A flexible software framework for magnetic measurements at CERN: a prototype for the new generation of rotating coils

P. Arpaia[1], L. Bottura[2], V. Inglese[3], G. Spiezia[3]

[1] Department of Engineering, Universityof Sannio, Corso Garibaldi 107, 82100 Benevento, Italy.
Ph : +39 0824 305804-17, Fax: +39 0824 305840, E-mail: arpaia@unisannio.it
[2] CERN, Dept. AT (Accelerator Technology), Group MTM, CH 1211, Genève 23, Switzerland,
Ph : +41 22 76 76635, Fax: +41 22 76 76230, E-mail: Luca.Bottura@cern.ch
[3]Department of Engineering, University of Naples - Federico II, Via Claudio 21, 80125 Napoli, Italy;
CERN, Dept. AT (Accelerator Technology), Group MTM, CH 1211, Genève 23, Switzerland,
Ph : +41 22 76 76635, Fax: +41 22 76 76230, E-mail: {Vitaliano.Inglese-Giovanni.Spiezia}@cern.ch

Abstract- The new software platform FFMM (Flexible Framework for Magnetic Measurements) under development at CERN (European Organization for Nuclear Research) in cooperation with the University of Sannio is presented. The FFMM is aimed at facing the new test requirements arising after the production series of the Large Hadron Collider magnets. In particular, the basic concepts of the FFMM, its architecture, and the experimental implementation of a proof demonstrator are illustrated in order to show how the quality requirements of software flexibility and scalability are met.

I. Introduction

At CERN (European Organization for Nuclear Research), the design and the implementation of the LHC (Large Hadron Collider), the biggest particle accelerator over the world, required a big effort in all the engineering fields. The test of the LHC superconducting magnets stimulated new stricter requirements for magnetic measurements [1]. In particular, the big effort required for the test of the all LHC magnets brought to an incremental development of the measurement software, without concentrating on its quality, namely flexibility and reusability. At the moment, the end of the test on the LHC magnets, marks a change in the requirements: a standard platform is required to put together all the magnetic measurement applications. In addition, now test engineers need to perform specialized tests on magnet batches of small-medium entity. As a consequence, a flexible software framework is demanded in order to satisfy the new magnetic measurement requirements highly evolving during time. Jan Bosch was one of the first to apply the concept of framework in the measurement field by proposing an object-oriented project capable of satisfying a wide range of applications [2]. At commercial level, National Instrument (NI) proposes the product NI TestStand [3] for supporting the user in designing new test applications, by integrating software modules developed in different programming languages (C, C++, LabView^{TM}). However, NI TestStand does not help the user in developing single software modules, by not assuring standard development and reusability intrinsically. So far magnetic measurements were performed at CERN using a LabView-based application (MMP, Magnetic Measurement Program) [4]. As said above, MMP underwent an incremental development whose plan had to be adapted over several years from an initial prototyping phase to the present status of general control and acquisition software for a multitude of devices interconnected in a magnetic measurement system. One of the main drawbacks observed now is that his software, as any other of the same type, does not allow the user to change easily test and analysis algorithms. Specifically, changes in the code needed to adapt MMP to the new requirements must be performed by software specialists. A number of alternative solutions have been examined. The first is the FESA (Front-End Software Architecture) paradigm adopted at CERN for the LHC controls [5]. FESA has been developed to provide a suitable front-end for all the PCs interfacing the LHC control instruments. However, the analysis of this software showed that a strong collaboration and involvement at the lowest software level of FESA would be required in order to adapt the architecture to the abovementioned applications. Outside CERN, at Fermi National Accelerator Laboratory, a new software system to test accelerator magnets has been developed to handle various types of hardware, as well as to be extensible to all measurement technologies and analysis algorithms [6]. At to date, this software is still under development and not worldwide accessible. Finally, other sub-nuclear research centres (Alba, Soleil, Elettra, and ESRF) are consociated in order to develop a suitable software framework for testing

accelerator magnets [7]. This Consortium proposes TANGO, an object-oriented system to handle different measurement applications.
In this paper, a Flexible Framework for Magnetic Measurements (FFMM), under development at CERN in cooperation with the University of Sannio with the aim of proving the feasibility of a flexible and reusable system to face the new test requirements arising after the production series of the LHC magnets, is presented. In the following sections, the FFMM basic ideas, the architecture, and the implementation of a proof demonstrator are described.

