UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola di Dottorato in Ingegneria Industriale

Corso di Dottorato in Ingegneria Elettrica - XXII Ciclo

A Flexible Framework for Magnetic Measurements

Relatori:
Ch.mo prof. Nello POLESE
Ch.mo prof. Pasquale ARPAIA

Candidato:
Vitaliano INGLESE

Co-relatore:
Dott. Marco BUZIO

Coordinatore:
Ch.mo prof. Guido CARPINELLI

ANNO ACCADEMICO 2009

To my family

Acknowledgements

This experience at CERN represented for me a great opportunity of growth, both from the professional and the human point of view. I worked with many persons who allowed me to improve my skills and my knowledge, so I would like to thank them.
I would like to thank my supervisor Prof. Pasquale Arpaia for his full-time guide. His enthusiasm encouraged me a lot by providing a strong motivation to do well my work during the thesis work as a whole.
I would like to express also my gratitude to my two CERN supervisors, Luca Bottura, who launched the FFMM project, and Marco Buzio, who followed the last part of my work as Ph.D. Student. Our discussions were always prone of clever suggestions.
I would like to acknowledge Louis Walckiers for his help and participation to the project.
I would like to express my appreciation to my tutor from the University of Naples, Prof. Nello Polese. His guide, helpfulness, and patience were important to achieve this final result.
I would like to acknowledge also the coordinator of the Doctoral School in Naples, Prof. Guido Carpinelli, for his careful guide during my work.
My gratitude also goes to Walter Scandale for the trust he had in me, from our first meeting at CERN.
My work is a part of a project that involved the work of many persons. I had the opportunity to work with consultants from the University of Sannio. I would like to thank Prof. Giuseppe Di Lucca, MarioLuca Bernardi, Giuseppe La Commara, and Felice Romano for their precious contribution to the development of FFMM.
I would like to thank Stefano Tiso and Domenico Della Ratta, my adventure at CERN

started with them. I will never forget the year we spent together.
I would like to thank Giovanni Spiezia, with whom I worked most of the time I spent at CERN, for his contribution to the project and for his friendship.
I would like to thank Giancarlo Golluccio, Giuseppe Montenero, Ernesto De Matteis, Lucio Fiscarelli, Carlo Petrone, Nicola Cardines, Fabio Corrado, Cosimo Iadanza. We worked and we had a lot of fun together.
A special thanks also goes to Laurent Deniau, J. Garcia Perez, David Giloteaux, and Peter Galbraith from CERN, and Nathan Brooks, from the University of Texas, for their precious help.
I would like to thank Oana for her support and her patience during the last months. A very special thanks also goes to Alessandro Masi, Pasquale Cimmino, Pier Paolo and Isabella, Emmanuele Ravaioli, Alberto Ferro, Phat Srimanhobas, Filippo Liberati, Tiziana Miani. I cannot make a list of all my friends, because I am afraid to forget someone. So, I express my gratitude to all of them, and particularly the many wonderful people I met in the Foyer Saint Justin, my house for the last years. Without them my stay in Geneva would not have been as pleasant as it was.
Finally, I would like to express my gratitude to my parents, my brothers, my grandmothers and my whole family, to whom I owe all. This work is dedicated to them.

Contents

List of Figures

List of Tables

Summary

The work presented in this Ph.D. thesis covers the specification, design, prototyping, and validation of a new version of a magnetic measurement control, acquisition, and data analysis software package: the *Flexible Framework for Magnetic Measurements* (FFMM). FFMM constitutes the software part of the new platform for magnetic measurements, including also new high-performance hardware, developed at the European Organization for Nuclear Research (CERN) in cooperation with the Department of Engineering of the University of Sannio. FFMM is conceived as a unified solution to drive all the existing and future park of measurement systems (mainly magnetic but also optical, mechanical, etc.).

The effort for the series test of the LHC superconducting magnets highlighted limitations in the measurement control and acquisition programs, mainly associated with the relatively long time needed for a development iteration (the cycle of specification-programming-debugging-validation). Moreover, the software capabilities needed to be extended to manage the challenges of the new hardware, namely fast rotating-coil transducers (Micro Rotating Unit) and high-performance digital integrators (Fast Digital Integrator), and the need to perform more specialized tests on small/medium magnet batches. FFMM was developed to address these issues. Implemented in C++, it is based (*i*) on *Object-Oriented Programming* (OOP), and (*ii*) on an innovative technology, the *Aspect-Oriented Programming* (AOP). AOP extends the object-oriented paradigm in order to encapsulate features that are transversal to several functional units (*crosscutting concerns*) by means of new software modules, the *aspects*. The framework supports the user in producing measurement applications for a wide range of requirements by limited effort

and development time.
FFMM includes utilities for (*i*) fault detection, (*ii*) software synchronization, (*iii*) automatic generation of user interfaces, and (*iv*) a *Measurement Domain Specific Language* (MDSL) to provide the test engineer with an easy and fast way to write measurement scripts containing formal descriptions of the test protocols.
The higher sampling rate of the new transducers and acquisition hardware, capable of increasing by three orders of magnitude the bandwidth of harmonic measurements with respect to the previous ones, produced an exponential rise in storage requirements. Thus, a data reduction algorithm was conceived as part of the system in order to decrease the size of measurement results by controlling the quality loss simultaneously. In addition to that, the framework includes an algorithm for field harmonic resolution enhancement allowing to overcome the limitation of the CERN standard analysis procedure, for the tests on magnets in non-stationary conditions.
A numerical study was performed to assess the performance of both the techniques for data analysis (reduction and resolution enhancement). The results highlighted the good performance achieved by the proposed approaches, and in particular their suitability for application to the magnetic measurements carried out at CERN.
For both algorithms, these conclusions were confirmed on the field through dedicated test campaigns. The tests on the field were performed with different protocols and measuring equipments. The framework proved its effectiveness in developing software for measurements with very different requirements. The results highlighted the resolution improvement (up to a factor 100) attained in the harmonic estimation thanks to the new platform and the capability of producing quickly and with a limited effort the acquisition and control software for new applications.
The framework was designed to be flexible, maintainable, reusable, efficient. To assess the fulfillment of these project goals on the release 3.0 of FFMM, the internal quality of its source code was evaluated by means of suitable metrics according to the reference model defined in the standard ISO 9126. The results highlighted a good average quality level, with possibilities of interventions to decrease the complexity of some hot spots and exploit

more profitably the concepts of object-oriented programming. A supplementary analysis was carried out on the aspect-oriented component handling the fault detection to state the advantages of using such a software design. The proposed architecture proved to grant a high level of flexibility, maintainability, and reusability, without affecting significantly the run-time performance.

Finally, an experimental approach to the software flexibility assessment of measurement frameworks was proposed and applied in the context of FFMM. The results highlighted that the framework achieves increasing degrees of flexibility moving from the point of view of the developer to that of the test engineer. The highest flexibility is attained for the changes involving the measurement procedure, namely at the level where it was mainly required.

Introduction

At the European Organization for Nuclear Research (CERN), the design and realization of the particle accelerator Large Hadron Collider (LHC) [1] has required a remarkable technological effort in many areas of engineering. In particular, the tests of LHC superconducting magnets disclosed new horizons to magnetic measurements [2, 3].
In the last years, several fast transducers (rotating units [4]) have been developed in order to achieve an increase of two orders of magnitude in the bandwidth of harmonic measurements (10 to 100 Hz), when compared to the standard rotating coil technique (typically 1 Hz or less), and still maintaining a typical resolution of 10 ppm [5, 4].
Standard magnetic measurements on accelerator magnets are mostly based on the integration of a voltage signal in order to get the magnetic flux, according to Faraday's law (such as in rotating coils, fixed coils, stretched wire, and so on)[6, 7, 8, 9], complemented also by other techniques (such as Hall plates) [10]. A multi-purpose numerical measurement instrument, the Fast Digital Integrator (FDI), has been therefore developed at CERN with the aim of reducing the flux acquisition time down to $4\,\mu s$ while increasing the metrological performance [11]. The new integrator was conceived with the specific aim of being general-purpose, as much as possible, in order to become a sound basis for satisfying a wide range of magnetic measurement requirements over the years.
Furthermore, after the end of the LHC series tests, and on the medium term, the expectation is to have a number of very specific tests to be rapidly adapted and performed on single prototypes or relatively small batches of magnets [12]. These tests require the control of various devices, such as transducers, actuators, trigger/timing cards, power supplies, and other devices not yet completely specified. Moreover, for different measure-

ment techniques, different algorithms have to be implemented.

All these conditions demand for re-engineering the control and acquisition software in order to be adequate to the new measurement requirements and to manage the challenge of the new hardware. In practice, the ideal situation would be to have a flexible software framework, providing a robust library to control remotely all the instrumentation involved in the tests, including the new high-performance hardware, as well as the tools to help the user in the design of new measurement algorithms.

A number of developments worldwide try to address these issues. At commercial level, National Instrument (NI) proposes the product NI TestStand [13] for supporting the user in designing new test applications by integrating software modules developed in different programming languages (C, C++, LabVIEW®). However, NI TestStand does not support the user in developing single software modules, and as a result standard development and reusability are intrinsically limited. The Front-End Software Architecture (FESA) paradigm, adopted at CERN for the LHC controls [14] was 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 level of FESA would be required in order to adapt the architecture to the abovementioned applications. At the Fermi National Accelerator Laboratory (FNAL), a new software system to test accelerator magnets was developed to handle various types of hardware, as well as to be extensible to all measurement technologies and analysis algorithms [15]. Also other sub-nuclear research centres (Alba, Soleil, Elettra, and ESRF) collaborate in order to develop a suitable software framework for testing accelerator magnets [16]. This Consortium proposes TANGO, an object-oriented system to handle different measurement applications. The software of FNAL and the object-oriented system Tango are still under development and not yet worldwide accessible.

The work presented in this Ph.D. thesis covers the specification, design, prototyping, and validation of a new version of a magnetic measurement control, acquisition, and data analysis software package: the *Flexible Framework for Magnetic Measurements* (FFMM). FFMM constitutes the software part of the new CERN platform for magnetic measurements, including also the new high-performance hardware, and is conceived as a unified solution to drive all the existing and future park of measurement systems (mainly magnetic but also optical, mechanical, etc.).

FFMM aims at maximizing the measurement software quality, in terms of flexibility, reusability, maintainability and portability, by simultaneously keeping high efficiency levels. Moreover, FFMM can be configured for satisfying a large set of measurement applications in the magnetic measurement field. It is characterized by (*i*) flexibility for rapid and cost effective realization of “scriptable” applications, including prototyping in an R&D context, (*ii*) a modular architecture to mix and reuse components chosen from an incremental library, and (*iii*) high performance to exploit the increased throughput of the new transducers and acquisition systems.

The framework, implemented in C++, is based (*i*) on *Object-Oriented Programming* (OOP), and (*ii*) on an innovative technology, the *Aspect-Oriented Programming* (AOP) [17], extending the objects capabilities in order to encapsulate features that are transversal to several functional units (*crosscutting concerns*) by means of new software modules, the *aspects*.

After developing the kernel and the main components of FFMM, it was necessary to provide the test engineer with an easy and fast way to write measurement scripts. To achieve this goal, a *Measurement Domain Specific Language* (MDSL) was developed.

Furthermore, suitable means were developed for the automatic generation of user interfaces. 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.

The higher sampling rate of the new generation of fast transducers and integrators, capable of increasing by three orders of magnitude the bandwidth of harmonic measurements, increased the amount of resulting data by producing an exponential rise in storage requirements. Thus, a data reduction algorithm was needed as a vital part of the system in order to decrease the size of measurement results by controlling the quality loss simultaneously.

In addition to that, the framework includes an algorithm allowing harmonic resolution enhancement in rotating coil measurement data in order to overcome the limitation of the standard analysis procedure [18] in non-stationary conditions, typical of tests on superconducting magnets.

In chapter 1, an overview of the main methods for magnetic measurements is given. Then the automatic measurement system so far employed at CERN for magnets harmonic analysis is described. Finally, a state of the art of the frameworks for measurement systems proposed in literature and employed in the main research centers is presented, concluding with the rationale for a custom development of a new system.

In chapter 2, the requirements of FFMM are presented, with a particular emphasis on the need for flexibility coming from the experience of other measurement systems previously employed at CERN. FFMM is part of a wider project aiming at developing a platform for magnetic measurements, including also new high-performance hardware. For the sake of completeness, a brief overview of the main hardware components is also provided.

In chapter 3, the design of the FFMM is presented. In particular, (*i*) the framework core design, with details on its overall structure and on its main software components (including tools for fault detection and software synchronization realized through AOP), (*ii*) the approaches proposed for the development of a *Measurement Domain Specific language* and the automatic generation of user interfaces, and (*iii*) the design principles of the algorithms for data reduction and harmonic resolution enhancement are presented.

In chapter 4, the results of the numerical analysis carried out on the algorithms for data reduction and harmonic resolution enhancement are presented. Simulations were performed to verify the fulfillment of the goals for which they were conceived, in conditions typical of measurement on superconducting dipole magnets.

In chapter 5, the quality characterization of the release 3.0 of FFMM is presented. The object-oriented and the aspect-oriented parts of the system are evaluated separately. First, an approach based on the standard ISO 9126 [19] is chosen as a reference model for the quality evaluation of the object-oriented part, along with a more practical analysis aiming at finding possible points of intervention and propose improvement actions. Only the internal software quality, related to static properties of the code, is considered. Subse-

quently, a modularity and performance analysis of the aspect-oriented part is presented to prove the benefits deriving from the introduction of this technology into the system.
In chapter 6, the experimental validation of the framework in some scenarios typical of LHC-related measurements is presented. Two applications are described: the former, the tracking test [20], is based on the rotating coil technique and aims at estimating and compensating of the field errors due to non-ideality of the LHC superconducting dipoles; the latter aims at measuring the permeability of a sample of the soft steel used for the LHC magnet yokes, through a fixed coil system [21]. The setup of the measurement stations and the results obtained by means of the software produced through FFMM are described.
In chapter 7, the validation on the field of the algorithms for data reduction and harmonics resolution enhancement is presented. It aims at proving on the field the effectiveness of their design principles and assessing their performance in actual working conditions. To this purpose, measurements through rotating coils were carried out at the CERN magnet test facility SM18, both on superconducting and on resistive dipoles.
In chapter 8, the flexibility test of FFMM is presented. As a part of the wider work aiming at the characterization of the framework started in chapter 5, the twofold purpose of this chapter is (*i*) to introduce specific metrics suitable for assessing the degree of flexibility achieved by the framework, and (*ii*) to present experimental results for some typical application scenarios of the current release 3.0 of FFMM.

Chapter 1

Automatic systems for magnetic measurements

In this chapter, an overview of the main methods for magnetic measurements is provided. Subsequently the automatic measurement system so far employed at the European Organization for Nuclear Research (CERN) for magnets harmonic analysis is described. Finally, a state of the art of the frameworks for measurement systems proposed in literature and employed in the main research centers is presented, concluding with the rationale for a custom development of a new system.

1.1 Methods for magnetic field measurements

The main issues of High Energy Particle (HEP) accelerators are (*i*) to explore matter at small scale, by means of radiations of wavelength smaller than the the dimension to be resolved, (*ii*) to produce new, massive particles in high-energy collisions, (*iii*) to reproduce locally the very high temperatures occurring in stars or in the early universe, and investigate nuclear matter in these extreme conditions, by imparting energy to particles and nuclei, (*iv*) to exploit the electromagnetic radiation they emit when accelerated, particularly when the beam trajectory is curved by a magnetic field (centripetal acceleration).
CERN, one of the most important HEP laboratories, located at Geneva in Switzerland, was founded in 1953 with the motivation of providing a deeper understanding of the matter and its contents. The last CERN project is the Large Hadron Collider (LHC): a circular accelerator that will collide proton beams, but also heavier ions up to lead. It is

installed in a 27-km long underground tunnel.

Superconductivity played a crucial role in the development of the LHC. It is a powerful means to achieve high-energy particle beams and keep compact the design of the machine. Making a machine compact means not only saving capital cost, but also limiting the beam stored per energy. Besides capital cost and compactness advantages, superconductivity reduces electrical power consumption. High-energy, high-intensity machines produce beams with MW power, so that conversion efficiency from the grid to the beam must be maximized, by reducing ohmic losses in RF cavities and in electromagnets [22]. In d.c. electromagnets, superconductivity suppresses all ohmic losses, thus the only power consumption is related to the associated cryogenic refrigeration. On the contrary, the wall resistance of superconducting RF cavities subject to varying fields does not drop to zero, but varies exponentially with the ratio of operating to critical temperature T_c [22]. This imposes to operate at a temperature well below T_c, in practice as the result of a trade-off between residual dissipation and thermodynamic cost of refrigeration.

The coils of the LHC superconducting magnets are wound with NbTi cables (7000 km in total), working in superfluid helium either at 1.9 K or at 4.5 K. A vertical dipole field B of 8.33 T is required to bend the proton beams, whereas the quadrupole magnets are designed for a gradient of 223 Tm^{-1} and a peak field of about 7 T.

In storage rings like the LHC, stable beams have to run as long as possible on the circular orbit in order to increase the number of collisions between the counter-rotating beams. This imposes strong constrains on the tolerable field perturbations along the trajectory. Deviations from the dipole and quadrupole fields, even if short in both space and time, can induce instabilities reducing the beam life-time. Higher-order multipoles correctors are required to compensate the unavoidable imperfections of dipole and quadrupole magnets.

The production of magnets with high field quality has been invariably assisted by a spectrum of various measurement, based on different methods depending on the goal and accuracy of the desired analysis.

The quantities of relevance for the magnetic field produced by accelerator magnets are the strength and direction of the field produced, the errors with respect to the ideal field profile, and the location of the magnetic center in the case of gradient fields. For all the LHC magnets, the above quantities are required as integral or average over the magnet length.

In the following, an overview of the main methods for magnetic measurements is provided. For its importance in the following chapters of this thesis, more details are provided about the rotating coil method and the related harmonic analysis.

1.1.1 Rotating coils

The rotating coil method [6, 7] is widely used for magnets with cylindrical bore owing to its capability at measuring all properties of the magnetic field (field strength, multipoles, angle, direction) integrated over the coil length. An induction coil is placed on a circular support and is rotated in the field to be mapped. The coil angular position is measured by an angular encoder, rigidly connected to the rotating support. The coil rotating in the field cuts the flux lines and a voltage is induced at the terminals. The voltage is integrated between predefined angles obtaining the flux change as a function of angular position (Fig. 1.1).

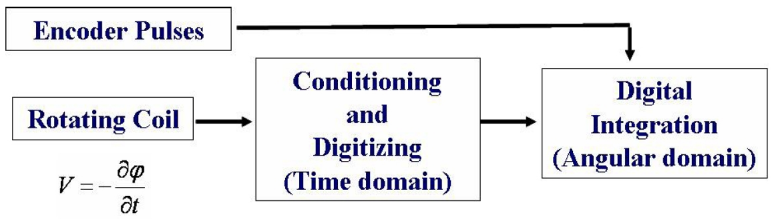

Encoder Pulses
Rotating Coil
$V = -\frac{\partial \varphi}{\partial t}$
Conditioning and Digitizing (Time domain)
Digital Integration (Angular domain)

Figure 1.1: Rotating coils measurement principle.

Magnetic field harmonic analysis

The LHC dipoles are 15-meters long with a beam aperture of 50 mm in diameter, giving the possibility to consider the coils as infinitely long, and to evaluate the magnetic field in the x-y complex plane by neglecting the z component. This 2-dimensional approximation is very convenient to describe $\vec{B}$ in terms of a complex variable z. In the central part of the dipole taking into account the properties of the analytical functions, it can be postulated that the magnetic field generated $\vec{B}$ can be expanded in the complex plane in a power series [23]:

$$B(z) = B_1 \sum_{n=1}^{\infty} \frac{C_n R_{ref}^{n-1}}{B_1} \left(\frac{z}{R_{ref}}\right)^{n-1} = B_1 \sum_{n=1}^{\infty} c_n \left(\frac{z}{R_{ref}}\right)^{n-1} \cdot 10^{-4} \tag{1.1}$$

where C_n is in *units* of $T \cdot m^{1-n}$ while $c_n = C_n \frac{R_{ref}^{n-1}}{B_1}$ are the multipoles normalized respect to the main dipole field and referred to a reference radius $R_{ref} = 17\ mm$. In this way, all the series coefficients c_n result dimensionless and are expressed in so called *units* of the main field at the reference radius. They are then multiplied by the scaling factor 10^4 that is the order of the ratio between the main field and the field errors. In the complex plane, the C_n coefficient can be decomposed in its normal and skew term, i.e. real and imaginary part respectively, as follows [18]:

$$C_n = B_n + iA_n \tag{1.2}$$

By using the decomposition above, and by applying the scaling factor to the normal and skew field components deduced from equation 1.2, the field components in units of the main field B_1 can be expressed:

$$\begin{cases} a_n = A_n \dfrac{R_{ref}^{n-1}}{B_1} \cdot 10^4 \\ b_n = B_n \dfrac{R_{ref}^{n-1}}{B_1} \cdot 10^4 \end{cases} \tag{1.3}$$

The existence of non-zero b_n and/or a_n coefficients reflects the fact that the magnetic field generated by the superconducting coil in a dipole is not a pure dipole and is affected by higher order of multipoles (quadrupole, sextupole, etc.). The multipole components are generated by the difference between the ideal and the actual current distribution in the coil. All undesired multipole components other than the main field are referred as *field errors*. They can be associated with the geometry approximation of the superconducting coils, but also they can have origins that depend on the different elements and materials

used.
The rotating coil method eliminates the time dependence [2], and, in particular, the influence of variations of the rotation speed, greatly relaxing requirements for uniform rotation. Differential measurements are also beneficial to increase the resolution of high-order multipoles, several orders of magnitude smaller than the main field. This is realized by using a set of compensation coils mounted on the rotation support [24]. The signal from the compensation coils is used to suppress analogically the strong contribution from the main field. The compensated signal is analyzed in Fourier series together with the absolute signal of the outermost rotating coil in order to obtain the main field, as well as the higher order multipoles [18]. The overall uncertainty on the integral field strength and on the harmonics depends on the shaft type. The new system developed at CERN can reach a bandwidth of harmonic measurement up to 100 Hz maintaining a resolution of ± 10 ppm.

1.1.2 Stretched wire

The stretched-wire technique is also based on the induction method [8, 9]. A thin wire, with a diameter of 0.1 mm, is stretched in the magnet bore between two precision stages. A motion results in a voltage at the two ends of the wire, whose integral is the magnetic flux through the area scanned by the motion. The method, a robust null technique with very high resolution, provides a measurement of the integral field, of the field direction, and of the magnetic axis.
The uncertainty depends on the accuracy of the precision stages driving the wire motion (± 1 μm), on the effectiveness of the sag correction, and on the alignment errors during installation. The overall uncertainty on the integrated strength and on the angle measurement was estimated at ± 5 $units$ and ± 0.3 $mrad$, respectively [8, 9].

1.1.3 Magnetic resonance technique

The nuclear magnetic resonance technique is considered as the primary standard for calibration. It is frequently used, not only for calibration purposes, but also for high accuracy field mapping. Based on an easy and accurate frequency measurement, it is independent

of temperature variations. Commercially-available instruments measure fields in the range from 0.011 T up to 13 T with an accuracy better than ± 10 *ppm*.
In practice, a sample of water is placed inside an excitation coil, powered from a radiofrequency oscillator. The precession frequency of the nuclei in the sample is measured either as nuclear induction (coupling into a detecting coil) or as resonance absorption [25]. The measured frequency is directly proportional to the strength of the magnetic field with coefficients of 42.57640 MHz/T for protons and 6.53569 MHz/T for deuterons.
The advantages of the method are its very high accuracy, its linearity, and the static operation of the system. The main disadvantage is the need for a rather homogeneous field in order to obtain a sufficiently coherent signal.

1.1.4 Hall probes

Hall probes exploit the Hall effect to measure magnetic fields [26]. When a current is flowing in a solid penetrated by a magnetic field, this field generates a voltage perpendicular to the current and the field itself. This voltage is large enough to be practical only for semiconductors [27]. The main uncertainty factor is due to the temperature coefficient of the Hall voltage.
The Hall probes permit the analysis of inhomogeneous fields because they measure the field locally. Conversely, the integral measurement, over the entire magnet length, is more difficult since the Hall sensors are quite small requiring either long and complex probes or many measurements steps.

1.2 Software for magnetic measurements at CERN

Many magnetic measurement systems are currently used at CERN (rotating coils, stretched wire, etc.), and different software packages are employed for control, data acquisition, and analysis. These systems were developed incrementally during the years without focusing on their quality, namely flexibility and reusability.
An example of such a software is the *Magnetic Measurement Program* (MMP) [28], used in the past for the series tests of the LHC superonducting magnets. For its importance

in the test activities carried out at CERN, mainly based on the rotating coil technique [29], and for its characteristics representative of the previous generation of control and acquisition systems, more details on MMP are provided in the following.

1.2.1 The Magnetic Measurement Program

The *Magnetic Measurement Program* (MMP) [28] was internally developed at CERN in LabVIEW® with the main aim of measuring the field in the LHC magnets by means of the rotating coils method. The software includes a control and a measurement system. The control system drives the hardware used for the measurement (motors, power supplies, etc.) and monitors the main parameters of the system allowing proper operation to be verified. The user interacts with the system through a Graphical User Interface. The measurement system reads out and saves the measured field and other parameters to be used in the magnet analysis. A principle layout of a magnetic field measurements system is shown schematically in Fig. 1.2. The software delivers as a result the measured raw data for each measurement and integrates analysis routines to compute the main field, the main field direction, the higher order harmonics and the magnetic axis coordinates. At the end of each measurement run, the collected raw data can be transferred into a database.

The system provides a predefined set of measurement procedures, adjustable to the current needs only through the definition of a limited number of parameters for hardware configuration. As a consequence, the system shows a remarkable lack of flexibility, since it implements fixed measurement algorithm and analysis procedure, both based on rotating coils, and requires changes in the LabVIEW® code in order to modify it. Each modification therefore requires a long time and the work of expert programmers.

The same approach was used also for the development of the other programs exploiting different measurement techniques.

As a consequence, nowadays a plurality of systems is employed, which result to be rigid, especially in the test protocol, and difficult to adapt to measurement requirements other from those for which they were originally designed. This major limitation pushed the

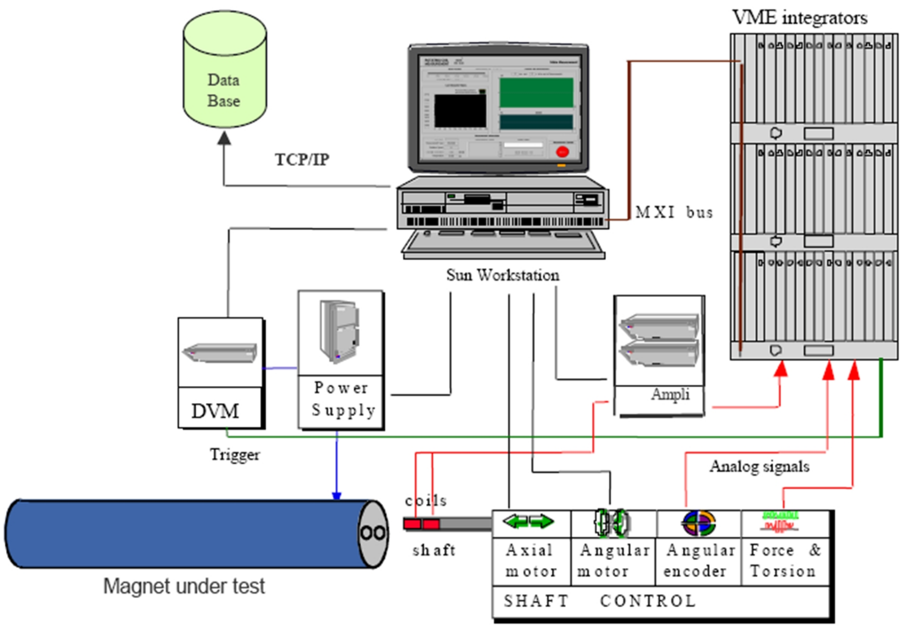


Figure 1.2: Layout of the rotating-coil based measurement system controlled by MMP.

research activities, carried out at CERN in the field of magnetic measurements, to move from standalone measurement programs towards the more modern and useful concept of framework.

1.3 Software frameworks

In software development, a *framework* is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project. In other words, frameworks are designed with the aim of facilitating software development, by allowing users to spend more time on meeting the application requirements rather than dealing with the low level details of providing a working system.
According to the object-oriented paradigm, a framework can be seen as a partial design and implementation for an application in a given domain [30], described by a set of

abstract classes and the way instances of those classes collaborate. The functionalities and the architecture of the system can be tailored and combined to create complete applications. Thus, frameworks achieve the most reuse for object-oriented systems and reduce the effort necessary for the construction of new applications.

1.3.1 Frameworks for measurement applications

Whilst test programs have been designed so far to solve specific problems with extremely limited capability to evolve, a framework for magnetic measurement, suitably conceived [31] in order to be configurable for satisfying a wide range of requirements, could constitute a unified solution to drive all the existing and future park of measurement systems. A number of developments worldwide try to address this issue. 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 [32]. At commercial level, National Instrument (NI) proposes the product NI TestStand® [13] for supporting the user in designing new test applications by integrating software modules developed in different programming languages (C, C++, LabVIEW®). However, NI TestStand® does not support the user in developing single software modules, and as a result standard development and reusability are intrinsically limited. The *Front-End Software Architecture* (FESA) paradigm, adopted at CERN for the LHC controls [14] was 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 level of FESA would be required in order to adapt the architecture to the aforementioned applications. At the Fermi National Accelerator Laboratory (FNAL), a new software system to test accelerator magnets was developed to handle various types of hardware, as well as to be extensible to all measurement technologies and analysis algorithms [15]. Also other sub-nuclear research centres (Alba, Soleil, Elettra, and ESRF) collaborate in order to develop a suitable software framework for testing accelerator magnets [16]. This Consortium proposes TANGO, an object-oriented system, to handle different measurement applications.

The most advanced solutions, namely the software of FNAL and the object-oriented system TANGO, are still under development and not yet worldwide accessible. Therefore, to address the issues introduced above and detailed in chapter 2, the development of a new framework for magnetic measurements, main topic of this thesis, was launched at CERN in cooperation with the Department of Engineering of the University of Sannio.

Chapter 2

Requirements

This chapter presents the requirements of the *Flexible Framework for Magnetic Measurements* (FFMM), and in particular the need for flexibility coming from the experience of other measurement systems previously employed at CERN.
Although the main topic in this thesis is the software for acquisition, control, and data analysis, FFMM is part of a wider project aiming at the development of a platform for magnetic measurements, including also new high-performance hardware. For the sake of completeness, a brief overview of the main hardware components is also provided.

2.1 Past experiences and need for flexibility

At CERN, the series field tests for the LHC superconducting and resistive magnets were carried out by means of a control and acquisition measurement system, developed during the past years under highly-variable conditions of evolving hardware and software configurations and measurement requirements. The result was implemented in all major test locations and used successfully for the warm and cold tests at CERN, as well as in industry.
The software bears a long heritance of the evolution from the original version of the magnetic measurement program (implemented in C language for a VME bus-based station) to the present version in use in the test stations (in excess of 1000 LabVIEW® VI's running on Sun workstations). The data acquisition throughput of this system is too slow for the commissioning period of the LHC with beams and calls for immediate streamlining, as

soon as series tests will be completed.
Furthermore, new hardware (i.e. digital integrators [33], rotating units [4]) was developed in order to provide new standards for magnetic measurements. In addition to its improved metrological performance, the new integrator was conceived with the specific aim of being open and general-purpose, as much as possible, in order to become a sound basis for satisfying a wide range of magnetic measurement requirements over the years.
All these conditions demand strongly for re-engineering the control and acquisition software in order to be adequate to the new measurement requirements and to manage the challenge of new hardware.

2.2 The platform for magnetic measurements at CERN

The above discussion highlights the reasons leading to launch the development of a new platform for magnetic measurements. This shall evolve from the accumulated knowledge of the developments pursued in the past, by allowing the measurement capability to be extended in harmony with the new available hardware and the new measurement and test requirements. Although directly aimed at flux measurements (fixed and rotating coils), the new control and acquisition software shall bear a large degree of generality to allow extension to other type of measurements (e.g. fast voltage signals from quench, Hall plates, etc.), which will be highly beneficial to unify the diverse systems presently used for superconducting magnets tests.
The aim of the work presented in this Ph.D. thesis is to prototype a flexible platform for acquisition, control, and data analysis, integrating the new hardware and a new software developed suitably, for satisfying a wider range of measurement requirements, variable and evolvable during the time.

2.2.1 Hardware overview

As far as the hardware is concerned, in the past years fast transducers [4, 5] have been developed at CERN in order to achieve an increase of up to three orders of magnitude in the bandwidth of harmonic measurements (10 to 100 Hz), when compared to the standard rotating coil technique (typically 1 Hz or less), and still maintaining a typical resolution of 10 ppm.

In particular, a new *Micro Rotating Unit* (MRU) [4] was designed to turn faster and provide harmonic measurements at rates in the range from 1 to 10 Hz. Fast measurements require that the coils rotate continuously in one direction and at higher speeds. The MRU system is capable to turn continuously in one direction up to 8 Hz thanks to 54-channel slip rings. The available coils are connected in series arbitrarily by means of a patch panel. This permits changes in the compensation schemes or combination of several coils in virtual super segments, used to measure the integral field.

These developments pave the way for a major improvement of the theoretical and experimental analysis of superconducting accelerator magnets. However, at the same time, they push the performance demands on the digital instrumentation used for acquisition [34, 35, 36]. Standard magnetic measurements of accelerator magnets require fast and accurate data acquisition with integrating voltmeters. So far, the standard de facto in most sub-nuclear research centers has been the *Portable Digital Integrator* (PDI [34]). The core of this instrument is a voltage-to-frequency converter, whose resolution is intrinsically limited by the counting frequency. As a result, this instrument cannot follow the evolution of the test requirements arising from the new generation of magnetic transducers described above [4], especially considering the increasing need to measure superconducting magnets supplied by high frequency current cycles and pulses [37]. A number of developments worldwide try to address this issue [35, 36, 33]. At CERN, a multi-purpose numerical measurement instrument, the Fast Digital Integrator (FDI), was developed and constitutes one of the main components of the platform hardware. Besides the increased metrological performance [11], the FDI is capable of reducing the flux acquisition time down to 4 μs.

For the development of the FDI, a new generation of high-resolution (18 bit) and high sampling rate (500 kS/s) ADCs, Successive Approximation Register (SAR) was employed. A DSP was added for on-line processing, thus allowing the decimation of the input samples, with a *Signal-to-Noise Ratio* improvement by means of oversampling [11]. After the realization of a first prototype, a digitizer model was developed on the results of experimental tests in order to enhance the design and further improve the performance [38].

2.2.2 Software requirements

As far as the software is concerned, the effort for the series test of the LHC superconducting magnets at CERN highlighted limitations in the measurement control and acquisition programs, mainly associated with the relatively long time needed for a development iteration (the cycle of specification-programming-debugging-validation). As an example, the *Magnetic Measurement Program* (MMP) [28] used at CERN has a large spectrum of pre-programmed configurations accessible to the user, but requires software specialists for extending the set of configurations to cover new test and analysis requirements. For this reason, more advanced design principles in the field of software engineering have to be considered [14, 15, 16].

Furthermore, after the end of the LHC series tests, and on the medium term, the expectation is to have a number of very specific tests to be rapidly adapted and performed on single prototypes or relatively small batches of magnets. These tests require the control of various devices, such as transducers, actuators, trigger/timing cards, power supplies, and other devices not yet completely specified. Moreover, for different measurement techniques and tests, different algorithms have to be implemented. In practice, the ideal situation would be to have a flexible software framework, providing a robust library to control remotely all the instrumentation involved in the tests, as well as the tools to help the user in the design of new measurement algorithms.

The platform software is based on the *Flexible Framework for Magnetic Measurements* (FFMM) in order to make easy the development of new measurement programs, allowing simultaneously easy modification and extension of existing test software. Given a set of measurement requirements, suitable to be satisfied by the new hardware, the flexible and reconfigurable platform to be developed will allow an effective automatic measurement system to be generated by low-cost and development time.

The new system, besides reproducing key operating capabilities of the previous software (reference for comparison is MMP [39]), has to (*i*) extend the acquisition and control capabilities to the new hardware, and (*ii*) allow user-driven and traceable configuration of the hardware as well as of the test protocol, in order to bear a maximum capability to evolve.

FFMM aims at maximizing the measurement software quality, in terms of flexibility, reusability, maintainability and portability, by simultaneously keeping high efficiency levels. In particular, the flexibility, the modification easiness of a system or component for use in applications or environments other than those for which it was specifically designed [40], is definitely one of the most desirable properties of any system to face changes in operational environment during its life. This is particularly true for software systems, both because they are often subject to extremely rapid technological development, and because some of them are specifically conceived to be employed in environments spanning a wide range of functional requirements, not fully predictable at the design stage. This is the case of FFMM, which should be easy to configure for satisfying a large set of measurement applications in the magnetic measurement field.

The users that will interact with the software system can be classified in different categories:

- the *developer/administrator user* has knowledge of the framework internal structure and can access it at any level;
- the *test engineer* has knowledge of the framework functionalities through its interface, and can therefore provide a formal description of the measurement protocol to be translated transparently into a suitable executable application by the framework;
- the *end user* interacts with the resulting executable measurement application in order to perform the tests.

The roles of the different users should not be rigidly divided. For example, if the test engineer needs to modify or add a component, he should be able to access the internal structure of the framework.

The main goals of FFMM (flexibility, maintainability, reusability, efficiency) are meant to

Software characteristic	**User**
Flexibility	Test engineer
Maintainability	Developer/administrator user
Reusability	Developer/administrator user, test engineer
Efficiency	Test engineer, end user

Table 2.1: Main software characteristics and users they address.

satisfy the various needs of the different users, according to the classification provided in Tab. 2.1.

2.2.3 Data analysis requirements

The higher sampling rate of the new generation of fast transducers[4] and integrators [11] increases the amount of resulting data by producing an exponential rise in storage requirements. Thus, a data reduction algorithm is needed as a vital part of the platform in order to decrease the size of measurement results by controlling the quality loss simultaneously. In addition to that, an algorithm allowing harmonic resolution enhancement in rotating coil measurement data 1.1.1 should be available in order to overcome the limitation of the standard analysis procedure [18] in non-stationary conditions, typical of tests on superconducting magnets.
Suitable algorithms are therefore to be conceived and integrated into the framework as two of its most important components. The proposed procedures of data analysis should be thought to be suitable also for an implementation at DSP level, so that they could be eventually moved into instruments provided with autonomous computing capabilities, such as the FDI.

Chapter 3

Framework design

This chapter presents the design of the *Flexible Framework for Magnetic Measurements* (FFMM). According to the requirements discussed in chapter 2, the main components of the framework can be classified in three groups: (*i*) the *kernel*, including the framework infrastructure and its main software components, developed with the aim of assuring flexibility, reusability; (*ii*) a *Measurement Domain Specific Language* (MDSL) and tools for the automatic generation of user interfaces, to increase the ease of use; (iii) the *algorithms* developed for data reduction and harmonics time resolution enhancement. In the following, the framework core design, with details on its overall structure and on its main software components, the MDSL, the automatic interface generation, and the design principles of the two proposed algorithms are presented.

3.1 FFMM kernel

The FFMM is a software framework for magnetic measurement applications based on Object-Oriented Programming (OOP), and Aspect-Oriented Programming (AOP) [17], conceived to make simple and cost-effective the development of new measurement programs, allowing simultaneously easy modification and extension of existing test software. In the following, (*i*) the *basic ideas*, (*ii*) the *architecture*, (*iii*) and the *main components* of the FFMM kernel are presented.

3.1.1 Basic ideas

FFMM is based on the following basic ideas:

1. a group of interfaces and abstract classes represents a *white-box layer* defining the high-level structure of FFMM for generating new parts of the framework, this allows potentiality and flexibility of FFMM to be extended;

2. a group of modules represents a *black-box layer*, allowing both module reusability and use easiness to be achieved, even by test engineers without knowledge of internal FFMM mechanisms;

3. Aspect-Oriented Programming (AOP) improves the reusability and the maintainability of FFMM [17]: 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 and enhance maintainability;

4. 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 progressively further requirements in the same domain to be satisfied by a limited effort;

5. a suitable definition of the code structure allows standard modules to be developed: such modules represents a sound basis of a library both for implementing new components and for extending old ones.

3.1.2 Architecture

On the basis of these ideas, the FFMM architecture shown in Fig. 3.1 was conceived. The test engineer produces a description of the measurement application, the *User Script*, whose 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

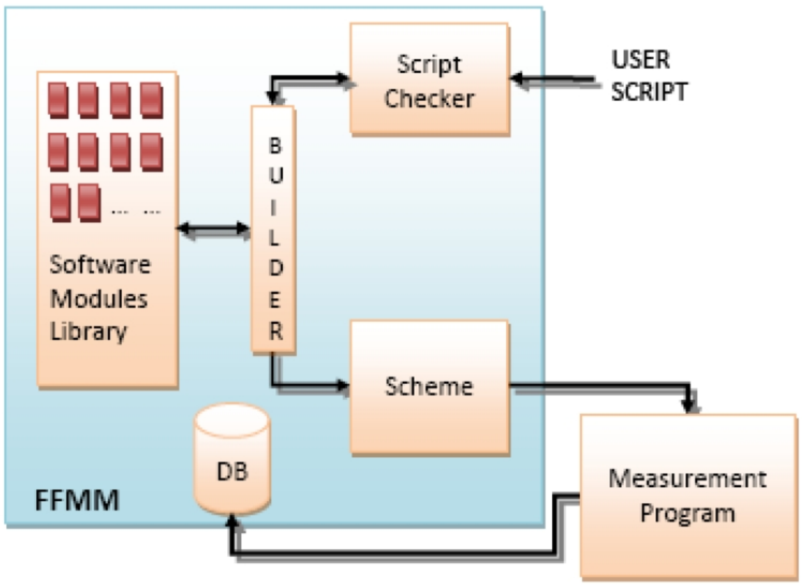


Figure 3.1: FFMM architecture.

