XIX IMEKO World Congress
Fundamental and Applied Metrology
September 6–11, 2009, Lisbon, Portugal

AUTOMATICALLY-GENERATED USER INTERFACES FOR MEASUREMENT SOFTWARE FRAMEWORKS: A CASE STUDY ON MAGNETIC PERMABILITY AT CERN

Pasquale Arpaia[1,2], *Marco Buzio*[2], *Lucio Fiscarelli*[1,2], *Vitaliano Inglese*[1,3], *Giuseppe La Commara*[1]

[1] Dipartimento di Ingegneria, Università del Sannio, Benevento, Italy, arpaia@unisannio.it
[2] Dept. of Technology, Group of Magnets Superconductors Cryostats, CERN, Geneva, Switzerland, {marco.buzio-lucio.fiscarelli-vitaliano.inglese}@cern.ch
[3] Dipartimento di Ingegneria Elettrica, Università degli Studi di Napoli – Federico II, Napoli, Italy

Abstract – A technique for generating user interfaces in software frameworks for automatic measurement systems is proposed. The user interface is separated from the application logic with the aim of enhancing flexibility and reusability of the software. A Model-View-Interactor paradigm focuses on the “interaction” between the automatic measurement system, executing test script written by a test engineer, and the application user. This approach has been applied to the flexible software framework for magnetic measurements at the European Organization for Nuclear Research (CERN). Experimental results on the application of the proposed technique to a case study of permeability measurement are reported.

Keywords: software measurement systems, magnetic measurements, automatic test equipment (ATE)

1. INTRODUCTION

In the industrial area, the need for software products, supporting users in developing measurement applications is increasing progressively. Building measurement test utilities through the suitable assistance of a specific framework allows resources to be saved during the development stage as a whole.

At CERN, the European Organization for Nuclear Research, the design and the implementation, first of the Large Hadron Collider (LHC) and now of the LINAC4 accelerator, has required a big effort to test the high performance magnets and stimulated new requirements for magnetic measurement software [1]. In cooperation with the University of Sannio, the Flexible Framework for Magnetic Measurements (FFMM) was designed to satisfy a wide range of measurement demands and to integrate more performing flexible hardware [2]. FFMM software applications can pilot several devices, such as encoder boards, digital integrators, motor controllers, transducers, and so on, as well as synchronize [3]-[4] and coordinate different measurement tasks and actions [5]. Such as the new generation of measurement frameworks [6], in addition to satisfy all the functional requirements, FFMM has to provide tools to generate the Graphical User Interface (GUI) automatically [7].

During the development of a measurement application through a software framework [1] in a first phase, the Test Engineer describes the measurement process formally by a suitable script [11]. From this input, the framework generates an executable measurement application as output. In a second phase, the Application User executes the measurement application, interacts with the system at runtime, by providing the required input and by configuring the hardware setup, and finally starts the measurement process on the devices.

As for most interactive applications, producing an attractive GUI for a measurement software framework is not an easy task. The powerful GUI libraries offered by the operating system can be used of course, but the offered level of abstraction is rather low in general. Therefore, a visual editor, such as offered by many commercial programming environments [8], should be used. Such tools turn out to be very user-friendly at the expense of offering limited functionality. Inherently, graphical representations depending on run-time data cannot be drawn in advance. Summarizing, a visual editor is a useful tool for simple GUI applications, but for more complicated ones, the Test Engineer still has to struggle with low-level programming code. In addition, the quality of manual GUI development depends strongly on the experience of the designers and their skills in the platform and development tools.

User interface generation has been the subject of research for many years, sometimes under the name of model-based user interfaces generation [8]. In fact, interfaces are generated by dividing the application domain in models. The main goal of automatic techniques for generating interfaces is to allow the designer to specify them at a very high level, with the details of the implementation to be provided by the system [9].

Nevertheless, this approach is very unspecific and a further effort is required to tailor the model to a definite context, such as in frameworks for measurement software applications.

In this paper, an evolution of model-based user interface generation for a measurement software framework, the

Model-View-Interactor paradigm, shifting the test Engineer from designing interfaces to designing interactions, is proposed. The practical goal is to allow programmers, as Test Engineers, who are not typically trained to design interfaces, to produce easily good GUIs for their applications. Designing interaction rather than interfaces attempts to control the quality of the interaction between user and computer, according to the main paradigm that "user interfaces are the means, not the end" [10].

2. THE *MODEL-VIEW-INTERACTOR* PARADIGM

The proposed approach to the generation of interfaces in measurement system frameworks starts from a fundamental consideration: usually Test Engineers are not trained to design interfaces, but at the same time they would like to maintain a high level of usability in measurement applications. They are responsible for preparing Test Scripts [2], where the interaction between Measurement Application and final user are described at high level [11], without any indication of GUI aspects.