II. Proposal

The FFMM is a software framework for magnetic measurement applications based on Object Oriented Programming (OOP), and Aspect-Oriented Programming (AOP) [8]. In particular, FFMM aims at supporting the user in developing software maximizing quality in terms of flexibility, reusability, maintainability, and portability, without neglecting efficiency, vital in test applications. Moreover, the requirements for a wide range of magnetic measurement applications, as required for the test of superconductive magnets for particle accelerators, have to be satisfied.
In the following, (i) the basic ideas, and (ii) the architecture of FFMM are described.

A. Basic Ideas

FFMM is based on the following basic ideas:

(i) a group of interfaces and abstract classes represents a *white-box layer* defining the high-level structure of FFMM used to generate new parts of the framework. This allows potentiality and flexibility of FFMM to be extended;

(ii) a group of modules, already available to the test engineer (end user), represents a *black-box layer*, allowing both module reusability and use easiness to be achieved, even by test engineers without deep knowledge of internal FFMM mechanisms;

(iii) Aspect-Oriented Programming (AOP) improves the reusability and the maintainability of FFMM: in large projects, several concepts are transversal to many modules (*cross-cutting concerns*). They are extrapolated form the native units and implemented in separated modules (*aspects*), in order to improve the system modularity (maintainability enhancement);

(iv) a suitable definition of the code structure (normalization of structures and software modules) gives rise to standard modules, representing the basic library for the realization of new components and the extension of already existing ones;

(v) a library of reusable modules is built incrementally during the start-up of the framework up to a “saturation” condition inside an application domain, allowing further requirements in the same domain to be satisfied by a limited effort.

B. FFMM Architecture

On the basis of the above basic ideas, the FFMM architecture shown in Fig. 1 was conceived. The test

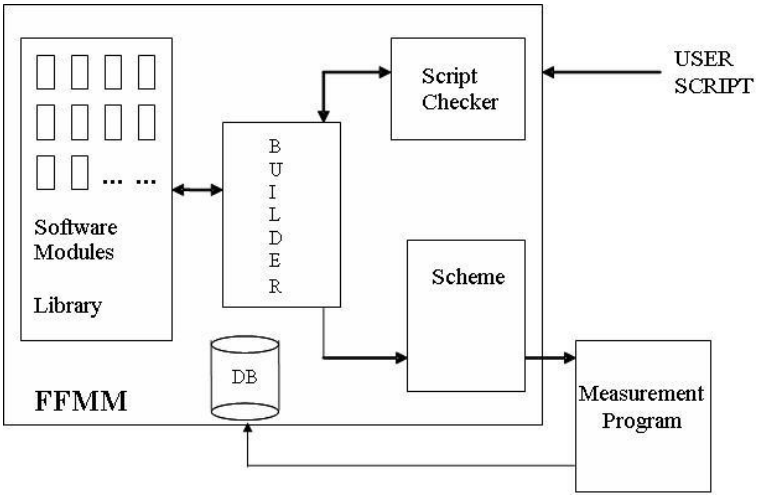

Figure 1. The FFMM architecture.

engineer (end user) produces a description of the measurement application, *User Script*, whose semantic and syntactic correctness is verified by the *Script Checker*. Then, from the *User Script*, the *Builder* assembles the *Measurement Program*, according to the architecture of the *Scheme*, by picking up suitable modules from the *Software Module Library*. If some modules are not available in the library, a template is provided to the user (administrator user) in order to implement them according to a suitable predisposed structure. Once debugged and tested, the *Measurement Program* will be stored in the *Database* in order to be reused.
According to the analysis of typical use-case tests on superconductive magnets, the generic *User Script* is organized into the following phases:

- definition of the measurement components;
- specification of mechanical and electrical connections;
- definition of dynamic parameters, i.e. configurable during run-time of the *Measurement Program*;
- component checking;
- configuration of measurement devices;
- description of the measurement procedure;
- preliminary data analysis;
- data saving.

The architecture of the FFMM core, the *Scheme*, is shown in Fig. 2: the *TestManager* organizes the test by knowing the *Unit Under Test*, the *Quantity* to be measured, the measurement configuration, and the measurement procedure. *TestManager* has an association with the *Devices* (software representation of the measurement devices). Among *Devices*, *Virtual Devices* can be controlled remotely by PC through a *Communication Bus*.
The *Synchronizer* and the *FaultDetector* are units managing critical topics in a measurement application. The *Synchronizer* manages the software temporization in the measurement procedure, while the *FaultDetector* intercepts malfunctions and errors. The *Synchronizer* and the *FaultDetector* can be considered cross-cutting concerns, because they are transversal to many software modules. As a matter of fact, the synchronization policy involves all the measurement devices and all the test procedures. Furthermore, the fault detection is a fundamental part of all the devices, as well as of the measurement system as a whole. Then, the *Synchronizer* and the *FaultDetector* are encapsulated in *Aspects* according to AOP approach. Therefore, the policy for managing synchronization actions and faults can be extrapolated from the single modules and handled separately. In this way, further modifications will affect only those two components, without any need for code changes in all the modules related to the fault detection or to the synchronization.
The *Logger* class handles the stock up of configuration and measurement data, as well as system warnings and exceptions.