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 superconducting magnets, the generic User Script is organized into the following phases: (*i*) definition of the measurement components; (*ii*) specification of mechanical and electrical connections; (*iii*) definition of dynamic parameters, i.e. configurable during run-time of the Measurement Program; (*iv*) component checking (fault detection); (*v*) storing of measurement conditions; (*vi*) configuration of measurement devices; (*vii*) description of the measurement procedure; (*viii*) preliminary data analysis; (*ix*) data saving.

The architecture of the FFMM kernel, the Scheme, is shown in Fig. 3.2 where its main components and the relations among them are highlighted. The *TestManager* organizes the test by knowing the device under test (*UnitUnderTest*), the measurands (*Quantity*), the measurement configuration, and the measurement procedure. *TestManager* has an association with the *Devices* (software representation of the measurement devices). Among Devices, the PC can control remotely the *VirtualDevices* through a *Communication Bus*.

The system models also *Sensors* and *Transducers* in dedicated class hierarchies.
The *Synchronizer* and the *FaultDetector* are critical modules for a test application: the

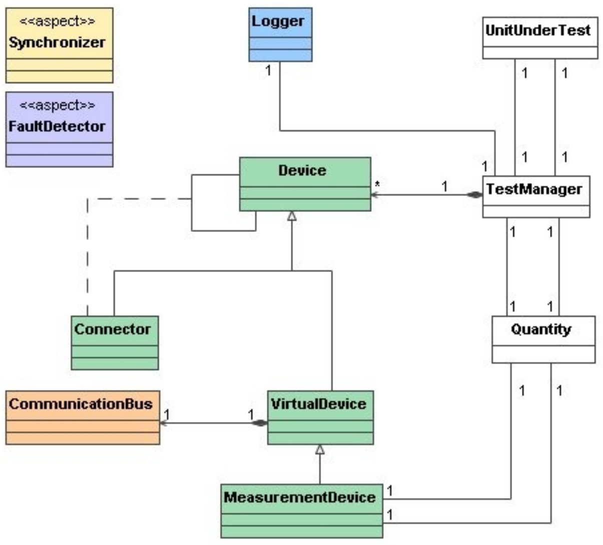


Figure 3.2: *Scheme* architecture.

former allows the measurement algorithm timing, while the latter fosters the identification and the location of failures and faults transparently to the user. Such features are transversal to several functional units (*cross cutting concerns*): the synchronization policy involves all the measurement devices and all the test procedures, the fault detection is a fundamental part of all the devices, as well as of the measurement system as a whole. The *Synchronizer* and the *FaultDetector* are, therefore, encapsulated in *Aspects* according to the AOP approach, as detailed in sections 3.1.4 and 3.1.5. The synchronization policy and fault management strategy can be extrapolated from the single modules and handled separately. In this way, future changes related to these topics will affect only the *Synchronizer* and the *FaultDetector* modules, without involving all the classes related, directly or indirectly, to the fault or synchronization events.
The *Scheme* architecture is further detailed in Fig.s 3.3 and 3.4, where its three layer structure is shown. The overall *Scheme* has a layered architecture, whilst single layers have an object-oriented internal organization. In other words, the interaction among objects takes place horizontally, among entities of the same level. The features of the level

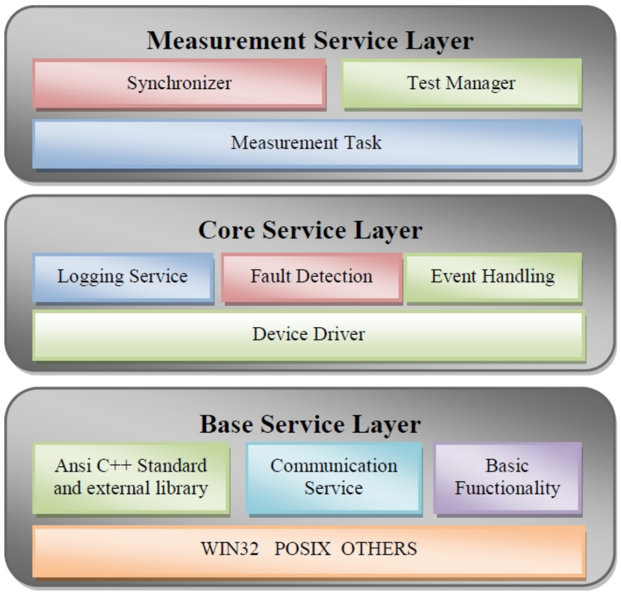


Figure 3.3: The multilayered *Scheme* architecture.

i are realized by the objects of that layer, which in turn can use the capabilities offered by the level *i+1*, through a suitably defined interface, as typically happens in layered systems. The layered structure is not rigid. Even though it would be better to use, while programming in a level, only functionalities implemented in the next lower layer, the user is allowed to call the functionalities of all the underlying levels.
In the *bottom layer*, all the basic services, needed to implement high-level logic are placed. This layer includes subcomponents for environment abstraction, memory management, error handling, filesystem abstraction, processes and threads handling. It also defines abstract communication services to high-level components within the FFMM layers, in order to extract data from actual devices and external interfaces, by allowing the exchangeability of the communication mechanisms without incurring into performance penalties. The interface *ICommunicationBus* is used to send and receive data to/from components in an abstract way. Concrete implementations of such interface are required to handle specific communication devices.
The *middle (core) layer* includes several packages exposing main functionalities related to components (in particular measurement devices), event handling infrastructure, fault

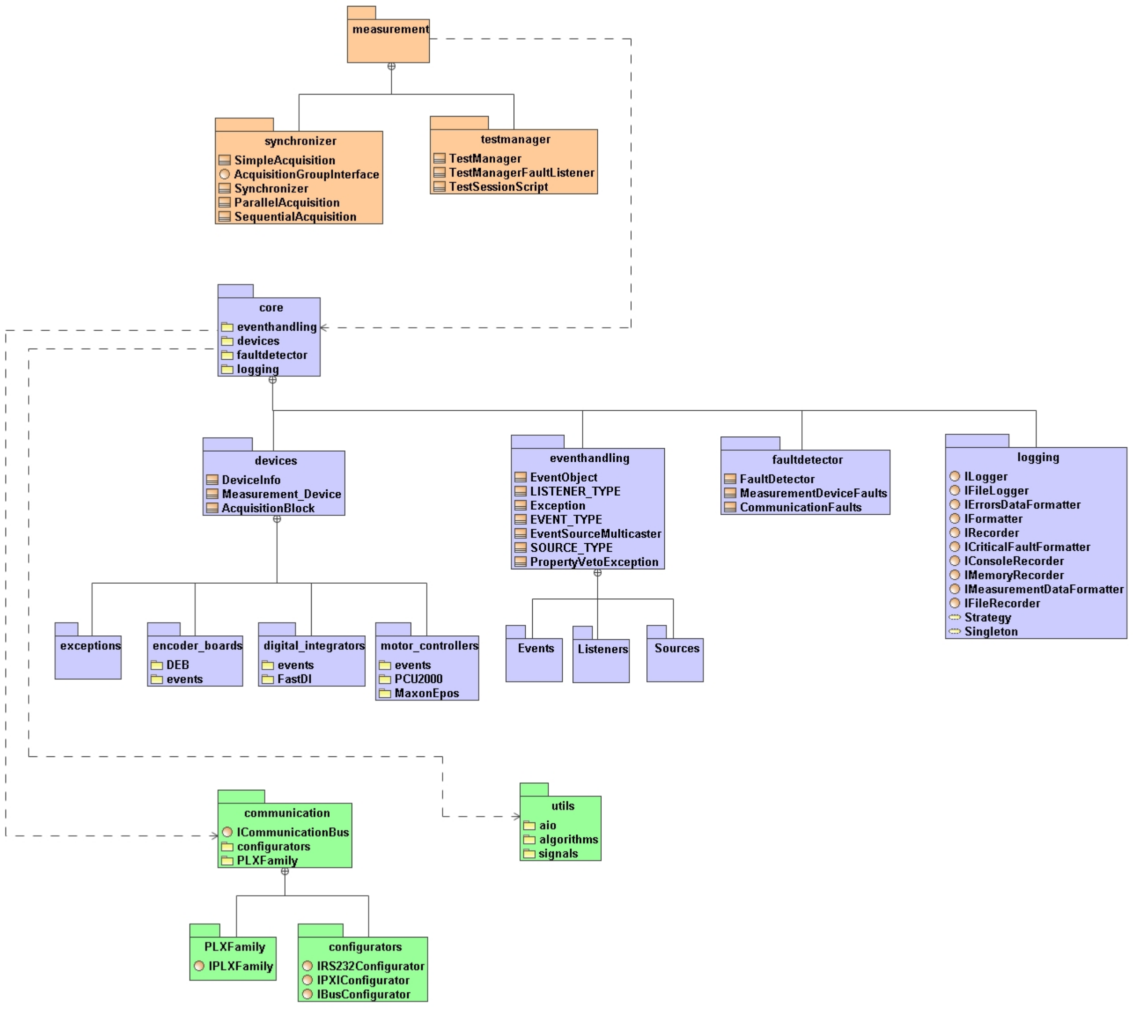


Figure 3.4: UML diagram of the multilayered *Scheme* architecture.

detection, and logging. In the design of the event handling architecture, a variant of the Observer design pattern [41] was used in order to keep synchronized the state of cooperating components (e.g. *VirtualDevices*). The Observer enables one-way propagation of changes: one publisher notifies any number of subscribers about changes of its state, thus providing a form of loosely coupled signaling from publisher to subscribers. In Fig.s 3.5 and 3.6, the main architecture and the related template based listener infrastructure, respectively, are shown. For this reason, the *FaultDetector* can be considered as a

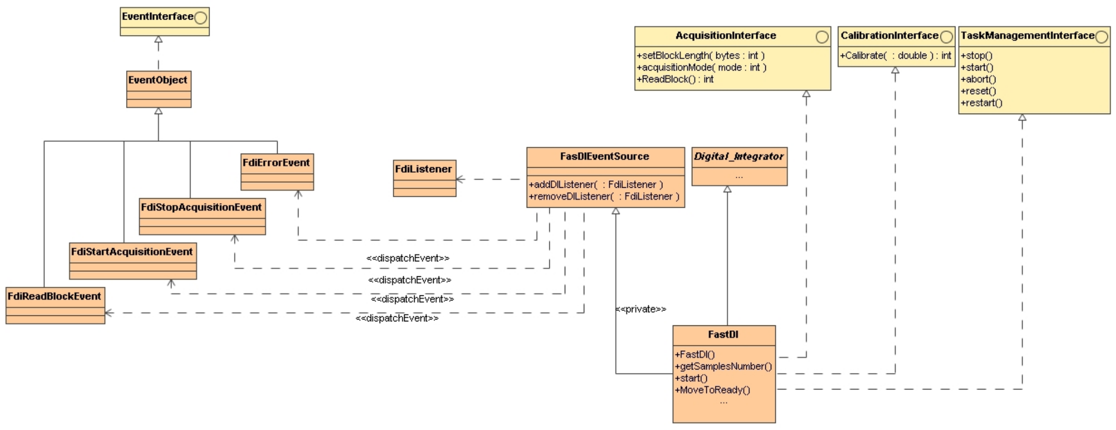


Figure 3.5: FFMM event handling architecture.

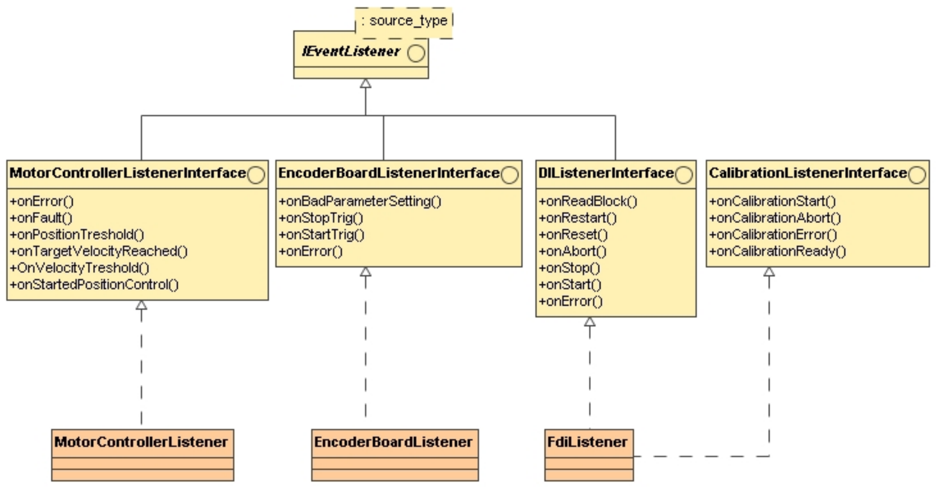


Figure 3.6: FFMM actions and listeners infrastructure.

cross-cutting concern, and an experimental release was conceived according to an aspect-oriented approach. Logging facilities are also provided at this level of the architecture (Fig. 3.4). The *Logger* class handles the storage of configuration and measurement data, as well as system warnings and exceptions. The logger architecture is depicted in Fig. 3.7. Data can be stored in a text or binary file. In any case, the final destination of the

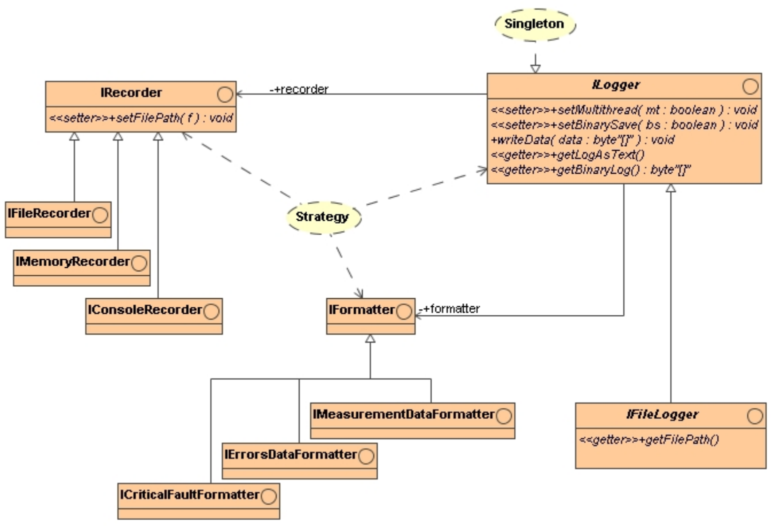

Singleton
IRecorder
<<setter>>+setFilePath(f) : void
-+recorder
ILogger
<<setter>>+setMultithread(mt : boolean) : void
<<setter>>+setBinarySave(bs : boolean) : void
+writeData(data : byte"[]") : void
<<getter>>+getLogAsText()
<<getter>>+getBinaryLog() : byte"[]"
IFileRecorder
IMemoryRecorder
IConsoleRecorder
Strategy
IFormatter
-+formatter
IMeasurementDataFormatter
IErrorsDataFormatter
ICriticalFaultFormatter
IFileLogger
<<getter>>+getFilePath()

Figure 3.7: FFMM logger architecture.

logged messages has to be kept decoupled with the format of the messages themselves. With this aim, two different responsibilities arise: logged message formatting, and logged message recording. The formatter does not take care about where the message is recorded, and the recorder does not care about the format of the message. Therefore, the *Logger* class implements the Strategy design pattern [41]: the concrete logger can be configured with the right formatter and the right recorder keeping them decoupled. Based on the services provided at this level of architecture, the measurement layer (top of Fig. 3.4) implements a minimal but extensible infrastructure, based on the class *TestManager*, in order to handle and to perform measurement session. For the measurement layer, two main features are needed:

- a test session director, the *TestManager*, encapsulating the user script (Fig. 3.1) and executing it in a controlled environment. Within the user scripts, core services are made available to the user in order to implement its measurement process;

- the capability of creating groups of data acquisition tasks (*measurement tasks*) to be synchronized to well defined events (e.g. start and stop, or device events). In the FFMM framework, the component realizing this high-level software synchronization of data acquisition is the *Synchronizer* Fig. 3.4.

In the following, the most important components of the FFMM kernel are discussed. Some of them rely on an aspect-oriented approach, thus AOP basic concepts are first recalled.

3.1.3 AOP basic concepts

The development of an automatic system, with reference to its software parts, is usually developed by exploiting *object-oriented* [32], *component-based* [15], and *agent-based* techniques [42]. They aim at organizing the software system in modules, each one responsible of specified functionalities, reducing their coupling and maximizing their internal cohesion. Anyway, *crosscutting concerns* can negatively affect the quality of even well modularized systems implemented by these techniques [43]. Crosscutting concerns are related to issues transversal to many modules, such as the synchronization and fault detection tasks in an automatic measurement system. They cause the duplication of portions of code in several different modules, by negatively affecting the maintainability and reusability.
This means that at run-time each component must encapsulate its part of information related to the cross-cutting concern even when it does not need it (thus wasting memory resources). The aspect-oriented architecture enforces, as much as possible, a centralized design in which synchronization state is maintained in the related aspects: components dealing with the cross-cutting concern are involved by the aspect encapsulating it for all components. When a component does not need that feature, no data is stored for it in the aspect and no memory is wasted at all.
Aspect-oriented programming (AOP) [17] is an extension of the object-oriented paradigm that provides new constructs for improving the separation of concerns and supporting their crosscutting. AOP defines a kind of program unit, the *aspect*, for specifying concerns separately, and rules for weaving them to produce the overall system to be run. Like a class of objects, an aspect introduces a new user-defined type into the system's type

hierarchy, with its own methods, fields, and static relationships to other types. Usually, an AOP system can be seen as composed by two parts: (*i*) one consisting of traditional modularization units (e.g. classes, functions) and referred as the *base system* or *core concern*, and (*ii*) the other one consisting of aspects, encapsulating the crosscutting concerns involved in the system, and usually referred as the *secondary concerns*. The features AOP provides for implementing crosscutting concerns in aspects can be classified in: (*i*) *dynamic crosscutting features*: implementation of crosscutting concerns by modifying the runtime behaviour of a program; (*ii*) *static crosscutting features*: modification of the static and structural properties of the system.

Dynamic crosscutting is implemented by using pointcuts and advice. An *advice* is a code fragment executed in specified points at the program runtime. The points in the dynamic control flow where the advice code is executed are called *join points*. A *pointcut* defines the events (such as method call or execution, field get and set, exception handling and softening) triggering the execution of the associated advices. A pointcut is an expression pattern matched during execution to join points of interest. Every advice is associated to a pointcut defining the joinpoint(s) at which it must be applied. Advice code can be executed either before, after, or around the intercepted joinpoint. A *join point shadow* is the static counterpart, in the code, of a join point; equivalently, a joinpoint is a particular execution of a joinpoint shadow. Aspects and base program are composed statically by a weaving process. The *weaver* is the component of an AOP programming language environment (such as AspectJ [44]) responsible for the weaving process. The weaver inserts instructions at join point shadows to execute the advice to be applied at the corresponding join points. The weaver may need to add runtime checks to the code inserted at a join point shadow in order to perform parameters binding and other requested computations.

The static crosscutting features of AOP implement crosscutting concerns by modifying the static structure of the system. An aspect can introduce new members (i.e. fields, methods, and constructors) to a class, or interface; change or add parents for any class or interface; extend a class from the subtype of the original super-class or implement a new interface. These features are called intertype declarations. An example of a straightforward AOP program (an AO version of the 'Hello World' program implemented in AspectJ [45]), enlightening AO basic working, is reported in Fig. 3.8. In the figure, the code of

```
// HelloWorld.java
public class HelloWorld {

    public  static  void  tell(String
message) {
       System.out.println(message);
    }

    public        static        void
tellPerson(String   message,   String
name) {
       System.out.println(name  +  ",  "
+ message);
    }
}
```

```
// GreetingsAspect.java
public aspect GreetingsAspect {
    pointcut    callTellMessage()     :
call(public        static         void
HelloWorld.tell*(..));

    before() : callTellMessage() {
       System.out.println("Good
morning!");
    }

    after() : callSayMessage() {
       System.out.println("Bye Bye!");
    }
}
```

Figure 3.8: A simple AO program.

the class *HelloWorld* and the aspect *GreetingsAspect* are reported. The aspect defines a pointcut and two advices. The *callTellMessage()* captures calls to all public static methods with names that start with tell. In the example, the pointcut captures the calls to *tell(...)* and *tellPerson(...)* methods in the *HelloWorld* class taking any arguments. The two advices, one before and one after, associated to the *callTellMessage()* pointcut will cause, respectively, the printing of the "Good morning!" and "Bye Bye!" text strings just before and after each message printed by the *tell()* and *tellPerson()* methods.

3.1.4 Fault detector

Nowadays, test automation is a must for product quality and reliability, both in industry and in research. Intelligent measurement systems are used deeply in delicate tests involving several instruments. One of their key issues is the capability of assuring a proper termination to the test process. With this aim, a suitable fault detection software turns out to be an adequate reaction to anomalous working [46]. Devices provide information about their status continuously and, in case of abnormal working, a fault condition is pointed out. Software implementation of fault detection is a well-known strategy for dealing with failures caused by both hardware and software faults [47]. Compared to hardware implementation, it has the advantage of higher flexibility and cost effectiveness. Today, it is a widely used technique, and emerging application areas for cost-effective dependable systems will further increase its importance [48]. Thus, the software implementation of a fault detector affects the overall system quality, in particular maintainability and reusability.

The analysis of state-of-the-art automatic measurement systems highlighted that fault detection is usually *scattered* all over different software components, mainly with reference to devices' hierarchy. This means that often the concrete classes of virtual devices contain duplicated code for fault detection, thus making harder their comprehension, testing, and maintenance. The aspect-oriented programming is proposed for the development of software components for fault detection in order to overcome the drawbacks due to their cross-cutting nature. The cross-cutting concerns related to fault detection of a large measurement software project are separated and handled better by encapsulating them into aspects. In this way, the reusability of system modules improves.

In the following, the proposed AOP-based approach to the development of software for fault detection in automatic measurement systems is described. In particular, (*i*) the *measurement fault analysis*, and (*ii*) the *fault detector architecture* are highlighted.

Measurement Fault Analysis

Most common faults in an automatic measurement system can be classified according to the sources and the synchronization of the related handling operations. According to the *sources*, faults can be classified as arising from:

- hardware devices, when devices are in a faulty internal state due to hardware anomaly or to an external condition. Device internal fault detection can scale from very basic internal information to very complex routines forcing the device in different states. Correspondingly, concrete aspects of the fault detection subsystem must be capable of intercepting relevant changes in the device status, decoding them, and broadcasting high-level faults description to the interested components.

- the measurement environment, when the measurement environment is compromised by external or internal alterations.

- software components, when software components are in any non consistent state, owing to an incorrect use violating pre-conditions and/or post-conditions, or to the presence of unresolved or undiscovered bugs.

According to the *synchronization of the related handling operations*, faults can be classified as:

- synchronous, when an anomalous operation is attempted. In this case, the following policies can be applied, according to the criticality level and the kind of the fault:
  - k-times retry: some operations are retried until the device goes back in a consistent state, without any performance constraint on the operation. As an example, an initialization reset tried several times during a slow start up of a multimeter.
  - multicast warning and continue: for operations requested in wrong conditions, requests can be ignored by issuing only a notification warning. As an example, a digital scope is triggered when previous data digitization is not ended, or when a stop or an abort is issued on a already stopped instrument.
  - multicast fault and deny operation: for operations to not be executed when specified faults occur. In this case, the operation is denied and the fault information is sent to the pertinent components in order to be properly handled.
  - multicast an immediate shutdown request and deny operation: for the most critical situation when a fault on an critical operation in a risky device should be blocked at the lowest level. Moreover, since the system as a whole is to be shutdown gracefully as fast as possible, a high priority request of system shutdown is sent to the fault handler component. These faults are handled suitably by wrapping operations through concrete pointcuts, bounded to around advices defined by abstract aspects of the fault detector component,
- asynchronous: when hardware or environment anomalies, in a whatever moment not synchronized with the measurement operations generate faults forcing devices

in faulty states usually detected by suitable monitoring. The detection is based on field access pointcut expressions bound to the decoding logic used to detect changes in the status of devices.

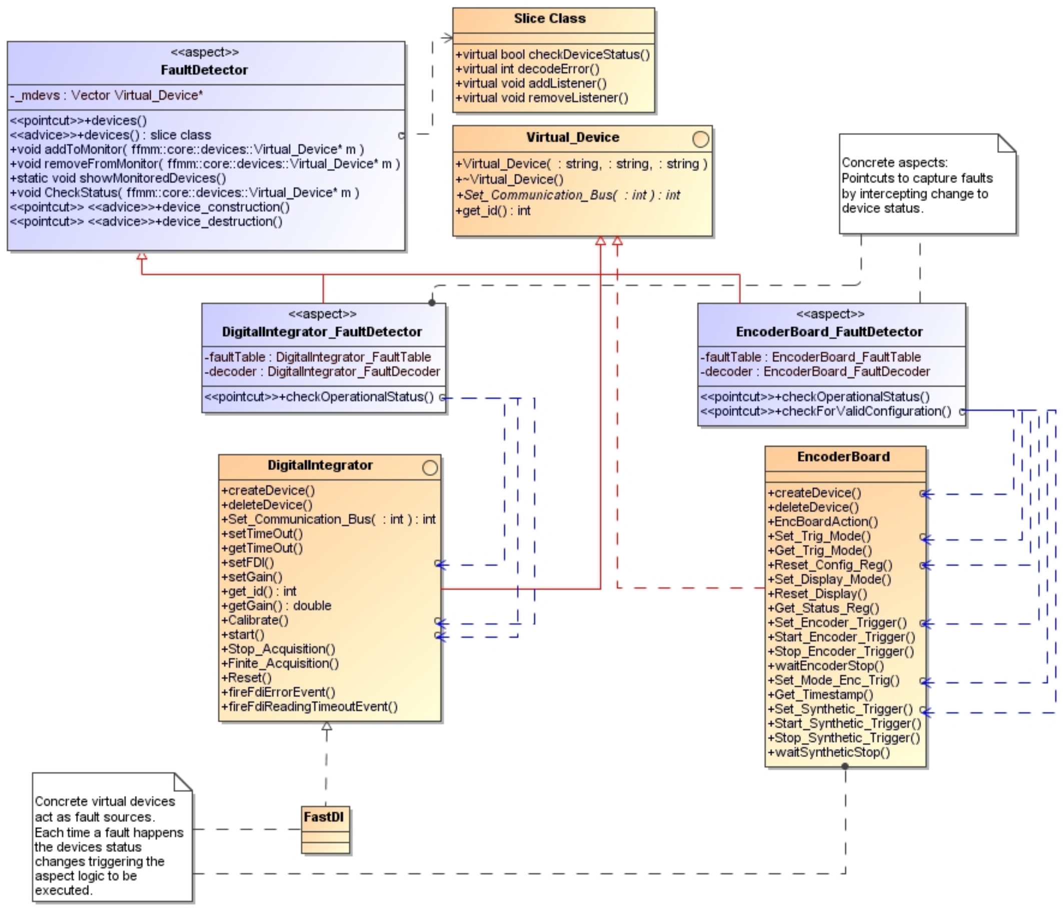
AOP architecture

The proposed architecture is based on:

- a fault detection subsystem, designed for: (*i*) monitoring the "health" state of the measurement system's component devices; (*ii*) catching software faults such as stack overflow, live-lock, deadlock, and application-defined faults, as soon as they occur;
- a fault notification subsystem, responsible for: (*i*) receiving the sequence of occurring faults from all the system components constantly; (*ii*) storing the diagnostic history and providing access to other components or to external humans in order to react to faulty events adequately.

These two subsystems exploit three key components: (*i*) a *FaultDetector* aspect hierarchy, allowing the code related to the fault detection logic to be removed from the modules implementing the virtual devices; (*ii*) *FaultDecoder* tables, needed by concrete aspects for decoding status representation specific of concrete *VirtualDevices*; (*iii*) *FaultListeners* in order to dynamically and to bind (obliviously) components responsible for the fault management to the ones acting as fault sources.
The aspects in the *FaultDetector* hierarchy intercept faults by means of the *FaultDecoder* classes. The decoders are capable of handling groups of similar devices and knowing internal state structure and encoding. They provide the aspect logic by *FaultTable* instances encapsulating fault information to be sent to the interested components through the *FaultNotifier* layer.
In Fig. 3.9, the proposed *FaultDetector* hierarchy is depicted, by highlighting the static relationships among *VirtualDevice* classes, *FaultDetector* aspects, and some concrete virtual devices. The figure shows the role played by the *FaultDecoder* and *FaultTable* for the Fast Digital Integratot (*FastDI*) device. Encoded fault information is extracted from the

FastDI device by context interception and is decoded by a concrete *FastDIFaultDecoder*. The decoded information is then provided to the *DigitalIntegratorFaultDetector*, responsi-



<<aspect>>
FaultDetector
-_mdevs : Vector Virtual_Device*
<<pointcut>>+devices()
<<advice>>+devices() : slice class
+void addToMonitor(ffmm::core::devices::Virtual_Device* m)
+void removeFromMonitor(ffmm::core::devices::Virtual_Device* m)
+static void showMonitoredDevices()
+void CheckStatus(ffmm::core::devices::Virtual_Device* m)
<<pointcut>> <<advice>>+device_construction()
<<pointcut>> <<advice>>+device_destruction()
Slice Class
+virtual bool checkDeviceStatus()
+virtual int decodeError()
+virtual void addListener()
+virtual void removeListener()
Virtual_Device
+Virtual_Device(: string, : string, : string)
+~Virtual_Device()
+Set_Communication_Bus(: int) : int
+get_id() : int
Concrete aspects: Pointcuts to capture faults by intercepting change to device status.
<<aspect>>
DigitalIntegrator_FaultDetector
-faultTable : DigitalIntegrator_FaultTable
-decoder : DigitalIntegrator_FaultDecoder
<<pointcut>>+checkOperationalStatus()
<<aspect>>
EncoderBoard_FaultDetector
-faultTable : EncoderBoard_FaultTable
-decoder : EncoderBoard_FaultDecoder
<<pointcut>>+checkOperationalStatus()
<<pointcut>>+checkForValidConfiguration()
DigitalIntegrator
+createDevice()
+deleteDevice()
+Set_Communication_Bus(: int) : int
+setTimeOut()
+getTimeOut()
+setFDI()
+setGain()
+get_id() : int
+getGain() : double
+Calibrate()
+start()
+Stop_Acquisition()
+Finite_Acquisition()
+Reset()
+fireFdiErrorEvent()
+fireFdiReadingTimeoutEvent()
EncoderBoard
+createDevice()
+deleteDevice()
+EncBoardAction()
+Set_Trig_Mode()
+Get_Trig_Mode()
+Reset_Config_Reg()
+Set_Display_Mode()
+Reset_Display()
+Get_Status_Reg()
+Set_Encoder_Trigger()
+Start_Encoder_Trigger()
+Stop_Encoder_Trigger()
+waitEncoderStop()
+Set_Mode_Enc_Trig()
+Get_Timestamp()
+Set_Synthetic_Trigger()
+Start_Synthetic_Trigger()
+Stop_Synthetic_Trigger()
+waitSyntheticStop()
FastDI
Concrete virtual devices act as fault sources. Each time a fault happens the devices status changes triggering the aspect logic to be executed.

Figure 3.9: An excerpt of the hierarchy of the proposed fault detector.

ble for enforcing fault management policies according to the fault kind. Moreover, it sends the fault data to the interested software components. The *FaultDetector* is responsible for defining high-level pointcuts capturing relevant operations affecting the state of devices. In the measurement system, the *VirtualDevice* hierarchy models and organizes all the physical devices involved in the measurement process. Each device has an internal status; modifications to such status are captured by means of concrete sub-aspects executing the logic needed to decode it, as well as detecting if and where a device notified an internal fault. In each *FaultDetector* sub-aspect, associated to main devices categories, the mapping logic towards concrete devices classes belonging to the same family is defined and the common behaviours can be factorized, such as needed. The coarseness of the mapping among aspects and concrete devices allows a very flexible reuse of fault detection logic for similar devices by encapsulating it in few modules (instead of spreading it all over the device classes). Fig. 3.10 depicts the different levels of fault interceptions, according to the fault types. The bottom level takes care about very specific issues and features of concrete devices to encapsulate in dedicated sub-aspects. At the middle level, concrete aspects, by using decoders, perform continuous monitoring of devices' status. The top

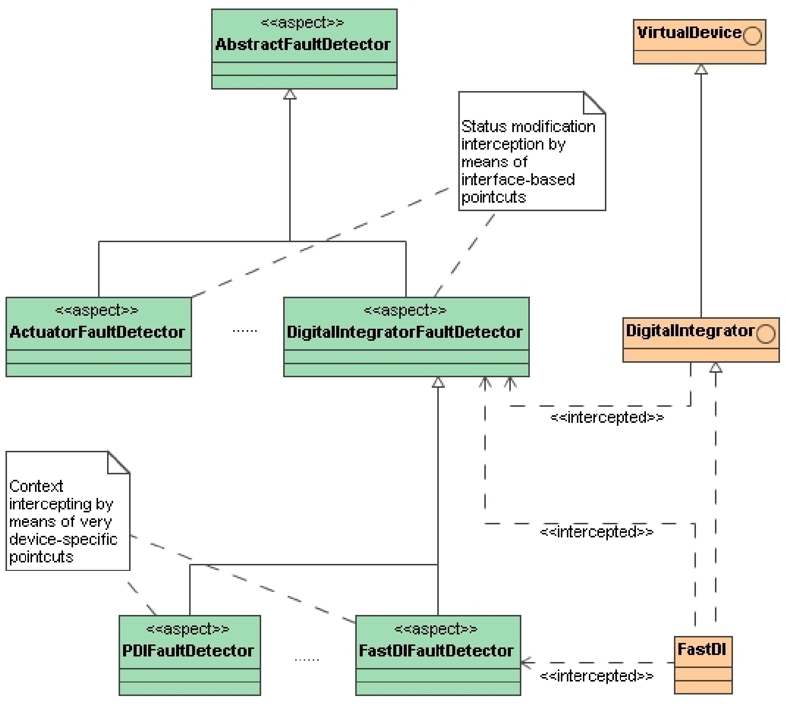


Figure 3.10: Levels of faults interception.

level includes abstract aspects implementing the fault detection logic reusable in concrete sub-aspects. In Fig. 3.11, the aspect mapping layer of the fault notification is shown. The services to dynamically associate handlers to fault sources in the measurement system are provided. The sub-aspects of the *FaultHandler* aspect have the responsibility of making aware the concrete classes (like the *TestManger* responsible of performing the test session) of the faults that happens in the system. This solution allows fault handling logic to be reused in the super-aspects and does not force concrete classes in the system to implement

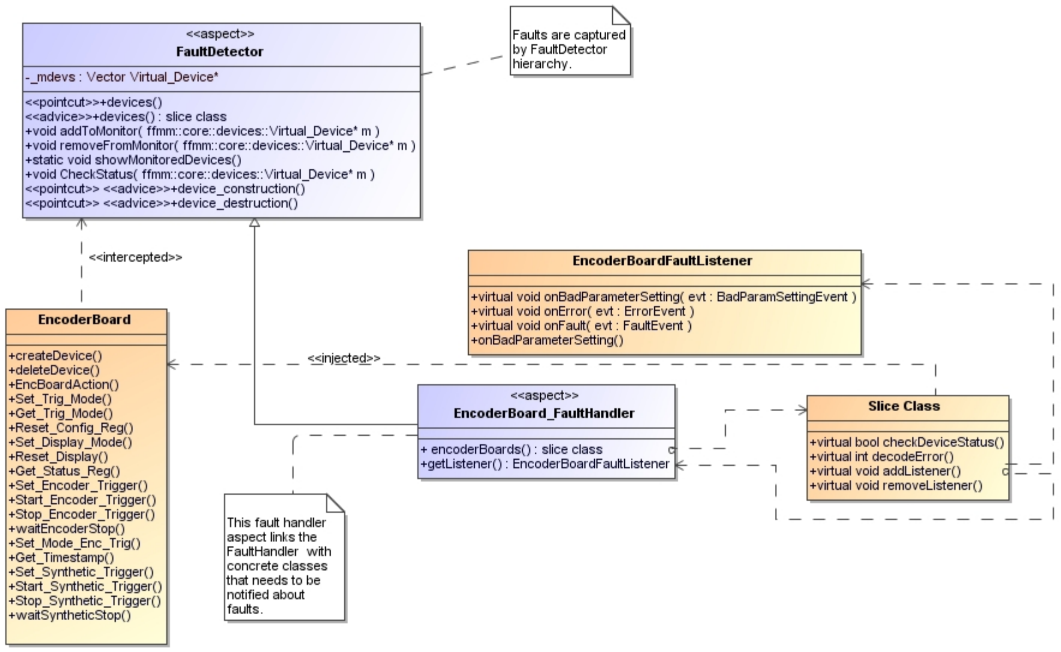

<<aspect>>
FaultDetector
-_mdevs : Vector Virtual_Device*
<<pointcut>>+devices()
<<advice>>+devices() : slice class
+void addToMonitor(ffmm::core::devices::Virtual_Device* m)
+void removeFromMonitor(ffmm::core::devices::Virtual_Device* m)
+static void showMonitoredDevices()
+void CheckStatus(ffmm::core::devices::Virtual_Device* m)
<<pointcut>> <<advice>>+device_construction()
<<pointcut>> <<advice>>+device_destruction()
Faults are captured by FaultDetector hierarchy.
<<intercepted>>
EncoderBoard
+createDevice()
+deleteDevice()
+EncBoardAction()
+Set_Trig_Mode()
+Get_Trig_Mode()
+Reset_Config_Reg()
+Set_Display_Mode()
+Reset_Display()
+Get_Status_Reg()
+Set_Encoder_Trigger()
+Start_Encoder_Trigger()
+Stop_Encoder_Trigger()
+waitEncoderStop()
+Set_Mode_Enc_Trig()
+Get_Timestamp()
+Set_Synthetic_Trigger()
+Start_Synthetic_Trigger()
+Stop_Synthetic_Trigger()
+waitSyntheticStop()
EncoderBoardFaultListener
+virtual void onBadParameterSetting(evt : BadParamSettingEvent)
+virtual void onError(evt : ErrorEvent)
+virtual void onFault(evt : FaultEvent)
+onBadParameterSetting()
<<injected>>
<<aspect>>
EncoderBoard_FaultHandler
+ encoderBoards() : slice class
+getListener() : EncoderBoardFaultListener
Slice Class
+virtual bool checkDeviceStatus()
+virtual int decodeError()
+virtual void addListener()
+virtual void removeListener()
This fault handler aspect links the FaultHandler with concrete classes that needs to be notified about faults.

Figure 3.11: Fault notification publish-subscribe architecture.

fault handling code. Any component in the system can react to specific faults that occur anywhere in the system and perform the needed actions to handle them. Moreover, since concrete classes (*TestManager* or any other components interested in monitoring faults) are oblivious of being faults' handlers, the monitoring relationships can be changed by simply acting on aspect mapping. Commonalities among different fault handling logics can be factored out in the aspects while multiple observations of different kinds of faults can be easily accomplished by defining several mapping aspects for a single concrete class.

3.1.5 Synchronizer

A key issue in automatic measurement systems is the capability of assuring a proper software synchronization to the test procedure. Usually critical measurement constraints are satisfied by running several asynchronous tasks contemporaneously on the same platform, by maximizing the degree of parallelism in the system. This improves efficiency in concurrent measurements, but, on the other hand, imposes synchronization constraints among the interacting processes. Whereas severe time constraints in a measurement procedure require dedicated hardware, the abovementioned software interaction often requires programming strategies capable of dealing with events asynchronously generated and notified to the processes once a synchronization point is reached [49, 50, 51]. Today, *software synchronization* is a widely used technique, and emerging application areas for cost-effective dependable systems will further increase its importance. Moreover, the implementation strategies of task synchronization not only affect the system performance, but also its quality, in particular the modularity, maintainability and reusability.

AOP architecture

The proposed AOP-based architecture for task synchronization in automatic measurement systems aims at modularizing concurrency and synchronization concerns, as well as guaranteeing system correctness, while increasing performance and safety. It is an abstract aspect layer composed by a simple aspect framework to be reused in the development of synchronization control in different application domains. The modularization achievable by the proposed architecture makes the synchronization control easy to evolve and simplifies the complexity of the remaining parts of the software, such as devices, fault detection or logging modules, by decoupling concurrency and synchronization control code from them.
Synchronization is a crosscutting concern particularly hard to modularize through object-oriented programming and design patterns. The proposed AOP-based architecture is the result of the analysis of several existing object-oriented software systems implementing synchronization and concurrency, that revealed some typical deficiencies of OOP implementation for synchronization. In particular, the deficiencies are related mainly to extensibility, modularity, encapsulation, reusability.