Therefore, the main concept underlying the proposed approach is the interaction. Interaction is a kind of action occurring when two or more objects have an effect upon one another. Examples of simple interaction in measurement software are reading an user input, or displaying a value.

To avoid that Test Engineers have to deal with raw graphical characteristic of software measurement system, the proposed architecture is organized through a three-way decomposition, by separating functional from look aspects of the interface: (i) the parts representing the model of the underlying application domain, (ii) the way the model is presented to the user, and (iii) the way the user interacts with it.

This proposal is called the *Model-View-Interactor* approach (Fig.1), derived as an evolution of the model-based approach [7] [8].

The *Model* is composed by the data structures and the classes of the framework involved in the GUI generation and subject to change by them. Typical example is offered by the device classes concerned to the configuration step of the measurement procedure.

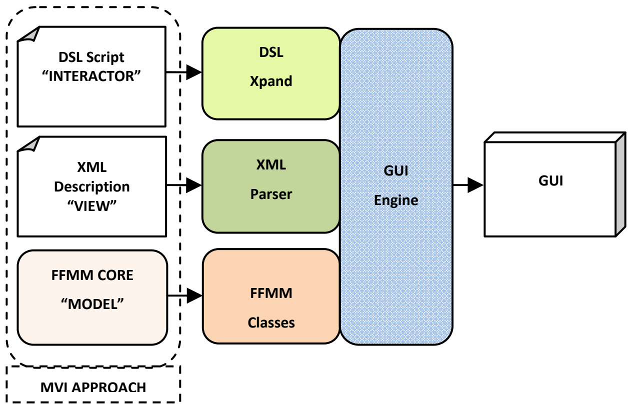


Fig.1. MVI Architecture.

The *View* consists in the aspect of the generated GUI, defined by the GUI expert in the View Description, a XML-file containing all the presentation features of the GUI and handled by a XML Parser, completely transparent to the Test Engineer using the framework. In particular, the user interfaces content may be organized in rectangular areas, or areas suitably described by a rectangular bounding shape (referred here as boxes). Graphical user interface layouts can be seen as a container subdivided in boxes, where graphic components (text editor component, buttons, menu item, and so on) are placed. A box can contain others boxes, and so on. A Layout Manager is responsible to arrange all the components in the resulting form [12].

The Interactor represents the tie between model and view, by making available different components specifying the GUI desired behaviour. In the measurement script writing phase [11], the Test Engineer defines the component contained in the GUI and the type of input/output data by means of the Interactive Components. Then, after the building process of the script made by the DSL-Xpand component, the framework is able to generate the application with the desired the GUI.

In this way, the Test Engineer can define the interaction Measurement Application-User by means of the Interactor Component objects only.

3. THE GUI ENGINE

The main aim of the proposed Model-View-Interactor Paradigm is to permit the Test Engineer to develop GUI applications by a minimal effort and without graphical knowledge. This aim is achieved mainly through the GUIengine structure (Fig.2), allowing all the GICs (Graphic Interactor Components), encapsulating all common aspect of graphical components [13] [14], to be built.

The GUI engine architecture is composed by several classes: (i) *GIC*, providing to *TestManager* the input/output features without graphical details, (ii) *GenericWindow*, giving the interface for all the frames, (iii) *InputWindow* and *OutputWindow*, the concrete windows, and (iv) *LayoutManager*, responsible for instantiating concrete windows defining the graphical features parsing the View Description file and computing the dimension and position parameters [12]. The View is kept clear-cut from the Interactor by implementing the GUIengine complying with the Abstract Factory Pattern, often employed to separate the details of GUI implementation from its general use.

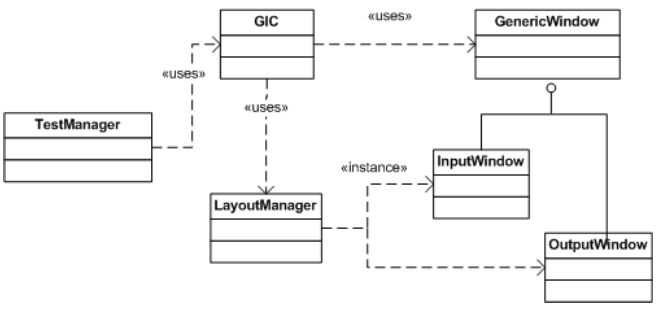


Fig.2. Abstract Factory Pattern for the GUIengine.

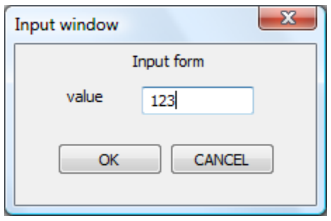


Fig.3. Final form aspect.