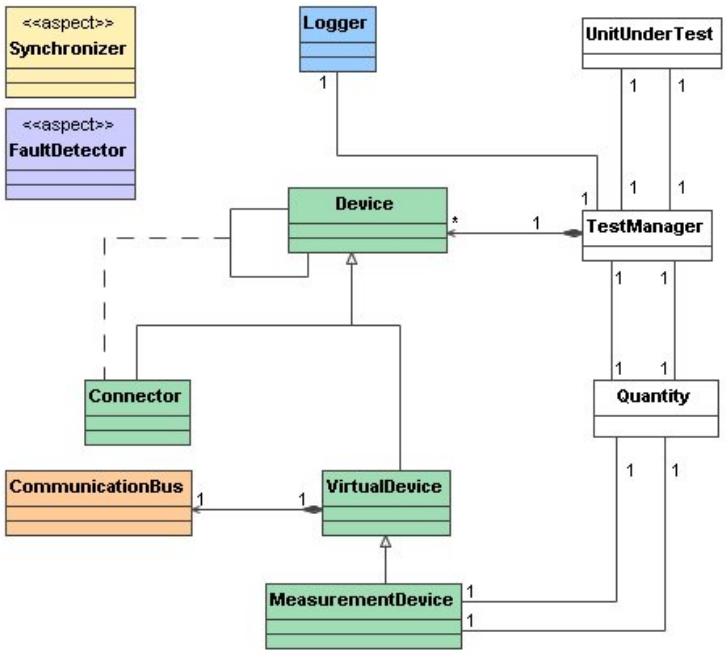

Figure 2. Architecture of the *Scheme*.

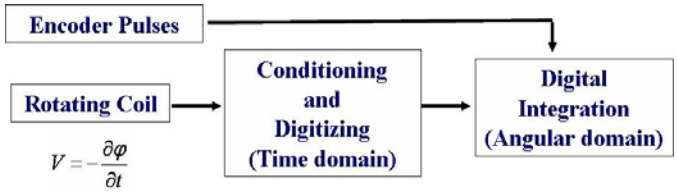

Figure 3. Rotating coil measurement principle.

III. Experimental results

According to this architecture, a proof demonstrator was implemented in order to verify the FFMM principle feasibility: the framework "generates" a measurement program according to specifications, written by a test engineer who must be familiar with the main framework interfaces.
The first FFMM test was carried out for a measurement application based on rotating coils [9], one of most accurate technique for superconductive magnet testing.
The *Scheme* was implemented in C^{++} by individuating the most suitable design patterns for the different phases of the measurement procedure. The low-level code, directly related to the hardware components, was structured according to the most suitable real-time pattern in order to assure efficiency and reliability.
In the following, (i) *the rotating coil technique and the demo bench*, and (ii) *the measurement application description* are illustrated.

A. Rotating Coil Technique and Demo Bench

In the rotating coils test technique (Fig. 3) [9], a coil turns into the magnet under test and its output signal is proportional to the flux derivative, according to Faraday's law. The coil signal is integrated in the angular domain by means of the output pulses of an encoder mounted on the rotating coil shaft. A Fourier analysis of the flux finally yields the multipoles of the magnetic field generated by the magnet under test.
The realized demo bench is based on the following main parts:

- a motor controller MAXON PCU 2000, accessible through RS232, for handling the shaft motor turning the coil inside a permanent magnet;
- an encoder board: a CERN proprietary PXI board, for managing the encoder pulses and feeding the trigger input of the FDI;
- a Fast Digital Integrator (FDI): a CERN proprietary PXI general-purpose digitalization board, configured for the coil signal acquisition and numerical integration [10].