In Fig. 3.12, a UML class diagram of the proposed architecture conceived as an AOP-based variant of the Synchronization Manager design pattern is shown. The stereotype "aspect" is used to distinguish aspects and classes. All the aspects in the diagram are abstract ones. The synchronization aspects can be easily integrated/reused in other architectures and software systems. Indeed, just the components and services to be syn-

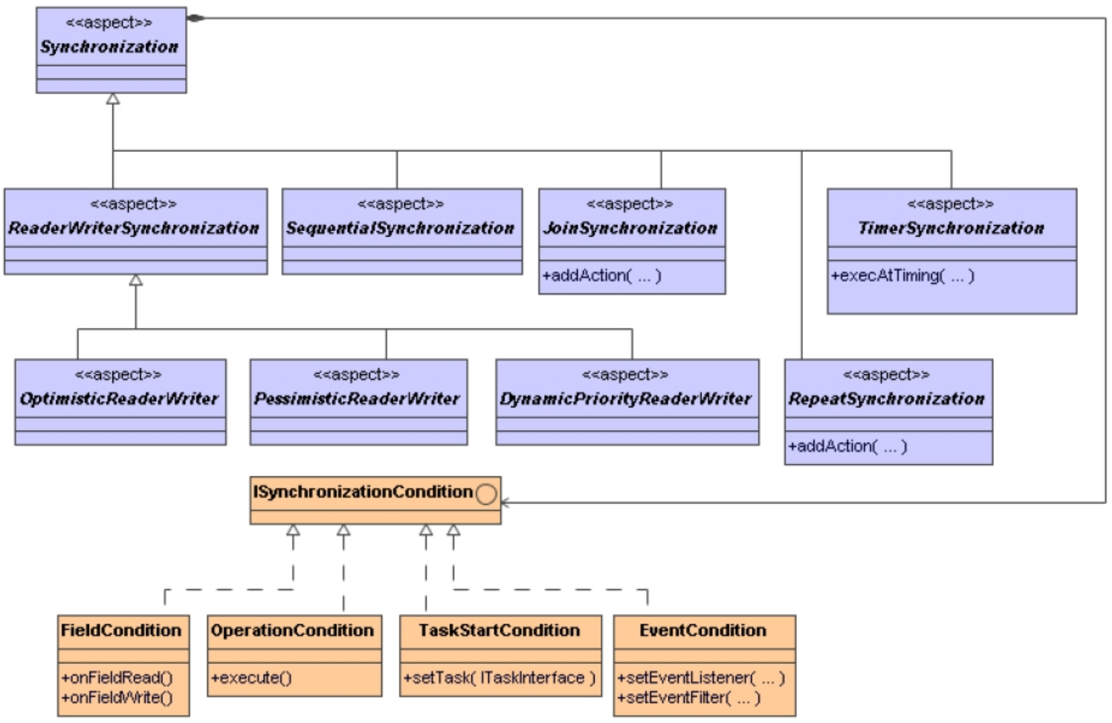


Figure 3.12: The proposed AOP-based architecture of synchronizer for an automatic measurement system.

chronized, as well as the policy for their synchronization, have to be identified.
The abstract *Synchronizer* aspect provides reusable code and behaviour for implementing and modularizing the synchronization logic and policies.
Concrete aspects have two main responsibilities: (*i*) intercept components and services interactions to be synchronized; (*ii*) enforce the right synchronization policy in the right context.
Three main issues are related to the synchronization management: the synchronization *policy* to be adopted, the synchronization *condition* to be defined, and the specification of the *context* using synchronized elements. The proposed architecture separates these three components allowing the synchronization logic to be reused in the super-aspects, without forcing concrete classes in the base system to implement synchronization handling code. Concrete classes are oblivious to synchronization scenarios: thus, the synchronization policies can be changed by simply acting on the aspects. In the proposed architecture, several synchronization policies are available in order to support the most interesting scenarios arising in a measurement sessions. In particular, the following policies were defined to synchronize data transfers among devices: (*i*) joined, repeated, sequential task execution, (*ii*) optimistic and pessimistic readers/writers, (*iii*) dynamic priority readers/writers, (*iv*) producer/consumer, and (*v*) support for active devices synchronization (scaling from single internal thread to a cooperating pool of k threads).
As far as the synchronization conditions are concerned, the architecture provides the basic conditions to be aggregated in order to build more complex conditions. They can be used in association with existing or new synchronization policies. The basic implemented conditions are related to field read/write events, operation execution or invocations and well-defined role operation execution events. For each new device added to the station, the related synchronization code is added to the synchronization hierarchy.
The resulting architecture is extensible since concrete aspects implementing specific synchronization policies can be added easily and designed to implement new kind of policies such as needed. The added policies only need to implement interface and concrete mapping logic to intercept the client contexts. The architecture also fosters reusability since

existing policies can be reused in several different contexts and the synchronization logic is completely decoupled from the client code.

3.2 Domain specific language

After developing the kernel and the main components of FFMM, it was necessary to provide the test engineer with a easy and fast way to write measuremet scripts. To achieve this goal, a *Measurement Domain Specific Language* (MDSL) was developed.

3.2.1 Proposed approach

A language is a set of terms and expressions which are bounded by a set of syntax and semantic rules and used for communication within a domain. *General Purpose Languages* (GPLs) are not specialized and are suited for a wide area of applications from business processing up to scientific computing.
Conversely, *Domain Specific Languages* (DSLs) are explicitly tailored to a target domain: *"DSLs are languages tailored to a specific application domain. They offer substantial gains in expressiveness and ease of use compared with GPLs in their domain of application"* [52]. Complex constructs and abstraction of the domain are offered within the language, thus increasing its expressiveness in comparison to GPLs. The higher abstraction level, the compactness, and consequently the better readability and writability enable a group of people larger than expert programmers to be productive using the DSL. This improves productivity, since it is possible to express solutions for domain problems with a smaller effort, and decreases maintenance costs.
A DSL has also potential shortcomings. One drawback is the high development effort which is needed for a new language. The language developer needs at least experience in language design and knowledge about the target domain. He has to find suitable abstractions, the right scope and balance between GPL and DSL constructs. Furthermore the language must be implemented and maintained.
Other problems are tool availability, user training costs and performance. While general purpose languages have a strong tool support, corresponding tools for a new DSL have to

be created. Proper development methodology and suitable tools have to be chose to avoid that DSL development costs surpass the estimated saving by using a DSL. Widely used IDEs like *Eclipse* or *Visual Studio* offer deep integration with these languages like powerful editors with syntax highlighting and checking, integrated compilers and advanced debuggers.

Finally, a DSL might lead to performance loss with respect to other languages. If performance is not critical, the other DSL benefits will make this a minor problem. Otherwise, special attention has to be paid to the possible optimizations that potentially make the performance loss negligible.

In FFMM, test engineers are not necessarily skilled programmers and have to produce concise and bug-free FFMM specific applications (Fig. 3.13). Thus, a new *Measurements*

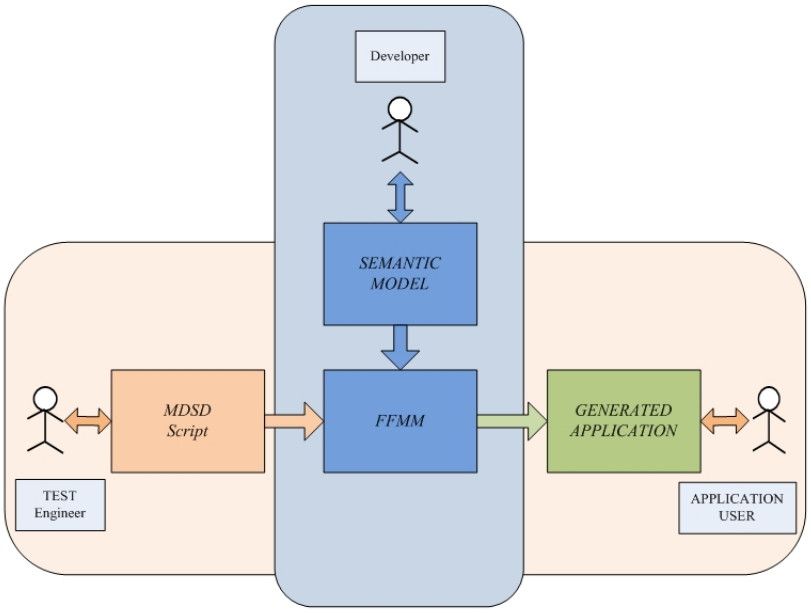


Figure 3.13: User roles in FFMM.

Domain Specific Language (MDSL) with specialized constructs was designed in order to: (*i*) define logical, numeric, and temporal conditions; (*ii*) perform conditional branching, immediate verification of conditions, verification of conditions within a time period, and continuous verification of conditions; (*iii*) be able to define events based on measurement value and attribute changes, time changes, external event notifications, and user inputs;

(*iv*) subscribe and unsubscribe to events, and respond to them with behaviors that include sending text messages to users or commands and generate measurements; (*v*) enable, configure and disable framework service; (*vi*) be able to interact with the user through a command prompt; (*vii* compare measurement data against specified criteria within a specified time period, and compute results that are numeric and Boolean functions.

3.2.2 Architecture

The proposed MDSL is based on a *Semantic Model*, seen as a part of the FFMM domain model. It captures the Measurement Test Procedure core structure and behavior. In Fig. 3.14, the proposed approach for the transformation of the Measurement Domain-Specific Description (MDSD) into the final code is shown.

The DSL script, written by the Test Engineer, is parsed to create an internal file treated

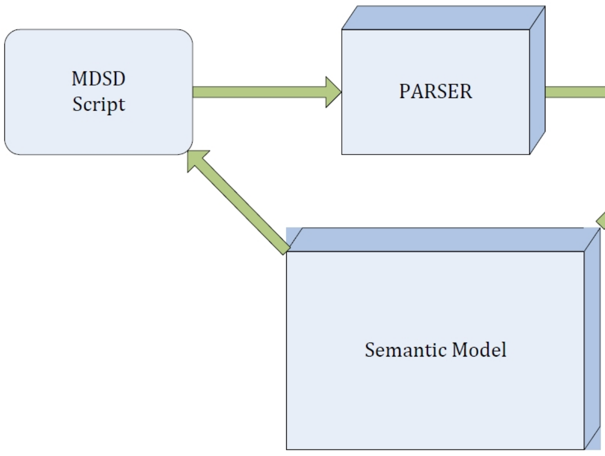


Figure 3.14: Code generation process.

by the semantic model (Fig. 3.14).The parser reads the script and populates the Semantic Model.

3.2.3 DSL in FFMM

A DSL can be thought of as a form of user interface [53]. It therefore provides an additional view of FFMM, specifically conceived for the test engineer. As depicted in Fig. 3.15, through the DSL it is possible to separate the developer view (interacting with the system in C++) from the test engineer view (interacting through the DSL). The developer can

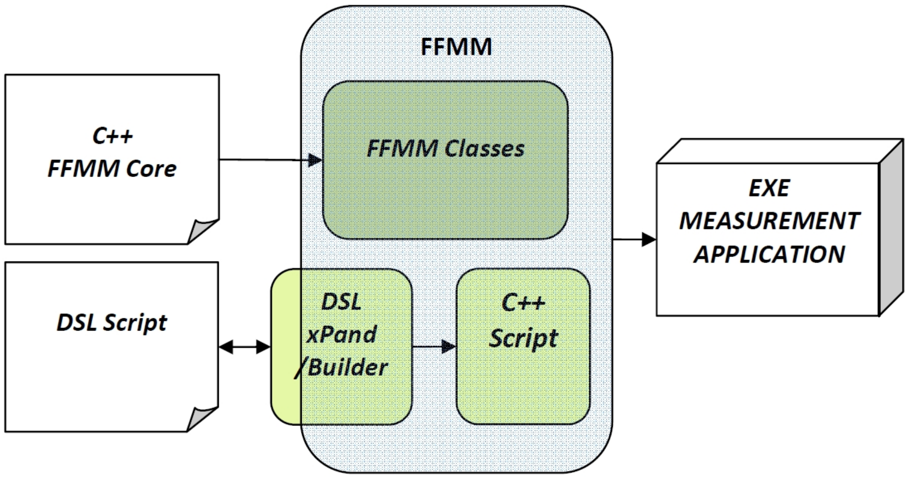


Figure 3.15: Developer view versus test engineer view of FFMM through DSL.

operate with C++ at any level in the system, including the definition of a measurement script. On the other hand, the test engineer, with limited effort and programmation skills, can operate at script level by means of the DSL, defining a procedure that the builder will translate into C++ through an interaction with FFMM classes.

3.3 Automatic generation of user interfaces

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 level of abstraction they offer is in general rather low. Therefore, a visual editor, such as available in many commercial programming environments [54], 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.

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 [55].
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.

3.3.1 The Model-Viewer-Interactor paradigm

To avoid that test engineers have to deal with raw graphical characteristic of software measurement system, the proposed architecture is organized by separating functional from appearance aspects of the interface through a three-way decomposition: (*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. 3.16), derived as an evolution of the model-based approach [54].
The Model is composed by the data structures and the classes of the framework involved

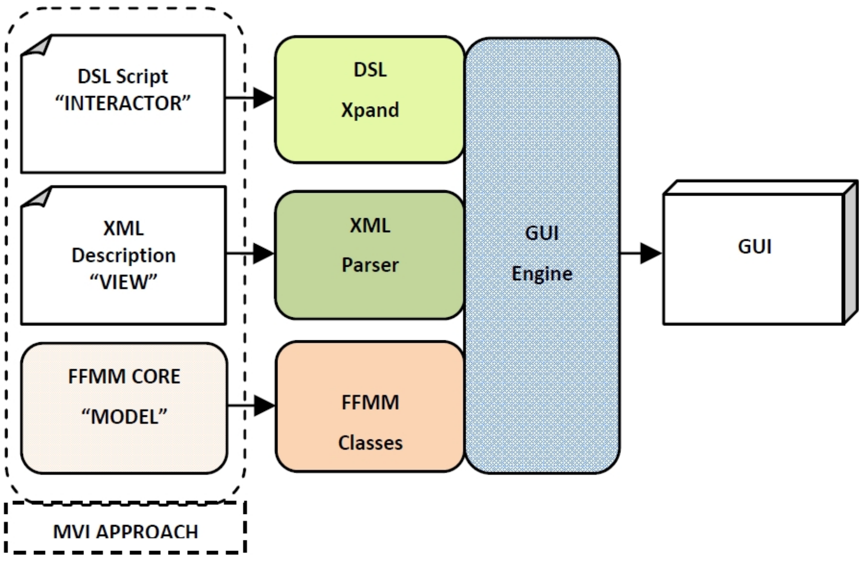


Figure 3.16: Model-Viewer-Interactor approach.

in the GUI generation and subject to change by them. A typical example is offered by the *Device* classes involved in the configuration step of a measurement procedure.

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. 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 [56].
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, the test engineer defines the components 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 Graphic Interactor Component objects only.

3.3.2 The GUI engine

The main aim of the proposed Model-View-Interactor paradigm is to allow the test engineer to develop GUI applications with a minimal effort and without graphical knowledge. This aim is achieved mainly through the GUI engine structure (Fig. 3.17), allowing all the GICs (*Graphic Interactor Components*), encapsulating all common aspect of graphical components [57, 58], to be built.

The GUI engine architecture is composed by several classes: (*i*) GIC, providing the *TestManager* with 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

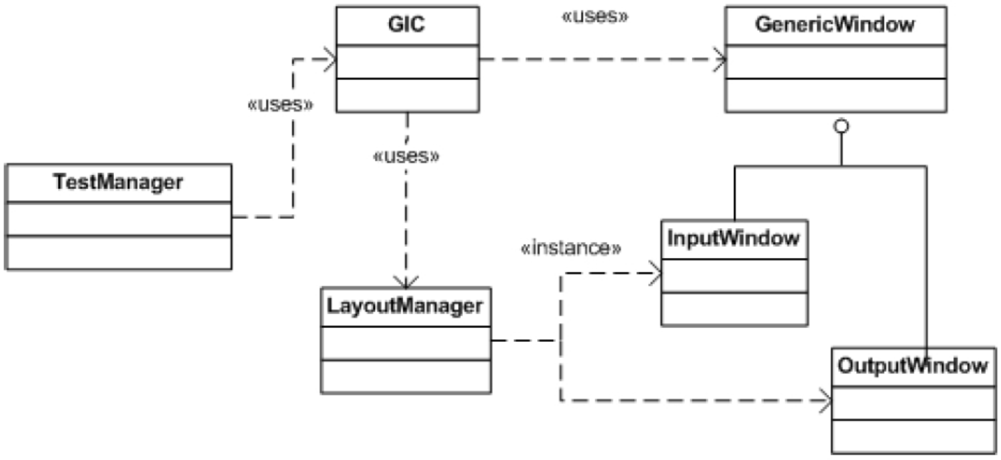


Figure 3.17: Abstract factory pattern for the GUI engine.

the dimension and position parameters [56].

As an example, if the test engineer needs to ask as input an integer value at runtime, he will use the *capture()* method of GIC object in the measurement script:

int a;

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

By inserting in the script only this instruction, a form is displayed (Fig. 3.18), and the value entered by the user is stored in the variable pointed.

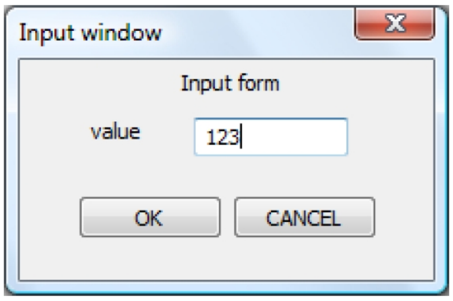


Figure 3.18: Final form aspect.

3.4 Analyzer

The framework includes libraries of software modules available to the user for developing quickly measurement applications. Often, the measurement results need to undergo a process of analysis in order to be understandable and useful. Therefore a specific component, the *Analyzer*, was included in the project in order to store and make available the required data analysis routines. Despite the simple structure of the component, specific conceptual work was devoted to the choice of the most convenient approach and to the definition of the working principles of algorithms for data reduction and harmonic resolution enhancement, presented in the following.

3.4.1 Data compressor

A real-time algorithm of data reduction, based on the combination of two lossy techniques specifically optimized for high-rate magnetic measurements in two domains (e.g. time and space), is proposed. The first technique exploits an adaptive sampling rule based on the power estimation of the flux increments in order to optimize the information to be gathered for magnetic field analysis in real time. The tracking condition is defined by the target noise level in the Nyquist band required by post-processing procedure of magnetic analysis. The second technique uses a data reduction algorithm in order to improve the compression ratio while preserving the consistency of the measured signal. The allowed loss is set equal to the random noise level in the signal in order to force the loss and the noise to cancel rather than to add, by improving the signal-to-noise ratio.

Data reduction techniques

Higher and higher sampling rate of measurement setups produces an exponential rise in data storage requirements. Thus, data reduction algorithms are needed in order to decrease the size of measurement results by controlling the quality loss simultaneously.
Lossless compression algorithms, such as Huffman coding [59], arithmetic coding [60], or lossless JPEG [61], exploiting statistical redundancy for a more concise representation without error, are the first logical choice. However, when original data contains sufficient

redundancy, and suitable assumptions on approximation can be made, better compression ratios are achieved by lossy methods. Various approaches to dimensionality reduction, such as principal components analysis, entropy measures for ranking features, and methods to discretize data were reviewed in [62]. In creating CAD geometry data from existing parts by surface laser scanning, uniform and non-uniform grid methods were proposed [63]. In biomedical applications, adaptive sampling algorithms are widely used to reduce data size at their source, while preserving clinical acceptability of the reconstructed signal [64]. In particular, significant-point-extraction algorithms, retaining only samples with key information such as Turning Point (TP) [65], Amplitude Zone Time Epoch Coding (AZTEC) [66], and Fan [67], are proposed. In sensor networks, the distribution led to the minimization of energy and network bandwidth, by adapting sampling rates to actual conditions [68, 69, 70, 71]. However, such applications have been developed mainly for measurements with low sampling rates (typically below 100 S/s). Furthermore, the proposed methods turn out to be highly complex, by requiring for example the solution of a constrained optimization problem [70]. Computational burden limits application to fast real-time data reduction above a few tens of samples per second. Other techniques, such as histogram equalization [72] and entropy-based adaptive sampling [73], needs for the data set as a whole, thus preventing their use for fast real-time applications.
In this section, a real-time two-domains data reduction algorithm based on adaptive sampling and significant point extraction, specifically optimized for high-rate magnetic measurements, is presented. In particular, in the following the proposed algorithm is illustrated by highlighting the basic ideas, the procedure, and main design criteria for magnetic flux measurements.

Proposed approach

In the following, (*i*) the basic ideas, (*ii*) the strategy, and (*iii*) the procedure of the proposed algorithm are illustrated.

Basic ideas The algorithm was based on the following design concepts:

1. Data are reduced by analyzing the results in real time during the measurement and

by storing consequently only the necessary ones. Two different lossy techniques, applied in different domains (e.g. time and space) and suitable for the signal features in those domains, are applied. The first technique, the *adaptive tracking sampler*, analyzes the acquired data on the basis of a Nyquist power threshold mechanism of tracking, in order to adapt the sampling rate for capturing only the significant features of the signal. The second technique, the *noise-cancelling compressor*, is based on a significant point extraction: only the samples containing significant information are retained by specifying the maximum error allowed on the reduced signal.

2. In the *adaptive tracking sampler*, according to the fundamental data reduction principle of gathering only the minimum amount of information sufficient to any kind of further analysis, a rule for adapting the sampling by tracking the signal power in the Nyquist bandwidth is defined [71]. Under the assumption of signal stationarity with respect to the algorithm adaptation time, sampling rate is modified by analyzing the signal in the frequency domain through a general technique for determining its spectral content. The sampling rate is adapted in real time when a limit condition is approached. The limit condition is based on the sampling rate defining the Nyquist bandwidth including the minimum power necessary to capture the required features of the signal. The power is estimated in a frequency band whose upper limit is the Nyquist frequency, compared with thresholds representing the power level in the band for which the current sampling rate is considered adequate. In particular, the sampling rate is increased/decreased when relatively high-frequency terms are present/absent into the power estimator.

3. In the second technique, the *noise-cancelling compressor*, between two observations where the signal is monitored and the sampling rate adapted at evenly spaced time intervals, the resulting signal is decomposed in a set of linear problems, to which classic lossy algorithms for significant point extraction, such as the Fan [65, 67], are applied. A suitable mechanism to control the related error allows a proper trade-off

among compression speed, compressed data size, and quality loss. In particular, under the assumption of random noise (i.e. white but not necessarily Gaussian) and deterministic signal, if the allowed loss is set equal to the noise level, loss and noise tend to cancel rather than to add, by increasing signal-to-noise ratio [74]. In other words, by means of a suitable design strategy, the loss mainly involves the noise content of the signal, by improving both the signal quality and the compression ratio simultaneously. This approach does not require a priori knowledge of the spectral properties of the noise-free signal.

Strategy In the *adaptive tracking sampler*, if the sampling rate is f_s, the power estimator is unable to adequately treat a signal with frequency approaching $f_s/2$. Moreover, if the sampling rate is only f_s, the presence of signals with frequency greater than $f_s/2$ can not be revealed. However, an increase of the estimated power in the observed frequency band [f_0, $f_s/2$], with f_0 ¡ $f_s/2$ to be suitably chosen, at a level above a suitable threshold, can reveal higher frequency components approaching the limit of the current observable band. Hence, an increase in resolvable frequencies can be obtained by triggering an increase in the sampling rate. Likewise, the absence of large terms in the band [f_0, $f_s/2$] is used to trigger a decrease in the sampling rate. In this way, error penalties incurred by slightly reducing the sampling rate in places where the signal is relatively uninteresting are not likely to be high. The tracking mechanism is explained graphically in Fig. 3.19.

This is true if the variation of the power content of the signal can be considered stationary with respect to the algorithm adaptation time. The frequency f_0 is to be chosen as a function of f_s, thus both the extremes of the observed frequency band can vary while the sampling rate is adapted.

While the tracking mechanism adapts the sampling frequency by analyzing subsequent batches of data, the *noise-cancelling compressor* discards possible redundant points between each pair of consecutive observations. The information loss is controlled by means of the tolerance ε: the approximation maximum error. The *noise-cancelling compressor* operates on the acquired samples, by approximating them by means of straight line seg-

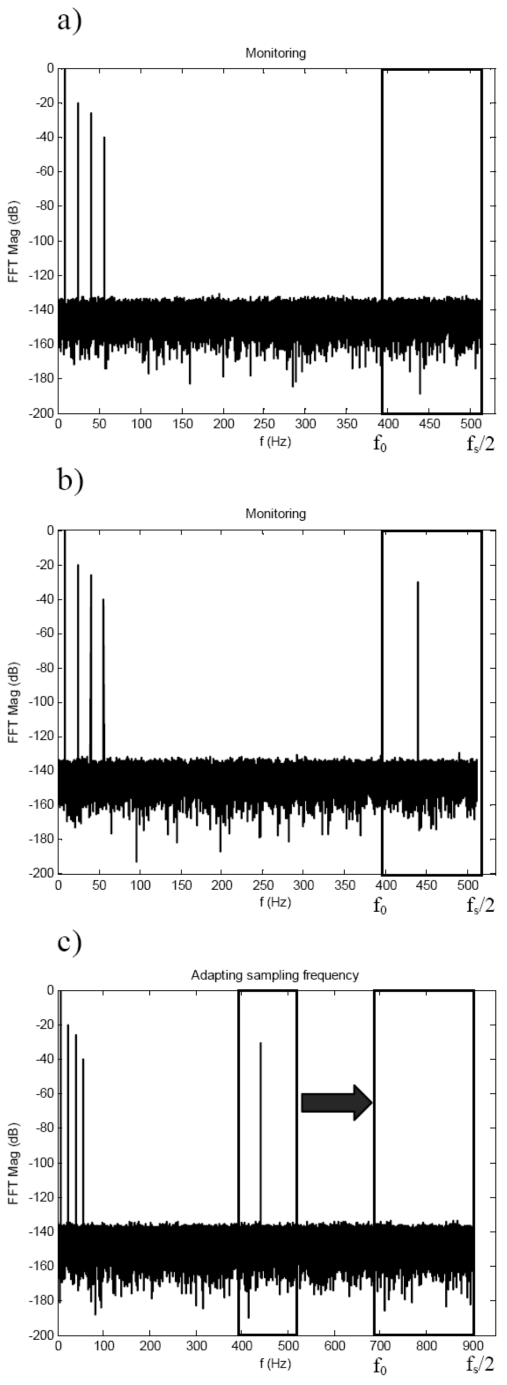


Figure 3.19: Tracking mechanism of the proposed algorithm, observed band are highlighted; a) the observed band is monitored, b) significant spectral content appears in the observed band, c) an update of the sampling frequency is triggered and observed band changes accordingly.

ments and removing redundant points along the way. Good results are obtained if the algorithm is applied to each of the linear data subsets. The end points of the segments are determined on the basis of a maximum-error criterion requiring that the results of an approximation always fall within a user-specified range. The error range can be adaptive and not necessarily symmetrical.

Procedure The procedure of the proposed algorithm acts in two different domains by exploiting the *adaptive tracking sampler* and the *noise-cancelling compressor*, respectively. The *adaptive sampling* algorithm can be summarized in the following steps (Fig. 3.20):

1. acquire a batch of samples at the current rate f_s;
2. estimate the signal power in the frequency band $[f_0,\ f_s/2]$;
3. compare the estimated power to the threshold terms, and if necessary update f_s;
4. if new data income then go to point 1, else exit.

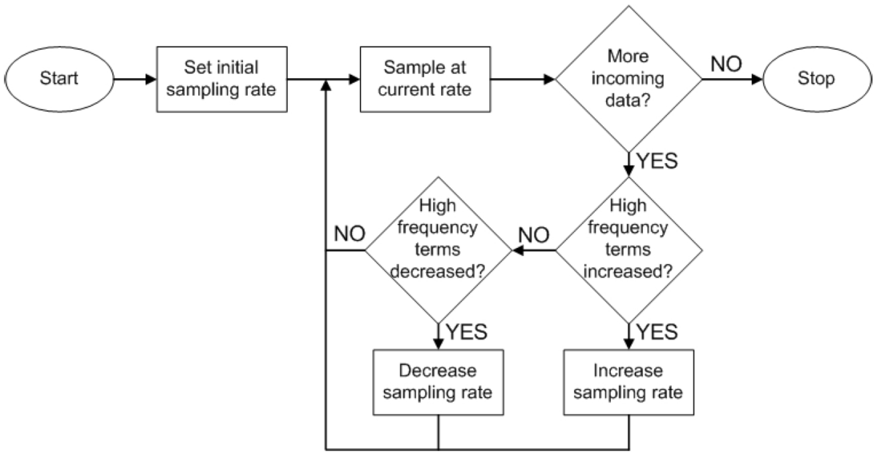


Figure 3.20: Flow chart of the tracking mechanism.

The algorithm is modular and generic: different techniques can be used for power estimation, thresholds computation, low-pass filtering, and sampling rate updating.

Subsequently, the *noise-cancelling compressor* is applied to each subset of linear data.

Once the tolerance ε has been specified, the approximation method works on a discrete signal *f(k)* through the following main steps [75]:

1. The starting point *k* is a non-redundant point by definition. Points *k* and *k+1* are used to draw two straight lines starting at *k* and crossing *f(k+1)+*ε and *f(k+1)-*ε, respectively, in order to check the redundancy of the point *k+1*.

2. The resulting lines are extended to point *k+2*. If point *k+2* lies outside them, point *k+1* becomes a non-redundant point and the process is repeated by starting from point *k+1*. If point *k+2* lies between the lines, it is found to be redundant and point *k* is retained as the starting point.

3. Then, new lines are created by using points *k* and *k+2*. These lines are compared with the previous ones and the more restrictive are kept. If point *k+3* lies outside the new lines, point *k+2* becomes a non-redundant point and the process starts over with point *k+2* as the starting point. If point *k+3* lies between them, point *k+2* also becomes redundant and new and narrower cones are created until a non-redundant point is found.

In this way, the original sampled signal is approximated by straight line segments sequentially. All the non-redundant points and all the lengths between each pair of consecutive non-redundant points are stored.

3.4.2 Harmonic resolution enhancer

The measurement method most widely used for the measurement of the integrated field and higher order harmonics of the LHC magnets is based on the rotating coils (section 1.1.1), well adapted to the measurement of integral steady-state or slow varying fields. Rotating coils are mainly used . The harmonics are obtained through Fast Fourier Analysis on the flux samples collected during a complete coil turn. The standard procedure [18], based on the application of traditional DFT algorithm to flux samples acquired through a rotating coils system, has proven to be unreliable when magnets are supplied with non-stationary current ramping up to the value corresponding to the nominal operative magnetic field.

Differently from traditional conductive magnets, superconducting magnets suffer from dynamic effects when supplied with non DC currents, like those characterizing the standard LHC cycle [76]. Non-stationary excitation currents give rise to additional dynamic field contributions, namely magnetization imperfections which worsen field quality inside the magnet.

The aim of this section is to present a digital signal processing approach to overcome standard procedure limitations and harmonic resolution enhancement.

At CERN, an approach based on interpolation techniques had already been proposed [77]. Here, an enhancement of this algorithm, allowing also the improvement of the harmonic resolution, is presented.

Standard analysis limitations

For varying magnetic fields, the coefficients obtained by means of the standard analysis [18] are significantly different from the mean value during a single coil turn and two effects have to be considered:

1. the magnetic flux due to the main component can be seen as an AM modulation of a sinusoidal carrier at frequency equal to the coil frequency rotation, where Cn(I(t)) represents the modulating signal.

2. for higher order multipoles, each one characterized by a variation law Cn(I(t)), the same effect have to considered. In addition, interference due to the modulation produced by the dipolar field variation has to be taken into account.

Furthermore, the standard analysis provides only an approximation of the harmonic coefficients over each coil turn. No possibility of tracking the instantaneous value of the coefficients is given.

In the hypothesis of a superconductive magnetic LHC dipole excited by a current ramping linearly versus time [77], intolerable differences, i.e. greater than 1 unit ($1 \cdot 10^{-4} T$), have been experienced even in the best operating condition (lowest ramp rate, 10 A/s). Yet worse results have been achieved for the other coefficients [78].

Proposed approach

The standard procedure for field quality measurement operates with success only in magnetic field stationary conditions (i.e. dipole magnets powered by a constant current), when flux samples show a periodic evolution versus angular position of the rotating coils. To overcome the limitations of the standard analysis due to the non stationary measurement conditions, a signal processing approach based on a simple interpolation of the magnetic flux samples stored over more coil turns was proposed. This approach is capable of granting as good results in the presence of non-stationary magnetic fields [79]. The key idea underlying the proposed approach is illustrated in Fig. 3.21 [77], The magnetic flux

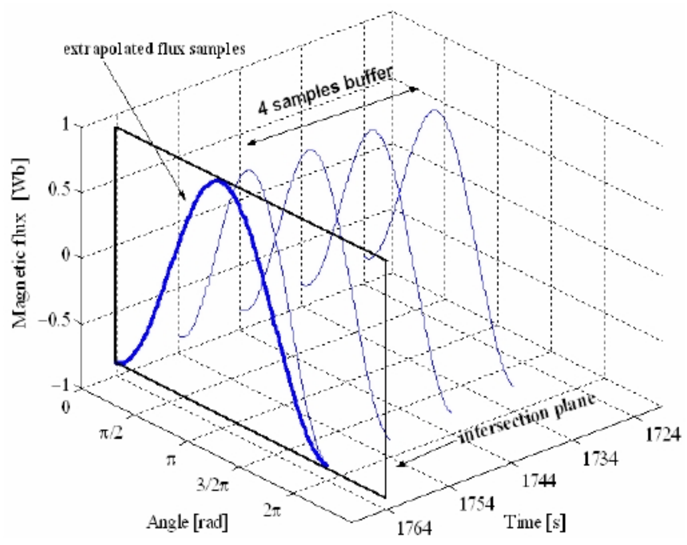


Figure 3.21: Representation of the extrapolation method.

samples are acquired in different angular positions, so they can be represented as belonging to P-points curves in a three dimensional diagram (time-angle-flux). When a single coil turn is completed, the angular position θ returns to 0, while time t is updated in order to take into account the expired coil rotation period. If a suitable set of completed turns is retained, it can be used to fit the surface $\varphi(t, \theta)$ by means of a straightforward polynomial regression algorithm [77]. The right trade-off between accuracy and computational load is obtained by choosing the order of the adopted polynomial equal to 3. This choice is further discussed in section 4.2.

According to the approach proposed in [77], a set of P-extrapolated flux samples, all related to the same time instant, $t*$, and different values of angular position can be considered as generated by a constant current equal to the current at time $t*$. The standard Fourier analysis can then be applied on the set of extrapolated flux samples characterized by the same time instant. The problems deriving from the amplitude modulation of the flux signal is therefore solved.
In the new implementation proposed in the following, the extrapolation was eliminated. The problem does not pose any causality constraint, in other words the harmonic do not have to be extrapolated at a time t^* greater then the times at which the flux samples were acquired. Consequently, t^* can be chosen in any possible way inside the observation window. For the sake of comparison with the results provided by the standard analysis procedure, t^* was chosen as the middle point of the temporal window. In fact the standard procedure can be seen as a zero order interpolation, providing the middle value on a single turn referred to the middle time on the same turn.
From an operating point of view, the algorithm consists of the following steps:

1. acquisition of flux samples related in the latest four coils turns, and their collection in a buffer;

2. computation of sets of interpolated flux samples, accounting for different angular positions uniformly distributed within $[0, 2\pi]$, for any time instant.

3. application of the standard procedure to the attained sets in order to gain the evolution versus time of harmonic coefficients;

4. after the completion of the new coils turn, the first flux sample in the acquired buffer is substituted by the last acquired sample, according to a first-in-first-out queue management;

5. steps 2, 3 and 4 are repeated until the whole flux samples are covered.

The above-described procedure solves the problems due to the amplitude modulation of the flux signal. Anyway, an intrinsic limitation of the rotating coils measurements remains, namely the impossibility of tracking the instantaneous value of the harmonic coefficients, since the analysis is performed over each coil turn. Before, the limitations of the existing hardware, namely the old rotating unit [80], made it difficult to overcome this problem. A measurement had to be performed by interrupting the acquisition between subsequent turns. With the new fast rotating unit, capable of continuous rotation, a new approach can be followed. In particular the algorithm can be run at each new flux sample to produce an undate of the harmonics estimation at the same rate at which the flux is acquired. This approach requires two main improvements in the algorithm:

- The proposed approach was implemented in order to scan the acquisition buffer point by point or in general m by m, with $m \in N^+$. So it is possible overcome the limitation of a turn by turn analysis, and attain maximum flexibility in the choice of the frequency at which the interpolation is run. This possibility allows in any situation the achievement of the best trade-off between computational burden and harmonics resolution.

- A phase correction was introduced to compensate the rotation of the reference frame during the analysis of subsequent points in a coil turn. In fact, the results of the analysis always provide the harmonics as real and imaginary part of the complex coefficient (section 1.1.1) in the reference frame corresponding to the initial position of the coil [18]. The algorithm can be run every m points, and not necessarily every turn as before, thus if m is not a multiple of the number of points per turn the reference frame rotates during the analysis. This rotation has to be recuperated by means of the previously mentioned phase correction.

The improved algorithm is more general, and gives the previous procedure when the parameter m is set equal to number of points per turn.
Another important remark about the parameter m concerns the link between this algorithm and the problem of data reduction previously explained. The interpolation procedure could in general also be seen as a method of data reduction, by expressing a sequence of acquired samples in terms of the computed coefficients of the interpolating polynomial.

Anyway, if an interpolation of order n is employed, the algorithm for harmonic resolution enhancement would produce $n + 1$ coefficient for each new point. This would cause an increase in the amount of data to be stored. At the same time, also the computational burden increases. A possible approach allowing the achievement of a better tradeoff between amount of data/computational burden and resolution improvement can indeed be allowed by the introduction of the parameter m. It is to be considered though that a big value of m can cause significant fit errors.
Moreover, there is a similarity between the algorithms proposed for harmonics resolution enhancement and data reduction: both operate on subset of the original data by exploiting the periodicity of the signal (see section 4.1). A reason to prefer the approach in section 3.4.1 for the data reduction is that with this technique, when the points lie outside the allowed tolerance (e.g. on an exponential profile), there is no reduction and the original data set is kept. Here, instead, the original signal is represented by means of a set of coefficients even when the order of interpolation does not allow to reconstruct perfectly the field profile. The loss is not considered and therefore is not under control.

3.5 Discussion

A batch algorithm for data reduction through adaptive sampling and significant point extraction has been proposed. It is fast, reliable, cost effective and can be implemented in real-time in order to extract significant points from a sampled signal, with an information loss within an error range specified by the user through a threshold mechanism. The algorithm is modular and generic: different techniques can be used for power estimation, thresholds computation, low-pass filtering, and sampling rate updating.
In the second part, a signal processing algorithm for improving measurements of non-stationary magnetic field through a rotating coils system was presented. An interpolation-based algorithm had already been developed at CERN. The proposed approach further enhances the performance of the previous one by allowing a remarkable improvement in the resolution with which field harmonics evolution versus time can be reconstructed.

Chapter 4

Numerical analysis

The algorithms for data reduction (3.4.1) and harmonic resolution enhancement (3.4.2) were tested in simulation to verify the fulfillment of the goals for which they were conceived, in conditions typical of a rotating coil-based measurement on superconducting dipole magnets.

This chapter presents the results of the simulation characterization, highlighting that, in the considered domain, the proposed approaches are suitable for data reduction and harmonic resolution enhancement, respectively.

4.1 Algorithm for data compression

The algorithm for data reduction (section 3.4.1) was tested in simulation, by evaluating the effect of the frequency adaptation on dynamic metrological performance, without other perturbation sources.

In the following, numerical results of (i) the *static tests*, and (ii) the *dynamic tests* of the algorithm are illustrated.

4.1.1 Static tests

In static tests, the algorithm is fed with a constant frequency input signal, the sampling frequency is set to the optimal value, and then the reduced data acquired in steady state are used to estimate the indices of the reduced signal and to compare them with those of the input signal.

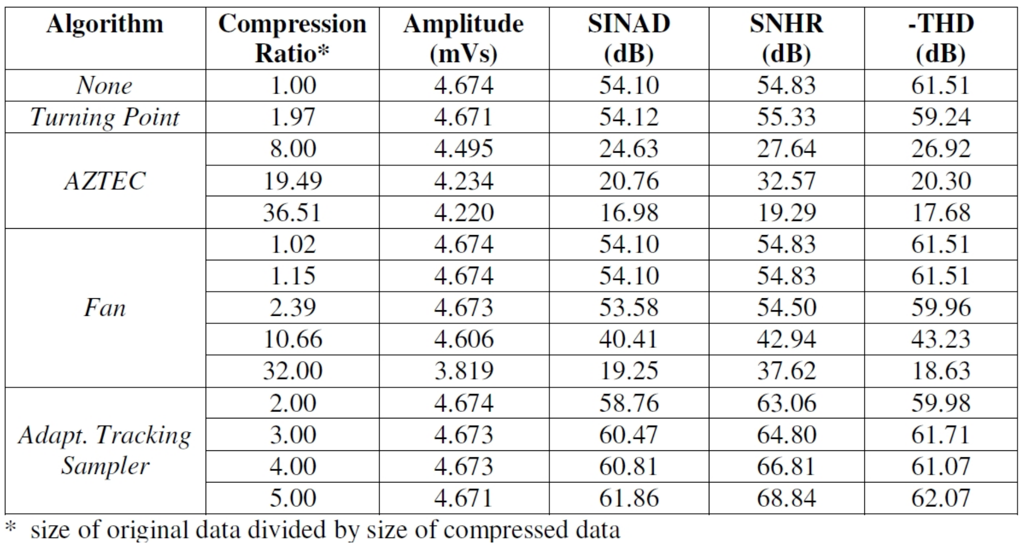
The proposed algorithm was compared to the following state-of-the-art solutions [59, 60, 61, 62, 63, 64, 65, 66, 67]:

1. The TP, a simple and fast algorithm producing a fixed reduction ratio of 2:1. It processes three points at a time, stores the first one and retains one of the next two samples depending on which one preserves the turning point (i.e., the slope change) of the original signal.

2. The AZTEC algorithm decomposes raw sample points into plateaus and slopes, thus representing the original signal through a piecewise-linear approximation formed by a sequence of line segments.

3. The pure Fan algorithm, as said before, also uses a piecewice-linear approximation of the original signal, but unlike AZTEC draws lines between pairs of starting and ending points so that all intermediate samples are within some specified error tolerance.

Simulation tests were carried out on an ideal input signal defined on the basis of the actual signals acquired on the field in a typical setting of a rotating coil measurement of a superconducting dipole magnet [76]. After the reduction, some performance indices were estimated and subsequently compared to those computed on the original signal.
The Compression Ratio (CR), defined as ratio between the size of original data and the size of compressed data, assesses the data reduction, while SIgnal-to-Noise And Distortion ratio (SINAD), Signal-to-Non Harmonic Ratio (SNHR), and Total Harmonic Distortion (THD) [81] the loss. Finally, the main component amplitude is evaluated to express the closeness of the reduced signal to the original one.
Test results are summarized in Tab. 4.1. for different settings of the algorithms' parameters.
Results show that the *adaptive tracking sampler* provides the best tradeoff between CR and error on the reconstructed signal. Lossy data compression algorithm, when applied to a noisy signal with the allowed loss set equal to the noise strength, produces a filtered signal with reduced noise content [74]. In addition to that, since the *adaptive tracking*



Algorithm	Compression Ratio*	Amplitude (mVs)	SINAD (dB)	SNHR (dB)	-THD (dB)
None	1.00	4.674	54.10	54.83	61.51
Turning Point	1.97	4.671	54.12	55.33	59.24
AZTEC	8.00	4.495	24.63	27.64	26.92
	19.49	4.234	20.76	32.57	20.30
	36.51	4.220	16.98	19.29	17.68
Fan	1.02	4.674	54.10	54.83	61.51
	1.15	4.674	54.10	54.83	61.51
	2.39	4.673	53.58	54.50	59.96
	10.66	4.606	40.41	42.94	43.23
	32.00	3.819	19.25	37.62	18.63
Adapt. Tracking Sampler	2.00	4.674	58.76	63.06	59.98
	3.00	4.673	60.47	64.80	61.71
	4.00	4.673	60.81	66.81	61.07
	5.00	4.671	61.86	68.84	62.07

* size of original data divided by size of compressed data

Table 4.1: Comparison of algorithms' static performance on simulated flux increment signal.

sampler operates a decimation with an Over Sampling Ratio (OSR) equal to the decimation ratio, the SNR is improved while reducing the data size.
The additional comparison of the *adaptive tracking sampler* with the pure Fan algorithm applied to sine wave signals typical of a rotating coils-based measurement system [2], [76] highlighted satisfying results. Nevertheless, the Fan algorithm performs better than TP and AZTEC, in terms of compression capabilities and fidelity to the original signal, respectively.
Furthermore, the choice of the Fan algorithm inside the *noise-cancelling compressor* was checked. By exploiting the signal periodicity, the reduced sine wave can be decomposed in a set of quasi-linear time series, formed by points corresponding to the same angular position (Fig. 4.1). The results (Tab. 4.2) highlight the remarkable tradeoff between compression ratio and approximation error of the Fan algorithm when applied to a data set representing a linear signal. Fig. 4.2 shows the corresponding rate-distortion plot [74], allowing the noise strength to be determined by means of a heuristic method. This is used for tuning the tolerance ε inside the *noise-cancelling compressor*. In the conditions of Fig. 4.2, where a white noise of power σ^2 is applied, the method suggests the setting of ε to a value of 3σ.

