Requirements Analysis for Graphical Programming and a Practical Experience in a System for the Measurement of X-Rays [1]

E. GAYTÁN-GALLARDO[1,2], F. J. RAMÍREZ-JIMÉNEZ1[1,2], J. ORTIZ-HERNÁNDEZ[3], J. A. SEGOVIA DE LOS RÍOS[1,2], L. C. LONGORIA-GÁNDARA[1]
Instituto Nacional de Investigaciones Nucleares, InstitutoTecnológico de Toluca[2,] Centro Nacional de Investigación y Desarrollo Tecnológico[3]
MEXICO

Abstract:-The methodology presented in this paper provides analysis tasks for the development of systems with graphical programming. This methodology is based in requirements engineering, structured analysis, object-oriented analysis, analysis for reuse and UML notation. The requirements establish software constraints. The software engineer and customers take active roles in requirements analysis, the customer formulates a system level description and the software engineer refines the software and builds models that will be treated by software. The structured analysis relies on data modeling and flow modeling to create the bases for an analysis engineering activities result in the function, data and behaviour of the software, interfaces with other system elements, and model. The object-oriented analysis provides a concrete way to represent the requirements, to define the classes, objects, attributes, operations of objects and messages in a system. The analysis for reuse searches and creates reusable components. UML notation (Unified Modeling Language) is used for modeling the system, UML allows to express an analysis model using a modeling notation. In this work, a study case related with the development of a system for the measurement of X-rays is presented.

Key-Words: - Requirements analysis, Graphical programming, X-Rays, UML

1. Introduction

In systems developed with graphical programming a variety of reusable graphic objects can be integrated into an evolutionary prototype, it lets to develop software more efficiently. To produce complex high-quality software it is necessary to understand the software requirements and it is very important to apply the requirements engineering to perform the job properly. The analysis of requirements is a process of discovery, refinement and modeling. This work presents a methodology for requirement analysis of graphical programming systems, this methodology was obtained from the study of: requirements engineering [1], structured analysis, object-oriented analysis, analysis for reuse and UML notation.

Graphical programming started with the arrival of personal computers and was proposed to solve the necessity to have reusable software components for measurement and control systems. Hewlett Packard [2] and National Instruments develop reusable graphical objects. In 1983, National Instruments started a project of a graphical programming language, it was based in pictures and was called LabVIEW (Laboratory Virtual Instrument Engineering Workbench). LabVIEW was invented by Jeff Kodosky [3] and it was based in dataflow diagrams. In 1986 the first version was delivered for the development of measurement and control systems. Software modularity, maintainability, and reusability, key benefits of LabVIEW's hierarchical and homogeneous structure, are critically important to reducing the effort in software development. The pioneering concept of graphical programming, coupled with leading-edge computer science, and open connectivity to thousands of devices and instruments, has made LabVIEW a staple to many scientists and engineers.

[1] X-ray measurement system, is partially supported by International Atomic Energy Agency (ARCAL LIII project).

Systems developed with graphical programming can apply reusable objects in acquisition, processing and signal analysis, too in acquisition and image processing, process control, connectivity, etc. But, to produce complex high-quality software with graphical programming, it is necessary to apply a software process [4]. Requirements analysis is the first technical step in the software process.

2. Methodology

In the proposed methodology for requirements analysis; requirement engineering, structured analysis, object oriented analysis, reuse analysis and UML graphical notation were analysed and adapted to a graphical programming system. The requirements analysis starts with the meetings of the software developer with the customer and from this interview, the requirements specification results and a first prototype is proposed. The steps in the proposed methodology are: problem recognition, UML modeling (use cases, sequence diagrams, protocols and state machines), partitioning, domain analysis, prototyping, specification and documentation.

2.1 Problem recognition

In this task the following steps should be applied:
Step 1. To conduct an interview with the customer, asking about overall goals, software objectives, benefits and which kind of users of the system.
Step 2. To compile the system requirements and modeling with use case diagrams.
Step 3. To identify system actors and their relationship with use cases.
Step 4. To describe the problem scenario.