These components and the prototyped demo bench are shown in Fig. 4: the coil signal and the encoder

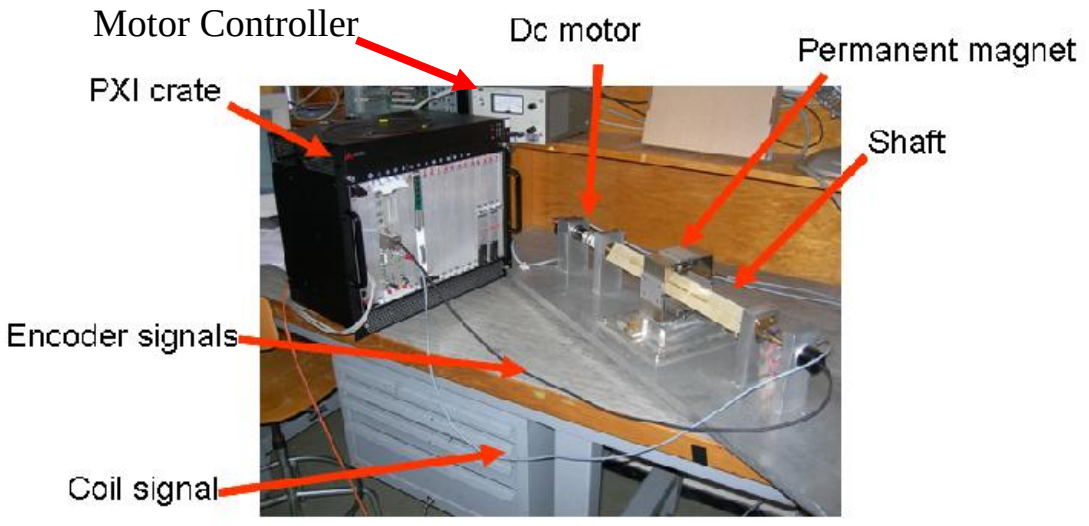

Figure. 4 The prototyped demo bench.

cables feed the FDI and the encoder boards installed in a PXI crate. The DC motor is controlled by the MAXON PCU 2000 in order to move the coil shaft at a constant speed. A permanent magnet was used as *unit under test* to generate the magnetic field.

B. Measurement Application Description

The *User Script*, describing the application measurement, is a method of the *TestManager* class, written by the test engineer. An example of the script syntax is shown in Fig. 5, limited for reasons of space only to the description of the measurement procedure. In future, a suitable user interface to simplify such a task will be provided.

According to the abovementioned phases, the *User Script* is organized into the following parts:

- definition of FDI, encoder board, and motor controller by using the *abstract factory* design pattern [11];
- the *Synchronizer* and the *Logger* are also instantiated;
- determination of the connection among the components and between the components and the PC (bus connection);
- definition of the dynamic parameters, configurable at the run-time of the *Measurement Program*: the test engineer defines as variables the parameters that will be inserted by the operator. These variables are read in at run-time and are used to specify the parameters that are not yet known by the test engineer at the moment of the preparation of the script (e.g., the device bus address), as well as parameters that are directly related to the specific aspects of the measurements (e.g., motor rotation speed, gain of the amplifiers in the FDI, and other);
- component check: the method *CheckDevices* of the *FaultDetector* class will be invoked by the test engineer. The check method of the instantiated devices will be performed. The results are logged and possible errors are printed out. This part is transparent to the test engineer;
- configuration of the measurement components: the test engineer configures the FDI, the Encoder board, and the motor controller, by using the interface services of the corresponding classes;
- description of the measurement procedure (Fig. 5): the test engineer has to specify: the measurement tasks to be carried out (lines 1-2, Fig.5), the actions (i.e. motor start) and their temporization. With this aim, the test engineer will use the interface services of the devices as well as the interfaces of some tools, such as the *Synchronizer*, and the *Observer* design pattern [11]. The test engineer specifies the measurement tasks and the devices that are its observers (lines 3-4). For each observer, the test engineer specifies the event related to the measurement task to which the observer must react (line 5), as well as the actions to be performed in case of such an event (lines 6-7). The *Synchronizer* provides the services for specifying action to be performed in parallel, by defining suitable groups (lines 8-13);
- data are saved through the services of the *ReportLogger* interface. This can be part of the measurement phase (e.g. in case of binary data saving for a continuous acquisition). This

```
//MEASUREMENT PROCEDURE

// Definition of the measurement tasks
MeasurementTask* mt1 = new MeasurementTask( fdi1 ) ;                                    //line 1
MeasurementTask* mt2 = new MeasurementTask( fdi2 ) ;                                    //line 2

//Definiton of the observers of mt1 measuremnt task: Measurement logger
mt1->addObserver(measurementLogger);                                                    //line 3
mt2->addObserver(measurementLogger);                                                    //line 4
// Definition of the condition which establishes if Measurement logger needs to be updated: logcondition is a
//function
measurementLogger->needUpdate(logCondition);                                            //line 5

//Definition of the action to be performed when Measurementlogger is updated; Writedata is a function
measurementLogger->setAction(mt1,writeData);                                            //line 6
measurementLogger->setAction(mt2,writeData);                                            //line 7

//Definition of the group of tasks and actions to be synchronized
Synchronizer.addGroup("gruppo");                                                        //line 8
Synchronizer.addTask("gruppo",mt1);                                                     //line 9
Synchronizer.addTask("gruppo",mt2);                                                     //line 10
Synchronizer.addAction("gruppo",new Action(mc,START));                                  //line 11
Synchronizer.addAction("gruppo",new Action(mul,START));                                 //line 12
Synchronizer.start("gruppo");                                                           //line 13
```