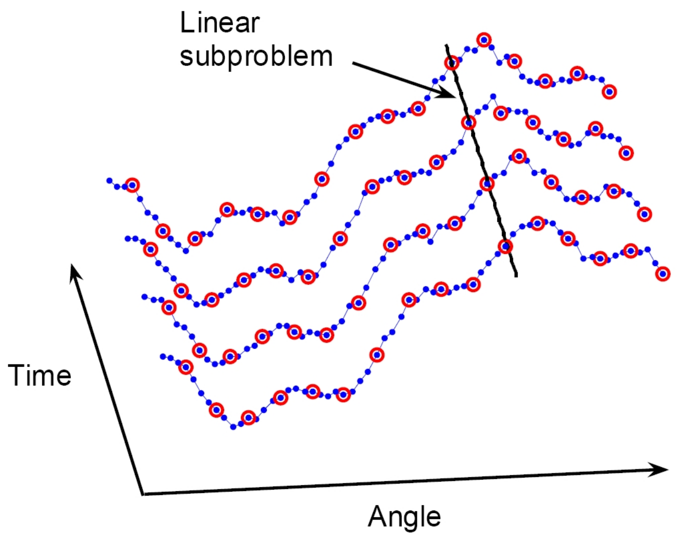


Figure 4.1: Combined approach to data reduction: original signal (dots) and signal reduced by means of *adaptive tracking sampler* (circles) as function of angular position and time.

Algorithm		Compression Ratio*	Amplitude (mVs)	SINAD (dB)	SNHR (dB)	-THD (dB)
none	σ_{noise} (Vs)					
	$8.47{*}10^{-6}$	1.00	4.674	54.10	54.83	61.51
Fan	ε (Vs)					
	$1.0{*}10^{-6}$	1.28	4.674	54.11	54.84	61.51
	$2.0{*}10^{-6}$	1.55	4.674	54.11	54.84	61.52
	$4.0{*}10^{-6}$	2.15	4.674	54.13	54.87	61.50
	$6.0{*}10^{-6}$	2.90	4.674	54.18	54.93	61.37
	$8.5{*}10^{-6}$	4.25	4.674	54.29	55.06	61.28
	$1.0{*}10^{-5}$	5.48	4.674	54.39	55.16	61.26
	$1.2{*}10^{-5}$	7.71	4.674	54.50	55.32	60.93
	$1.7{*}10^{-5}$	18.5	4.674	54.97	55.85	60.77
	$2.0{*}10^{-5}$	34.5	4.674	54.98	56.04	60.04
	$2.6{*}10^{-5}$	101.2	4.674	55.51	56.83	59.52
	$3.4{*}10^{-5}$	368.1	4.674	55.96	56.86	61.95
	$8.0{*}10^{-5}$	512.0	4.674	55.41	56.51	61.24
	$1.0{*}10^{-4}$	512.0	4.674	55.41	56.51	61.24

* size of original data divided by size of compressed data

Table 4.2: Performance of the Fan algorithm, employed by the *noise-canceller compressor*, for the reduction of nearly linear signals.

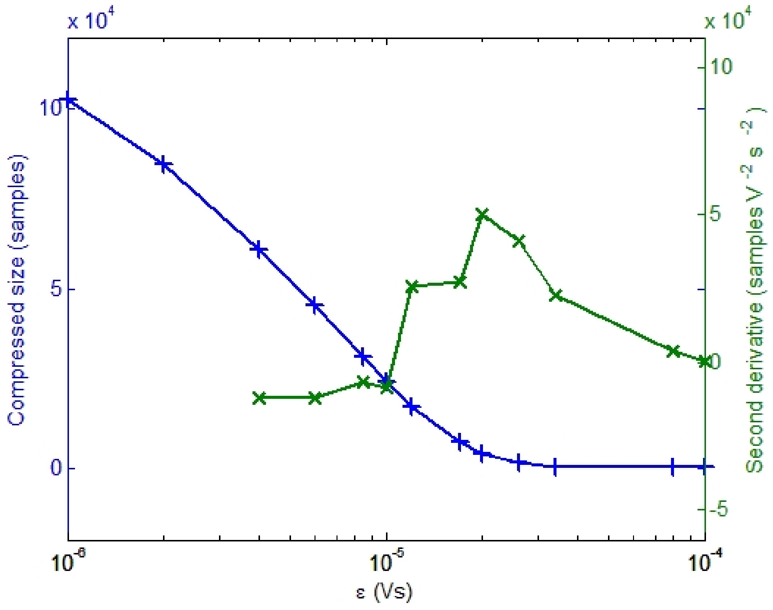


Figure 4.2: Rate-distortion plot and heuristic determination of the noise strength ($\varepsilon = 2 * 10^{-5}$ Vs), corresponding to the maximum of the second derivative.

4.1.2 Dynamic tests

Inside the *adaptive tracking sampler*, the proposed algorithm includes an *fft*-based tracking mechanism, working properly only if its adaptation time is negligible in comparison to the typical time variation of the signal. The algorithm tracking capabilities were verified for different settings of its parameters by a simulation with variable frequency input signal. Extremely fast changes were imposed by using step functions to define the time variation law of the input frequency. The algorithm had to follow this variation by updating suitably the sampling frequency. The algorithm dynamic performance is depicted in Fig. 4.3 for different values of α, the maximum change in the sampling frequency allowed in one step, expressed as a fraction of the current sampling frequency. The figure shows how adaptation times suitable for the typical magnetic measurements carried out at CERN can be achieved.

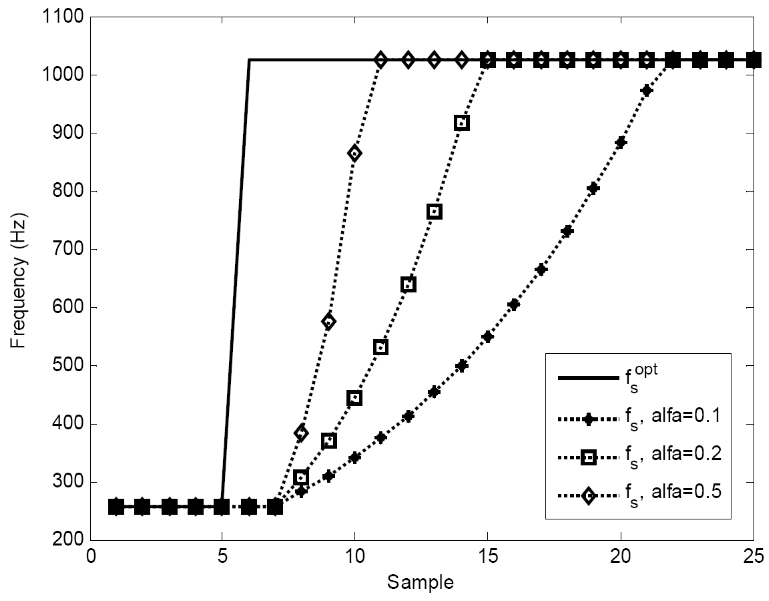


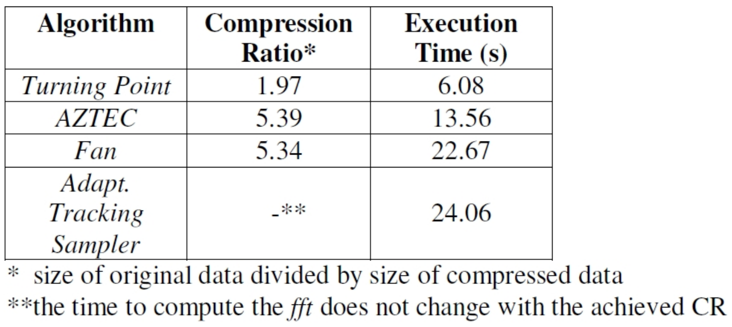
Figure 4.3: Sampling rate step responses for different values of α (the maximum change in the sampling frequency allowed in one step, expressed as a fraction of the current sampling frequency).

4.1.3 Algorithm performance

With the aim of checking the computational burden of the proposed solution, the algorithm performance were evaluated in simulation and compared to those of other state-of-the-art algorithms, by measuring the time for processing the same data set. Execution times were computed in MATLAB, with respect to a very easy and fast solution such as the TP. Tab. 4.3 shows the result of the comparison, proving that the computation required suits the capabilities of a system typically used for the application to magnetic measurements.

4.2 Algorithm for harmonic resolution enhancement

A numerical analysis was performed also on the algorithm for harmonic resolution enhancement (section 3.4.2. A number of tests on simulated flux samples were carried out to assess the performance of the proposed approach, by comparing it to that granted by the standard procedure. To this aim, suitable models for harmonic coefficient were adopted in order to generate simulated harmonic coefficients C_n (section 1.1.1), from which the



Algorithm	Compression Ratio*	Execution Time (s)
Turning Point	1.97	6.08
AZTEC	5.39	13.56
Fan	5.34	22.67
Adapt. Tracking Sampler	-**	24.06

* size of original data divided by size of compressed data
**the time to compute the *fft* does not change with the achieved CR

Table 4.3: Comparison of algorithms' computational burden expressed as execution time required to process the same amount of incoming data (a sine wave of 131072 points): computation performed in MATLAB on a Pentium IV-2.8 GHz processor.

flux samples were computed. Each effect has been quantified using data obtained from measurement series on the LHC magnet. The proposed approach has been applied to estimate the harmonic coefficients from the flux samples and compare their values with the nominal ones, during an LHC PELP (Parabolic Exponential Linear Parabolic) cycle [76]. The coefficient are computed for a fixed coil rotation speed ($8rps$), and fixed number of points per turn (256), by varying: 1)the field profile (linear, parabolic, exponential), 2) the field variation speed, 3) the rate at which the algorithm is executed, expressed through the parameter m (number of new flux samples to be acquired before each run of the algorithm). The interpolation order is set equal to 3. Former experiences at CERN have indicated this value as the best tradeoff between computational burden and accuracy in harmonics reconstruction. In the following, for the sake of conciseness, only the results obtained for the normal dipole harmonic B_1 are shown. The other harmonics exhibits a very similar behaviour and the related results are therefore omitted.

Linear field profile

For this profile, the algorithm was tested with different ramp rates and updating frequency of interpolation coefficients. A first test was run with m=1 (coefficients computed at the maximum rate, i.e. for every new flux sample) and with a ramp rate of 0.0628 T/s. The simulation was subsequently repeated at a higher ramp rate (0.4776 T/s), with two different values of the parameter m ($m = 1$, i.e. one update for every new flux sample and

$m = 256$, i.e. one update for every coil turn) in order to evaluate also the effect of this parameter on the harmonic coefficients estimation. Fig.s 4.4-4.7 show the reconstructed profile of the main normal harmonic (B_1) for the two different ramp rates, while Fig.s 4.8-4.13 report the error estimated with and without algorithm for the aforementioned cases.

As the ramp rate increases, as expected the absolute error without interpola-

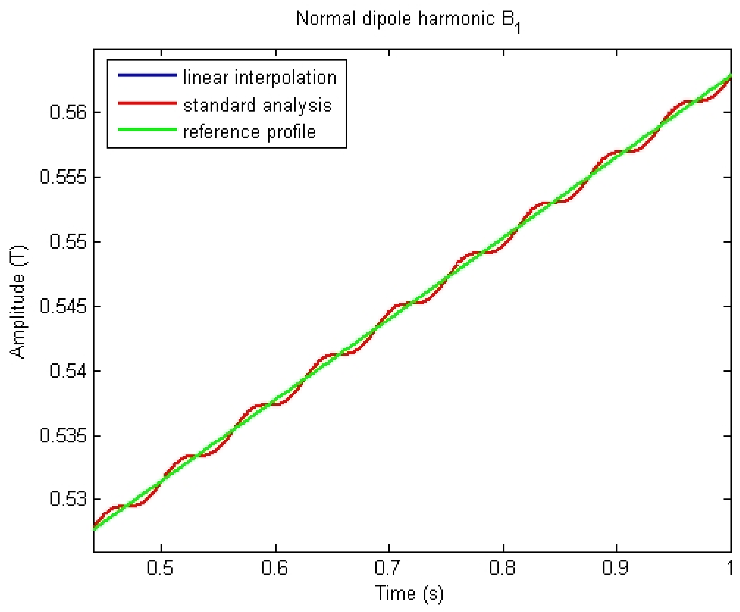


Figure 4.4: First normal harmonic (B_1) comparison: linear profile, m=1, ramp rate 0.0628 T/s.

tion increases proportionally, with the relative error remaining comparable in both cases. Conversely, when the algorithm is applied there is a perfect match between known input harmonics and their estimated value and there is no ramp rate dependence. The absolute error is always negligible, i.e. within the resolution of the 64-bit floating point environment used for the computations. As expected, it is therefore possible to conclude that with a pure linear field profile, the third order interpolation guarantees very good fidelity in the reconstruction of the field profile, no matter for the frequency at which the interpolation coefficients are updated.

Furthermore, varying m gives no remarkable change in the error amplitude, always negligible in the linear case if the algorithm is applied, but the computational complexity is

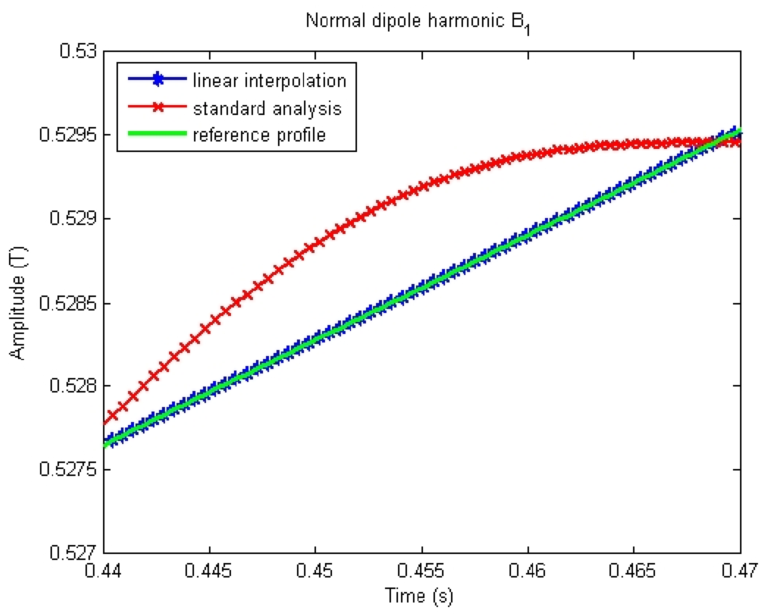


Figure 4.5: Detail of first normal harmonic (B_1) comparison: linear profile, m=1, ramp rate 0.0628 *T/s*.

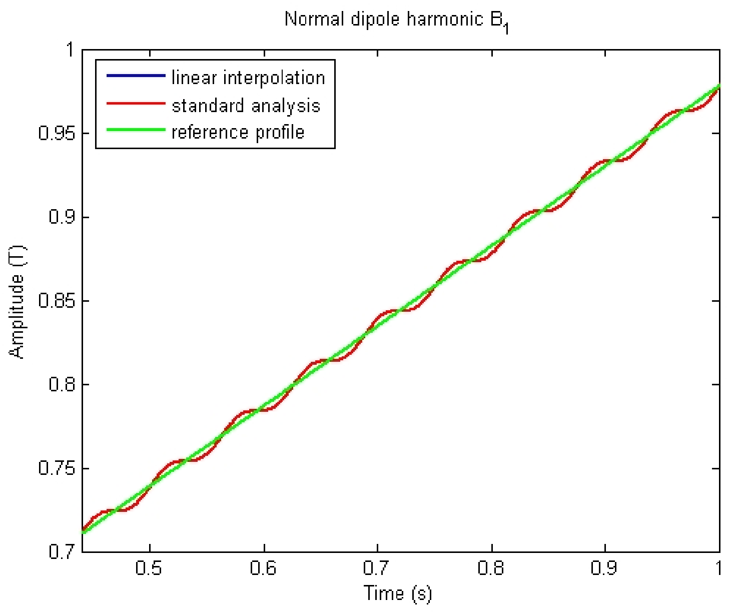


Figure 4.6: First normal harmonic (B_1) comparison: linear profile, m=1, ramp rate 0.4776 *T/s*.

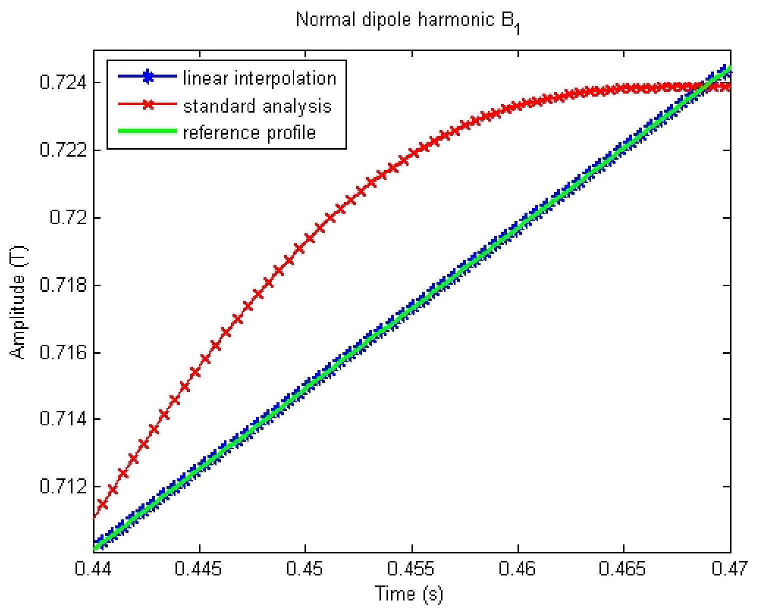


Figure 4.7: Detail of first normal harmonic (B_1) comparison: linear profile, m=1, ramp rate 0.4776 T/s.

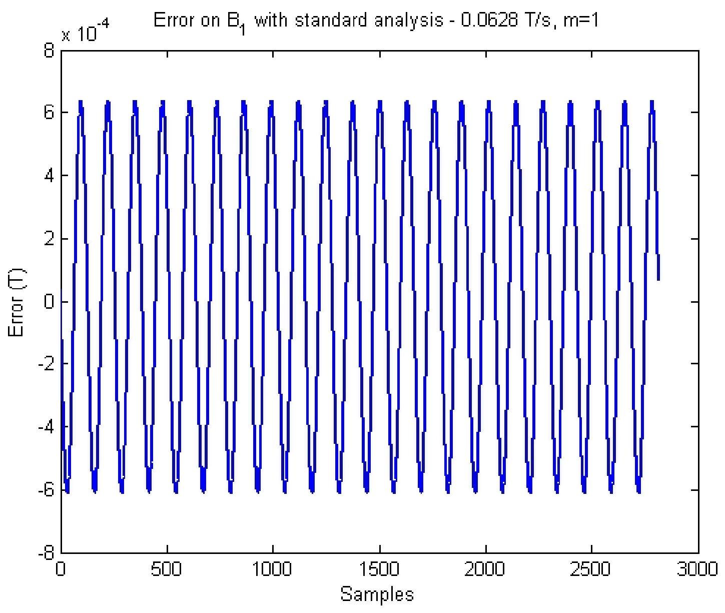


Figure 4.8: Error on B_1 with standard analysis: linear profile, m=1, ramp rate 0.0628 T/s.

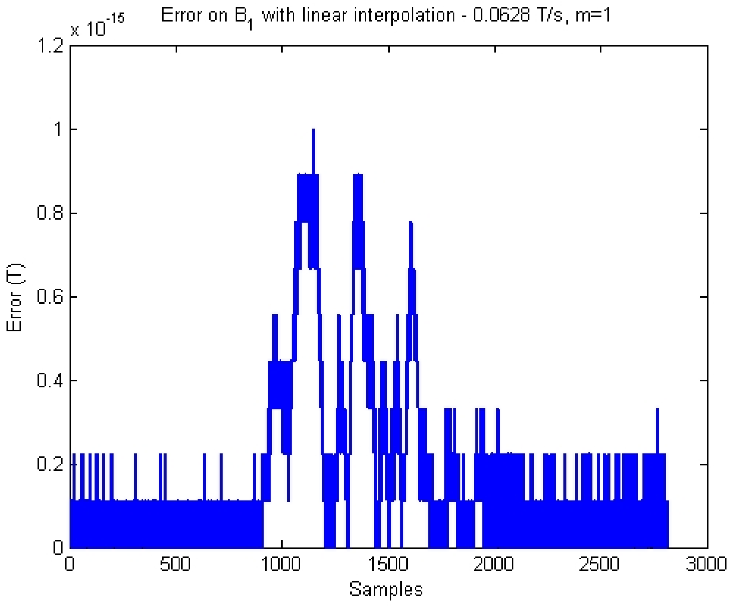


Figure 4.9: Error on B_1 with linear interpolation: linear profile, m=1, ramp rate 0.0628 T/s.

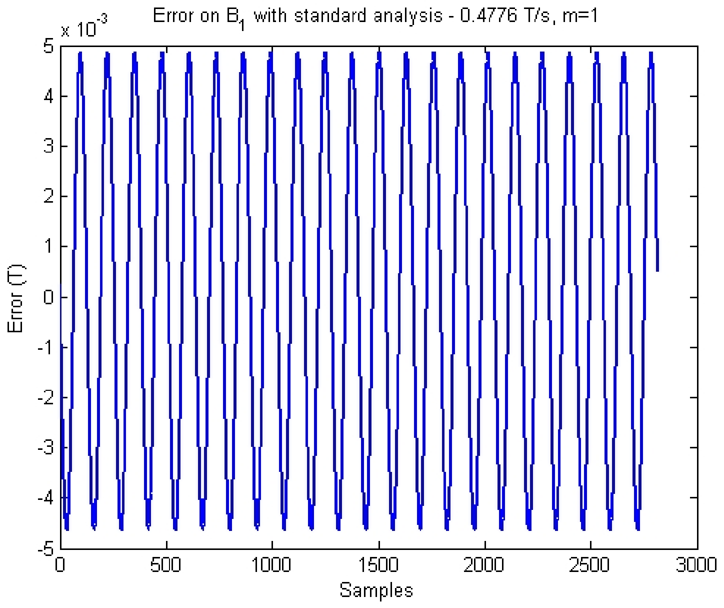


Figure 4.10: Error on B_1 with standard analysis: linear profile, m=1, ramp rate 0.4776 T/s.

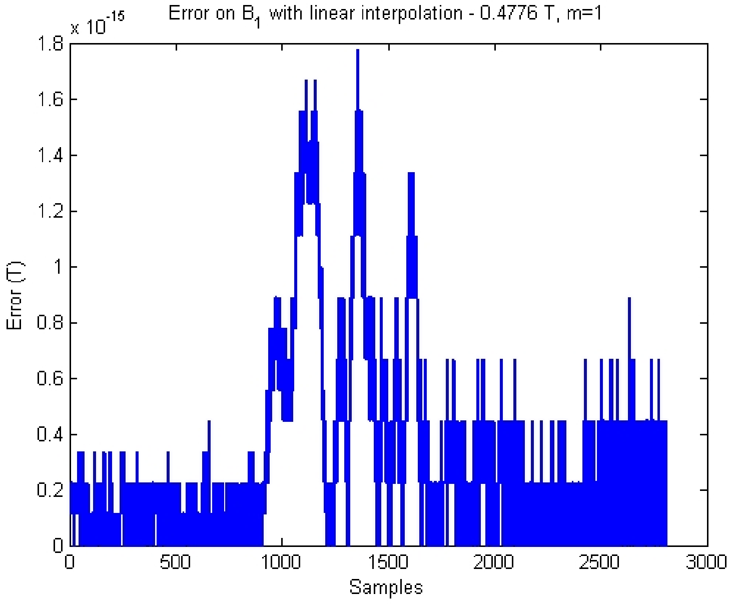


Figure 4.11: Error on B_1 with linear interpolation: linear profile, m=1, ramp rate 0.4776 T/s.

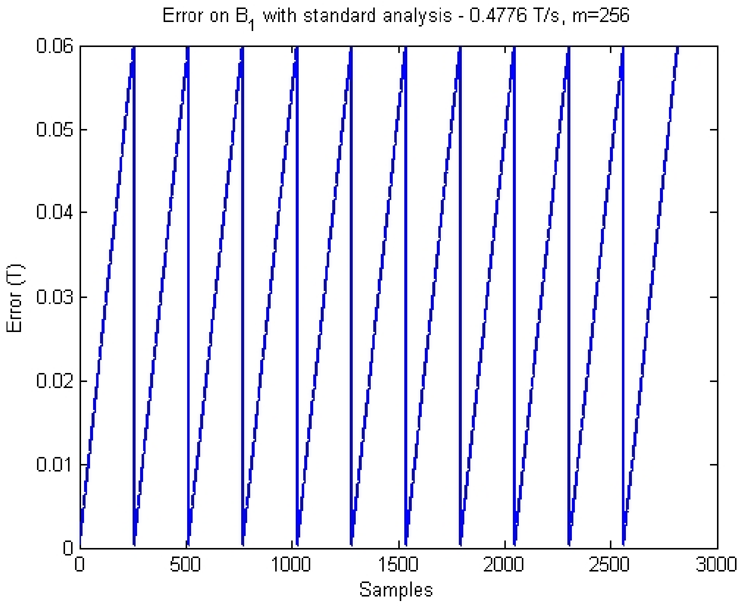


Figure 4.12: Error on B_1 with standard analysis: linear profile, m=256, ramp rate 0.4776 T/s.

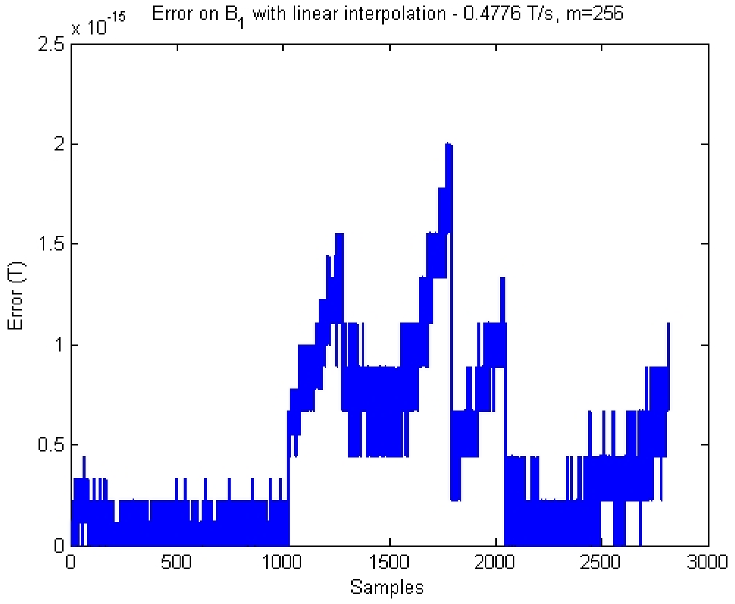


Figure 4.13: Error on B_1 with linear interpolation: linear profile, m=256, ramp rate 0.4776 T/s.

decreased for larger values of m.

With the standard analysis, the amplitude modulation of the flux signal produces an oscillation in the amplitude of the harmonic coefficient, with a period related to that of the input sine wave. In order to support this hypothesis, additional simulations were performed with different input field profile. Additional simulations were executed by keeping constant the main harmonic and imposing a linear variation of the higher order components. The spectral leakage is supposed to be basically due to the time variation of the main field harmonic, by far the most significant in module (higher order harmonics are chosen to be 4 orders of magnitude smaller than the main one, according to typical values found at CERN during superconductive magnet test campaigns). In this case the amplitude modulation involves only small terms with slow variations. In such a situation, a drastic reduction of the oscillation in the harmonics computed through the standard analysis is observed as shown in Fig.s. 4.14.

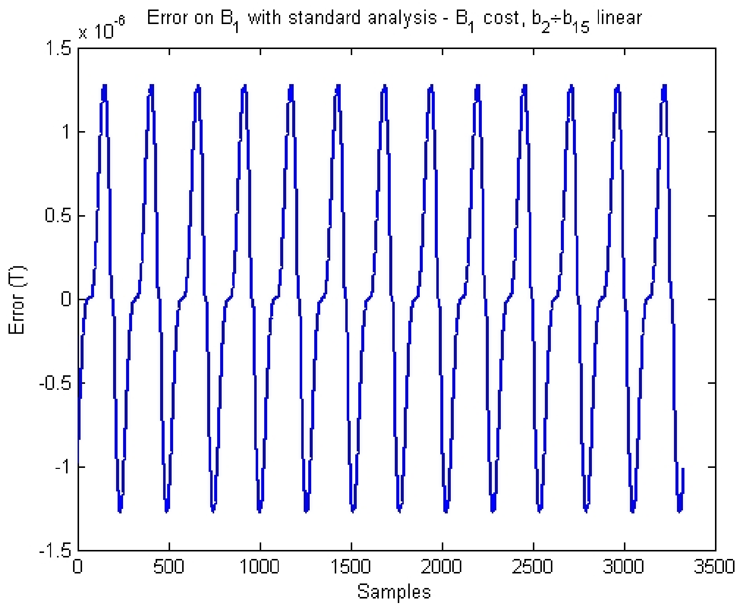


Figure 4.14: Error on B_1 with standard analysis: constant B_1, linear profile $b2 \div b15$, m=1.

Parabolic field profile

Similar tests were considered for a parabolic profile. This case is quite similar to the linear one, since in both cases the third order interpolation allows a perfect reconstruction of the field profile. The usefulness of showing the results of the harmonic computation lies in the possibility it provides of evaluating the effects of the algorithm with different interpolation orders, capable of perfectly matching or not the input field profile. In this way the effects of different interpolation orders can be highlighted. In particular, simulations with parabolic field profile were run with (i) linear interpolation and (ii) quadratic interpolation, and compared with the harmonics provided by the standard analysis, with $m = 1$. As expected, the field coefficients evaluated with the interpolation algorithm match the ideal ones, while the coefficients evaluated without algorithm show a significantly greater error, depending in absolute terms on the local field slope (i.e. the field variation during a turn), and a clearly visible oscillation.

Fig.s 4.15-4.18 depict the results, showing that with the first order interpolation, incapable of fitting perfectly the field profile, also by applying the algorithm there is an oscillation of the reconstructed field profile, whose period is related to the period of the input sine wave.

This oscillation is visible to a smaller extent with respect with the standard analysis,

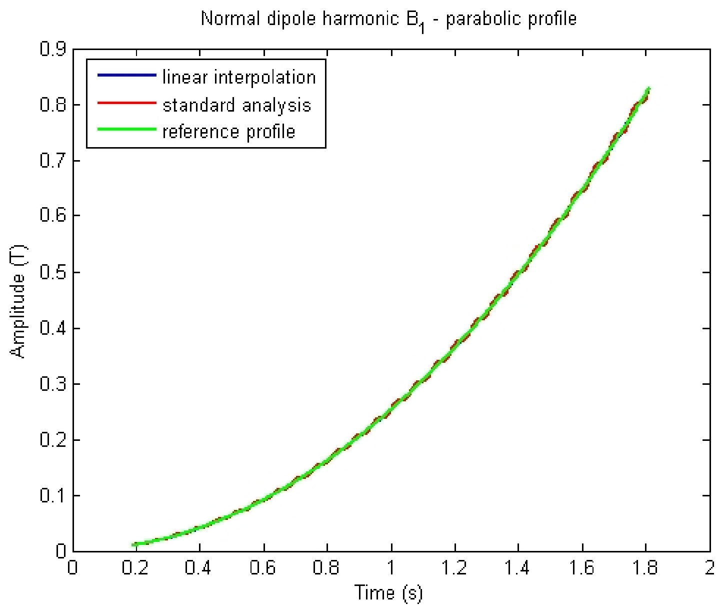


Figure 4.15: First normal harmonic (B_1) comparison: parabolic profile, linear interpolation, m=1.

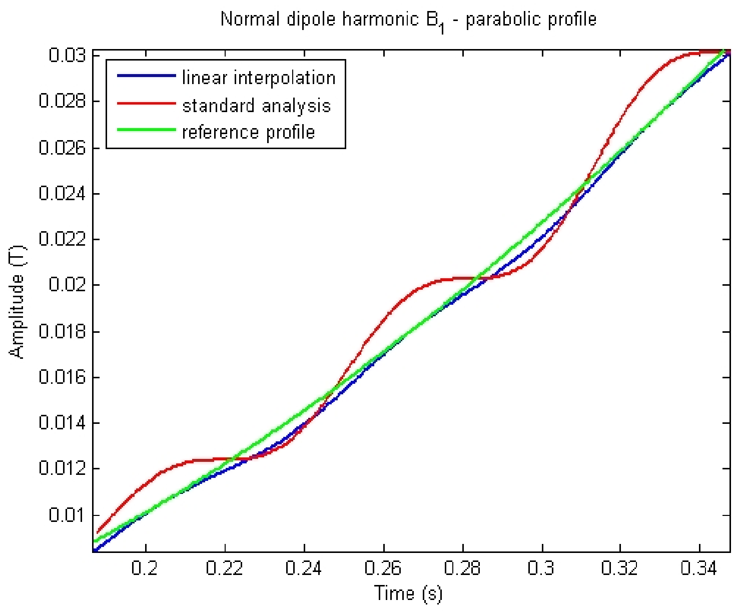


Figure 4.16: Detail of first normal harmonic (B_1) comparison: parabolic profile, linear interpolation, m=1.

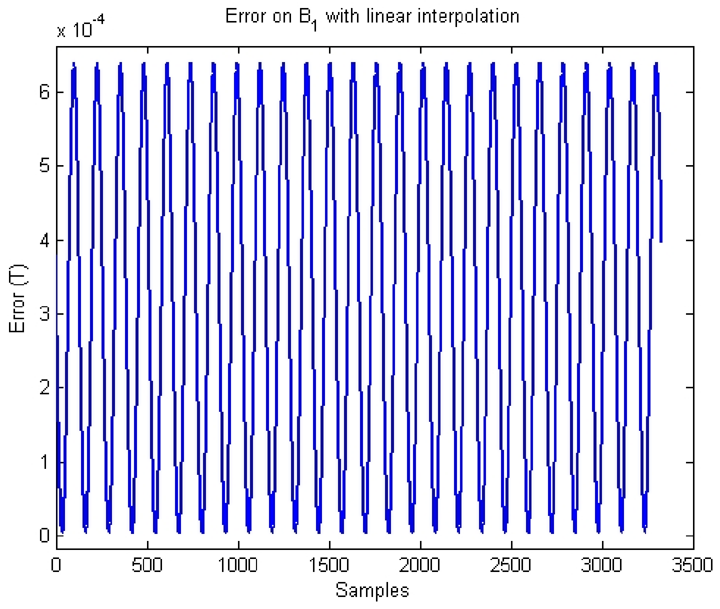


Figure 4.17: Error on B_1 with linear interpolation: parabolic profile, m=1.

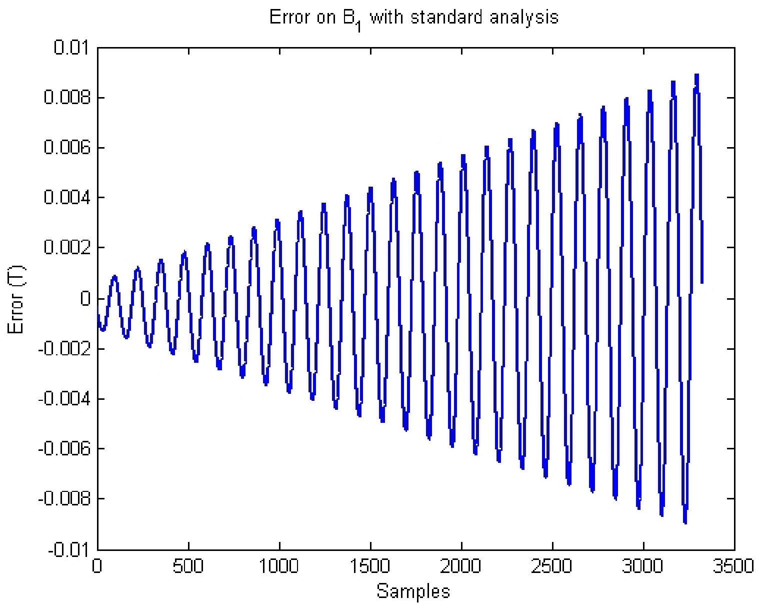


Figure 4.18: Error on B_1 with standard analysis: parabolic profile, m=1.

and even more important, unlike the results of the standard analysis it does not exhibits a dependence on the instantaneous slope of the field profile. Conversely, the second order interpolation adequacy to fit the input data results with a negligible reconstruction error is shown in Fig.s 4.19 and 4.20. Even though the absolute error is negligible, it is possible to see its dependence on the variation speed of the field, namely the faster the variation the bigger the error.

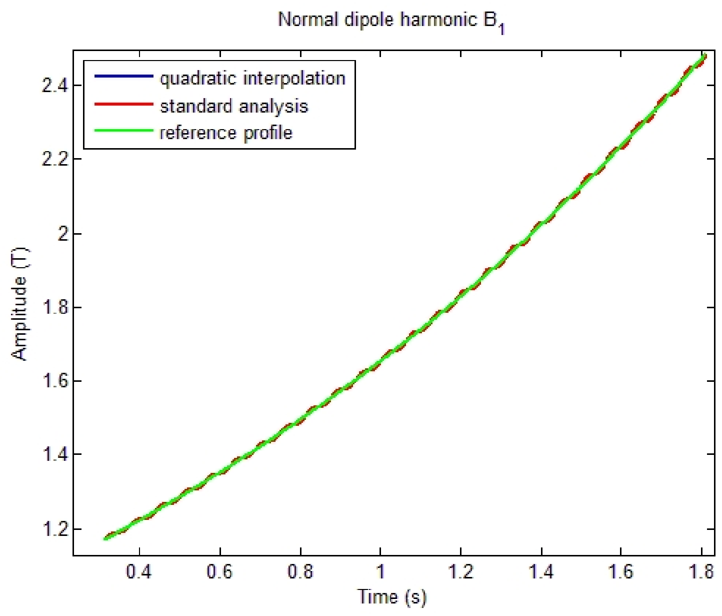


Figure 4.19: First normal harmonic (B_1) comparison: parabolic profile, quadratic interpolation, m=1.

Exponential field profile

In the following the results obtained for an exponential field profile case are shown. This profile allows some considerations to be made about the appropriate interpolation order to be chosen, since it is not polynomial and cannot be fit perfectly by any of the possible options. Fig.s 4.21-4.25 show the reconstruction of the main normal field harmonic and the estimation error obtained by applying both the standard analysis procedure and the interpolation algorithm, with $m = 1$. In particular, the latter was executed twice with different interpolation orders. Initially the second order was applied, leading to the results shown in Fig. 4.22, where the residual error not eliminated by the algorithm,

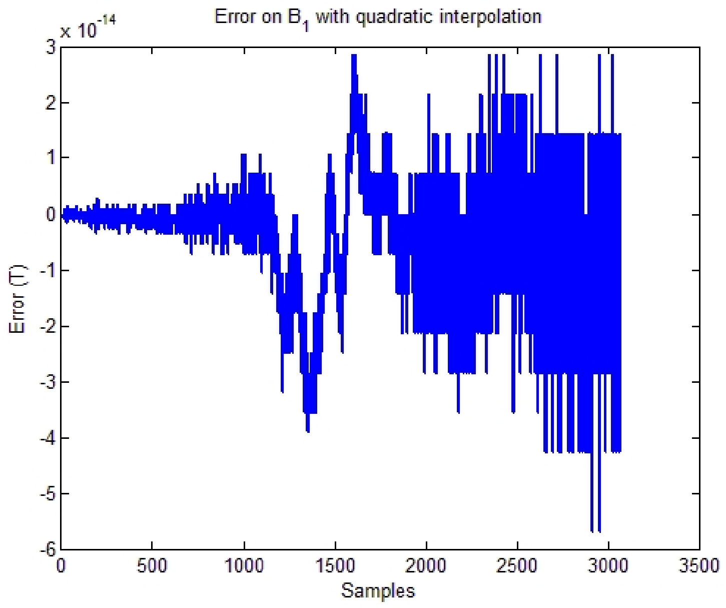


Figure 4.20: Error on B_1 with quadratic interpolation: parabolic profile, m=1.

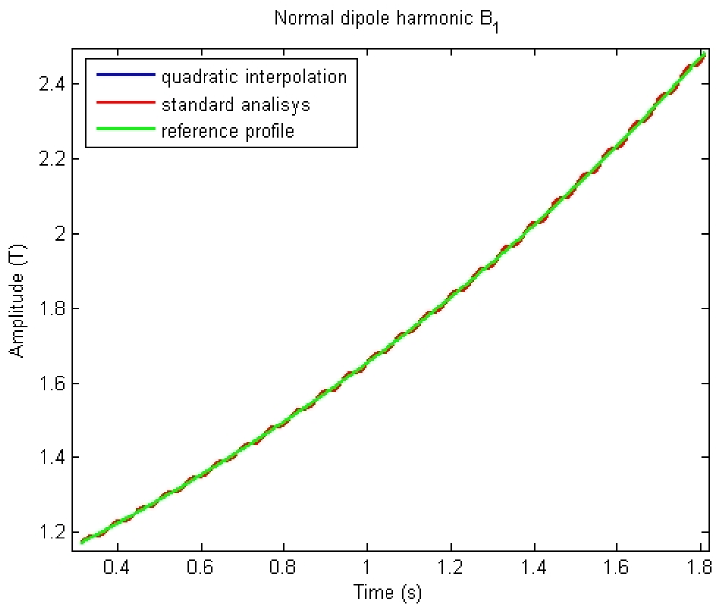


Figure 4.21: First normal harmonic (B_1) comparison: exponential profile, quadratic interpolation, m=1.