2.2. UML modelling

The heart of software problem solving is the construction of a model. The model abstracts the essential details of the problem from its usually complicated real world. Several modeling tools are wrapped under the heading of the **UML** [5]. The UML gives a common vocabulary to deal with software problems, it begins with the construction of a model. A **model** is an abstraction of the underlying problem. The **domain** is the actual world from which the problem comes. Models consist of **objects** that interact by sending **messages**. Objects have (**attributes**) and things they can do (**behaviours** or **operations**). The values of an object's attributes determine its **state**. At the requirement analysis we use the next kinds of modeling diagrams:

- **Use case diagrams:** They describe what a system does from the point of view of an external observer. The emphasis is on *what* a system does rather than *how*. *U*se case diagrams are closely connected to scenarios. A **scenario** is an example of what happens when someone interacts with the system. A **use case** is a summary of scenarios for a single task or goal. An **actor** is who or what initiates the events involved in that task. Actors simply are roles that people or objects play. The connection between actor and use case is a **communication**. Actors are stick figures. Use cases are ovals. Communications are lines that link actors to use cases. A **use case diagram** is a collection of actors, use cases, and their communications.
- **Sequence diagrams:** They are interaction diagrams that detail how operations are carried out, what messages are sent and when. Sequence diagrams are organized according to time. The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to the time when they take part in the message sequence.
- **Protocols**: They are used for modeling behaviour, a protocol is a specification of desired behaviour, an explicit specification of the contractual agreement between the participants in the protocol. A protocol comprises a set of participants, each of which plays a specific role in the protocol. Each one of such protocol roles is specified by a unique name, a set of signals that are received by that role as well as the set of signals that are sent by the role.
- **State machines:** The specification of valid protocols sequences is done using standard UML state machines

2.3 Partitioning

One of the tasks in this methodology is to divide a problem into its constituent’s parts [6]. Establishing a

hierarchical representation of function or information and then partition the uppermost element exposing, increasing detail by moving vertically in the hierarchy or decomposing the problem by moving horizontally in the hierarchy.

2.4 Domain Analysis

Software domain analysis consists of identification, analysis and knowledge of the application domain, typically for reuse on multiple projects within this application domain. The steps in the process are the following:

Step 1. Identification of the domain to be investigated.

Step 2. Identification of reusable graphical objects.

Step 3. Organization of reusable graphical objects.

Step 4. Analysis of the feasibility of reuse through modifications.

Step 5. Realization of modifications for reuse.

Step 6. Development of an analysis model.

For example the graphical programming language LabVIEW has a variety of reusable graphical objects applicable in domain as: data acquisition, signal processing, signal analysis, instrument control, image processing, mathematics, communication, report generation, fuzzy logic, PID control, etc.

2.5 Software prototyping.

Requirements Analysis should be conducted regardless of the software engineering paradigm that is applied, some circumstances require the construction of a prototype at the beginning of analysis, since the model is the only mean through which requirements can be effectively derived. The model then evolves into production software. The prototyping paradigm can be a throwaway prototyping or evolutionary prototype. The throwaway prototyping serves solely as a rough demonstration of requirements. The evolutionary prototype uses the prototype as the first part of an analysis activity that will be continued into design and construction.

It is always necessary to determine whether the system to be built is amenable to prototyping. Graphical programming language LabVIEW has libraries of reusable graphical objects and a rapid prototyping is assembled. To develop a prototype it is necessary: to understand the application domain, to model the problem and to know the basic system requirements.

2.6 The Software Requirements Specification

The specification has much to do with the quality of the solution, it may be viewed as a representation process. The software requirements specification is produced at the culmination of the analysis task. Because the specification forms the foundation of the development phase, extreme care should be taken in conducting the review.

2.7 Documentation

Documentation provides a foundation for successful requirements analysis and, more important, guidance for software support. A general outline for the contents is developed. Representations reveal layers of information, and diagram numbering indicates the level of detail presented.

3 Results

The main results of this work is the proposed requirements analysis methodology for graphical programming and its application in an acquisition and processing system as a case example of its application [7]. The requirements analysis in the development of a system for the measurement of X-rays is presented.

3.1 Problem recognition

After the interview with the customer, the problem is described:

“The Metrology Department at Nuclear Research National Institute, has three X-ray units, two for medical applications and another for industrial application. It is necessary to measure the main parameters of the X-ray units [8]. The detectors used in the measurements are PIN type silicon diodes and their output is a charge or current signal, this signal is

converted to voltage with a preamplifier. The output is proportional to the X-ray intensity and associated energy. The PIN type silicon diodes employed for the measurement of X-ray have different filters, one of tungsten and another of aluminum. The detection section is separated from the personal computer, and then it is necessary to develop a system for the X- ray measurements (dose and exposure) and to show its results in graphical form".

3.1.1 Main system operations

- Acquisition of signals from two detectors.
- Processing of acquired signals.
- Calculation of X-ray dose and exposure.
- Presentation of results in the personal computer screen.
- Record of X-ray equipment data.
- Saving acquired data in a file.

3.1.2 Innovatory system operations

- Presentation of X-ray measurement results in the WEB.