As an example, if the Test Engineer needs for asking as input a integer value at runtime, he will use the *capture()* method of GIC object in the measurement script [1]:

```
int a;
gic.capture(a,1,"Input form:","value");
```

By inserting in the script only this instruction, a form is displayed (Fig.3), and the value entered by the user is stored in the variable pointed.

4. EXPERIMENTAL RESULTS

The Flexible Framework for Magnetic Measurement (FFMM) is a software platform under development at CERN in cooperation with the University of Sannio [1], [5]. FFMM is aimed at generating in a systematic way all the measurement software applications for testing the particle accelerator magnets.

In the following, (i) the *case study of magnetic permeability measurement*, and (ii) the *measurement results* are reported.

4.1. Case Study of Magnetic Permeability Measurement

The proposed case study is aimed at illustrating how the proposed approach supports Test Engineers in generating the GUI automatically for a measurement procedure based on the methods of the split-coil permeameter (Fig. 4) [15].

The split-coil permeameter (Fig.5) is composed by two coils wound in a toroidal shape, that can be opened allowing to wrap a toroidal specimen. One coil is to excite the field and the other one to capture the flux.

A PC, hosting the FFMM Automatically-generated User Interface (AUI), is linked to a DAQ [18], in order to control the Voltage Controlled Power Supply of the excitation coil by the analog output.

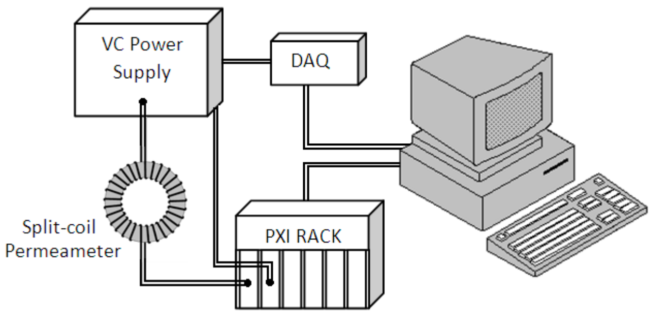


Fig.4. Permeameter bench

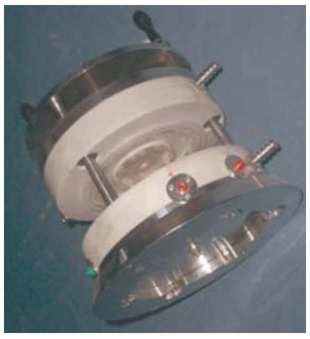


Fig.5. The split-coil permeameter.

A PXI crate containing:

- two CERN FDI (Fast Digital Integrator), a CERN proprietary PXI board general-purpose digitalization board, configured for the coil signal acquisition and numerical integration [2],
- a CERN Encoder board, a CERN proprietary PXI board, for managing the encoder pulses and feeding the trigger input of a digital integrator,

is also linked to the PC and used to acquire, through the FDIs, the value of the excitation current, the relative flux, and the trigger generated by the Encoder Board (Fig.4).

The measurement algorithm is composed by the following steps:

1) setup of all the device needed in the procedure;
2) demagnetization of the specimen [15];
3) start the acquisition of flux and current;
4) start the generation of the signal controlling the power supply;
5) wait for the reaching of the selected maximum current value;
6) stop the acquisition.

This procedure is codified in the application script and processed by the FFMM framework in order to produce executable file.

5.2. Measurement Results

To setup the devices involved in the measurement procedure, the AUI features of FFMM are used.

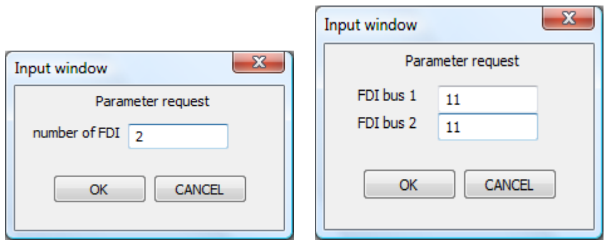
As an example, at the beginning of the measurement script, the Test Engineer needs to configure the FDIs: the number of FDI and their bus are required to start the acquisition. Thus, the Test Engineer puts in the script the following statements:

```
gic.capture(numFDI,1,"Parameter Request:","number of FDI");
gic.capture(bus, numFDI,"Parameter Request:","FDI bus");
```

Then, during the application execution, the forms are generated. In Fig.6, the examples of the GUI generated correspondingly are shown.

A steel specimen was tested and, according to the procedure explained in [15], the permeability characteristic curve may be obtained by analysing the data.

For validating the Automatically-generated User Interface by Model-View-Interactor paradigm, inn Fig. 7, the hysteresis curve, obtained by plotting the flux versus the feed current, is reported.


Input window

Parameter request

number of FDI 2

OK CANCEL

Input window

Parameter request

FDI bus 1 11

FDI bus 2 11

OK CANCEL

Fig.6. FDI configuring forms.