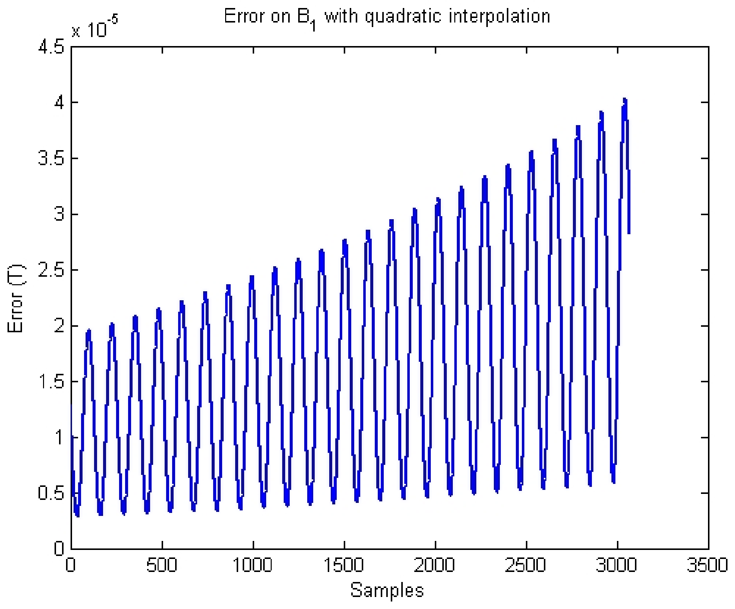


Figure 4.22: Error on B_1 with quadratic interpolation: exponential profile, m=1.

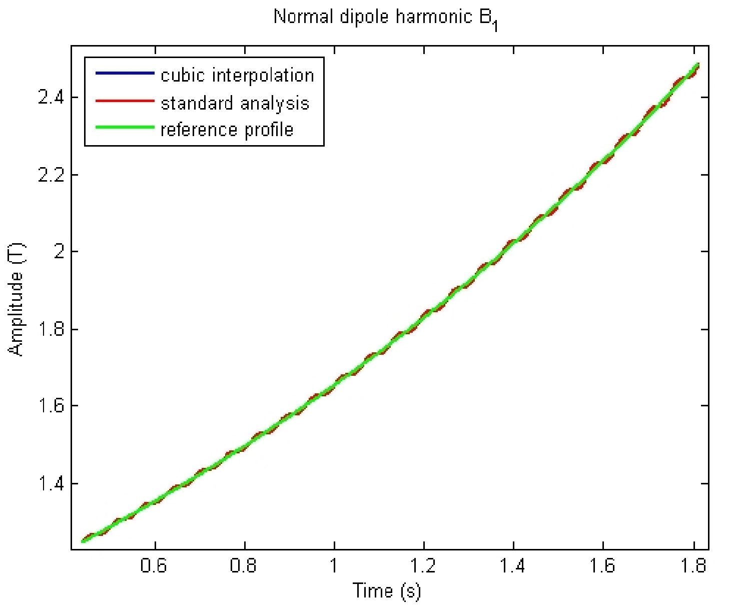


Figure 4.23: First normal harmonic (B_1) comparison: exponential profile, cubic interpolation, m=1.

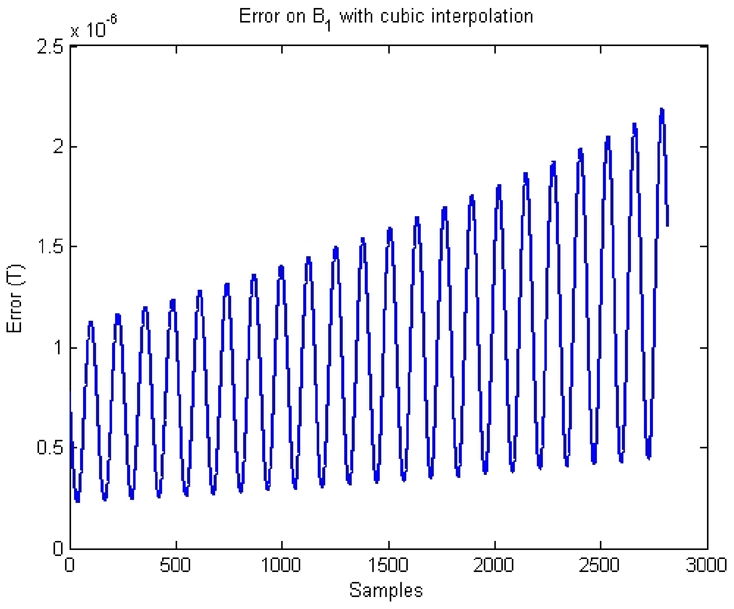


Figure 4.24: Error on B_1 with cubic interpolation: exponential profile, m=1.

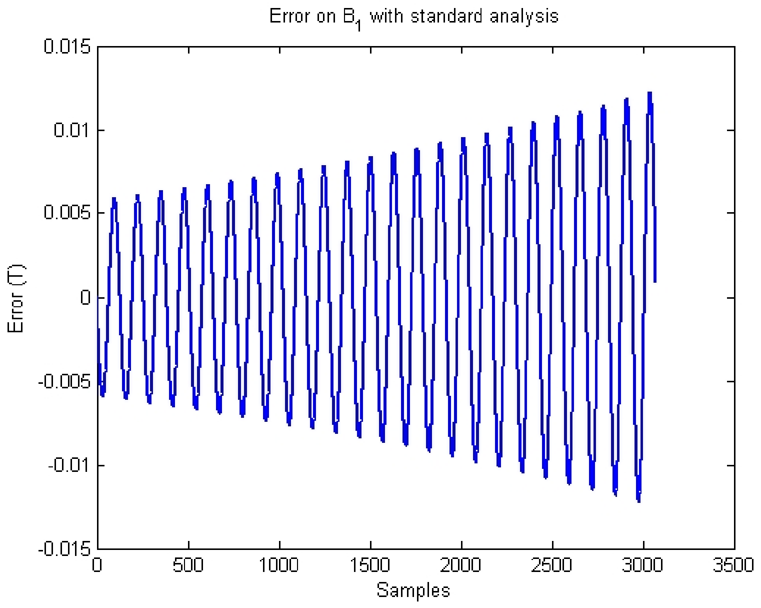


Figure 4.25: Error on B_1 with standard analysis: exponential profile, m=1.

although significantly reduced with respect to that produced by means of the standard analysis, still exceeds the requirements of the most demanding applications (error below $10^{-4} - 10^{-5}$ T). The quadratic interpolation therefore does not give satisfactory fidelity of the reconstructed field profile to the original one. The simulation was repeated employing a third order interpolation, yielding the results reported in Fig. 4.24. Also in this case, the polynomial interpolation is not capable of following the field profile with negligible error, but the results highlight that with this setting the algorithm can be usefully employed to increase the harmonic resolution of the short exponential parts of the LHC PELP current cycle within the tolerance of $10^{-4} - 10^{-5}$ T.

4.3 Discussion

Several tests have been conducted on simulated flux samples to assess the performance of the approaches proposed both for data reduction and harmonic resolution improvement. A far as the data reduction algorithm is concerned, the results highlight the better performance of the proposed combined approach, when compared to other techniques, and in particular its suitability for application where signals similar to those usually found in rotating coils based magnetic measurements, from the point of view of performance, reduction capabilities, and fidelity of the reconstructed signal to the original one.
Subsequently, the algorithm for harmonic resolution improvement was characterized. The results obtained for different field profiles and algorithm's settings are presented and compared to those granted by standard procedure. In all cases, the interpolation algorithms allows significant improvements of the harmonics resolution to be achieved. The simulations also highlighted that, during an LHC PELP cycle, a good tradeoff between computational burden and desired harmonics resolution improvement can be achieved when a cubic interpolation is used.

Chapter 5

Software quality assessment

In this chapter the quality characterization the release 3.0 of the Flexible Framework for Magnetic Measurements is presented. The object-oriented and the aspect-oriented parts of the system are evaluated separately. First, the approach proposed in the standard ISO 9126 is chosen as reference model for the quality evaluation of the object-oriented part, and experimental results are provided. Anyway, this evaluation follows an approach only recently proposed and not yet completely validated. Moreover, it does not provide clear indications about possible ways of improving the quality. Therefore a more practical analysis of the object-oriented part follows, in order to find design flaws and propose corrective actions. Finally, a modularity and performance analysis of the aspect-oriented Fault Detector is presented to prove the benefits deriving from the use of this technology.

5.1 Software quality

Quality is key issue in software development. The quality of a system is the result of the quality of its elements and their interactions. Quality in general can be defined as: (i) the degree to which a system, component, or process meets specified requirements; (ii) the degree to which a system, component, or process meets customer or user needs or expectations [82]. This definition offers the two most common interpretations of the word quality are (i) conformance to requirements ("inner quality"), and (ii) measure of user satisfaction ("outer quality"). Although it can be described from different perspectives [83, 84, 85], a specific definition has been provided for the software quality, intended as

"the capability of a *software product* to satisfy stated and implied needs when used under specified conditions'" [86]. Pursuing software quality is always worthwhile, since the cost of achieving a high quality level is widely overtaken by the cost of nonquality (having a software incapable of providing the required functionalities when needed).
The assessment of the software quality cannot be achieved without defining how to measure it in a quantitative way. For this reason, several metrics were introduced. The term *metric* is defined as a measure of the degree to which a process or product possesses a certain quality characteristic [87]. Tabs. 5.1 and 5.2 present all the metrics used in this chapter, providing also a brief description for each of them.

One of the major points when using metrics is that reference points (*thresholds*) are required in order to link them to a useful semantic, thus allowing an objective assessment. The main aim is not to look for perfect thresholds, but for values that can be useful in practice in order to detect possible software artifacts. Two major sources of threshold values can be identified [88]:

- *Statistical information*, leading to thresholds based on statistical measurements. One or more reference points are used to split the space of numbers into meaningful intervals. Applying simple statistical techniques to the data collected for each metric, the average AVG can be used to estimate the typical values, and the standard deviation STD to define higher/lower margins as $AVG \pm STD$.

- *Generally accepted semantics*, leading to thresholds based on information considered common widely accepted knowledge. This knowledge could be in its turn based on former statistical observations, but their values have become part of our culture and can be inferred without statistically measuring them.

Metrics need to be evaluated within the frame of a quality model to avoid their misuse. A *model* is an abstraction of reality, allowing to discard useless details and view an entity or a concept form a particular perspective [89], understanding the interactions among the parts forming the whole system of interest. A model can be used to predict or assess the quality, the latter being the aim of this chapter.

Metric	**Description**
Cyclomatic Complexity (CYCLO)	Logic complexity of a module, as number of linearly-independent paths.
Essential Complexity (ESS)	Cyclomatic complexity after replacing all well structured control structures with a single statement.
Class Depending Child (CDC)	Class depending at leat on one of its children.
Class Depth (DEPTH)	Depth of a class within the inheritance hierarchy.
Multiple Inheritance (FAN IN)	Number of immediate base classes.
Response for Class (RFC)	Number of methods, including inherited ones.
Coupling between Objects (CBO)	Number of other classes coupled to. Coupling means using a type, data, or member from that class. Any number of couplings to a given class counts as 1 towards the metric total.
Lack of Cohesion of Methods (LOCM/LCOM)	Cohesion between class data and methods.
Weighted Methods for Class (WMC)	Sum of cyclomatic complexity of all nested functions or methods.
Access to Forein Data (ATFD)	Number of attributes from unrelated classes accessed directly or through accessor methods.
Changing Classes (CC)	Number of classes in which the methods that call the measured method are defined.
Coupling Intensity (CINT)	Number of distinct operations called by the measured operation.
Changing Methods (CM)	Number of distinct methods that call the measured method.
Lines of Code (LOC)	The number of lines that contain source code.
Tight Class Cohesion (TCC)	Relative nuber of method pairs of a class that access in common at least one attribute of the measured class.
Weight of a Class (WOC)	Number of public methods divided by the total number of public members.
Number of Accessor Methods (NOAM)	Number of accessor (getter and setter) methods of a class.
Number of Public Attributes (NOPA)	Number of public attributes of a class.
Number of Accessed Variables (NOAV)	Number of variable accessed directly by the measured operation.
Locality of Attribute Accesses (LAA)	Number of attribute from the method definition class, divided by the total number of variable accessed.
Foreign Data Providers (FDP)	Number of classes in which the attributes accessed are defined.
Maximum Nesting Level (MAXNESTING)	Maximum nesting level of control structures within an operation.

Table 5.1: Metrics catalogue.

Metric	Description
Coupling Dispersion (CDISP)	Number of classes in which the operation called from the measured operation are defined, divided by CINT.
Number of Packages (NOP)	Number of high level packages (packages in Java, namespaces in C++).
Number of Classes (NOC)	Number of classes defined in the system, not counting library classes.
Number of Operations (NOM)	Number of user-defined operations (methods and global functions).
Number of Operation Calls (CALLS)	Number of distinct operation calls (invocations) made by all the user-defined operations.
Number of Called Classes (FANOUT)	Sum of the classes from which operations call methods, for all the user-defined operations.
Average Number of Derived Classes (ANDC)	Average number of subclasses of a class, tells how extensively abstractions are refined through inheritance. Interfaces are not counted.
Average Hierarchy Height (AHH)	Average path length from a root class to its deepest subclasses, tells how deep the class hierarchy is. Interfaces are not counted.
Degree of Focus (DOF)	Measure of the level of dedication of a component to every concern in the system.
Degree of Scattering (DOS)	Measure of the level of scattering of a concern within all the modules in the system.

Table 5.2: Metrics catalogue. ANDC, AHH, DOF, and DOS are proportions.

A considerable amount of work has been devoted to the formulation of so-called quality models. One of the first was proposed by Gilb [90], according to whom any quality characteristic can be measured directly. The quality concept is broken into component parts until each can be stated in terms of directly measurable attributes. Other models were proposed by Boehm [87] and McCall [91]. These hierarchical models are based on the assumption that there are a number of important high level quality factors that are determined by lower level criteria supposed much easier to measure than the corresponding factors. Actual measures, metrics, are proposed for the criteria. The model describes all the relationships between factors and criteria, so that the former can be quantified in terms of measures of their dependent criteria. This conception of modeling quality was more recently at the basis of international efforts that led to the development of a standard for software quality measurement, defining a software quality model (ISO 9126 [19, 92, 93, 94]), the software measurement process (ISO 15939 [95]), and the software evaluation process (ISO 14598 [96]). The standard ISO9126-1 [19] recommends six quality characteristics, further refined in subcharacteristics, as basic set for quality evaluation, and the standards ISO9126-2 -3 and -4 [92, 93, 94] define metrics for measurement of characteristics and subcharacteristics. Anyway, the metric list is not finalized and no clear indications are provided about their mapping to the quality characteristics. Given a particular problem, techniques like the Goal-Question-Metric [97] can help identify which measures are to be taken into account to monitor and improve quality in the specific case.

5.2 The standard ISO 9126

A definition of software quality, along with guidance for its evaluation, is provided by international standards [19, 92, 93, 94, 95, 96]. This section aims at assessing of the quality level achieved by the release 3.0 of FFMM according to the guidelines of these standards. The software quality model provided by the standard ISO 9126 defines six quality characteristics (Fig. 5.1):

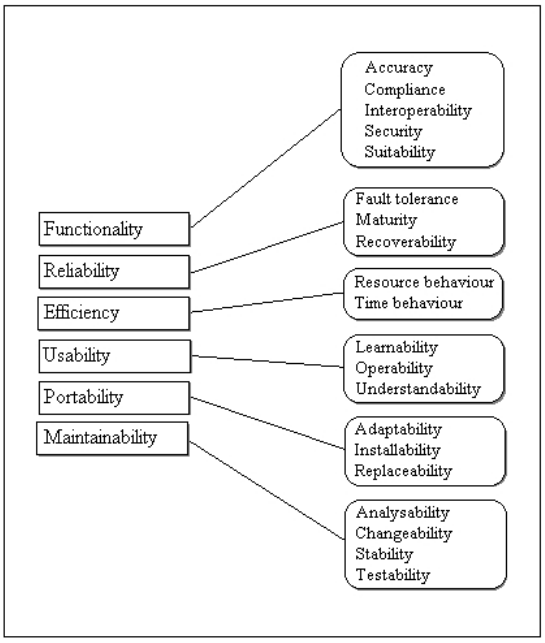


Figure 5.1: The ISO 9126 quality model.

- *Functionality*: the capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

- *Reliability*: the capability of the software product to maintain a specified level of performance when used under specified conditions.
- *Usability*: the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
- *Efficiency*: the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.
- *Maintainability*: the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
- *Portability*: the capability of the software product to be transferred from one environment to another.

The standard defines an additional quality characteristic:

- *Quality in use*: the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use.

The quality characteristics have defined sub-characteristics and the standard allows for user defined sub-subcharacteristics in a hierarchical structure. The ISO framework is completely hierarchical, each subcharacteristic is related to only one characteristic. The quality model defines three different views of quality: (*i*) *software quality in use*, (*ii*) *external software quality*, and (*iii*) *internal software quality*. The software quality in use view is related to application of the software in its operational environment, for carrying out specific tasks by specific users. External software quality provides a black box view of the software and addresses properties related to the execution of the software on computer hardware and applying an operating system. Internal software quality provides a white box view of software and addresses properties of the software product that typically are available during the development. Internal software quality is mainly related to static properties of the software and has an impact on external software quality, which again has an impact on quality in use (Fig. 5.2). As shown in the figure, internal and external

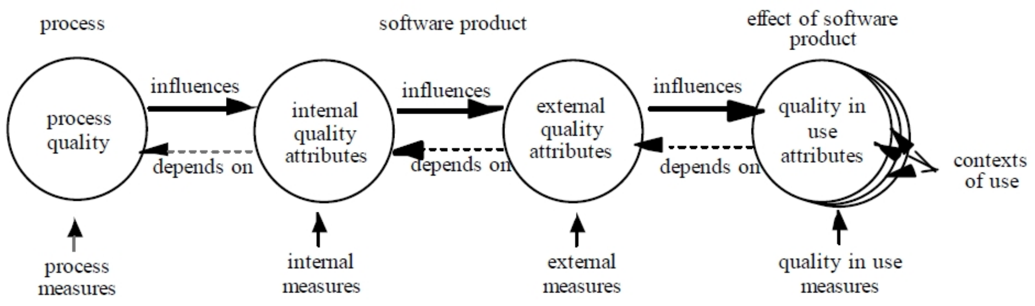


Figure 5.2: Approaches to software quality.

attributes refer to *product quality*, while quality in use reflects the *user view* of the product. The the process through which this product is obtained is not taken into account.
At the current stage of development of FFMM, the system is not yet widely employed by users others than the developers, thus making premature the evaluation of the quality in use. The following quality assessment is therefore product-oriented. More specifically, an analysis of the static properties of the software design and code (*internal quality*) is carried out by means of *pure internal metrics* [93].

5.2.1 Experimental results

The analysis of the software was carried out by means of the tool UnderstandC++ [98]. Heuristic thresholds were employed, as proposed in literature [88, 99, 100, 101], in order to define the metrics target values. An example of metrics and corresponding target values is reported in Tab. 5.3 [88, 102].

In particular, the complexity metrics (such as Essential Complexity and Cyclomatic Complexity [103]) measure the logic complexity of the software modules and hence the effort required for testing and maintain them. The object-oriented metrics, taken from well known metrics suites [104, 105, 106, 107] (LCOM, FAN IN, CBO, RFC, WMC, DEPTH) measure the extent to which features typical of object-oriented systems are exploited (e.g. inheritance) or achieved (e.g. lack of coupling and cohesion). Tab. 5.4 reports a short

Metric	**Target**
Cyclomatic Complexity (CYCLO)	≤ 10
Essential Complexity (ESS)	≤ 4
Class Depending Child (CDC)	$FALSE$
Class Depth (DEPTH)	≤ 7
Multiple Inheritance (FAN IN)	≤ 1
Response for Class (RFC)	$\leq (WMC * DEPTH) + 1$
Coupling between Objects (CBO)	≤ 2
Lack of Cohesion of Methods (LOCM/LCOM)	≥ 0.75
Weighted Methods for Class (WMC)	≤ 14

Table 5.3: Complexity and object-oriented metrics with their target values.

Blank Lines	4’115
Classes	96
Code Lines (LOC)	16’253
Comment Lines	6’977
Comment to Code Ratio	0.43
Declarative Statements	4’779
Executable Statements	8’642
Files	131
Functions	1’082
Inactive Lines	172
Lines	28’119

Table 5.4: FFMM 3.0 size metrics summary.

summary of size metrics computed on FFMM. At first glance, the complexity metrics (Essential Complexity and Cyclomatic Complexity) show that, although in FFMM the average complexities respect the heuristic upper bounds, the maximum values exceed them in a significant way (Tab. 5.5). This means that the complexity is concentrated in few points that need to be simplified in order to decrease the effort required for software testing and maintenance. Analogous remarks can be made from the analysis of the object-oriented metrics (Tab. 5.6): most of them show maximum values significantly exceeding

Metric	Average	Max	Std	Target	% OK (program units)
ESS	1.2	19	1.1	≤ 4	99
CYCLO	2.4	40	3.2	≤ 10	97

Table 5.5: FFMM 3.0 complexity metrics.

Metric	Average	Max	Std	Target	% OK (classes)
CDC	FALSE	FALSE	-	FALSE	100
DEPTH	1	4	1.2	≤ 7	100
FAN IN	0.7	2	0.7	≤ 1	85
RFC	20	138	31	$\leq (WMC*$ *DEPTH)+1	45
CBO	3.9	21	4.6	≤ 2	50
LOCM/LCOM	0.42	1	0.40	≥ 0.75	40
WMC	13	103	18	≤ 14	74

Table 5.6: FFMM 3.0 object-oriented metrics.

the heuristic thresholds, potentially causing problems to system developers and users. The values of the FAN IN metric exceeding the threshold are the result of a conscious design choice, since all the devices implemented in FFMM inherit from two abstract classes. These two classes are completely independent from each other, therefore the multiple inheritance is not expected to cause any undesired side effects.

An attempt to use internal metrics, computed on the source code, according to the model ISO 9126 for an *automatic metric based quality control* is presented in [108, 109]. This approach proposes a set of metrics and a quality matrix mapping them into factors and criteria of the model.

Heuristic thresholds [88, 99, 100, 101] were used to evaluate the percentages of classes exceeding the acceptable values for the considered metrics. The quality matrix allow the translation of these properties into the levels at which quality characteristics are achieved. The quality model presented in [109] refers to an adapted quality model where reusability replaces usability, evaluated on the basis of both non-object-oriented metrics (mainly referring to size) and to object-oriented metrics (mainly referring to cohesion, coupling and inheritance). The results obtained through this approach are shown in Fig.s 5.3 and 5.4. The values vary in the range [0,1], with 0 corresponding to the best quality level. It is worth pointing out that the aforementioned connections between static metric analysis and internal software qualities have been recently proposed, are only partially validated and and still need improvements and confirmations to be widely accepted by the scientific community. In other words, the automatic assessment of software quality defined by

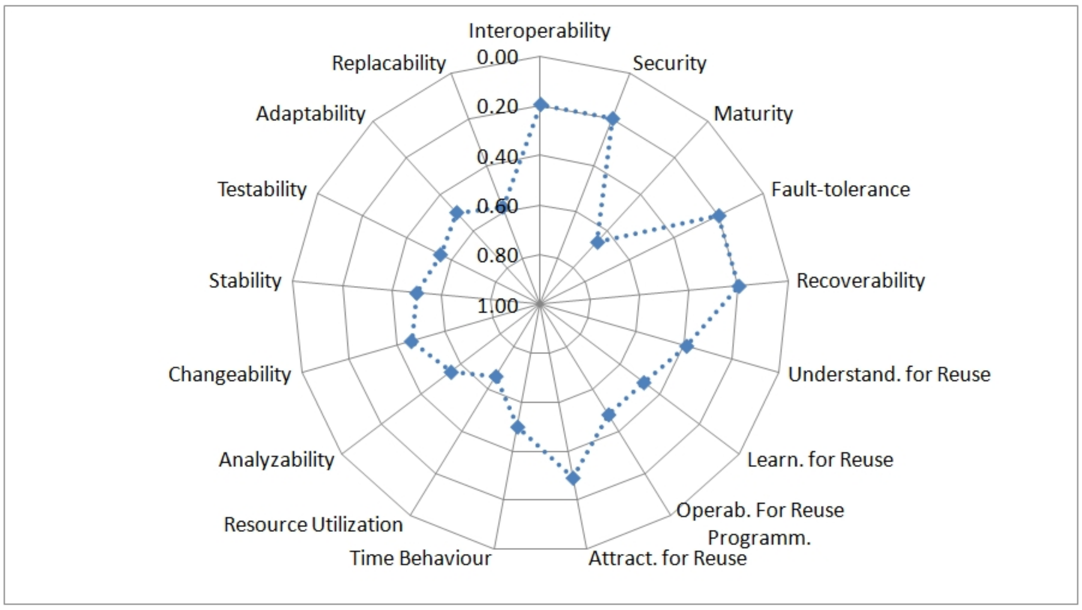


Figure 5.3: ISO 9126 subcharacteristics in FFMM 3.0 (0 indicates the best quality level).

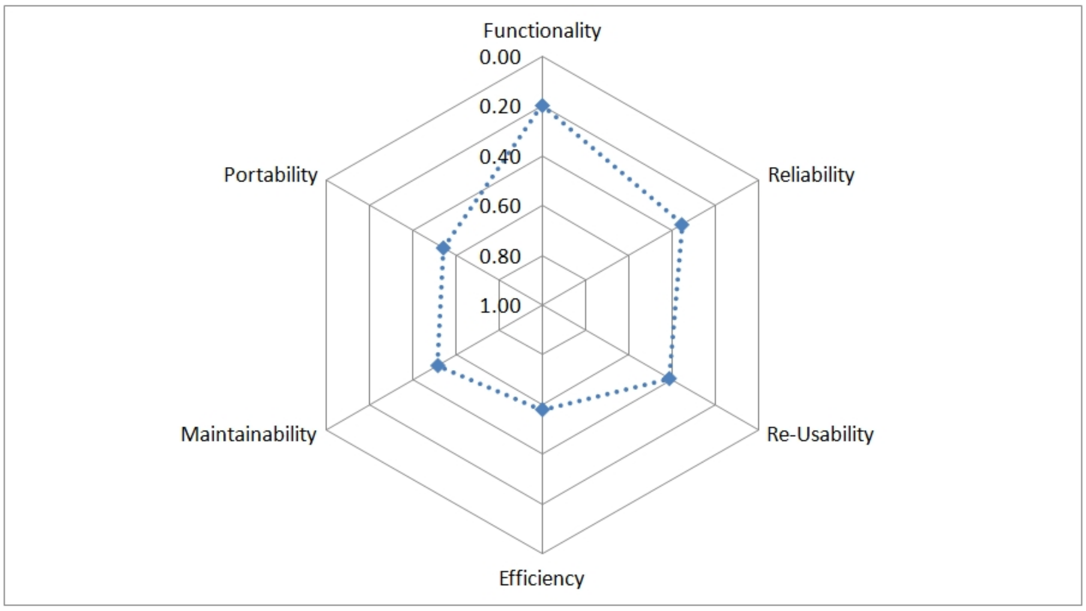


Figure 5.4: ISO 9126 characteristics in FFMM 3.0 (0 indicates the best quality level).

the ISO standards using internal metrics is still an open question. Therefore the results presented in this section are to be intended only as suggestive of the quality level achieved by FFMM. Furthermore, no direct indications are provided on how to improve quality. For this reason, in the following the quality analysis is complemented by introducing a more practical approach.

5.3 Quality pyramid

In [88], a more practical approach to software quality is presented to (*i*) characterize the design of the object-oriented part of FFMM, (*ii*) find the possible problems, and (*iii*) propose corrective actions.

According to these approach, some design metrics [110], namely metrics capturing the quality of the project's design at a certain point in the software development cycle, are used in the frame of the *Goal-Question-Metric* (GQM) [97] technique to effectively characterize and evaluate the design of an object-oriented system.

The characterization of an object-oriented system is a complex task that cannot rely on the evaluation of single metrics in a non-organized way. It requires to choose suitable metrics, compute their values and correlate them in a proper manner in order to draw significant conclusions. As previously said, the characterization of an object-oriented system must necessarily include metrics that reflect three main aspects (Fig. 5.5) [88]): (*i*) *Size and complexity*, to understand how big and complex a system is, (*ii*) *Coupling*, to know to which extent classes are coupled with each other, and (*iii*) *Inheritance*, to understand how much and how well the concept of inheritance is used. To understand these three aspects, the *Overview Pyramid* is introduced in [88]. The pyramid is a metric-based means to both describe and characterize the overall structure of an object-oriented system by quantifying complexity, coupling and usage of inheritance. It can be seen as a graphical template for presenting and interpreting system-level measurements in a unitary manner. An *Overview Pyramid* is composed of three parts, each concerning one of the above mentioned aspects (Fig. 5.6).

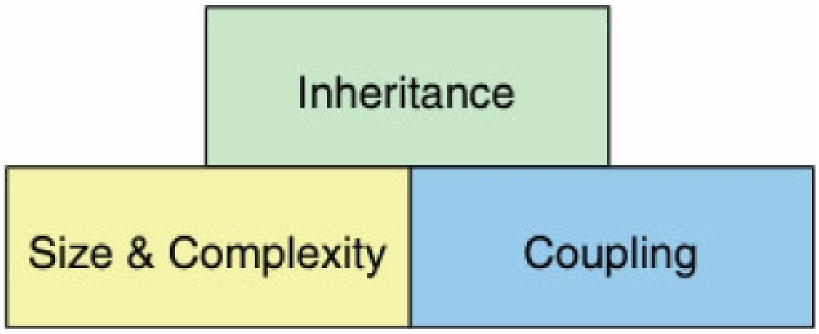


Figure 5.5: The three major aspects quantified by the Overview Pyramid.

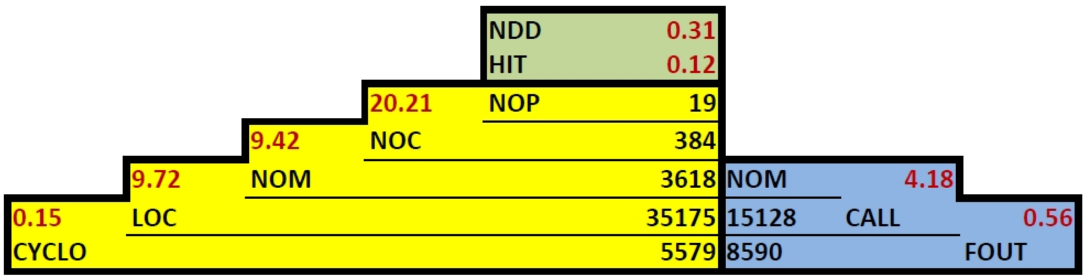


Figure 5.6: Example of a complete Overview Pyramid.

System size and complexity

The left side of the pyramid gathers information characterizing *size and complexity* of the system, provided by direct metrics computed on the source code. These simple and widely used metrics refers to the most significant modularity unit of an object-oriented system, from the highest level (packages or namespaces) to the lowest (lines of code). For each unit there is a metric in the pyramid measuring it. The considered metrics, placed one per line in a top-down manner, are: NOP, NOC, NOM, LOC[1], and CYCLO (Tabs. 5.1 and 5.2). From these basic absolute metrics, some proportions (the numbers on the left) can be computed as rations of the direct metrics, namely those placed immediately next to them by dividing the lower number by the next upper one (Fig. 5.6). The four computed proportions, unlike the direct metrics, have two important characteristics: (*i*) they are independent of each other, thus making each number a distinct characteristic of

[1]In the pyramid only lines of code belonging to methods are counted.

a specific aspect of the code organization, and (*ii*) they allow an easy comparison with other projects, independently of their size, being computed as ratios of absolute values. In the *Overview Pyramid*, the following proportions result: (*i*) **High-level structuring (NOC/Package)**, providing a clue on the package level of the system, i.e if the packages tend to be *coarse grained* or *fine grained*, (*ii*) **Class structuring (NOM/Class)**, providing a hint about the quality of class design, since it reveals how operations are distributed among classes, (*iii*) **Operation structuring (LOC/Operation)**, indicating how well the code is distributed among operations, i.e the structuring from the point of view of the procedural programming, (*iv*) **Intrinsic operation complexity (CYCLO/Code Line)**, characterizing the conditional complexity found in the operations (density of branches with respect to the lines).

System coupling

The right part of the *Overview Pyramid* provides information about the level of coupling in the system (Fig. 5.6), by means of operation invocations. Two direct metrics are used to establish how intensive and how dispersed coupling in the system is: CALLS and FANOUT (Tabs. 5.1 and 5.2). Again, these metrics describe the total amount of coupling of a system, bur are difficult to use to characterize the system. To this purpose, two proportion can be calculated using the number of operations (NOM): (*i*) **Coupling intensity (CALLS/Operation)**, denoting the level of collaboration (coupling) between the operations, namely how many how many other operations are called on average from each operation. Very high values suggest that there is excessive coupling among operations, i.e. that the calling operation do not collaborate with the right counterparts, (*ii*) **Coupling dispersion (FANOUT/Operation call)**, indicating how many classes are involved in the coupling.

System inheritance

The top part of the *Overview Pyramid* is composed of two proportion metrics that provide an overall characterization of the inheritance usage. They give a clue of the extent to which some typical object-oriented features (generalization and polymorphism) are used. The two proportions are: ANDC (Tabs. 5.1 and 5.2). These two metrics reveal the presence and the shape of class hierarchies by capturing two important complementary aspects: their width and height.

Interpreting the overview pyramid

The characterization of an object-oriented system is based on the eight computed proportions previously described. The proportions are preferred to the absolute metrics from which they are computed because they are independent of the project size, thus (*i*) making easier its evaluation, and (*ii*) making meaningful the gathering of statistical data and allowing the use of thresholds based on statistical measurements, as explained in section 5.1. In particular, in [88] metrics were collected from a statistical base of Java and C++ projects.

5.3.1 OOP design characterization

The Overview Pyramid obtained from the FFMM source code by means of the analysis tool inFusion [111], is presented in Fig. 5.7. In order to provide a graphical aid to its

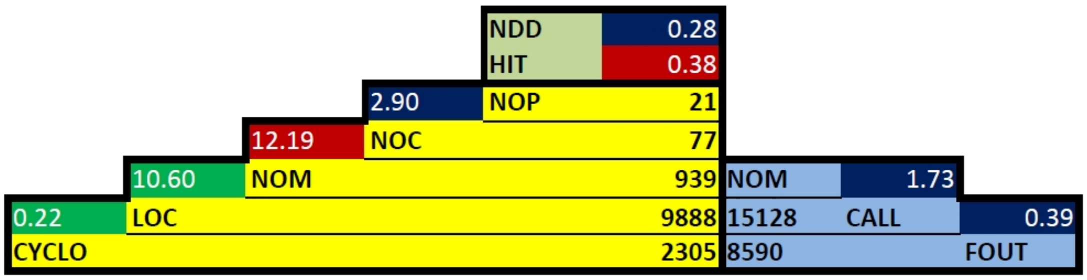


Figure 5.7: Overview Pyramid for the FFMM 3.0 source code.

interpretation, colors were associated to the computed proportions. In particular, a *blue* rectangle shows that the value is closer to the low threshold, a *green* rectangle that it is closer to the average threshold, and a *red* rectangle that it is closer to the high one. An analysis of the resulting computed proportions allows the following conclusions to be drawn for the FFMM source code:

- **Class hierarchies** tend to be *tall* and *narrow*, i.e. inheritance trees tend to have many depth-levels and base-classes with few directly derived subclasses.
- **Classes** tend to be rather *large* (they define many methods) and organized in *fine-grained packages* (few classes per package).
- **Methods** tend to be *average* in length and have an *average logical complexity* (conditional branches).

In the development of a software system it is often difficult to find the appropriate design, the responsibilities of objects and their distribution. Once this has been done, it is important to be able to state whether the complexity due to the design choices is balanced by the benefits they introduce.
Although the *Overview Pyramid* allows the graphic characterization of an object-oriented system through the quantification of some suitable chosen metrics, it is not enough to completely understand and evaluate the design. Metrics and thresholds must be meaningful and put in the right context [88], [112], in order to assess the quality level of the project deign and eventually to ameliorate it. An application, a class, a method should be implemented in an *harmonious* way, in terms of size, complexity and functionality, with respect to itself, its collaborators and its ancestors and descendants. In other words, the system has to achieve an overall *harmony*, composed of three distinct measurable parts:

- **Identity harmony**, related to the extent to which a software entity implements a specific concept and how well (is it doing too many things? is it not doing enough to exist as an autonomous entity?).
- **Collaboration harmony**, expressing the extent to which an entity collaborate with others, and how well (does an entity use other entities? how many?).
- **Classification harmony**, combining elements of the other two harmonies in the context of inheritance (does a subclass use inherited services? does it ignore some of them?).

The design evaluation, from the point of view of the harmonies presented above, aims therefore at stating if every software entity has *appropriate* place, size, complexity within the system.

Disharmonies are revealed by means of metric-based heuristics to detect and locate object-oriented design flaws from the source code. For the quantification of complex design rules, the evaluation of a single metric is not sufficient. Therefore *detection strategies*, composed logical conditions based on proper set of metrics and thresholds, are used to evaluate design quality of an object-oriented system through quantifying deviations from good design heuristics and principles. Design rules are in this way made quantifiable, so that is it possible to detect fragments of the source code with specified properties (typically denoting a design problem). The result of this stage of detection is a list of software entities *suspected* to be affected by some flaw. These entities need subsequently to be inspected to find those that cause the most severe problems and determine how to refactor them. This insight into the class structure is needed to understand the static structure of the class, i.e. the way attributes are accessed, methods called, inheritance used, and decide if there is need for intervention.

Among the 11 design disharmonies classified in [88] in the three categories presented above, those that were detected in FFMM are discussed in the following. It is worth pointing out that no significant code duplication was found. This is an important achievement of the FFMM design, since code duplication can be very harmful, breaking the uniqueness of entities with certain functionalities in the system, and causing an increase of size, complexity, error proneness, and problems of co-evolution of clones.

Identity disharmonies

Identity disharmonies are design flaws that affect single entities such as classes and methods. Three distinct aspects contribute to the disharmonies of a single entity: its size, its interface, and its implementation. These aspects can be summarized in three rules f identity harmony:

- *operations and classes should have an harmonious size*;

- *each class should present its identity (interface) by a set of services , which have one single responsibility and which provide a unique behaviour*;
- *data and operations should collaborate harmoniously within the class to which they semantically belong.*

The most frequent and easily recognizable sign if identity disharmony is is excessive size and complexity of a class (*proportion rule*), and one of the causes could be a massive presence of code duplication. Another sign of disharmony is the lack of cohesiveness of behaviour (*presentation and implementation rule*) and the tendency more and more features and services, thus producing a *God Class* [113, 114]. Generally, the more a class tends to become a God Class, the more other classes communicating with it tend to be *Data Classes* [113], simple data containers not providing much functionality. As a consequence, the methods of the God Class, which use foreign data, produce the Feature Envy [114], since they are more interested in attributes of other classes than those of their own class.

God Class The *God Class* design flaw refers to classes that tend to centralize the intelligence of the system [113]. A God Class performs too much work, delegating only minor details to a set of trivial classes and using the data from other classes, with negative impact on *reusability* and *understandability.*
In FFMM, the three classes were detected as God Classes because: (*i*) some of their methods access directly (or via getter/setters) attributes from external classes, (*ii*) the methods are very complex, i.e. have many branches, and (*iii*) the classes are non-cohesive with respect to the way methods use the attributes of the class. An analysis of the results shows that: (*i*) the methods accessing attributes from external classes are the same for the three God Classes, and are used for data type conversion, bus configurations, or thread handling. In all cases, they result from precise implementation choices and are not considered a source of problems. (*ii*) the cohesiveness of all the classes is only slightly below the limit of one third. As a consequence of these considerations, the problem represented by the God Classes can be classified as noncritical. Anyway it could be

useful to reduce the complexity of the methods, since this might affect *understandability*, *usability*, and *maintainability*.

Data Class A *Data Class* [114, 115] is a data holder without complex functionality, but with other classes strongly relying on it. The lack of functionality may indicate that related data and behaviour are not kept in one place. In other words, Data Classes provide almost no functionality through their interfaces, which mainly exposes data fields either directly or though accessors methods.
In FFMM the three detected Data Classes do not produce any Feature Envy. This implies that, although they do not provide complex functionality, the data they contain are not significantly accessed by methods of other classes. The problem represented by the detected Data Classes can be therefore reappraised if we put it into this perspective. They do not contain misplaced data needed by other parts of the system, but are the result of a precise design choice for handling the communication buses and their configurators (chapter 3).

Brain Class This design flaw refers to classes that tend to accumulate an excessive amount of intelligence, often concentrated in Brain Methods. It recalls the God Class, but the two disharmonies are distinct. A God Class, besides being complex and centralizing a large amount of the system's intelligence, breaks the encapsulation principle accessing attributes from other classes and shows a lack of cohesion. The Brain Class detection strategy is complementary to that of the God Class, catching very complex classes which do not break encapsulation and do not manifest a significant lack of cohesion.
The main characteristic of a Brain Class is that it is very likely to contain Brain Methods, therefore the first improvement action should be directed to these methods, as discussed in the relative section. On the other hand, if the class is detected as a Brain Class because of its lack of cohesion, it should be split into an appropriate number of more cohesive classes. However, it is often the case where a Brain Class does not cause any significant problem in the system, for example if it is simply a mature complex utility class. In such cases, if no maintenance problems arise during the system's history [116], it is not worth

starting a costly refactoring phase just to get better metric values.
In FFMM, the six Brain Classes contain only 1 or 2 Brain Methods, and are characterized by high complexity and low cohesion. As a corrective actions, the Brain Methods should first be split. In case of maintenance problems, the involved classes could also be split in smaller and more cohesive units.