3.2 Requirements modeling with UML

UML has been used for requirements modeling. Figure 1 shows the use case diagram of the X-ray measurement system. This use case refers to the data acquisition and results presentation and an use case description is presented. Figure 2 shows the relation between the actors within the system.

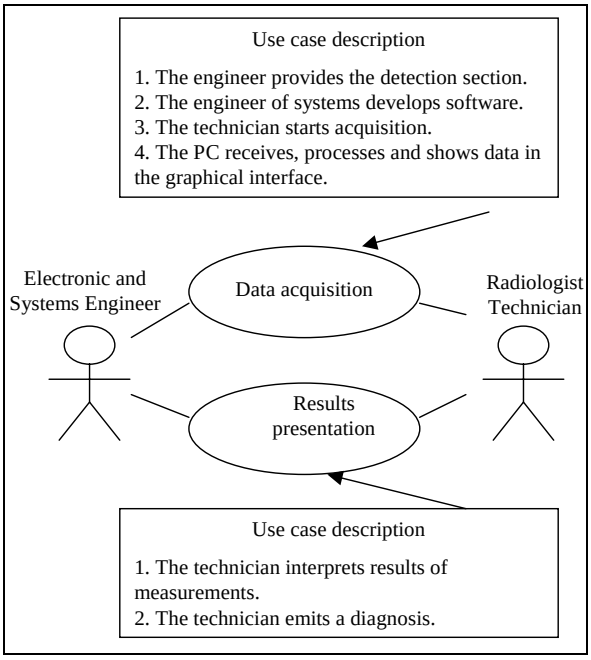

Figure 1. Use case diagram for the development of the X-ray measurement system.

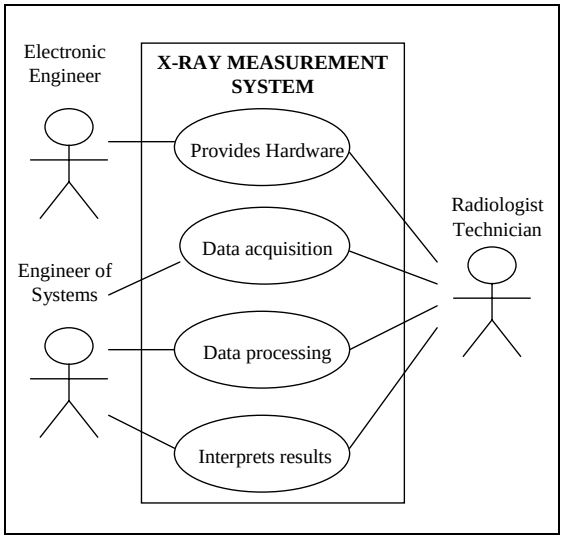

Figure 2. Relation between the actors within the system.

3.2.1 System Actors and their function. Table 1 shows the description of the functions performed by the actors, whose participate in the development of the X-ray measurement system.

Personal	Hardware
Electronic engineer. • Analyzes PIN detectors response. • Realises signal conditioning. *System engineer.* • Analyzes the signals from the detectors. • Develops the X-ray measurement system *Radiologist technician.* • Operates the X-ray equipment. • Places the radiation detectors in the exposition field. • Starts the X-ray measurement system • Interprets the results.	*Detectors.* • PIN diodes. • Filters. *Preamplifier*s. • Detects and prepares the X-ray signal *Data acquisition interface PCI-6024E* [9,10]. • Acquires the signals from the PIN detectors. *PersonalComputer.* • Presents the results from the X-ray measurement system.

Table 1. X-ray measurement system actors and their function.

3.2.2 Problem scenario. Figure 3 shows the problem scenario of the X-ray measurement system. Two PIN type radiation detectors are exposed to X-ray, which generate a photo-current, it is converted to a voltage in the preamplifiers. The voltage is measured by the acquisition card and the signal processing is realized. After that, the signals are presented in the graphical user interface.

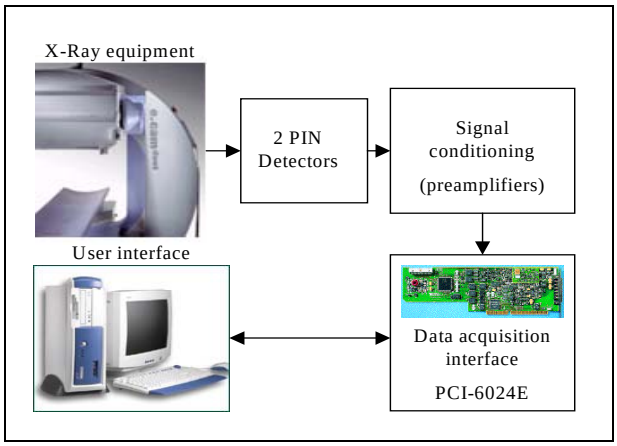

Figure3. X-ray measurement system scenario.