5. CONCLUSIONS

The Model-View-Interactor paradigm for Automatically-generated User Interface was proposed. The main purpose of this technique was to allow Test Engineers using FFMM to produce easily the GUI for their measurement applications. A complex and realistic case study, the magnetic permeability measurement, was treated and both the graphical and the measurement results were shown.

REFERENCES

[1] P. Arpaia, L. Bottura, M. Buzio, D. Della Ratta, L. Deniau, V. Inglese, G. Spiezia, S. Tiso, L.Walckiers, "A software framework for flexible magnetic measurements at CERN", in Proc. of IEEE IMTC 07, Warsaw, Poland, May 2007.

[2] P. Arpaia, A. Masi, G. Spiezia, "A Digital Integrator for Fast Accurate Measurement of Magnetic Flux by Rotating Coils", IEEE Transactions on Instrumentation and Measurement, Vol. 56, No. 2, April 2007.

[3] P. Arpaia, M. Bernardi, G. Di Lucca, V. Inglese, G. Spiezia, "An Aspect Oriented Programming-based approach to software development for measurement system fault detection", in press on Computer, Standards & Interfaces.

[4] P. Arpaia, M. Bernardi, G. Di Lucca, V. Inglese, G. Spiezia, "Aspect Oriented-based Software Synchronization in Automatic Measurement Systems", Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE Volume , Issue , 12-15 May 2008 Page(s):1718 – 1721.

[5] P. Arpaia, L. Bottura, V. Inglese, G. Spiezia, "On-field validation of the new platform for magnetic measurements at CERN", Measurement, Vol. 42, No. 1, January 2009, Pages 97-106.

[6] J. Bosch, "Design of an Object-Oriented Framework for Measurement Systems" *Domain-Specific Application Frameworks*, M. Fayad, D. Schmidt, R. Johnson (eds.), John Wiley, ISBN 0-471-33280-1, 1999, pp. 177-205

[7] K. Stirewalt, S. Rugaber, "Automating UI Generation by Model Composition", submitted to: Automated Software Engineering (ASE '98) 13th IEEE International Conference, 1998 .

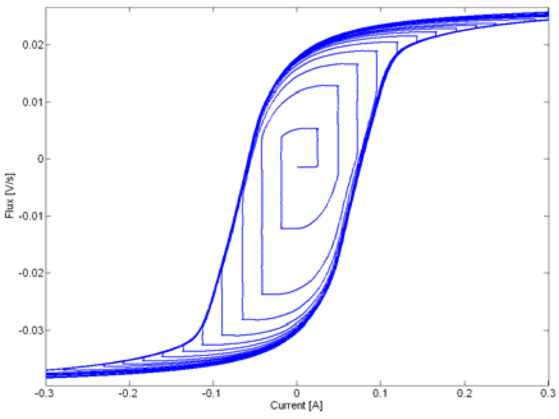


Fig.7. Flux vs current hysteresis curve.

[8] B. Mayers, S.E. Hudson, R. Pausch, "Past, Present and Future of User Interface Software Tools" ACM Trans. Computer-Human Interaction, vol. 7, no.1, pp 3-28, March 2000.

[9] T. P. Browne et al., "Using declarative descriptions to model user interfaces with MASTERMIND". In F. Paternò and P. Palanque editors, Formal Methods in Human Computer Interactions, Springer-Verlag, 1997.

[10] M. Beaudouin-Lafon, "Interactions as First-class Objects" In Proceedings of the ACM CHI 2005 Workshop on the Future of User Interface Design Tools. ACM Press 2005.

[11] P. Arpaia, M. Buzio, L. Fiscarelli, V. Inglese, G. La Commara, "Measurement-Domain Specific Language for Magnetic Test Specifications at CERN". Proc. I2MTC09, Singapore, May 5-7, 2009.

[12] C. Lutteroth, G. Weber, "Modular Specification of GUI Layout Using Constraints", Proceedings of ASWEC 2008 - 19th Australian Conference on Software Engineering, IEEE Press, 2008.

[13] P. Achten, M. van Eekelen, and R. Plasmeijer, "Compositional Model-Views with Generic Graphical User Interfaces". In Practical Aspects of Declarative Programming, PADL04, volume 3057 of LNCS, Springer, 2004.

[14] P. Achten, M. van Eekelen, and R. Plasmeijer, "Generic Graphical User Interfaces". In Greg Michaelson and Phil Trinder, editors, Selected Papers of the 15th Int. Workshop on the Implementation of Functional Languages, IFL03, volume3145 of LNCS. Edinburgh, UK, Springer, 2003.

[15] K. N. Henrichsen, "Permeameter", Proc. 2nd Int. Conf. on Magnet Technology, Oxford, 1967.

[16] http://sine.ni.com/nips/cds/view/p/lang/en/nid/1037