Feature Envy Objects are a mechanism for keeping together data and operations processing that data. The *Feature Envy* design flaw [114] refers to methods that access, directly or through accessor methods, more data of other classes than of their own class. The Feature Envy disharmony is often a sign that a method was misplaced and should be moved to another class. The problem can be solved if the method, or a part of it, is extracted and moved to the envied class [114, 115].
The Feature Envy problem is often due to the presence of Data Classes, which make the classes using them to envy their data. When a method is affected by Feature Envy, it is probable that there are data classes among the classes from which the method accesses data. The fact that in FFMM the two methods affected by Feature Envy do not access Data Classes highlights that the problem is isolated to two methods used to handle the polling tasks of a class. Even though this is detected as a violation of the encapsulation principle of the object-oriented design, it is the result of an implementation choice for the threads executing the polling methods. Moreover some metrics are very close to the detection threshold. It is therefore to be concluded that the observed Feature Envy does not represent a critical issue in the system, and that the cost of a possible code improvement action would not be balanced by proportionate benefits.

Brain Method *Brain Methods* tend to centralize the functionality of a class (proportion and implementation rules), in the same way as God Classes centralize the functionality of the whole system or of a subsystem, making it hard to *understand*, *maintain*, and *reuse* [88].
The 14 Brain Methods detected result rather complex (both for conditional branching and nesting) and long, and employ a very large number of variables. For the negative impact

of this design flaw on *understandability*, *maintainability* and *reusability*, these methods require a refactoring. In this case, the Brain Methods do no not exhibit either significant code duplication or Feature Envy, thus there is no cloned code to remove nor Data Classes to which some of the behaviour complexity can be moved. In literature it is suggested that in almost all cases a Brain Method should be split into one or more simpler methods [114], by finding appropriate cutting points.

Collaboration disharmonies

Collaboration disharmonies are design flaws that affect several entities at once in terms of the way they collaborate to perform a specific functionality. All the authors propose low coupling as a design rule for object-oriented system[2]. Anyway, a tradeoff needs to be found between the aim of low coupling and the need of a certain amount of collaboration among objects of the same system. The collaboration harmony consists in the achievement of a balance between the aforementioned opposing demands.
All this can be summarized in the following rule:

- *collaborations should be only in terms of methods invocations and have a limited extent, intensity and dispersion.*

where *extent* refers to the number of other classes, *intensity* to the number of services provided by other classes, and *dispersion* to the distance of collaborating classes (two classes can be in the same hierarchy, package, ...). The rule refers both to incoming and outgoing dependencies. Excessive outgoing dependencies are undesirable because they make a class more *vulnerable to changes and bugs of other classes.* On the other hand, excessive incoming dependencies are undesirable because they create the need of stability and therefore make the class *less evolvable*[3]. Collaboration disharmonies are captured using two detection strategies: *Intensive Coupling* and *Dispersed Coupling.* The former refers to the case where a method uses intensively a reduced number of classes, the latter

[2]To this purpose, an Event Handling Infrastructure (chapter 3) is used whenever possible in FFMM, minimizing the coupling introduced by the necessary communications among objects.

[3]It is to be noted that, at the same time, incoming dependencies could mean high degree of *code reuse* in the system, with the condition of having stable interfaces.

to situations where the dependencies are dispersed over many classes. Moreover, on the server method's side, it might happen that a method is excessively invoked by many methods located in many other classes (*Shotgun Surgery* [114]). In this case, a small change in a part of the system can cause lot of changes in many other classes.

Intensive Coupling The main reason for reducing coupling is that this is required in order to be able to use a component without the others or to make easier the replacement of a component with another one. A usual refactoring action that allows to solve the problem consists in defining a new more complex service in the provider class, and replacing the multiple calls with a single call to the new method. Anyway in some cases the design flaw might be due to misplaced operations. This is the case of FFMM, where the intensive coupling involves two methods contained in the classes modeling two motor controllers. These methods access many (11) methods of another class. The origin of the problem is the fact that the methods were developed at a too low abstraction level in the system, i.e. in a device. In fact each method involves two devices, and if inserted in one of them it has necessarily to call many methods of the other class. On the basis of this consideration, it is clear that it should be implemented at a higher level, as a measurement routine employing both the aforementioned devices. Placing the method into a library of measurements routines would solve the coupling problem, besides removing the code duplication by leaving only one instance of the method.

Shotgun Surgery Unlike the intensive coupling, the *Shotgun Surgery* refers to the cases where the incoming dependencies can cause problems, i.e. where a change in an operation implies many (little) changes to a lot of other components (methods and classes) in the system [114]. Similarly, if this method is erroneous (has bugs) this will have a significant negative ripple effect on the parts of the system that are using it. In such a situation, *maintenance* and *evolution* problems could arise.
A possible refactoring action to solve the problem consists in moving more responsibility to the classes containing Shotgun Surgeries, above all for small and non complex methods and classes with tendency to become Data Classes [88]. Anyway, this is not the case in

FFMM, where this design flaw involves five methods of two classes designed in order to *favour code reuse* for interacting with the user and handling the communication buses (and consequently massively accessed by methods in other classes). The characteristics of these methods come therefore from a conscious design choice, and do not cause any problems under the assumption that they are used through stable interfaces.

5.4 AOP Fault Detector characterization

Finally, a twofold analysis was performed on the aspect-oriented Fault Detector (chapter 3). The analysis is aimed at (*i*) assessing the modularity improvement deriving from its introduction (internal quality), and (*ii*) evaluating its performance to verify that these benefits do not introduce run-time side effects (external quality).

5.4.1 Modularity comparison

The software quality of the proposed AOP version of the fault detector was evaluated in comparison with the corresponding OOP version previously existing inside FFMM. With this aim, the software quality attribute of modularity was assessed for both the AOP and OOP versions by evaluating (*i*) the percentage of lines of source code related to fault detection logic present in each module with respect to the total lines of code (LOC) of the same module, and (*ii*) the Degree of Scattering (DOS) and Degree of Focus (DOF) metrics [117], for each module and fault detection concern. The analysis was focused on the most relevant fault sources of the FFMM, i.e. the modules implementing devices.
In Tab. 5.7, the analysis results along with the ratio of code duplicated in the different software modules (cloned code ratio) are reported. In the OOP version of the fault detector, a high level of cloned code exists, because in each device operation often the same tests against the internal status are requested. Conversely, in the AOP version, this ratio is drastically reduced. An ideal implementation of the Fault Detection concern would have a null DOS and a DOF equal to 1, for each device module (i.e. each device is focused only on the base concern and does not contribute at all to the Fault Detector concern). Tab. 5.7 shows that the OOP version has a very-low value of DOF, for each

Device	**OOP FD** %LOC	**OOP Cloned** %LOC	**OOP DOF**	**OOP DOS**	**AOP %LOC**	**AOP DOF**	**AOP DOS**
FastDI	15.75	10.93	0.17		0.81	0.97	
Maxon_Epos	18.04	9.78	0.21		0.73	0.97	
EncoderBoard	21.53	14.27	0.28	0.96	0.62	0.98	0.13
PowerSupply	18.36	12.70	0.24		2.64	0.90	
Transducer	21.15	8.24	0.27		1.79	0.93	
Keithley2k	18.48	11.32	0.16		0.77	0.97	

Table 5.7: Fault detection code in each device module and computation of percentage DOF and DOS metric for both OOP and AOP versions (OOP: object-oriented programming; AOP: aspectoriented programming; LOC: lines of code; DOF: degree of focus; DOS: degree of scattering).

module (i.e. all modules contribute to the fault detection concern), and a DOS for the Fault Detection concern near to the maximum (uniformly scattered). This means that the fault detection concern in the OOP version has very bad values of modularity. Thus, any not trivial *maintenance* (or *evolution*) is very difficult, because each modification could affect and require changes in many different software modules (i.e. mainly all device modules).

Instead, in Tab. 5.7, DOS values of the AOP version are near to the minimum: the fault detection concern is well modularized in one module (the FaultDetector aspect), and each device module is marginally involved in the concern (such as said before, this is due to the fault and error broadcasting methods not yet removed from the devices). This results is better highlighted in Figs. 5.8 and 5.9. In particular, in Fig. 5.8 the percentage LOC (%LOC) of the Fault Detection concern all over the device modules are compared for both AOP and OOP versions. In the figure, the ratio of the cloned LOCs in the OOP implementation, completely removed in the AOP version, is reported. Of course, the cloned code makes worst the maintainability and increases the probability of introducing bugs in the code. In Fig. 5.9, the level of DOS (a) and DOF (b) for each device module with respect to Base System and Fault Detection concerns is reported. The results show a radically increased modularity for the AOP version, because each device module is much more focused on the base concern with respect to the OOP version. Moreover, the fault detection concern is highly scattered in the OOP version (high values of DOS), while it is very focused in the AOP implementation (very low values for DOS).

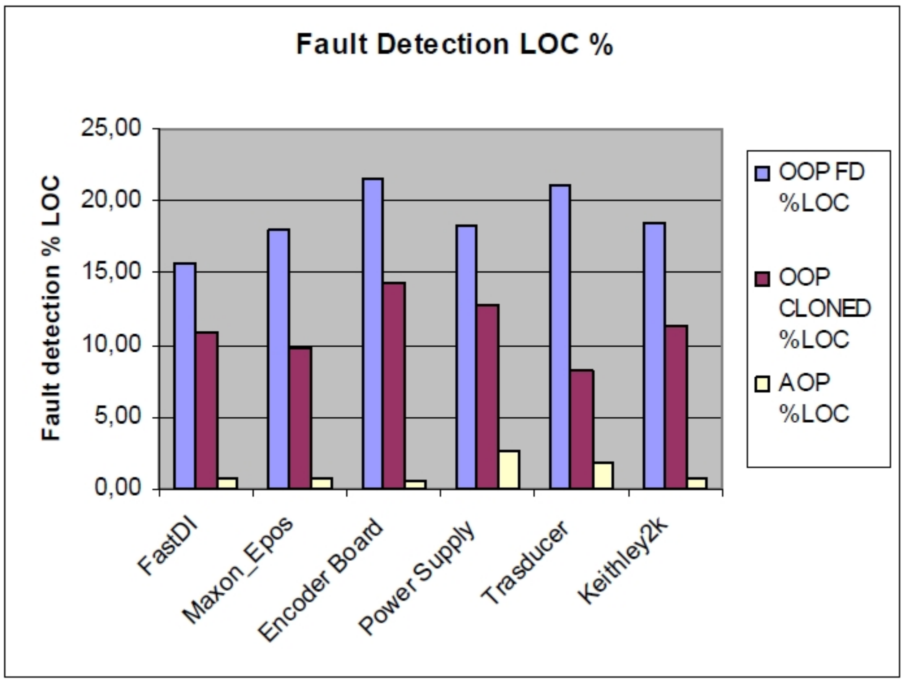
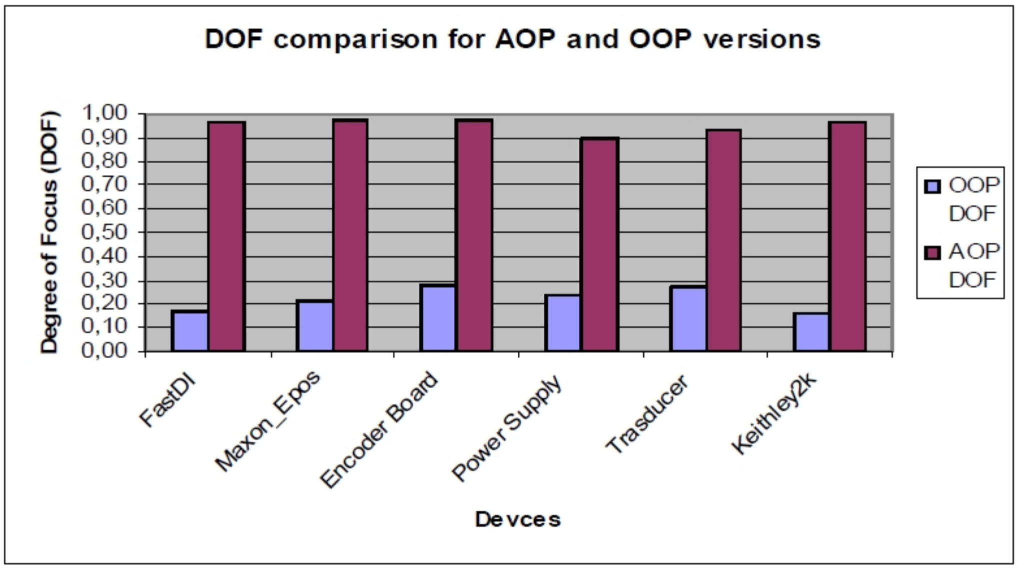


Figure 5.8: Percentage lines of code (LOC%) of fault detection concern in device modules for OOP and AOP versions.

5.4.2 Performance verification

The analysis was aimed also at verifying experimentally that the AOP architecture would not have a negative impact on run-time performance of the overall system (due to aspect runtime interception overhead). With this aim, the AOP system was instrumented in order to gather execution times of the aspect overheads. In both the versions, fault detection times related to fault decoding and handling, are present. They were filtered out from the analysis. Therefore, main attention was paid to evaluate the overheads added by AOP interception mechanism to the fault detection time in order to assess the effectiveness of the AOP architecture, i.e. that the AOP response times are not worst than the OOP version. The above described analysis was carried out by running the two versions of the software in the same conditions. The runs were performed by causing some faults in the measurement station previously described. Those faults were induced intentionally in different ways, for example by providing the devices with wrong parameter values, by interrupting the communication between the PC and the devices (device not found, or

(a)



(b)

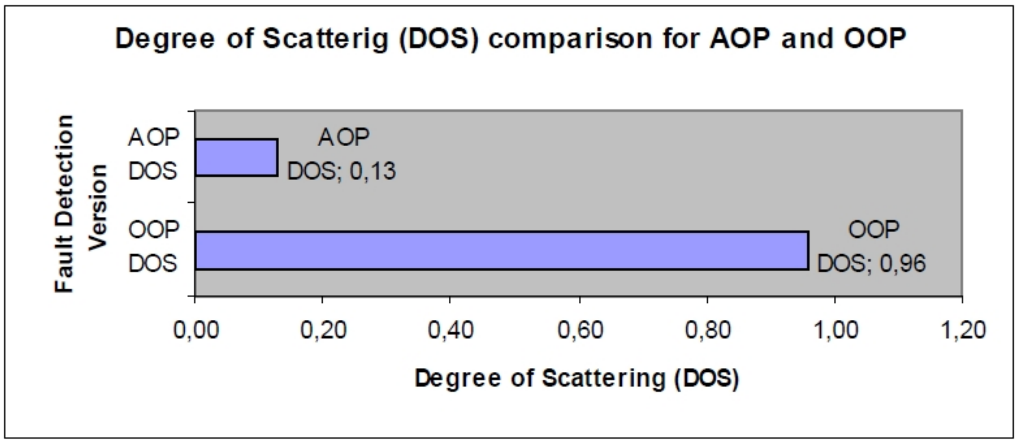


Figure 5.9: DOS (a) and DOF (b) comparisons of OOP and AOP versions with respect to Fault Detection concern.

communication timeout if the communication with device had already been established), by starting the FDI acquisition procedure without feeding the instrument with the required trigger signal (measurement timeout) and adding a delay in the execution of some commands (command timeout). The worse average times in several different categories of fault detection pointcut expressions (i.e. device creation/destruction, interception of device operations) were selected, and the time spent in the aspect runtime to jump to fault detection routines were collected. These are reported in the last column of Tab. 5.8[4] (in percentage of the total time spent in the aspect). For the sake of clarity, the results of Tab. 5.8 are reported also in Fig. 5.10. Times needed to handle creation/destruction of devices (pointcut expressions from rows 1 to 8) are greater than those required to handle faults during measurement tasks (pointcut expressions from rows 9 to 15). In the former cases, the fault detector infrastructure must be set up for devices being created. This requires more time than the other kind of pointcuts expressions, that have only to capture the context of an operation, issuing a fault event if necessary. These times are comparable to those of the OOP version, where listeners are explicitly registered with the created devices to handle faults. In these cases, the aspect overhead is particularly reduced with respect to the entire fault detection tasks. Pointcut expressions ranging from row 9 to row 15 are related to fault detection during normal device operations. Their goal is to capture all the context in which the device state changes to check its validity. In the developed AOP implementation, the worst overheads due to aspect interception mechanism (see the last column in Tab. 5.8) are always less than 1.5 % of the fault detection times (the worst case is for the interception of calls to EncoderBoard device operations with complex arguments matching expression to check preconditions; related to pointcut expression at row 9). Therefore, the suitability of such performance overhead in the concrete measurement scenario was assessed, and all the timing constraints were satisfied.

[4]The times refer to a Pentium IV 1.3GHz machine, with 512Mb of RAM running the instrumented AOP version.

	Pointcut expressions	Total Time (ms) spent in aspect	Time (ms) spent in matching advices	Time (ms) spent in aspect runtime	%Time spent in aspect runtime
1	EncoderBoard construction	13,029641	13,028840	0,000801	0,006%
2	FastDI construction	45,912234	45,893478	0,018756	0,041%
3	FDICluster (1 element) construction	59,062982	59,010519	0,052463	0,089%
4	Maxon_Epos construction	14,013891	14,011993	0,001898	0,014%
5	EncoderBoard destruction	10,282883	10,282652	0,000231	0,002%
6	FastDI destruction	18,511722	18,511302	0,000420	0,002%
7	FDICluster (1 element) destruction	6,034312	6,034096	0,000216	0,004%
8	Maxon_Epos destruction	11,045256	11,045013	0,000243	0,002%
9	within(EncoderBoard)&&call(%) && args(...)	2,8803261	2,842784	0,037542	1,303%
10	withincode(FastDI) && call(plx->write)	1,146675	1,144412	0,002263	0,197%
11	withincode(FastDI) && call(plx->read)	1,486675	1,476675	0,010000	0,673%
12	withincode(Maxon_Epos) && execution(set%)	0,982102	0,980102	0,002000	0,204%
13	withincode(Maxon_Epos) && execution(get%)	0,902345	0,899015	0,003330	0,369%
14	withincode(Maxon_Epos) && execution(start())	2,896735	2,886735	0,010000	0,345%
15	withincode(Maxon_Epos) && execution(stop())	2,416875	2,406875	0,010000	0,414%

Table 5.8: Worst average times spent in aspect runtime with respect to device creation/destruction and fault detection pointcuts.

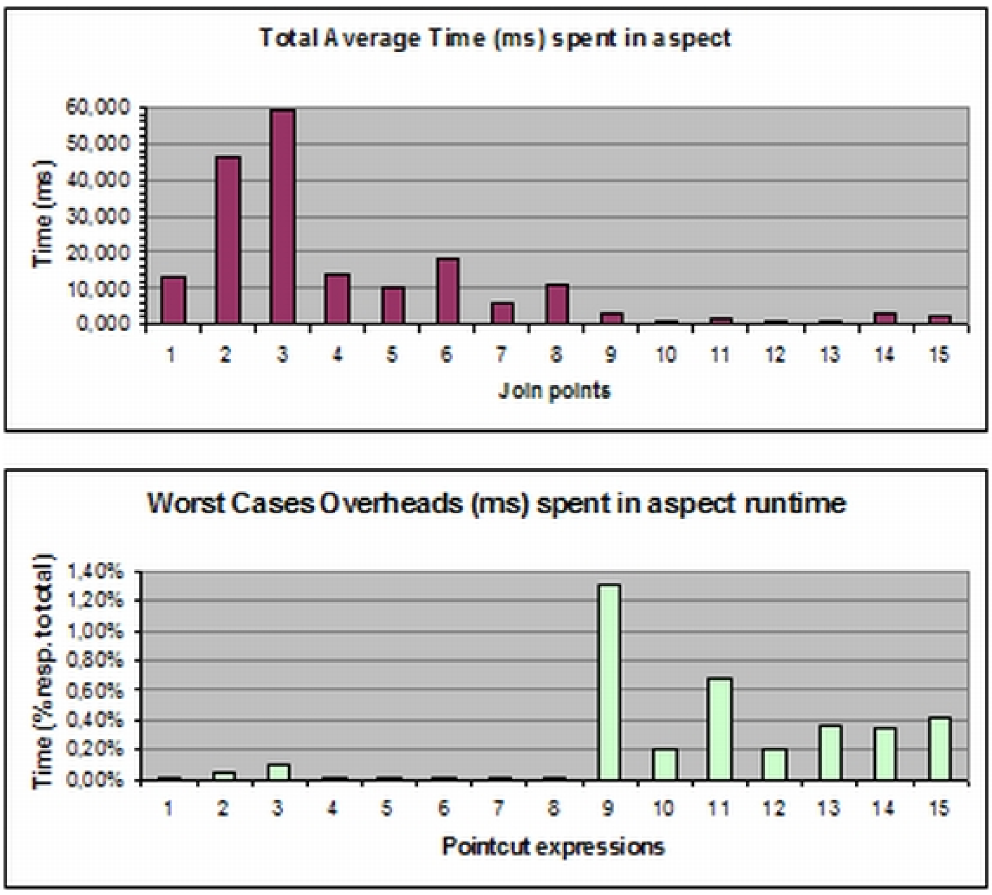


Figure 5.10: Times spent in aspect runtime. The pointcut expressions numbering refers to Tab. 5.8.

5.5 Discussion

This chapter presents the results of the software quality characterization of the object-oriented and aspect-oriented parts of the release 3.0 of the Flexible Framework for Magnetic Measurements. The characterization of the object-oriented part was carried out with reference to the quality model ISO 9126 developed by the International Standard Organization. Only the internal quality of FFMM source code was taken into account. As a consequence, the quality assessment relates more to the developer point of view than to that of the user. Both complexity and object-oriented metrics were evaluated. Although the results highlighted a good average quality level, improvements are possible in order to decrease the maximum complexity and to exploit more profitably the concepts of object-oriented programming. A supplementary analysis was carried out in order to find the main parts requiring improvement in the object-oriented design.
Subsequently the aspect-oriented Fault Detector was analyzed. Such a software design extends the Object-Oriented approach, by adding specific encapsulation of crosscutting concerns. The advantages of using AOP in the development of a fault detector were verified in the case of a measurement application based on rotating coils for testing a superconducting magnet. The proposed architecture allows a high level of flexibility by performing very complex and bendable run-time binding among sources and handlers of the faults, without affecting significantly the performance, while keeping the detection code well modularized in its hierarchy. Another main advantage of such a technique is the maintainability and the reusability of the code: for each new device added to the framework, the related fault detection code is added to the fault detection hierarchy. Since all fault detection code is well modularized in few sub-aspects, commonalities among different fault detection logic is well structured and factored out. As a consequence, the FaultDetector design, with respect to ‘traditional’ OOP version, exhibits a much more centralized design, reducing code duplication and greatly increasing the possibility of code reuse. Finally, the proposed AOP architecture is not targeted at a specific system component, and the same fault detector architecture can be reused to detect different kinds of faults in different components.

Chapter 6

Validation on LHC-related measurement applications

The framework was validated on the field in some scenarios typical of LHC related measurements. In particular, in the following two applications are described. The former, the tracking test, is based on the rotating coil technique and aims at the estimation and compensation of the field errors due to non-ideality of the LHC superconducting dipoles. The latter aims at measuring the permeability of a sample of soft steel through a fixed coil system. This chapter presents the setup of the measurement stations and the results obtained by means of the software produced through FFMM.

6.1 Application scenarios

The Flexible Framework for Magnetic Measurement was used to develop applications for different activities currently carried out at CERN. Different application scenarios are a good test bed for checking FFMM flexibility, namely its capability to offer an environment for a fast development of several measurement applications with different requirements. A specific discussion of FFMM flexibility is presented in chapter 8. Here, the discussion is focused on describing the measurement procedure and the test stations, and on presenting the first results obtained on the field.

	commissioning	**nominal operation**
b_3	0.35	0.02
b_5	-	0.1

Table 6.1: Injection harmonic tolerance (values are shown in *units*).

6.1.1 Tracking test

The LHC has unprecedented demands on the control of the field and its errors during injection, acceleration, and collision. One of the most stringent requirements during the energy ramp of the LHC is to have a constant ratio between dipole-quadrupole and dipole-dipole field so as to control the variation of the betatron tune and ensure that the beam orbit remains the same throughout the acceleration phase, hence avoiding particle losses. Furthermore, superconducting magnets for particle accelerators are affected by characteristic dynamic effects leading to field errors of the order of a few 10^{-4} relative to the main harmonic component. These errors, which were observed and studied systematically for the first time at the Tevatron [118], are due mainly to the diffusion of persistent magnetization currents through the strands composing the superconducting cable. In particular, LHC double-aperture 15 m long, 8.34 T main dipole magnets are affected by a slow decay of the sextupole (b_3) and decapole (b_5) components during the low-field phase of particle injection. Subsequently, as the field is ramped up for beam acceleration, these error components snap back suddenly to their initial value, unbalancing the beam orbit and giving rise to significant particle losses [119]. The tolerances of the sextupole and decapole correction are calculated from the beam requirements [20] and these hence provide a specification for the maximum allowed field errors. These calculations [20] yield the tolerances shown in Tab. 6.1 for the commissioning and the nominal operation phases.

For the reasons presented above, a specific test (*tracking test*) is performed with a twofold purpose:

1. generate the current ramps for the main superconducting magnets which would produce the expected magnetic fields;

2. generate the current ramps to supply the corrector magnets and compensate the

sextupole and decapole field errors in the main dipole.

It is known that the decay amplitude is affected by the power history of the magnet, and particularly by the pre-cycle flat top current and duration. A Field Description for the LHC (*FiDeL*) [120], modeling the field variations during injection, acceleration, and collision, was developed to cope with the difference between the test procedure and the expected cycles during the machine operation. *FiDel* is a feed-forward system used to forecast and compensate the field variations within the commissioning tolerance, so that subsequently suitable tools based on beam measurements can bring the beam to the nominal parameters. In practice the LHC ring is divided into 8 sectors, in each of which it is possible to actuate independently the compensation actions by means of power converters supplying the series of the magnets. An average field model for each sector is therefore necessary. This model is obtained from the measurements of (i) all the magnets at room temperature, and (ii) one third of the magnets at cryogenic temperature (1.9 K).
In this section, the application of FFMM to the rotating coil techinque was devoted to the second point, the field harmonic correction, since this problem strongly relies on field harmonic analysis, for which rotating coils (section 1.1.1) are one of most accurate techniques. In particular, in the following the compensation of the sextupole term for the commissioning phase is considered.
Past measurement campaigns [20] highlighted the need to improve the harmonic compensation of the third-harmonic (b_3) component of the main LHC dipoles. In particular, measurements had already been carried out by means of the standard measurement equipment [80], but the time resolution obtained on the field estimation was intrinsically limited by the acquisition hardware, so that a new harmonic value was available only every 20 s. A new, fast hardware was developed to overcome this limitation [4]. Currently the harmonic estimation can be updated at a rate of 8 S/s. Anyway, the new hardware still needed a proper acquisition and control software. This software was realized by means of FFMM. In the following, (i) the *measurement procedure*, (ii) the *test station architecture*, (iii) the *compensation mechanism*, and (iv) the *experimental results* are presented.

Measurement procedure

The test procedure for the harmonic compensation is composed of the following steps:

1. Measurement of the integral dipole field and error components during a nominal LHC machine cycle;

2. Measurement of the transfer function of the two superconducting sextupole corrector magnets (MCS) installed in line with each dipole aperture in the same cryoassembly;

3. Computation of the compensation current for the MCS from the results of points 1 and 2;

4. Measurement of the integral field when the main dipole performs an LHC cycle and the MCS are supplied with the compensation current, and estimation of the residual field errors.

The first step is aimed at characterizing the LHC dipole magnet during a nominal machine cycle (LHC cycle, Fig. 6.1) [76]. In particular, the measurement is carried out to get a

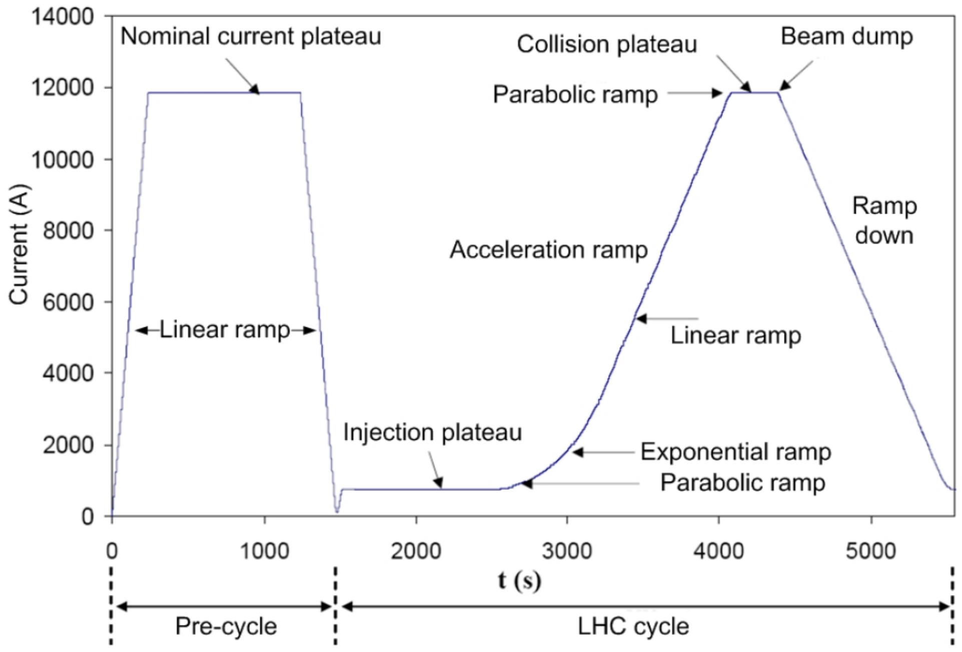


Figure 6.1: LHC standard current cycle.

reference behavior, without compensation, of the integral harmonic component $\overline{B}_3$. The nominal LHC cycle has a ramp-up at 10 A/s from 350 A to an injection current plateau at 760 A, lasting about 1000 s, to simulate the particle injection at constant field. This is followed by a Parabolic Exponential Linear Parabolic (PELP) [76] profile, a 1000 s flat top at nominal current of 11850 A, and a ramp-down at 10 A/s to the minimum current of 350 A. The LHC cycle is also preceded by a pre-cycle aimed at bringing the magnet into a reproducible magnetic state. The reproducibility of the sextupole during LHC cycles is better than 0.1 units [20]. The target value of 0.02 *units* for the b_3 compensation (Tab. 6.1) is therefore to be intended as an ideal target.

The second step is aimed at (*i*) computing the Transfer Function (TF), i.e. the ratio between the field and the current, for the two sextupole correctors, and (*ii*) verifying the linearity of such TF. The resulting TF allows the required sextupole excitation current to be computed for the compensation of the $\overline{B}_3$ field inside the dipole. The field is measured during several ramp cycles, i.e. from 0 A up to the nominal current of 550 A, down to -550 A and back to 0 A with a ramp rate of ± 10 A/s.

The third step is the main measurement procedure. The sextupole is fed with the current cycle computed via the TF and at the same time the LHC dipole is fed with the nominal LHC cycle, both cycles being tightly synchronized ($< 1ms$). The results of such a measurement highlight the quality of compensation for the third harmonic in the dipole.

Measurement station architecture

In Fig. 6.2, the architecture of the measurement system is shown. The core is the fast equipment for harmonic coils measurements. Rapidly varying magnetic fields are measured by designing the measurement station according to the main specification of improving the bandwidth. This was achieved by means of developing high-speed rotating units and associated electronics. The algorithm for harmonic resolution enhancement presented in section 3.4.2 was not employed, because the current limitations of the power converter control system make it pointless.

Both apertures of the cryoassembly are equipped with a rotating shaft, made by 12

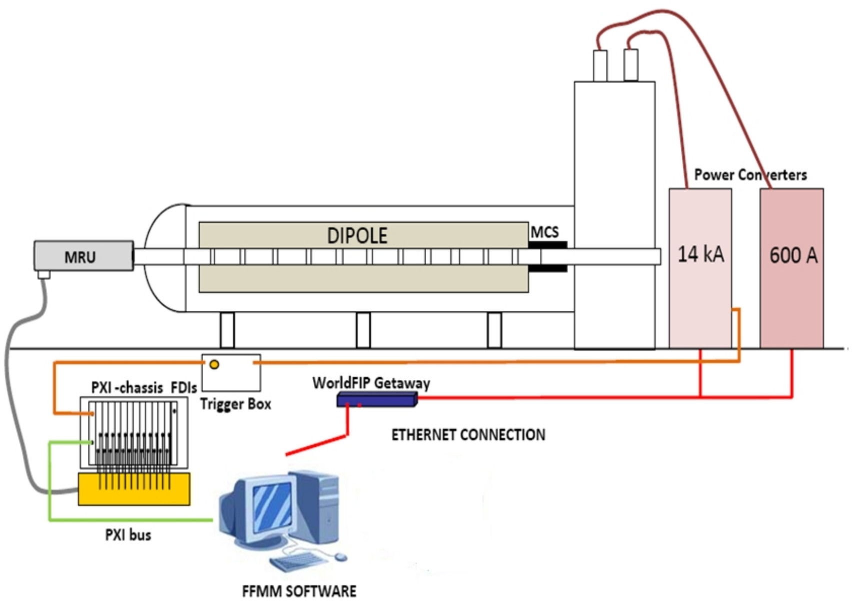


Figure 6.2: Architecture of the tracking test measurement station.

pivoting ceramic segments each holding three tangential, equal and parallel pick-up coils. One coil is normally used to measure the dipole field component (the so-called "absolute" signal), while the connection in series opposition with a second coil provides cancellation of the dipole ("compensated" signal) and ensures higher SNR for the measurement of harmonic error components. The shaft covers the whole length of the LHC dipole and the last segment captures the sextupole corrector field in its entirety. In the present arrangement, the signals from consecutive segments are connected in series by three groups of four segments, constituting three "super segments" with the purpose of limiting the number of necessary integrators to 6 per aperture. Two Micro Rotating Units[4] provide a rotation speed up to 480 *rpm* in order to get voltage signals with the desired time resolution. The absolute and compensated signals, from each super segment, are the input of a Fast Digital Integrator (FDI) [33] measuring the magnetic flux linked with the super segment coils. The pulses from the angular encoders trigger the integration time of the FDI's and the acquisition of the supply magnet current: the synchronization between magnetic flux sample and the current measurement is thus ensured. The software used to handle the station and to retrieve the current reading, via ethernet connection from the power supply controller, is obtained through FFMM.

Compensation mechanism

The sextupole harmonic component in the LHC dipole is compensated by applying to the MCS magnets a current computed from from (*i*) the integral $\overline{B}_3$ and (*ii*) the sextupole corrector's TF. The integral $\overline{B}_3$ (expressed in Tm) is defined as:

$$\overline{B}(t)_3 = \sum_{i=1}^{3} L_i B(t)_{i,3} \tag{6.1}$$

where L_i and $B(t)_{i,3}$ are the effective length and the measured third harmonic of the *i-th* super segment, respectively. The effective length is adjusted to take into account the contribution of the gaps between the coils. All harmonic field values are expressed in T measured at a given reference radius, which for the LHC is conventionally taken to be 17 mm.

The sextupole TF (expressed in Tm/A) is obtained by averaging the ratio of the measured fields and currents (point 2 of the measurement procedure) of both correctors:

$$TF = \frac{L_1}{2}(mean_t \frac{B(t)_{ap.1,3}}{I(t)} + mean_t \frac{B(t)_{ap.2,3}}{I(t)}) \tag{6.2}$$

where L_1 is the length of the super segment used for sextupole magnets measurement (Fig. 6.2).

Finally, by combining (6.1) and (6.2), the current for dipole harmonic compensation turns out to be:

$$I_{MCS}(t) = -\frac{B(t)_3}{TF} \tag{6.3}$$

In Fig. 6.3, the current curve for the corrector magnets computed through (6.3) is shown. The mean value of the TF (6.2) was $9.55 \cdot 10^{-5}$ Tm/A, consistent with the previous measurements of the sextupoles installed in the machine [20]. A standard deviation of 3 $\mu Tm/A$ proved also a satisfying TF linearity.

Harmonic compensation results

The preliminary results reported in this section aim at highlighting the capability of the new setup of attaining at the first iteration a compensation level of the integral sextupole

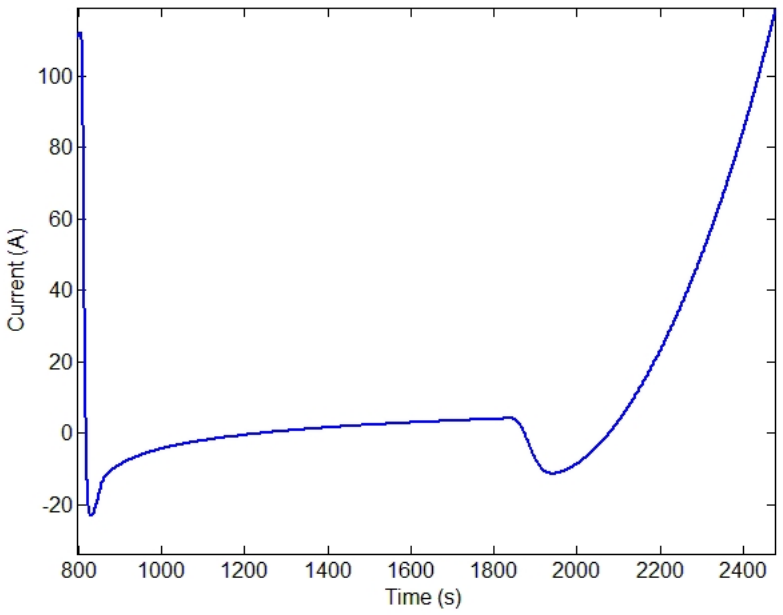


Figure 6.3: Computed MSCs powering current cycle for sextupole compensation.

harmonic very close to previous campaign [20].

The desired sextupole excitation cycle had to be approximated by interpolating the current curve of Fig. 6.3 with linear segments, owing to a limitation of the power converter control system that is going to be updated. The harmonic measurements were performed once every second, which is still well below the theoretical bandwidth of the instrumentation albeit 20 times faster than what was done during the series tests.

In Fig. 6.4, the integral b_3 in units measured during the reference measurement of the LHC cycle in the aperture 1 of the dipole *MB2425* is shown. In Fig. 6.4 the integral b_3 during the harmonic compensation measurement, with the linear interpolated current supplying the MCSs, is also shown. A more detailed view of the residual b_3 with compensation is provided in Fig. 6.5. A tolerance of 0.6 *units*, roughly corresponding to a reduction of a factor 3 of the snapback swing, was achieved already at the first iteration. The decay and snapback transient is detected with unprecedented detail, in particular considering the amplitude of the peak. The increased resolution is highlighted in the comparison with the results of the standard measurement equipment shown in Fig. 6.6, for an acquisition

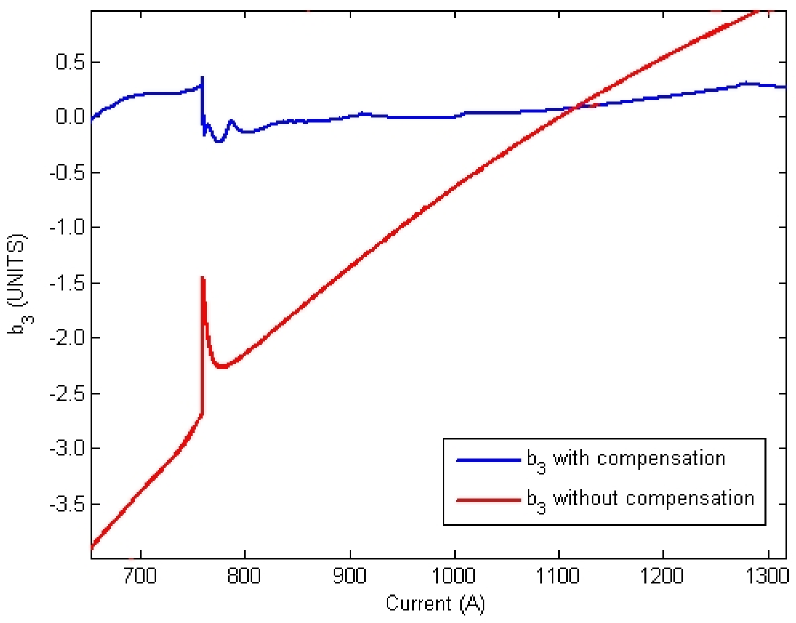


Figure 6.4: Integral b_3 component vs. I with and without compensation, in the dipole magnet *MB2524* during an LHC cycle.

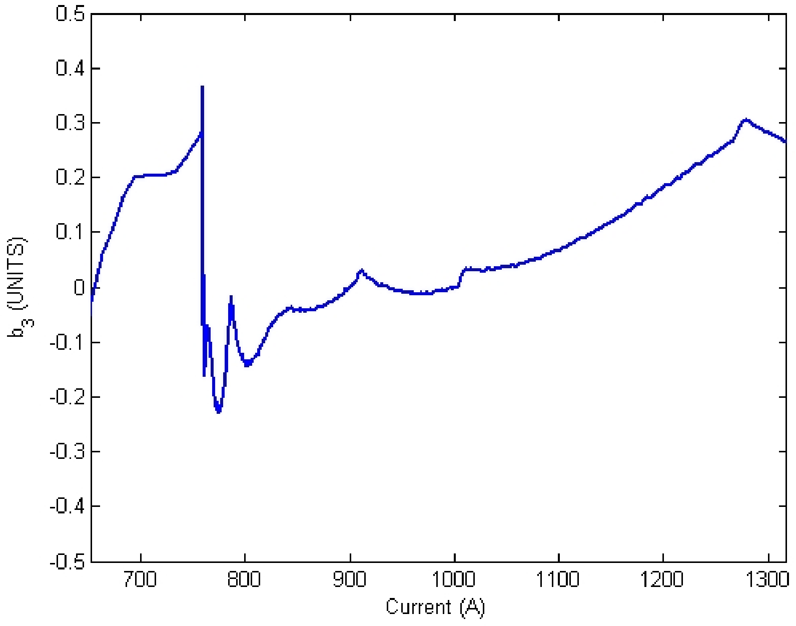


Figure 6.5: Residual integral b_3 component vs. I with compensation, in the dipole magnet *MB2524* during an LHC cycle.

on a single segment with the new acquisition system running at maximum speed (8 *rps*).

