & & f_{G2-Me} \\ f_{G3-ON} & f_{G3-F1} & f_{G3-F2} & \cdots & \cdots & \cdots & f_{G3-Me} \\ f_{G4-ON} & f_{G4-F1} & f_{G4-F2} & & & & f_{G4-Me} \\ f_{G5-ON} & f_{G5-F1} & f_{G5-F2} & & & & f_{G5-Me} \\ f_{G6-ON} & f_{G6-F1} & f_{G6-F2} & & & & f_{G6-Me} \\ f_{H1-ON} & f_{H1-F1} & f_{H1-F2} & & & & f_{H1-Me} \\ f_{H2-ON} & f_{H2-F1} & f_{H2-F2} & & & & f_{H2-Me} \\ f_{Me-ON} & f_{Me-F1} & f_{Me-F2} & \cdots & \cdots & \cdots & f_{Me-Me} \end{bmatrix} \quad (12)$$

The occurrence of mutations as a result of incorrect implementation can be calculated from the formula (4, 5) in following procedure:

$$Mut = \begin{bmatrix} m_{f(1,1)} & m_{f(1,2)} & \cdots & m_{f(1,j)} \\ m_{f(2,1)} & & & m_{f(2,j)} \\ \vdots & & & \vdots \\ m_{f(i,1)} & & & m_{f(i,j)} \end{bmatrix} \tag{13}$$

$$m_{f(i,j)} = 1 \text{ for } N_{t(i,j)} \neq 0 \tag{14}$$

III. Conclusion

The parametric method for functional testing of virtual instruments presented in this paper makes it possible to perform tests of the virtual instrument already on the stage of the design of user interface (information flow model), and on the stage when virtual instrument has been prepared. The parametric method for functional testing can fill the gap resulting form the lack of functional methods, which are dedicated to the testing of measuring devices with the special consideration of virtual instruments. The specifics of virtual instruments in the design of user interface has been taken into account in the method, because tests can be carried out not only within the measurement scenario, but also outside the measurement scenario. The determination of probability of selection combination of parameters permit to estimate the risk related to the implementation such combination of parameters, and thus, it is possible to perform works over the user interface already during the works on its prototype. The verification of the prototype implementation correctness permit to affirm, which mutations occur as a result of the algorithm implementation process in a defined programming language, as well as whether these mutations cause incorrect or instable operation of the virtual instrument. The parametric method does not force a definite number of tests, which must be performed to test the virtual instrument. The number of necessary tests depends on the required precision, on the time in which the test should be performed.

References

[1] W. Winiecki, P. Bilski, *"Time Analysis of Virtual Spectrum Analyzer",* International Scientific Journal of Computing, Vol. 3, Issue 2, pp. 913-918, 2004

[2] IEC60300-3-6

[3] N.Juristo, A.Moreno S.Vegas, Reviewing 25 Years of Testing Technique Experiments, *Empirical Software Engineering*, 9, (2004), s. 7–44,

[4] Ross K. Practical guide to software system testing, K. J. Ross & Associates Pty. Ltd., (1998).

[5] T.Clement, L.Emmet, P.Froome, S.Guerra, "*Best Practice Guide on the Development of Test and Measurement Software*", Adelard and Crown, 2002

[7] R.Baker, G. Parkin, "*Techniques for Validation of Measurement Software – without specialised tools"*, NPL Report CMSC 43/04, National Physical Laboratory, Queens Road, Teddington, Middlesex, 2004

[8] B.Beizer, "*Black-Box Testing: Techniques for Functional Testing of Software and Systems*", John Wiley & Sons,1995

[9] G.J.Myers, *The Art of Software Testing,* John Wiley&Sons, New York, 1976.

[10] T.Daboczi, I.Kollar, G.Simon, T.Megyeri, *"How to test graphical user interfaces*", IEEE Instrumentation & Measurement Magazine, vol. 6, pp. 27-33, 2003.

[11] T.Clement, L.Emmet, P.Froome, S.Guerra, "*Best Practice Guide on the Development of Test and Measurement Software*", Adelard and Crown, 2002

[12] M.Ramachandran, **"***Testing software components using boundary value analysis",* Proceedings of 29th Euromicro Conference, 1-6 Sept. 2003, Belek-Antalya, Turkey, pp. 94-98, 2003.