Figure 5. Measurement procedure of the *User Script* for the rotating coil measurement application.

choice is demanded to the test engineer. The data saving can be time consuming and is usually characterized by a lower priority than the acquisition process, thus the *ReportLogger* interface provides the services to launch the storing in a thread different from the main process by supporting multithreading operations. Also this turns out to be transparent to the test engineer.

IV. Conclusions

A new Flexible Framework for Magnetic Measurement (FFMM), based on Object-Oriented Programming (OOP) and Aspect-Oriented Programming (AOP) is under development at CERN. The main target of this development is to show that the new requirements of software flexibility and reusability that have arisen after the end of the series production of the LHC superconducting magnets can be met, and to produce a design for complete implementation in the coming year.
The core of the platform was implemented as a proof demonstrator on a prototype of magnetic measurement bench, based on rotating coil technique for the harmonic analysis of magnets. The proof demonstrator verified the feasibility of the proposed key concepts and the related architecture. Preliminary results show that the platform can be easily tailored by a test engineer to manage different magnetic measurement applications, and thus encourage to finalize the FFMM design and implementation.
Future work will be devoted to complement and improve the present tools. Special attention will be devoted to programming simplicity on the user side, as well as the detection of errors in the measurement application. A large part of this work will be accomplished in the development of the event manager. Finally, the drivers for other devices, widely used for magnetic measurements, will be integrated in the framework.

Acknowledgments

This work is supported by CERN trough the agreement K 1322 with the Department of Engineering, University of Sannio, whose supports authors gratefully acknowledge. Authors thank F. Cennamo, L. Walckiers, G. Di Lucca, M. L. Bernardi, and L. Deniau for their useful suggestions.

References

[1] S. Amet, L. Bottura, L. Deniau, and L. Walckiers, “The multipoles factory: an element of the LHC control”, *IEEE Trans. on Appl. Supercond.*, Vol. 12, pp. 1417-1421, 2002.
[2] 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.
[3] http://zone.ni.com/devzone/cda/tut/p/id/3238#toc0 Designing Next-Generation Test Systems Developers Guide.
[4] P.C. Ferreira, H. Reymond, “Sequence of tests and settings to start a magnetic measurement on MMP 6.5.0”, *Internal note*, CERN 2003.
[5] A. Guerrero, J-J Gras, J-L Nougaret, M. Ludwig, M. Arruat, S. Jackson, “CERN front-end software architecture for accelerator controls”, *Proceedings of ICALEPCS2003*, Gyeongiu, Korea, 2003.
[6] J. M. Nogiec, J. Di Marco, S. Kotelnikov, K. Trombly-Freytag, D. Walbridge, M. Tartaglia, “Configurable component-based software system for magnetic field measurements”, *IEEE Trans. On Applied Superconductivity*, Vol. 16, N. 2, Jun. 2006, pp. 1382-1385.
[7] http://www.tango-controls.org/.
[8] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. “An overview of AspectJ”, *Proc. of 15th Eur. Conf. on Object-Or. Prog (ECOOP 01), Vol. 2072 of Lecture Notes in Computer Science*, Budapest, Hungary, 2001.
[9] L. Bottura, K. N. Henrichsen, “Field Measurements”, CERN Accelerator School Proceedings, CERN, September 2004, pp. 118-151.
[10] Erich Gamma, Richard Helm, Ralph Johnson , John Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison Wesley, October 1994.
[11] P. Arpaia, A. Masi, G. Spiezia, “A digital integrator for fast and accurate measurement of magnetic flux by rotating coils”, *IEEE Trans. on Instrum. Meas.*, Vol. 56, No. 2, April 2007.