Considerably better results are expected from the upcoming test campaign with the

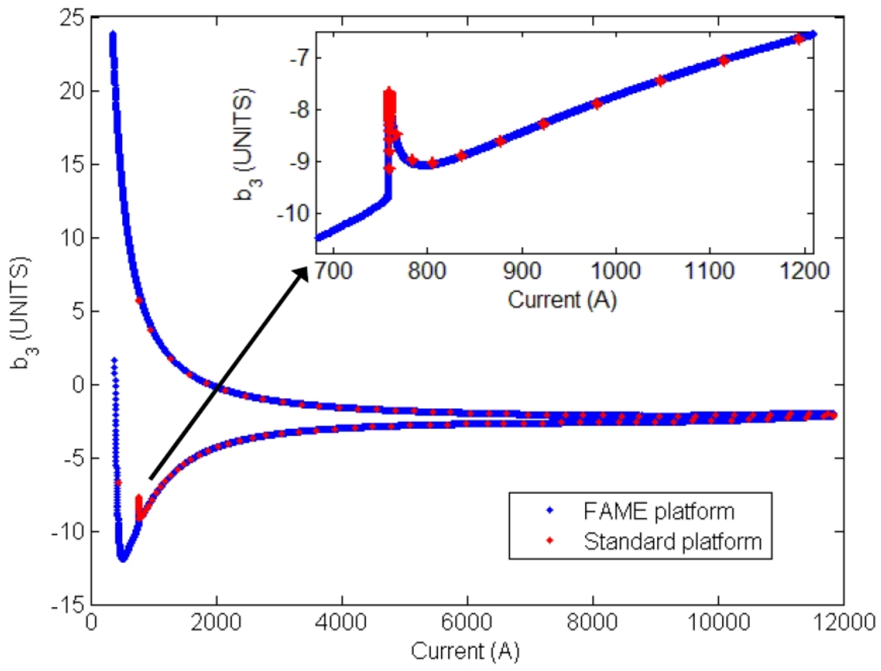


Figure 6.6: Estimation of the sextupole with the old and the new acquisition system.

updated power supply control system, which will provide much tighter synchronization of the two current sources and the possibility to overcome the limitations of the linear approximation of the compensation current cycle.

6.1.2 Permeability measurement

Measurements of the materials' magnetic permeability are of main interest in order to exploit the properties of innovative materials to improve accelerator technologies. In particular, for the LHC it is important to characterize the magnetic properties of the material used for the magnets' yokes, a laminated low-carbon steel.

In the practice, toroidal specimens for magnetic permeability measurements of soft materials are used in order to avoid the issues regarding testing of bars or strips samples [21]. As a matter of fact, a ring-shaped specimen avoids the problems concerning end effects and gaps or joints in the magnetic circuit, and allows the accurate computation of the

mean magnetizing force from a measurement of the magnetizing current, the dimensions of the specimen, and the number of turns in the magnetizing windings. They are therefore nearest to the ideal case when considering the testing principle, even if they are not particularly suitable when end usage aspects of certain specimen are considered.
A split-coil permeameter was built and used for magnetic property characterization at CERN (Fig. 6.7). The permeameter consists of three toroidal windings, which can be

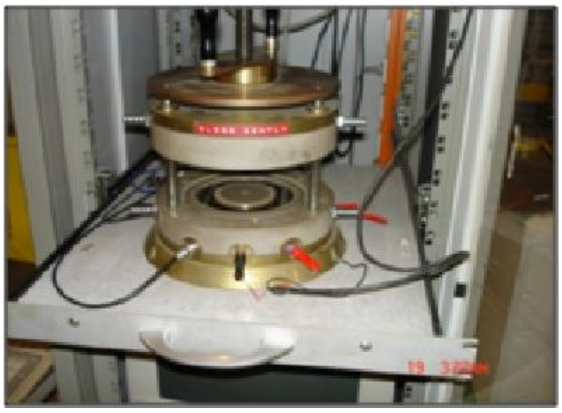


Figure 6.7: Split-coil permeameter.

opened for placing the sample. The two outer coils form the 180 turn excitation winding and the inner 90 turn coil forms the flux measurement winding. The maximum excitation current, passing through the coil, is limited to 40 A to avoid overheating, then the maximum magnetizing field is approximately 300 $Oersted$ or 24000 A/m at room temperature. Acquisition systems developed in the past for the aforementioned permeameter provide low sample rate of the hysteresis curve, as well as a low level of flexibility in the measurement definition. Moreover, an automatic measurement station with acquisition and control software has never been available, therefore this application represents a good test bed for checking the flexibility of FFMM.
The system exploits the state-of-the-art performance of FFMM and FDI in order to improve the accuracy of the whole measurement and to increase also the application domain of such tools.

Theoretical background

The principles of permeability measurements are here recalled for the particular case of tests on ring-shaped specimen by means of a ballistic galvanometer, when the material is subject to a particular steady state (magnetostatic) or is changed from one magnetostatic condition to another.

Points on the curve of first magnetization are measured by bringing a magnetic field (H) to bear on the sample. Switching the field to the opposite direction causes a change of the flux density, which equals twice the flux density (B). Repetition of the measurement with a gradual increase of the field will produce a set of values (B, H) determining the curve.

In order to properly retrieve the value of the average magnetic induction, a correction has to be used since the search coil detects also the flux contribution from the applied magnetic field outside the bulk of the sample. For this reason, two measurements are carried out and combined, the former without the specimen, the latter with the sample inside the permeameter. Subsequently the values of B and H can be calculated as [21]:

$$B = k_2\phi - k_3 I \tag{6.4}$$

$$H = \frac{N_1 I}{2\pi r_0} \tag{6.5}$$

with

$$\begin{aligned} k_1 &= \mu_0 \frac{H}{I} \\ k_2 &= \frac{1}{2N_2 S_s} \\ k_3 &= k_2 \frac{\phi_0}{I} - k_1 \\ 2\pi r_0 &= 2\pi \frac{r_{ext} - r_{int}}{\ln r_{ext} - \ln r_{int}} \end{aligned}$$

where N_1 is the number of excitation windings, N_2 the number of windings of the search-coil, r_{int} and r_{ext} the inner and outer radius of the sample, ϕ and ϕ_0 the integrated signal of the permeameter with and without sample, respectively, $2\pi r_0$ the sample average magnetic length, S_s the sample section. The section S_s might be difficult to estimate, especially when laminated samples are used. In these cases, by knowing the density of the material and by measuring its mass, it is possible to compute the sample volume V and hence S_s as $\frac{r_{int}+r_{ext}}{V}$.

Measurement procedure

The formulas presented above highlight that the estimation of the magnetic permeability requires two measurements to be carried out, one with and one without sample inside the split coils. Moreover, a preliminary demagnetization cycle is needed to bring the sample under test into a virgin state. The whole measurement procedure can be summarized as follows:

1. Measurement of the average magnetic induction without the specimen to retrieve the correction factor;
2. Demagnetization cycle;
3. Measurement of the average magnetic induction with the specimen inside the permeameter.

The demagnetization procedure is carried out by feeding the excitation windings of the permeameter with several current plateaus. The first current plateau (40 A) generates a magnetization field sufficient to bring the sample into a saturation state. Subsequently, the current is decreased from 40 A down to 1 mA, with three ranges of attenuation factors according to a geometric progression. From 40 A down to 0.2 A each plateau equals the previous divided by 1.5, then by 1.2 down to 85 mA, and finally by 1.1 to 1 mA. Each plateau lasts 4 s.
Each measurement cycle, with and without the sample, is carried out by powering the excitation windings with a current cycle made of increasing plateaus of opposite signs, linked by ramps of fixed slope (1.5 A/s). The voltage signal induced on the search coil is integrated to obtain the variation of the linked flux. The current is measured via a feedback signal of the Voltage Controlled Power Converter, synchronously with the flux by exploiting a common trigger signal.

Measurement station architecture

In Fig. 6.8, the architecture of the bench for magnetic permeability measurement is shown. A PC hosting the measurement application produced by FFMM is connected to a NI data acquisition board [121], in order to control the Voltage Controlled Power Supply of the excitation coil of the permeameter by the analog output. The PC controls also a PXI

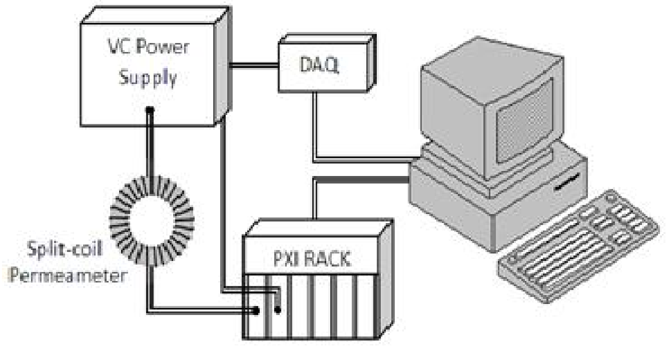


Figure 6.8: Architecture of the new permeability measurement bench.

crate containing (*i*) two FDIs (Fast Digital Integrator), configured for current acquisition and voltage integration, respectively; (*ii*) a board developed at CERN generating pulses used to trigger synchronously the acquisition of the FDIs.

Experimental results

The software application for magnetic permeability measurement is obtained from FFMM through a formal description provided in a user script. Besides the hardware synchronization of the two FDIs by means of trigger pulses generated by the Encoder Board, a software synchronization of the devices is handled by the FFMM Synchronizer (chapter 3). In particular, suitable constructs are used to schedule the execution of the following actions without worrying about the time synchronization of parallel or series tasks:

1. demagnetization of the specimen;
2. start acquisition of flux and current;
3. start generation of one cycle of the signal controlling the power converter;

4. wait for the completion of the present current cycle;
5. stop the acquisition of the flux;
6. start the generation of the next current cycle and go to 3) or, if the maximum value of current is reached, stop the acquisition;
7. convert the data obtained from the FDIs to a suitable format.

An excerpt of the high-level user script is provided in Fig. 6.9. The specimen is magnetized gradually by using a current waveform made by a series of linear ramps and plateaux with an exponential increasing amplitude. The current cycle referred at point 3 is composed by an initial plateau, a linear ramp with constant ramp rate, and a final plateau.

A test was performed on a laminated soft steel sample. As first step, a current cycle was defined and generated through the DAQ board. The current and the flux without the sample were acquired. In Fig. 6.10, an example of current cycle and field H computed from it are shown.

Subsequently, the current cycle was repeated with the sample inside the permeameter. By combining the results of the two acquisitions, as previously explained, the first magnetization curve of the sample material was obtained (Fig. 6.11). Finally, an estimation of the sample relative permeability was obtained from the points of this curve as ratio of the fields B and H (Fig. 6.12).

6.2 Discussion

The Flexible Framework for Magnetic Measurements was employed successfully on the field to produce the software applications required by the current needs at the CERN test facilities. In particular, the framework proved its effectiveness in developing software for measurements with very different requirements, based both on rotating and fixed coils. The former technique was employed for the estimation and compensation of the sextupolar component of the field generated by a superconducting LHC dipole. The latter technique was used to estimate the permeability of a soft steel sample.

```
BEGIN_SCRIPT main_Permeability_Measurement:

   //*************************************/
   //Variable declaration
   //*************************************/
   DEF_VAR numberOf_FDI              AS int =2;
   DEF_VAR surceStop                 AS int =1;
   DEF_ARRAY Cluster_slot            OF int   [2]={14,15};
   DEF_ARRAY Cluster_bus             OF int   [2]={4,4};
   DEF_VAR Cluster_abs_gain          AS float =1.0;
   DEF_VAR Cluster_comp_gain         AS float =10.0;
   [...]

   //*************************************/
   // Device Definition
   //*************************************/
   DEF FDI_CLUSTER:    FDI_Cluster_1  WITH (numberOf_FDI  );
   DEF DAQMX:          NI_Daq     WITH ( "1", "2", "NI") ;

   //*************************************/
   // Device Configuration
   //*************************************/
   CFG FDI_CLUSTER:   FDI_Cluster_1   WITH ( Cluster_bus , Cluster_slot ) ;
   CFG DAQMX:         NI_Daq      WITH ( Daq_channel_name, Daq_task_name, Daq_channel,  Daq_timeOut, Daq_generatioMode  ) ;

   //*************************************/
   // Measurement Task Definition
   //**************************************

   //-----------------------------------
   BEGIN_MTASK Device_setting:
   //-----------------------------------
   [...]
   END_MTASK
   //-----------------------------------
   BEGIN_MTASK Demagnetization_Procedure:
   //-----------------------------------
   [...]
   END_MTASK
   //-----------------------------------
   BEGIN_MTASK Flux_Measurement:
   //-----------------------------------
      PRINT "Start Flux_Measurement" ;

      USE FDI_CLUSTER:    FDI_Cluster_1;
      CMD FDI_CLUSTER: Acquisition (FDI_Cluster_1, path_name,AcquisitionBufferSize, 2);

      TRIG_EVENT start_ramp ;

      CMD FDI_CLUSTER: Wait_Acquisition (FDI_Cluster_1);

   END_MTASK
   //-----------------------------------
   BEGIN_MTASK Begin_Measurement_Procedure:
   //-----------------------------------
   [...]
   END_MTASK
   //-----------------------------------
   BEGIN_MTASK Set_Next_Measurement:
   //-----------------------------------
   [...]
   END_MTASK
   //-----------------------------------
   BEGIN_MTASK Conversion:
   //-----------------------------------
   [...]
   END_MTASK

   ADD_TASK Device_setting;
   ADD_TASK Flux_Measurement;
   ADD_TASK_AFTER_EVENT start_ramp Begin_Measurement_Procedure;
   ADD_TASK_AFTER_TASK Begin_Measurement_Procedure Set_Next_Measurement;
   ADD_TASK_AFTER_TASK Flux_Measurement Conversion;

END_SCRIPT
```

Figure 6.9: DSL script for permeability measurement.

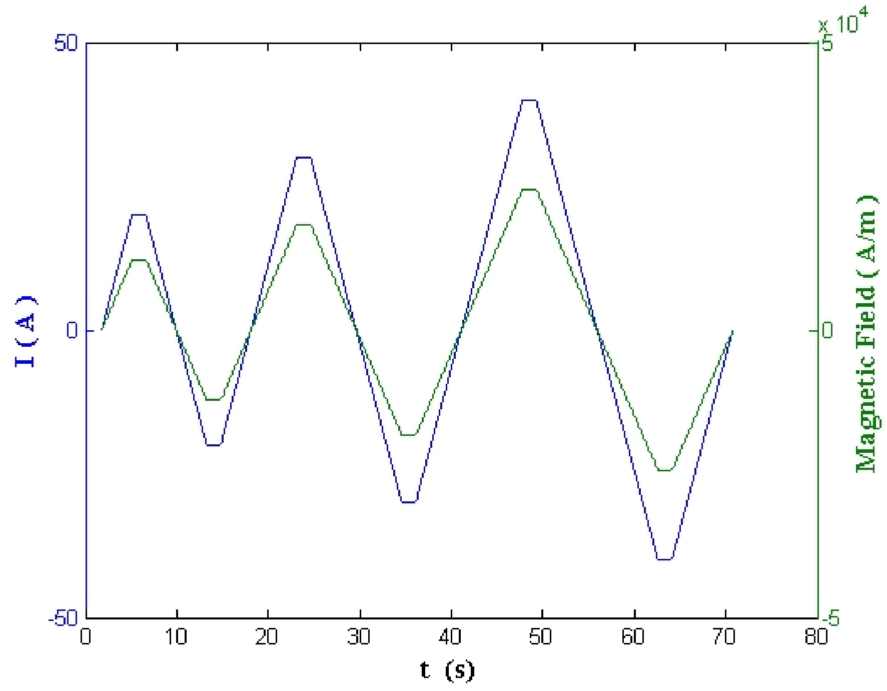


Figure 6.10: Measured current and computed magnetic field without sample.

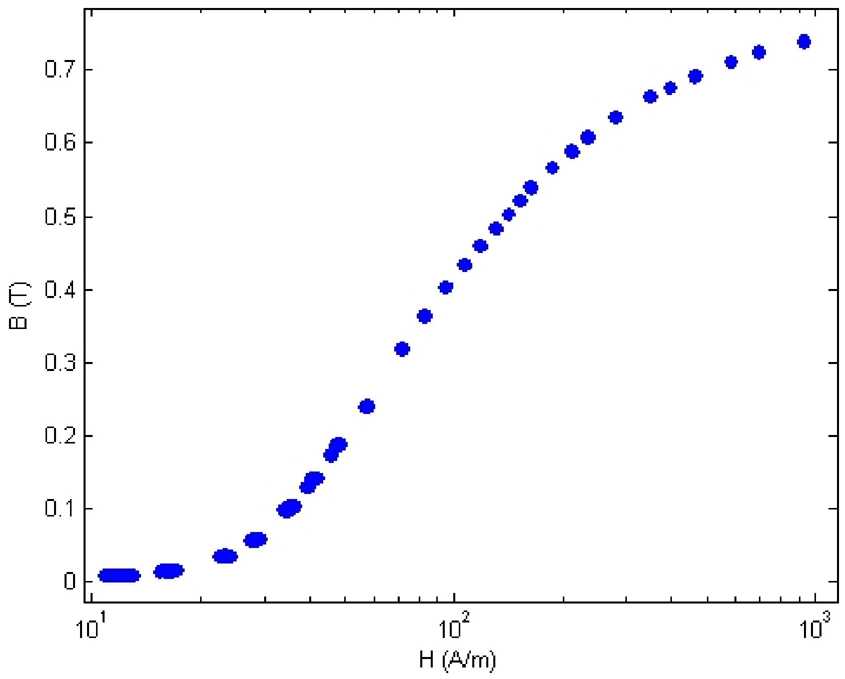


Figure 6.11: First magnetization curve of the soft steel sample.

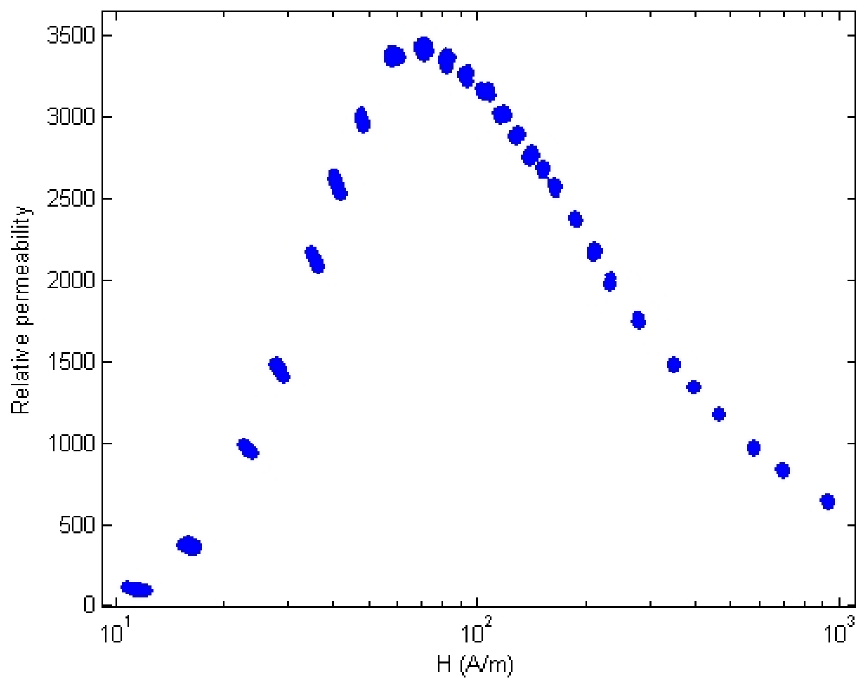


Figure 6.12: Relative permeability of the soft steel sample.

The results of harmonic compensation and permeability measurement were presented, highlighting the resolution improvement attained in the harmonic estimation and the capability of FFMM of producing quickly and with a limited effort the acquisition and control software for both applications, in particular for the split-coil permeameter for which an automatic test station had never been developed before.

Chapter 7

Data analysis algorithms' validation

This chapter presents the validation on the field of the algorithms for data reduction (section 3.4.1) and harmonics resolution enhancement (section 3.4.2), already characterized in simulation (chapter 4).
The aim of the validation is to prove on the field the effectiveness of their design principles and assess their performance in actual working conditions. To this purpose, measurements through rotating coils were carried out at the CERN magnet test facility SM18, both on superconducting and on resistive dipoles.
In the following, the results of the validation campaign are presented and discussed.

7.1 Algorithm for data compression

The experimental proof of the principle and the performance assessment of of the algorithms for data reduction (section 3.4.1) were carried out at the magnet test facility SM18 of CERN by exploiting two demonstrators. Tests were carried out on dipole magnets with the rotating coil method (section 1.1), one of most accurate techniques for magnet testing.

The aim to be achieved is to reduce the data size and simultaneously to be able of reconstructing the field with an acceptable approximation. In the most demanding applications, this means estimating the main field harmonic, in module and phase, with an uncertainty lower than $\pm 10 \div 100\ \mu T$, and the higher harmonics with an uncertainty of few ppm of the main component.
Methods for data reduction through transformation were proposed [122]: data are com-

pressed through a compact representation in a transformed domain, with a simultaneous noise suppression. The proposed algorithm exploits an *fft* transformation, and could be easily adapted to include such a mechanism for the rotating coils application. However, data reduction would be achieved only by releasing the flux harmonics at each fixed number of time samples, typically at every coil turn. Therefore, the instantaneous behaviour of the original signal can not be reconstructed and the resulting data can not used for an even slightly different analysis. For this reason, a more general solution based on reducing and storing raw data in time domain was preferred.

Rotating coil measurements have some peculiar features useful for customizing the proposed algorithm implementation and improve its performance. In particular, the main field component and its first m harmonics have to be found, thus inside the *adaptive tracking sampler*, the tracking is tuned to detect the main component frequency f_1, and the sampling rate is updated in order to extend the observable spectrum up to mf_1. Moreover, field harmonics have a spatial period along the circumference described by the coils during the measurement. The harmonics time period is therefore a function of their spatial period and of the motor rotation speed. Formally, for the m^{th} harmonic:

$$f_m = m\omega/(2\pi) \tag{7.1}$$

where ω is the angular speed in rad/s. On the other hand, as explained before, the flux sampling frequency is the trigger frequency, obtained by multiplying the number of points per turn of the angular trigger by the motor rotation speed:

$$f_s = N\omega/(2\pi) \tag{7.2}$$

where N is the number of flux samples acquired per turn. In other words, both harmonic frequencies and sampling frequency scale according to the same proportionality law with the coils shaft angular speed. Their ratio is therefore fixed by the number of samples per turn for any motor speed. In this case, according to the Nyquist criterion, the minimum number of points per turn required for the m^{th} harmonic to be resolved can be determined easily:

$$f_s \geq cmf_1, \text{ with } c \geq 2 \tag{7.3}$$

Therefore, the algorithm capability of approaching the theoretical optimal value can be checked.

Consequently, the *adaptive tracking sampler* operates in the angular domain, basically by adapting the number of points per turn to the signal characteristics (Fig. 7.1). The signal

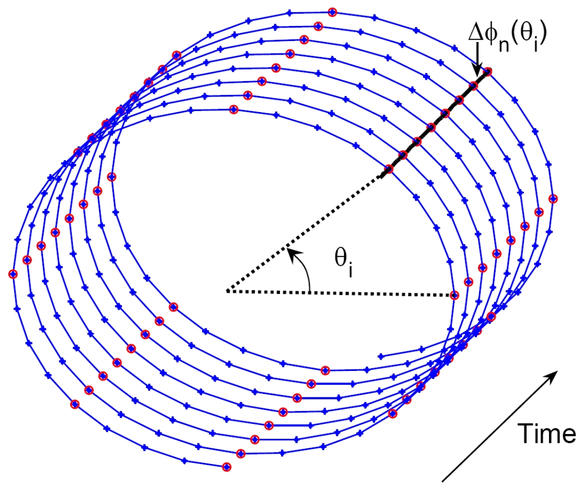


Figure 7.1: Combined approach to data reduction in rotating coils measurements: original flux variations (dots) and flux variations reduced by means of *adaptive tracking sampler* (circles) as function of angular position and time.

under analysis represents the flux over subsequent angular sectors, with width depending on the flux sampling frequency. Therefore, the sampling frequency adaptation implies a variation of the angular sectors where the flux increments are computed. Each of these values is obtained by integrating the voltage samples coming from a fixed rate ADC, thus the adaptation of the flux sampling frequency results in different OSR of the flux signal [11].

Then, the signal features allow the *noise-cancelling compressor* to be applied. The rotating coils-based technique typically produces a sinusoidal output. As shown in Fig. 7.1, if the series $\Delta\varphi_n(\theta_i)$ are considered as formed by flux samples corresponding to a given angular position (θ_i) and acquired for increasing time (expressed as the current turn number n), when the magnet is supplied with a constant or ramped current, they have a nearly linear pattern since they represent the flux variation on the same angular

sector in different time instants. For each point along the circle, there will be an almost linear time series $\Delta\varphi_n(\theta_i)$ to which the *noise-cancelling compressor* can be applied with satisfying results.
The combined reduction procedure as a whole can be summarized:

1. the *adaptive tracking sampler* is let run in order to reduce to its optimal value, compatibly with the constraints, the number of samples acquired on every single turn in correspondence to the angular positions θ_i;

2. for each one of these θ_i, a quasi-linear series $\Delta\varphi_n(\theta_i)$ is obtained, containing a point for each coil turn, and the *noise-cancelling compressor* is applied.

The signal is therefore reduced twice, in angular and time domains, thus increasing the overall achievable compression. This combined approach is capable of attaining a remarkable tradeoff between reduction ratio and fidelity of the reconstructed signal to the original one, since its two steps operate subsequent compressions by means of mechanisms fitting the signal features in their respective application domains.
Finally, from a computational point of view it has to be noted that, since every series $\Delta\varphi_n(\theta_i)$ has one point for each coil turn, at the maximum allowed rotation speed (8 rps) each of them requires to process only 8 S/s, thus keeping the additional computing power required to the measurement system within reasonable limits.
A twofold campaign of on-field tests was carried out on the proposed algorithm mainly aimed at (i) the performance characterization, and (ii) the data reduction validation.

7.1.1 Performance characterization

A test bench aimed at testing on the field the proposed algorithm was assembled in the CERN test facility SM18, by means of:

1. a motor controller MAXON EPOS 24, accessible through RS232, for handling the motor turning the coil inside a superconducting magnet at a constant rotation rate;

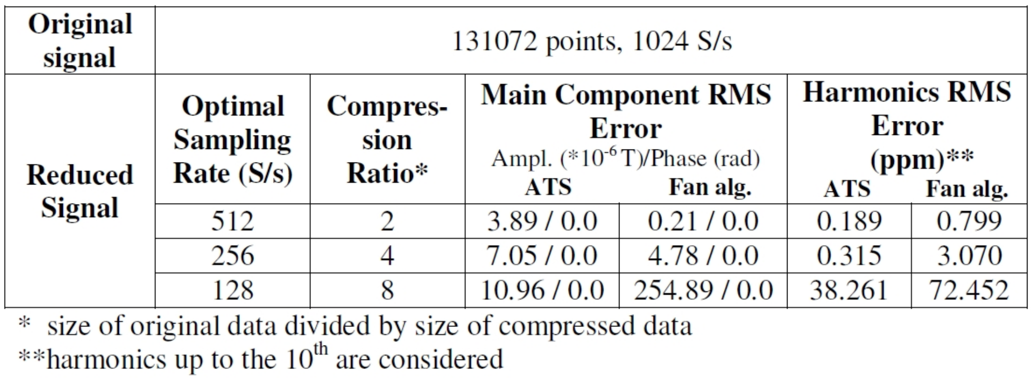
2. a Fast Digital Integrator (FDI), configured for the coil signal acquisition and numerical integration [33];

3. an encoder board for managing the encoder pulses and feeding the trigger input of the FDI;

4. a superconducting magnet at cryogenic temperature (1.9 K) used as unit under test, supplied with a current of 1500 A to generate the magnetic field;

5. a software developed by means of FFMM.

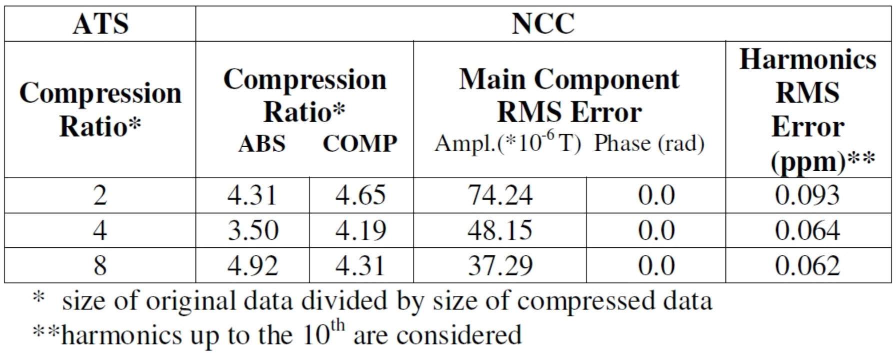
The proposed algorithm was characterized on the field for (i) performance analysis, by varying its settings, and (ii) improvement assessment, in comparison to state-of-the-art algorithms.
As far as the *performance analysis* is concerned, the *adaptive tracking sampler* was executed off line in order to find the optimal sampling rate for carrying out the flux measurement, and the signal was consequently reduced and saved. Afterwards, the multipole expansion of the magnetic field was determined by means of the CERN standard analysis process [18]. The procedure was repeated for different settings of the algorithm, corresponding to different optimal sampling rates and consequently to different compression ratios.
As an example, the results with the Fast Digital Integrator acquiring 128 points per turn from a coil rotating at a speed of 8 rps are shown in Tab. 7.1, in reference to the original signal. Amplitude and phase of the main component of the magnetic field were considered, along with harmonics of the multipole expansion [23] up to order 10. The latter are normalized with respect to the main field component and multiplied by a factor 10^6 (i.e. expressed in ppm). These results highlight how the algorithm is capable of achieving a remarkable trade off between reduction ratio and fidelity of the reconstructed signal.
As far as the *improvement assessment* is concerned, the reduction of the same sinusoidal measurement data by means of the *adaptive tracking sampler* and a classic reduction method, the Fan algorithm [75], were compared. In Tab. 7.1, the RMS error on the multipole expansion obtained from both the algorithms, are reported for different compression ratios. For similar ratios, the comparison shows a remarkable reduction of the error when the *adaptive tracking sampler* is employed. In particular, with a compression



Original signal	131072 points, 1024 S/s					
Reduced Signal	Optimal Sampling Rate (S/s)	Compres-sion Ratio*	Main Component RMS Error Ampl. (*10^{-6} T)/Phase (rad)		Harmonics RMS Error (ppm)**	
			ATS	Fan alg.	ATS	Fan alg.
	512	2	3.89 / 0.0	0.21 / 0.0	0.189	0.799
	256	4	7.05 / 0.0	4.78 / 0.0	0.315	3.070
	128	8	10.96 / 0.0	254.89 / 0.0	38.261	72.452

* size of original data divided by size of compressed data
**harmonics up to the 10th are considered

Table 7.1: Compression Ratio and RMS Error for different settings of the *adaptive tracking sampler* (ATS) on sinusoidal data of an LHC superconducting dipole.



ATS	NCC				
Compression Ratio*	Compression Ratio*		Main Component RMS Error		Harmonics RMS Error (ppm)**
	ABS	COMP	Ampl.(*10^{-6} T)	Phase (rad)	
2	4.31	4.65	74.24	0.0	0.093
4	3.50	4.19	48.15	0.0	0.064
8	4.92	4.31	37.29	0.0	0.062

* size of original data divided by size of compressed data
**harmonics up to the 10th are considered

Table 7.2: Compression Ratio and RMS Error of the *noise-canceller compressor* (NCC) run after the *adaptive tracking sampler* (ATS) on linear data of an LHC superconducting dipole.

ratio of 4, the proposed adapting sampling algorithm is still able to provide an estimation of the field harmonics within an RMS error of a few tenths of *ppm*.

The *noise-cancelling compressor*, according to combined approach explained before, was applied to the series $\Delta\varphi_n(\theta_i)$ obtained after reducing the data by means of the adaptive tracking sampler. The results achieved after the second step of reduction are reported in Tab. 7.2, in terms of compression ratio and RMS error on the computed harmonics. The results highlight, for similar compression ratio, the significant performance improvement of the *noise-cancelling compressor* if applied to a linear signal with respect to the reduction of a sine wave reported in Tab. 7.1. The value of the tolerance ε was chosen on the basis of the noise level of the flux signals. Obviously different values have to be set for the absolute and compensated signal. In particular, the noise level estimated on the compensated signal is two orders of magnitude lower than on the absolute signal ($\sim 10^{-7}$ and $\sim 10^{-5}\ Vs$, respectively), according to typical values observed on these signals.

7.1.2 Data reduction validation

The data reduction algorithm was subsequently validated on the same calibration bench with a reference resistive dipole magnet at room temperature, used as *unit under test*, supplied with a current up to 200 A (Fig. 7.2).

First, the reduction was performed during an acquisition with the magnet supplied by

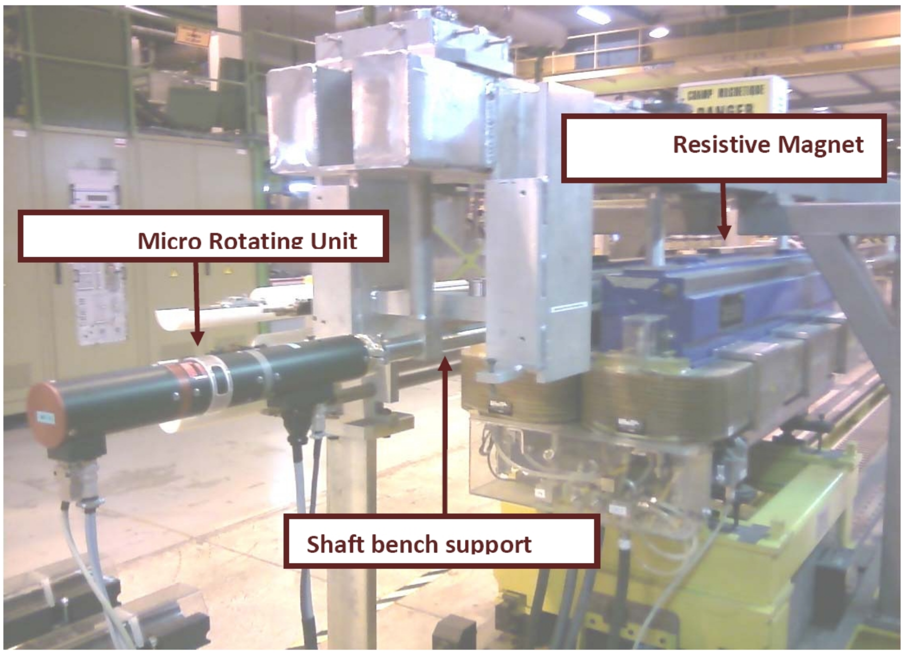
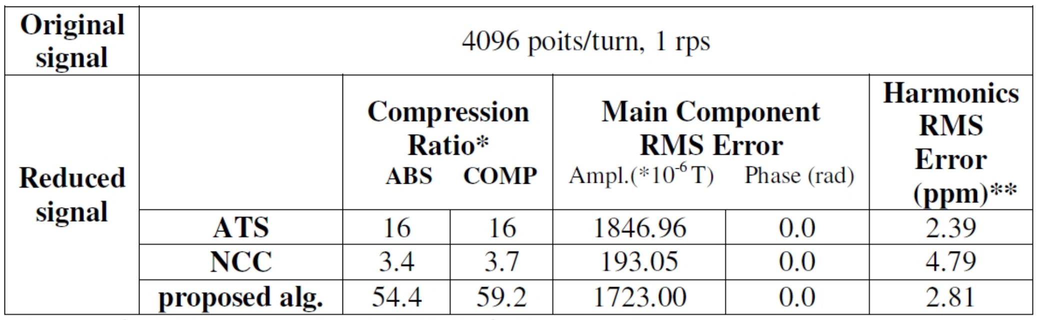


Figure 7.2: The reference dipole calibration bench at CERN.

a constant current, with the aim (i) of determining the noise levels of the absolute and compensated signals in order to tune the tolerance ε, and (ii) of testing the algorithm in stationary conditions. The results are shown in Tab. 7.3. The values of the parameter ε were chosen according to the noise level of the flux signal. The magnet was supplied with a current of 200 A, and 4096 points per turn were initially set. The tracking mechanism, used to detect harmonics up to the 15^{th}, reduced the number of points per turn to 256, thus achieving a compression ratio of 16. This value is in accordance with the theoretical



Original signal	4096 poits/turn, 1 rps					
Reduced signal		**Compression Ratio***		**Main Component RMS Error**		**Harmonics RMS Error (ppm)****
		ABS	**COMP**	Ampl.(*10^{-6}T)	Phase (rad)	
	ATS	16	16	1846.96	0.0	2.39
	NCC	3.4	3.7	193.05	0.0	4.79
	proposed alg.	54.4	59.2	1723.00	0.0	2.81

* size of original data divided by size of compressed data

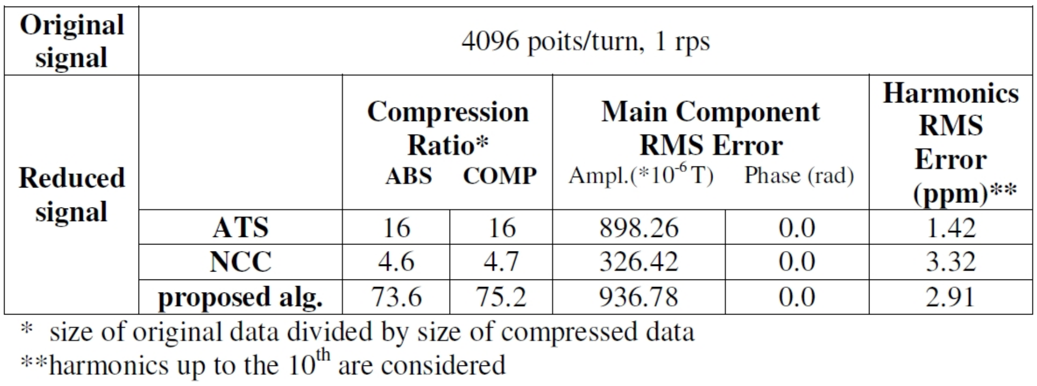
**harmonics up to the 10th are considered

Table 7.3: Compression Ratio and RMS Error for different settings of the algorithm on data of a resistive reference dipole at constant current (200 A).

value derived by considering eq. 7.3 with $m = 15$ and $c = 10$. In fact, this yields:

$$\text{Number of points per turn} = f_s/f_1 \geq cm = 150 \tag{7.4}$$

Since the number of points per turn is to be a power of 2, its minimum value is 256. Subsequently, the *noise-canceller compressor* further reduces the obtained data with a compression ratio of 3.4 and 3.7 for the absolute and compensated signals, respectively. As a whole, the proposed technique proves therefore to be capable of achieving the remarkable compression ratios of 54.4 and 59.2 for the absolute and compensated signal, introducing an acceptable error on the main component and on the harmonics of the resulting signals. Afterwards, the compression was performed on the data acquired during an acquisition with the current in the magnet ramping at a rate of 10 A/s. The Fan algorithm was executed with the values of ε determined before. Tab. 7.4 shows the results obtained in these measurement conditions. The figures in table highlight the good capabilities of the proposed approach also in non-stationary conditions. In particular, in this case, a higher compression ratio is obtained with a smaller approximation error. The reason for that could be that, when the current is low at the beginning of the ramp, the signal is significantly affected by the noise. The results therefore highlights that the algorithm is capable of filtering most of the noise, thus achieving high compression without increasing the related error.



Original signal	4096 poits/turn, 1 rps					
Reduced signal		**Compression Ratio***		**Main Component RMS Error**		**Harmonics RMS Error (ppm)****
		ABS	COMP	Ampl.(*10^{-6}T)	Phase (rad)	
	ATS	16	16	898.26	0.0	1.42
	NCC	4.6	4.7	326.42	0.0	3.32
	proposed alg.	73.6	75.2	936.78	0.0	2.91

* size of original data divided by size of compressed data
**harmonics up to the 10th are considered

Table 7.4: Compression Ratio and RMS Error for different settings of the algorithm on data of a resistive reference dipole at variable current (ramp from 15 to 200 A at 10 A/s).

7.2 Algorithm for harmonic resolution enhancement

The algorithm for harmonics resolution enhancement (section 3.4.2) has already proven its effectiveness in simulation (section 4.2). Here, some tests are performed on signals of actual acquisitions carried out on an LHC superconducting dipole in non-stationary conditions. In particular, the dipole was supplied with a current varying according to the Parabolic-Exponential-Linear-Parabolic (PELP) profile of an LHC current cycle [76], where the ramp rate of the linear part was set to 10 A/s. The normal and skew harmonic coefficients (section 1.1.1) were obtained in the following measurement conditions: 128 points per turn, motor speed 1 rps, absolute gain 1, compensated gain 50. It is possible to choose a suitable value for the parameter m (section 3.4.2), thus giving the best trade-off between computation burden and accuracy of the estimated harmonics (estimation quality in fact improves for a little m but computational load is smaller for a great m). Here, the harmonics estimation was executed with $m = 4$. Unlike the simulations presented in chapter 4, in this and in the following tests the ideal harmonic coefficients are not available. Therefore only the harmonic coefficients obtained from the experimental flux with and without algorithm are presented and compared.

Fig. 7.3 shows a comparison of the main normal harmonics computed through standard analysis and interpolation algorithm in a linear-parabolic part of the current profile. Their difference over a linear profile with a ramp rate of 10 A/s is reported in Fig. 7.4. The

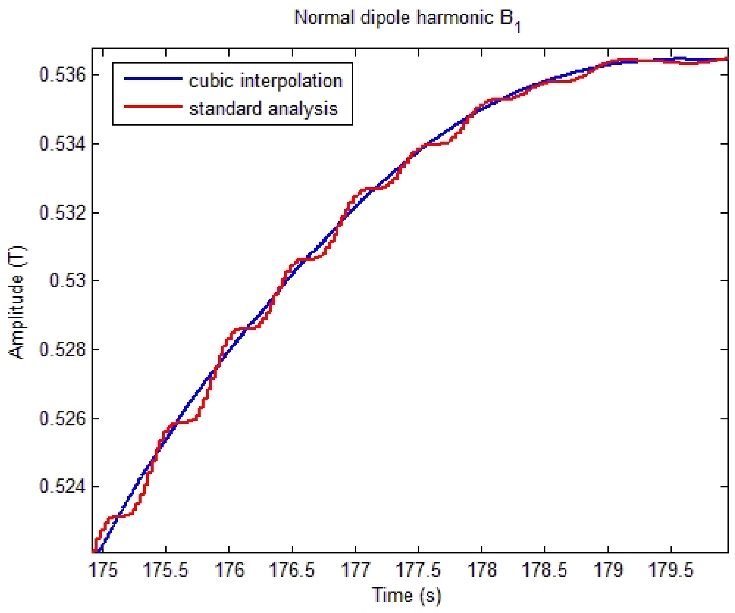


Figure 7.3: Main normal harmonic (B_1) over a linear-parabolic current profile, $m = 4$.

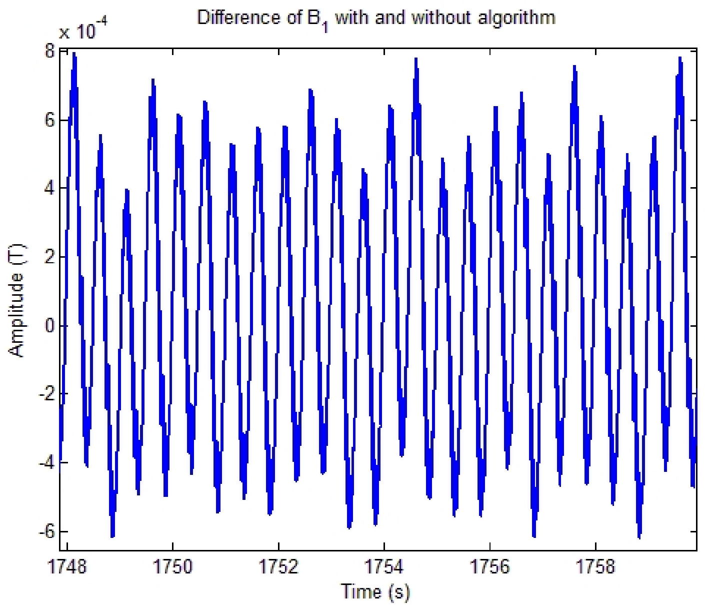


Figure 7.4: Difference of the main normal harmonics (B_1) computed through cubic interpolation and standard analysis, over a linear current profile at 10 A/s, $m = 4$.

difference is significant since it exceeds 10^{-4} T, usually considered as the threshold for many applications based on harmonic coil measurements.

In Fig. 7.5, the resolution improvement obtained in the phase of decay and snapback of

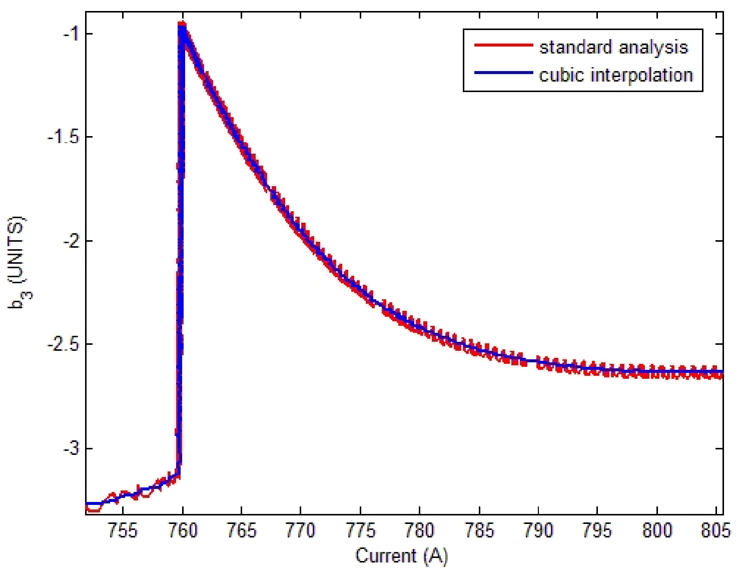


Figure 7.5: Decay and snapback of the normal sextupole harmonic (b_3), $m = 4$.

the normal sextupole harmonic b_3 is shown.

Finally, the main skew harmonic A_1 is reported in Fig. 7.6. The measurement was carried out in a normal reference frame, where A_1 should ideally be 0. In the figure, the systematic deviation from 0 is due to a non perfect alignment of the reference frame to the normal direction. Moreover, the oscillations of the harmonic computed by means of the interpolation algorithm, where the effect of the flux amplitude modulation is not visible, can be interpreted as the results of the mechanical vibrations of the coil shaft during the rotation.

The results presented above highlight the algorithm's capability of improving the harmonic coefficients estimation, enhancing their time resolution in non-stationary conditions.

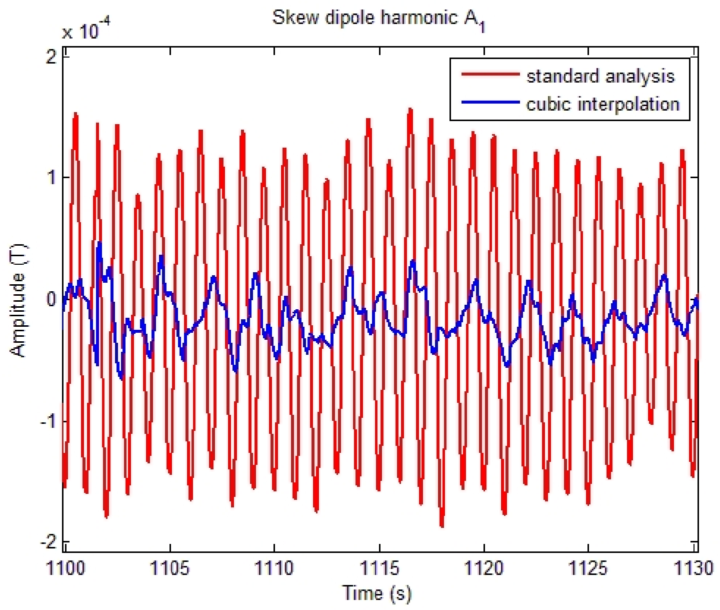


Figure 7.6: Main skew harmonic (A_1) over a linear current profile, $m = 4$.

7.3 Discussion

The experimental validation of the algorithms for data reduction and harmonic resolution enhancement was performed at the CERN test facility SM18 both on a superconducting LHC dipole and on a calibration resistive dipole magnet.

As far as the data reduction is concerned, the results highlight the remarkable tradeoff between fidelity to the original signal and achieved reduction allowed by the proposed two-domain combined approach. The results also highlighted a synergic interaction between the two steps of the proposed reduction procedure: the overall error of the algorithm never exceeds those of each of its subprocedures.

Finally, it is worth pointing out that the absolute and compensated signals previously analyzed have different harmonic contents (typically first harmonic in the absolute and higher order harmonics in the compensated), thus the algorithm's performance could be further improved by means of a separate reduction in the angular domain of such signals.

The algorithm for harmonic resolution enhancement was also validated on the field through a test on a superconducting LHC dipole in non-stationary conditions. The algorithm proved its effectiveness in enhancing the time resolution of the harmonic estimation during an LHC current cycle.

Chapter 8

Flexibility experimental tests

This chapter deals with the flexibility test of software frameworks for measurement applications, and, in particular, of the Flexible Framework for Magnetic Measurements (FFMM). After prototyping, experimental applications, and analysis of code quality, a flexibility characterization of the proposed framework is also needed. As part of the wider work aimed at the characterization of the framework started in chapter 5, the twofold purpose of this chapter is (*i*) to introduce specific metrics suitable for assessing the degree of flexibility achieved by a software framework for measurement applications, (*ii*) to present experimental results for some typical application scenarios of the current release 3.0 of FFMM.

8.1 The generalized evolution cost metric

Classic and contemporary literature in software design recognize the central role of flexibility in software design and implementation. Structured design, modular design, object-oriented design, software architecture, design patterns, and component-based software engineering, among others, seek to maximize flexibility.

During its life cycle, a flexible software system is forced to face variable requirements. As a consequence, the implementation has to be adapted to provide a solution to problems in new application domains. An *evolution step* is defined as the unit of evolution with relation to a particular change in the implementation.

It has been observed that predicting the class of changes is the key to understanding software flexibility. During the phases of design and development of the software, initially the changes that are likely to occur over the lifetime of the product are characterized . Since it is impossible to predict the actual changes, the predictions will be about classes of changes [123].

The notion of evolution step can be used for estimating software flexibility [124]: *a* is more flexible than *b* towards a particular evolution step, if the number of changes required for *a* is smaller than the number of changes required for *b*. Thus, the complexity of an evolution step measures how inflexible the implementation is towards a particular class of changes: the less changes are required, the more flexible it is.

Therefore, it is useful to organize the software so that the items that are most likely to change are confined to a small amount of code, so that if those things do change, only a small amount of code would be affected [123]. In other words, flexibility (measured in terms of the cost of the evolution process) is directly linked to the amount of code affected by the changes required in a particular evolution phase. Thus, a first approximation to measuring the cost of an evolution step ε is given by the evolution cost metric counting the number of modules affected by ε. Under the assumption that the costs of adding, removing, or changing each modular unit commensurate, the *evolution cost metric* can be obtained by calculating the number of modules added, removed, or adjusted as a result of the evolution. This number is obtained by calculating the symmetric set difference between the sets of classes in the old (i_{old}) vs. the adjusted ($i_{adjusted}$) implementations. Formally [124]:

$$C_{Classes}(\varepsilon) = |(Classes(i_{old}) - Classes(i_{adjusted})) \bigcup \bigcup (Classes(i_{adjusted}) - Classes(i_{old}))| \tag{8.1}$$

This evolution cost metric is inadequate in some situations: when the evolution of different modules do not commensurate, when the modules are not implemented yet, and when the programming language does not support classes at all or adds other programming units (such as in the case of AOP). Therefore, the metric is to be accommodated for varying degrees of modular granularity, as well as for varying degrees of information on each module. This leads to the definition of the *generalized evolution cost metric* [124]:

$$C^{\mu}_{Modules}(\varepsilon) = \sum_{m \in \Delta Modules(i_{old}, i_{adjusted})} \mu(m) \tag{8.2}$$

where $\Delta Modules(i_{old}, i_{adjusted})$ is the symmetric set difference between the set of modules in i_{old} and the set of modules in $i_{adjusted}$.
The generalized metric is parameterized by the variables *Modules* and μ:

- *Modules* represents any notion of module that is appropriate for the circumstances, such as class, procedure, method, aspect, and package;

- μ represents any software complexity metric meaningful in relation to a particular module m.

Finally, evolution complexity is a *measure of growth*, not an absolute value, and therefore it does not measure the actual cost of the evolution process but how it grows.

8.2 Experimental results

The proposed approach to flexibility assessment has been applied at CERN in the context of the Flexible Framework for Magnetic Measurements. The platform was designed in order to satisfy the requirements for a wide range of magnetic measurement applications, thus the most probable scenarios to face are the different techniques currently used for testing magnets for accelerators, besides those developed in the future.
The framework is based on OOP and AOP, therefore the modules involved in these scenarios are *methods*, *classes*, and *aspects* (section 3.1.3). In a preliminary analysis phase, the classes of changes due to the different measurement techniques were classified (by increasing flexibility) as: (*i*) adding/modifying software modules implementing the devices, (*ii*) changing the strategies for handling the services provided by the framework (e.g. fault detection, logging, synchronization), and (*iii*) implementing new measurement algorithms. The abovementioned classes of changes involve different users of the framework, namely (*i*) and (*ii*) the developer, and (*iii*) the test engineer.
In the following, some preliminary experimental results of the flexibility assessment are

Increasing flexibility ↓↓↓↓↓	**Class of change**	**User involved**	$C^{CYCLO}_{Modules}$
	Add device	Developer	$\propto$ #methods,events,faults
	Change device interface	Developer	$\propto$ depth inheritance hierarchy
	Change fault detection strategy	Developer	$\propto$ #faults involved
	Change measurement procedure	Test engineer	0

Table 8.1: Generalized evolution cost metric for different classes of changes in FFMM.

illustrated. The tests were carried out at CERN on the release 3.0 of FFMM for different measurement methods. The experimental results are summarized in Tab. 8.1.

The generalized evolution cost metric is obtained by fixing $\mu = CYCLO$ (Cyclomatic Complexity [103], a measure of the number of linearly independent paths through a program's source code and therefore of its logical complexity), thus yielding the metric $C^{CYCLO}_{Modules}$. This metric is used to compare the degree of flexibility of the different classes of changes, and not as an absolute measure of flexibility. A high cyclomatic complexity (>10 [88, 99, 100, 101]) denotes a complex procedure hard to understand, test, and maintain. Therefore, the lower the cyclomatic complexity (and consequently $C^{CYCLO}_{Modules}$), the higher the flexibility.

8.2.1 Adding/modifying a device

When new devices are required by a measurement application, the effort for their implementation cannot be avoided completely. In this case, the flexibility is therefore limited intrinsically. Nevertheless, FFMM is fairly flexible towards this class of changes, because it helps the user effectively in developing the related new components. Namely, it provides services, such as event handling and fault detection (chapter 3), whose infrastructure is accessible easily and whose implementation is customizable with limited effort.

The possible changes at device level can be classified as (i) adding the device into the framework from scratch, and (ii) modifying it to satisfy new requirement when it already exists, by adapting its interface or some method implementation. The cost of adding a new device strongly depends on its size. Formally, rather than through the lines of code (LOC), this cost can be expressed as the sum of the cyclomatic complexity of all its methods, including additional code devoted to events and faults handling, and computed as the average cyclomatic complexity of a software unit (method) multiplied by the number of units implemented. The generalized evolution cost metric results therefore proportional to the number of member functions, events and faults of the new class (Tab. 8.1). The member functions of the class are likely to be more complex than the methods handling events and faults, thus the generalized evolution cost metric usually depends more on the former set of functions. If a class interface has to be modified, for example by adding/removing a method, the change will involve many modules since typically a device is part of a hierarchy of classes in a generalization relationship (chapter 3). The effort to add/remove a method is fixed and determined by its own complexity, thus the growth of the evolution cost metric depends only on the depth of the inheritance hierarchy (Tab. 8.1). In the design phase of FFMM, the maximum depth was kept to a reasonable value (4), so this class of changes requires a limited effort. The evolution cost estimation strongly depends on the device considered.

In order to provide a quantitative example, in the following the driver for the device Encoder Board, developed at CERN and employed in different scenarios typical of the magnetic measurements (see section 1.1), is taken into account. The device is part of the hierarchy of classes (chapter 3). Adding the device requires a considerable programming effort, anyway FFMM provides support in the following ways: it provides libraries implementing communication features on different buses, so that all the required functionalities are already available and accessible through a suitable interface. Furthermore, FFMM already implements and makes available infrastructures for event handling and fault detection. The tasks of exploiting events and improving system fault tolerance are therefore extremely simplified for the user. He just needs to add few small modules to extend the event structure and the fault detection logic. The generalized evolution cost metric, computed as the sum of the total cyclomatic complexity of the modules to be added, has a value of 301 in the particular case considered. This value is provided just as an example, since it strongly depends on the evolution step under analysis. Anyway, to give insight on its meaning, it can be observed that in FFMM an average device has 35 member functions, and an average device member function has $CYCLO = 2$. Therefore, since the complexity of a device lies mainly in its member functions, on average adding a new device costs $35 \cdot 2 = 70$. With respect to this value, the Encoder Board results to be significantly above the average complexity level.

8.2.2 Changing service strategies

FFMM provides many services to help the user employ the framework and enlarge its application domain. The choice of OOP reduces the number of modules affected by possible changes, thus assuring a good level of flexibility. Moreover, some services (chapter 3) were implemented by means of AOP. As an example, here only the fault detection is considered, because it is a fundamental part of all the devices (section 3.1.4). By this solution, the classes of FFMM are oblivious of triggering the execution of specific code in the related aspects providing the services. Classes and aspects are therefore completely decoupled, further increasing software flexibility. Namely, a change of the fault detection strategy typically involves only one module, without affecting in any way the corresponding device. The complexity of such a change can be estimated as the average complexity of a fault handling method multiplied by the number of methods to be modified, and is therefore proportional to the number of faults involved in the change (Tab. 8.1). To provide a quantitative estimation, for the fault detection code specific of the Encoder Board one gets $C^{CYCLO}_{Modules} = 4$, while for fault detection code common to other devices one gets $C^{CYCLO}_{Modules} = 25$, for a total generalized evolution cost of 29.

8.2.3 Implementing new measurement algorithms

Several measurement techniques are currently employed for the test of accelerator magnets, such as fixed and rotating coils, as well as stretched wire [125].
FFMM was designed to reduce drastically the amount of code affected by modification to the measurement procedure. The test engineer interacts with the framework through the User Script mainly, a formal description of the measurement procedure. All the changes required by a new measurement algorithm are focused in the User Script, without af-

fecting any other modules. In this case, the framework provides the highest degree of flexibility, with $C^{CYCLO}_{Modules} = 0$ (Tab. 8.1). This result was proven experimentally by developing the application for permeability measurements described in section 6.1.2 by means of devices already developed and previously employed for the rotating coil benches. It is worth pointing out that the system for permeability measurement was developed at CERN in the 1960s, and since then used by means of a semiautomatic test station [21]. It is therefore remarkable the possibility to develop quickly from scratch the required control and acquisition software through FFMM.

8.3 Discussion

In this chapter, an experimental approach to the software flexibility assessment of measurement frameworks is proposed. In particular, this approach is meant to be applied in the context of FFMM at CERN. FFMM was designed to be flexible, reusable, maintainable, and portable. A complete release of FFMM is available and its effectiveness on the field has already been proved in chapter 6, thus the evaluation of its degree of flexibility completes the more comprehensive phase of software quality assessment, aimed at stating the fulfillment of the challenging project goals.
The flexibility of the system cannot be stated in absolute terms, but only with respect to specified classes of changes, involving different users. The results highlight that the framework achieves increasing degrees of flexibility moving from the programming level to the user script level, and at the same time from the point of view of the developer to that of the test engineer. The highest flexibility is attained for the changes involving the measurement procedure, namely at the level where flexibility was mainly required.

Conclusions

A new software system, the *Flexible Framework for Magnetic Measurements* (FFMM), was designed, developed and validated. Implemented in C++ and based on object-oriented (OOP) and aspect-oriented (AOP) programming, FFMM aims at supporting the user in developing measurement software maximizing its quality, in terms of flexibility, reusability, maintainability and portability, by simultaneously keeping high efficiency levels. Given a set of measurement requirements, formally described by the test engineer in a script and suitable to be satisfied by the available hardware, FFMM allows an effective automatic measurement system to be generated with limited effort and development time. It can be easily configured for a large set of measurement applications, mainly magnetic but also optical, mechanical, etc.
Tests on the field were performed at CERN with different protocols and measuring equipments, including also the new high-performance hardware, namely fast rotating-coil transducers (Micro Rotating Units) and digital integrators (Fast Digital Integrators). FFMM was employed successfully to produce the software applications required by the current needs of the CERN test facilities. In particular, the framework proved its effectiveness in developing software for measurements with very different requirements, based both on rotating and fixed coils. The former technique was employed for the estimation and compensation of the sextupolar component of the field generated by a superconducting LHC dipole. The latter technique was used to estimate the permeability of a sample of the same material (soft steel) used for the LHC magnet yokes. The results highlighted the resolution improvement (up to a factor 100) attained in the harmonic estimation and the capability of FFMM of producing quickly and with a limited effort the acquisition

and control software for both applications.

The framework includes also two algorithms, for data reduction and harmonic time resolution enhancement, respectively. A numerical study was first performed to assess the performance of both the proposed techniques.

A far as the data reduction algorithm is concerned, the results highlighted the better performance of the proposed approach, when compared to other techniques, from the point of view of computational burden, reduction capabilities, and fidelity of the reconstructed signal to the original one. In particular they showed its suitability for application where signals are similar to those typical of rotating coils based magnetic measurements.

As far as the algorithm for harmonic resolution enhancement is concerned, the results obtained in non-stationary conditions for different field profiles and algorithm's settings highlighted the achievement of a significant improvement in the harmonics time resolution when compared to those granted by the CERN standard procedure.

For both algorithms, these results were confirmed on the field through dedicated test campaigns carried out at the CERN test facilities.

A software quality characterization of the object-oriented and aspect-oriented parts of the release 3.0 of FFMM was performed. The characterization of the object-oriented part was carried out with reference to the quality model ISO 9126. The internal quality of FFMM source code was evaluated by means of suitable metrics. Although the results highlighted a good average quality level, improvements are possible to decrease the maximum complexity and exploit more profitably the concepts of object-oriented programming. A supplementary analysis was carried out identify the possible improvements in the object-oriented design. Subsequently, the aspect-oriented component handling the fault detection was analyzed. Aspect-oriented programming extends the object-oriented approach, by adding specific encapsulation of crosscutting concerns. The advantages of using such a software design in the development of a fault detector were verified in the case of a measurement application based on rotating coils for the test of superconducting magnets. The proposed architecture proved to grant a high level of flexibility, maintainability, and reusability, without affecting significantly the run-time performance.

Finally, an experimental approach to the software flexibility assessment of measurement frameworks was proposed and applied in the context of FFMM. A complete release of FFMM was available and its effectiveness on the field had already been proved, thus the evaluation of its degree of flexibility completed the more comprehensive phase of software quality assessment, aimed at stating the fulfillment of the challenging project goals. The flexibility of the system was stated with respect to specified classes of changes, involving different users. The results highlighted that the framework achieves increasing degrees of flexibility moving from the programming level to the user script level, and at the same time from the point of view of the developer to that of the test engineer. The highest flexibility is attained for the changes involving the measurement procedure, namely at the level where it was mainly required.

Bibliography

[1] CERN. Lhc design report. In *CERN 2004 003*, 2004.

[2] L. Bottura and K.N. Henrichsen. Field measurements. CERN Accelerator School Proceedings, September 2004.

[3] S. Amet, L. Bottura, L. Deniau, and L. Walckiers. The multipoles factory: an element of the lhc control. *Applied Superconductivity, IEEE Transactions on*, 12(1):1417–1421, Mar 2002.

[4] N.R. Brooks, L. Bottura, J.G. Perez, O. Dunkel, and L. Walckiers. Estimation of mechanical vibrations of the lhc fast magnetic measurement system. *Applied Superconductivity, IEEE Transactions on*, 18(2):1617–1620, June 2008.

[5] M. Haverkamp, L. Bottura, E. Benedico, S. Sanfilippo, B. ten Haken, and H.H.J. ten Kate. Field decay and snapback measurements using a fast hall plate detector. *Applied Superconductivity, IEEE Transactions on*, 12(1):86–89, Mar 2002.

[6] W.C. Elmore and M.W. Garrett. Measurement of two-dimensional fields, part i: theory. *Review of Scientific Instrument*, 1954.

[7] I.E. Dayton, F.C. Shoemaker, and R.F. Mozley. Measurement of two-dimensional fields, part ii: study of a quadrupole magnet. *Review of Scientific Instruments*, 1954.

[8] J. DiMarco and J. Krzywinsky. Mtf single stretched wire. *Technical report, Fermi National Accelerator Laboratory*, March 1996.

[9] J. DiMarco, H. Glass, M.J. Lamm, P. Schlabach, C. Sylvester, J. C. Tompkins, and J. Krzywinsky. Field alignement in quadrupole magnets for the lhc interaction region. *IEEE Transactions on Applied Superconductivity*, 10(1):127–130, 2000.

[10] L. Bottura, L. Larsson, S. Schloss, M. Schneider, and N. Smirnov. A fast sextupole probe for snapback measurement in the lhc dipoles. *IEEE Transactions on Applied Superconductivity*, 10(1):1435–1438, 2000.

[11] P. Arpaia, V. Inglese, and G. Spiezia. Performance improvement of a dsp-based digital integrator for magnetic measurements at cern. *Instrumentation and Measurement, IEEE Transactions on*, 58(7):2132–2138, July 2009.

[12] P. Arpaia, L. Bottura, M. Buzio, D. Della Ratta, L. Deniau, V. Inglese, G. Spiezia, S. Tiso, and L. Walckiers. A Software Framework for Magnetic Measurements at CERN. In *Proc. of the IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Poland, May 1-3 2007.

[13] Designing next-generation test systems developers guide. http://zone.ni.com/devzone/cda/tut/p/id/3238toc0 .

[14] A. Guerrero, Jj Gras, Jl Nougaret, M. Ludwig, M. Arruat, and S. Jackson. Cern front-end software architecture for accelerator controls. In *Proceedings of ICALEPCS2003, Gyeongiu, Korea*, 2003.

[15] J.M. Nogiec, J. DiMarco, S. Kotelnikov, K. Trombly-Freytag, D. Walbridge, and M. Tartaglia. A configurable component-based software system for magnetic field measurements. *Applied Superconductivity, IEEE Transactions on*, 16(2):1382–1385, June 2006.

[16] http://www.tango-controls.org/ .

[17] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proc. of the 11th European*

Conference on Object-Oriented Programming (ECOOP), Springer-Verlag, volume 1241, pages 220–242, 1997.

[18] L. Bottura and K.N. Henrichsen. Standard analysis procedures for field quality measurement of the lhc magnets - part i: Harmonics. Internal note EDMS 313621, 1997.

[19] International Standard ISO/IEC 9126-1. *Software Engineering - Product Quality - Part 1: Quality Model.* International Organization for Standardization, International Electrotechnical Commission, 2001.

[20] P. Xydi, N. Smmut, R. A. Fernandez, L. Bottura, G. Deferne, M. Lamont, J.Miles, S. Sanfilippo, M. Strrzelczy, and W. Delsolaro. A demonstration experiment for the main field tracking and the sextupole and decapole compensation in the lhc main magnets. In *LHC Project Report 1083*, CERN, Geneva, Switzerland, 2008.

[21] K. N. Henrichsen. Permeameter. In *Proc. 2nd Int. Conf. On Magnet Technology*, Oxford, 1967.

[22] J. Gareyte. Impact of superconductors on lhc design. In *CERN 96-03*, pages 335–346, CERN, Geneva, Switzerland, 1996.

[23] A. K. Jain. Harmonic coils. CERN Accelerator School Proceedings, April 1997.

[24] S. Bidon, J. Billan, F. Fischer, and C. Sanz. New technique of fabrication of search coil for magnetic field measurement by harmonic analysis. In *CERN Internal Note AT-MA 95-117*, CERN, Geneva, Switzerland, 1995.

[25] N. Bloenbergen, E.M. Purcell, and R.V. Pound. Relaxation effects in nuclear magnetic resonance absorption. *Physical Review*, 73, 1948.

[26] E.H. Hall. On a new action of the magnet on electric currents. *American Journal of Mathematics*, 2:287–292, 1879.

[27] G.L. Pearson. A magnetic field strength meter employing the hall effect in germanium. *Review of Scientific Instruments*, 19:263–265, 1948.

[28] L. Madaro, A. Rijllard, R. Saban, L. Walckiers, L. Bottura, and P. Legrand. A vme-based labview system for the magnetic measurements of the lhc prototype dipoles. In *Proc. of EPAC 96*, Barcelona, Spain, 1996.

[29] L. Walckiers. The harmonic coil method. In *CERN Accelerator School on Magnetic Measurments and Alignment*, CERN, Geneva, Switzerland, May 1992.

[30] J. Bosch, P. Molin, M. Mattson, and P. Bengtsson. *Object-oriented frameworks - problems and expectations.* In Building application frameworks: object-oriented foundation of framework design, Eds. Wiley and Sons, 1999.

[31] J. van Gurp and J. Bosch. Design, implementation and evolution of object oriented frameworks: concepts and guidelines. *Software Practice and Experience*, 31:277–300, 2001.

[32] J. Bosch. Design of an object-oriented framework for measurement systems. *In Domain-Specific Application Frameworks, M. Fayad, D. Schmidt, R. Johnson (eds.), John Wiley, ISBN 0-471-33280-1*, pages 177–205, 1999.

[33] P. Arpaia, A. Masi, and G. Spiezia. Digital integrator for fast accurate measurement of magnetic flux by rotating coils. *Instrumentation and Measurement, IEEE Transactions on*, 56(2):216–220, April 2007.

[34] P. Galbraith. Portable digital integrator. In *Internal Technical Note 93-50, AT-MA/PF/fm*, CERN, Geneva, Switzerland, 1993.

[35] C. Evesque. A new challenge in magnet axis transfer. In *Proc. of Int. Magnetic Measurement Workshop IMMW11*, Brookhaven National Laboratory (USA), September 1999.

[36] R. Carcagno, J. DiMarco, S. Kotelnikov, M. Lamm, A. Makulski, V. Maroussov, R. Nehring, J. Nogiec, D. Orris, O. Poukhov, F. Prakoshin, P. Schlabach, J.C.

Tompkins, and G.V. Velev. A fast continuous magnetic field measurement system based on digital signal processor. In *Proc. of 19th Magnet Technology Conference*, Genoa, 18-23 September 2005.

[37] W. Pellico and P. Colestock. Pulsed magnetic field measurement using a ferrite waveguide in aphase bridge circuit. In *Proceedings of the Particle Accelerator Conference*, volume 3.

[38] P. Arpaia, V. Inglese, G. Spiezia, and S. Tiso. Surface-response-based modeling of digitizers: A case study on a fast digital integrator at cern. *Instrumentation and Measurement, IEEE Transactions on*, 58(6):1919–1928, June 2009.

[39] P.C. Ferreira and H. Reymond. Sequence of tests and settings to start a magnetic measurement on mmp 6.5.0. In *Internal note EDMS num. 399822*, CERN, Geneva, Switzerland, 2003.

[40] IEEE. *Standard Glossary of Software Engineering Terminology 610.12-1990, Vol. 1*. Los Alamitos: IEEE Press, 1999.

[41] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, October 1994.

[42] N.R. Jennings. Agent-based computing: promises and perils. In *Proc. of the fifth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 3.

[43] C. Pfister and C. Szyperski. Why objects are not enough. In *Proc. First International Component Users Conference (CUC)*, volume 3.

[44] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of aspectj. In *Proc. of 15th Eur. Conf. on Object-Or. Prog (ECOOP 01)*, volume 2072, pages 220–242, Budapest, Hungary, 2001.

[45] http://www.eclipse.org/aspectj/ .

[46] O. Postolache, J. M. Dias Pereira, M. Cretu, and P.S. Girao. An ann fault detection procedure applied in virtual measurement systems case. In *Proc. of IEEE Instrumentation and Measurement Technology Conference, IMTC/98, Vol. 1*, volume 3.

[47] M. Catelani and S. Giraldi. A measurement system for fault detection and fault isolation of analog circuits. *Measurement*, 25(2):115–122, March 1999.

[48] P. Arpaia, G. Lucariello, and A. Zanesco. Automatic fault isolation by cultural algorithms with differential influence. *IEEE Trans. on Instrumentation and Measurement*, 56(5):1573–1582, Oct. 2007.

[49] R. K. Gupta, C. N. Coelho, and G. De Micheli. Synthesis and simulation of digital systems containing interacting hardware and software components. In *29th ACM/IEEEDesign Automation Conference*, 1992.

[50] G. Graunke and S. Thakkar. Synchronization algorithms for shared-memory multiprocessors. *IEEE Computer*, 23(6):68–69, June 1990.

[51] C. von Praum, H. W. Cain, J. Choi, and K. D. Ryu. Conditional memory ordering. In *in Proc. of the 33th International Symposium on Computer Architecture (ISCA), IEEE*, 2006.

[52] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, December 2005.

[53] J. Bosch and G. Hedin. Editors's introduction. In *In Proceedings ALEL'96 Workshop on Compiler Techniques for Application Domain Languages and Extensible Language Models, Technical Report LU-CS-TR:96-173*, Lund University, April 1996.

[54] B. Mayers, S.E. Hudson, and R. Pausch. Past, present and future of user interface software tools. *ACM Trans. Computer-Human Interaction*, 7(1):3–28, March 2000.

[55] 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.

[56] G. Weber C. Lutteroth. Modular specification of gui layout using constraints. In *Proceedings of ASWEC 2008 - 19th Australian Conference on Software Engineering, IEEE Press*, 2008 1996.

[57] P. Achten, M. van Eekelen, and R. Plasmeijer. Compositional model-views with generic graphical user interfaces. *In Practical Aspects of Declarative Programming, PADL04, LNCS, Springer*, 3057, 2004.

[58] 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, LNCS. Edinburgh, UK, Springer*, 3145, 2003.

[59] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[60] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[61] W.B. Pennebaker and J.L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.

[62] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms.* Wiley-IEEE Press, 2002.

[63] K.H. Lee, H. Woo, and T. Suk. Data reduction methods for reverse engineering. *The International Journal of Advanced Manufacturing Technology*, 17(10):735–743, May 2001.

[64] W.J. Tompkins. *Biomedical digital signal processing.* Prentice Hall, New Jersey, 2000.

[65] W.C. Mueller. Arrhythmia detection program for an ambulatory ecg monitor. *Biomed.Sci. Instrument.*, 14:81–85, 1978.

[66] J. R. Cox, F. M. Nolle, H. A. Fozzard, and G. C. Oliver. Aztec, a preprocessing program for real-time ecg rhythm analysis. *Biomedical Engineering, IEEE Transactions on*, BME-15(2):128–129, April 1968.

[67] L.N. Bohs and R.C. Barr. Prototype for real-time adaptive sampling using the fan algorithm. *Medical and Biological Engineering and Computing*, 26(6):574–583, November 1988.

[68] A. Djafari Marbini and L.E. Sacks. Adaptive sampling mechanisms in sensor networks. In *London Communications Symposium*, London, 2003.

[69] Ankur Jain and Edward Y. Chang. Adaptive sampling for sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 10–16, New York, NY, USA, 2004. ACM.

[70] Johnsen Kho, Alex Rogers, and Nicholas R. Jennings. Decentralized control of adaptive sampling in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):1–35, 2009.

[71] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–6, Oct. 2007.

[72] O.O. Fadiran, P. Molnar, and L.M. Kaplan. Adaptive sampling via histogram equalization using an active walker model. In *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on*, pages 424–432, July 2006.

[73] J. Rigau, M. Feixas, and M. Sbert. Entropy-based adaptive sampling. In *Graphics Interface*, Halifax, Canada, June 2003.

[74] B.K. Natarajan. Filtering random noise from deterministic signals via data compression. *Signal Processing, IEEE Transactions on*, 43(11):2595–2605, Nov 1995.

[75] L. Kok-Fung. *Biomedical Digital Signal Processing*, chapter Data Reduction Techniques, pages 193–215. Prentice Hall, 2000.

[76] S. Sanfilippo, L. Bottura, M. Buzio, and E. Effinger. Magnetic measurements for 15-m long dipoles - extended program of tests. Internal note LHC-MTA-IN-2002-183, 2002.

[77] L. Angrisani, L. Bottura, A. Masi, and R. Schiano Lo Moriello. Digital signal processing approach for measurements of non-stationary magnetic field through a rotating coils system. In *Instrumentation and Measurement Technology Conference, 2006. IMTC 2006. Proceedings of the IEEE*, pages 747–752, April 2006.

[78] A. Jain. Measurements of field harmonics at very high ramp rates. In *Proceedings of the 14th International Magnet Measurement Workshop (IMMW-XIV)*, Geneva, September 26-29 2005. CERN.

[79] C. Daniel and F.S. Wood. *Fitting Equations to Data.* John Wiley & Sons, 1980.

[80] J. Billan, L. Bottura, M. Buzio, G. D'Angelo, G. Deferne, O. Dunkel, P. Legrand, A. Rijllart, A. Siemko, P. Sievers, S. Schloss, and L. Walckiers. Twin rotating coils for cold magnetic measurements of 15 m long lhc dipoles. In *16th International Conference on Magnetic Technology*, Ponte Vedra Beach, USA, 1999.

[81] IEEE standard for digitizing waveform recorders. *IEEE Std 1057-1994*, pages –, Dec 1994.

[82] IEEE Std 610-1990. IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries. (ANSI).

[83] D. Garvin. What does "Product Quality" really mean? *Sloan Management Review*, pages 25–45, Fall 1984.

[84] B. Kitchenham and S. L. Pfleeger. Software Quality: the Elusive Target. *IEEE Software*, 13(1):12–21, 1996.

[85] I. Tervonen and P. Kerola. Towards deeper Co-Understanding of Software Quality. *Information and Software technology*, 39:995–1003, 1998.

[86] International Standard ISO 8042. *Quality Management and Quality Assurance - Vocabulary.* International Organization for Standardization, Geneva, second edition, 1994.

[87] B. W. Böhm, J. R. Brown, and M. Lipow. Quantitative Evaluation of Software Quality. *Information and Software technology*, 39:995–1003, 1998.

[88] Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[89] N. E. Fenton. *Software Metrics: A Rigorous Approach.* Chapman & Hall, 1991.

[90] T. Gilb. *Principals of Software Engineering Management.* Addison-Wesley, Reading, Mass., 1987.

[91] J. A. McCall, P. K. Richards, and G. F. Walters. *Factors in Software Quality*, volume 1, 2, and 3. US. Rome Air Development Center Reports NTIS AD/A-049 014, NTIS AD/A-049 015 and NTIS AD/A-049 016, U. S. Department of Commerce, Springfield, Va., 1977.

[92] International Standard ISO/IEC 9126-2. *Software Engineering - Product Quality - Part 2: External Metrics.* International Organization for Standardization, International Electrotechnical Commission, 2003.

[93] International Standard ISO/IEC 9126-3. *Software Engineering - Product Quality - Part 3: Internal Metrics.* International Organization for Standardization, International Electrotechnical Commission, 2003.

[94] International Standard ISO/IEC 9126-4. *Software Engineering - Product Quality - Part 4: Quality in Use Metrics.* International Organization for Standardization, International Electrotechnical Commission, 2004.

[95] International Standard ISO/IEC 15939. *Software Engineering - Software Measurement Process.* International Organization for Standardization, International Electrotechnical Commission, 2002.

[96] International Standard ISO/IEC 14598. *Software Engineering - Product Evaluation.* International Organization for Standardization, International Electrotechnical Commission, 2000.

[97] R. van Solingen and E. Berghout. *The Goal/Question/Metric Method: A practical guide for quality improvement of software development.* McGraw-Hill Education, 1999.

[98] Understand c++. http://www.scitools.com/products/understand/.

[99] V. A. French. Establishing software metric thresholds. In *WSM 99: Int. Workshop on Software Measurement*, pages 43–50, Lac Supérieur, Canada, September 1999. IEEE.

[100] Le metriche e il loro utilizzo nello sviluppo del software (*in Italian*). http://www.dia.uniroma3.it/ torlone/sistelab/annipassati/sbavaglia.pdf., 2005.

[101] Nasa. software metrics. http://satc.gsfc.nasa.gov/metrics/codemetrics/index.php.

[102] Metrics available in understand c++. http://www.scitools.com/documents/metrics.php.

[103] Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, December 1976.

[104] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994.

[105] W. Li and S. Henry. Maintenance metrics for the object oriented paradigm. In *IEEE Proceedings of the First International Software Metrics Symposium*, pages 52–60. IEEE, May 1993.

[106] J.M. Bieman and B.K. Kang. Cohesion and reuse in an object-oriented system. In *Proceedings of the ACM Symposium on Software Reusability*. ACM, April 1995.

[107] M. Hitz and B. Montazeri. Measure coupling and cohesion in object-oriented systems. In *ISAAC'95: Proceedings of International Symposium on Applied Corporate Computing*, pages 24, 25, 274, 279, October 1995.

[108] R. Lincke and W. Löwe. Validation of a standard- and metric-based software quality model. In *10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE)*, July 2006.

[109] Rüdiger Lincke and Welf Löwe. Compendium of software quality standards and metrics. http://www.arisa.se/compendium/, April 2007.

[110] Mark Lorenz and Jeff Kidd. *Object-Oriented Software Metrics: A Practical guide*. Prentice-Hall, 1994.

[111] infusion - an integrated environment for performing in-depth code and architectural reviews of object-oriented and procedural software systems. http://www.intooitus.com/infusion.html.

[112] M. Salehie, S. Li, and L. Tahvildari. A metric-based heuristic framework to detect object-oriented design flaws. In *ICPC 2006: Proceedings of the 14th IEEE International Conference on Program Comprehension*, pages 159–168. IEEE, 2006.

[113] Arthur Riel. *Object-Oriented Design Heuristics*. Addison Wesley, Boston, MA, 1996.

[114] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Addison Wesley, 1999.

[115] Serge Demeyer, Stéphane Ducasse, and Oscar Nierstrasz. *Object-Oriented Reengineering Patterns*. Morgan Kaufmann, Boston, MA, 2002.

[116] Daniel Rajiu, Stéphane Ducasse, Tudor Girba, and Radu Marinescu. Using history inforation to improve design flaws detecion. In *CSMR'94: Proceedings Eighth Euromicro Working Conference on Software Maintenance and Reengineering*, pages 223–232, Los Alamitos, CA, October 2004. IEEE Computer Society.

[117] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho. Do Crosscutting Concerns Cause Defects? *IEEE Trans. on Software Engineering*, 34(4):497–515, July/August 2008.

[118] D. A. Finley, D. A. Edwards, R. W. Banft, R. Johnson, A. D. MC Inturff, and J. Strait. Time dependent chromaticity changes in the tevatron. In *Il. 60510*, Fermilab, Batavia, 1987.

[119] G. Ambrosio at alt. A scaling law for the snapback in superconducting accelerator magnets. *IEEE Transaction On Applied Superconductivity*, June 2005.

[120] N. Sammut, L. Bottura, and J. Micallef. The lhc magnetic field model. In *Proceedings of Particle Accelerator Conference*, pages 2648–2650, Knoxville, Tennessee, 2005.

[121] Ni m series multifunction daq for pci, pxi, and usb, http://sine.ni.com/nips/cds/view/p/lang/en/nid/14114.

[122] D. Wenzel, N. Borsi, and E. Gockenbach. Noise suppression and data reduction for partial discharge measurements using orthogonal transformations. In *Electrical Insulation, 1994., Conference Record of the 1994 IEEE International Symposium on*, pages 292–295, Jun 1994.

[123] D. L. Parnas. Software Aging. In *Proc. Int'l Conf. Software Engineering-ICSE*, pages 279–287, Los Alamitos, May 1994. IEEE Computer Society Press.

[124] A. H. Eden and T. Mens. Measuring software flexibility. *IEE Proc. Softw.*, 153(3):113–125, June 2006.

[125] K.N. Henrichsen. Overview of magnet measurement methods. CERN Accelerator School Proceedings, April 1997.