3.2.3 Sequence diagrams for the development of the X-ray measurement system. Figures 4 and 5 shows the interaction between actors and scenario about operation details. Figure 4 describes the personnel working in the system. The electronic engineer designs and builds the radiation detection section. The systems engineer analyzes, designs and develops an acquisition and processing system and the radiologist technician tests the system, reviews and interprets the results. The figure 4 shows that the detectors and preamplifiers supply and prepare X-ray signals respectively, these signals are carried to the acquisition interface and the personal computer. In the personal computer the X-ray measurement system is programmed and results are presented in an user interface. The objects involved in the operation are listed from left to right according to when they take part in the message sequence and the sequence diagrams are organized according to time.

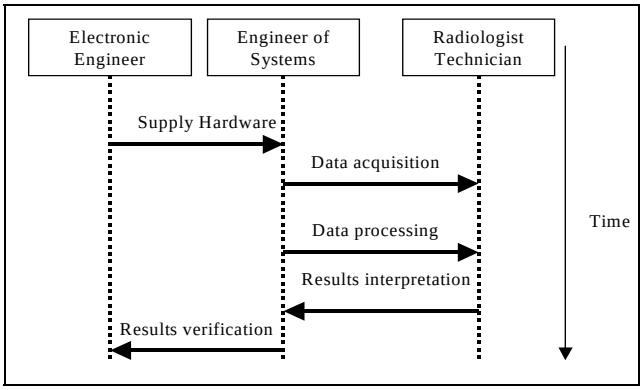

Figure 4. Sequence diagram about personal interaction within the X-ray measurement system.

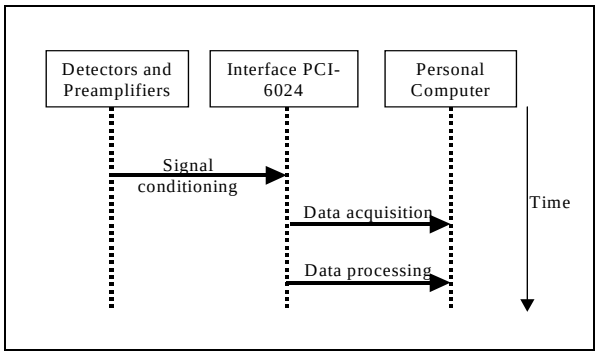

Figure 5. Sequence diagram showing the hardware interaction within the X-ray measurement system.

3.2.4 X-ray measurement system protocol. Figure 6 is a protocol [11] that shows a specification of desired behaviour of the X-ray measurement system, here the incoming and outgoing signals are exhibited. Each protocol role is specified by a unique name and a set of signals that are received by that role as well as the set of signals that are sent by the role. Consider the abstract machine in figure 7, it is representative of the most abstract level of behaviour of the X-ray measurement system and shows how a simple state machine represents the behaviour of a real-time system.

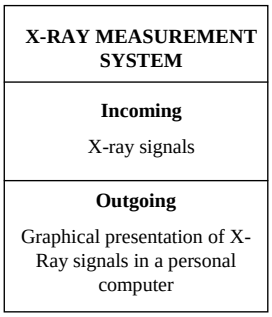

Figure 6. X-ray measurement system protocol.

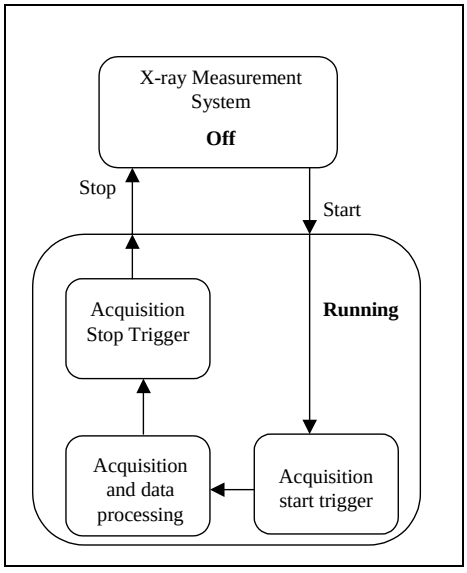

Figure 7. State machine of the X-ray measurement system.

3.3 Partitioning

The X-ray measurement system showed in figure 8 is divided in three subsystems and deals with the configuration, trigger monitoring and user interfaces development.

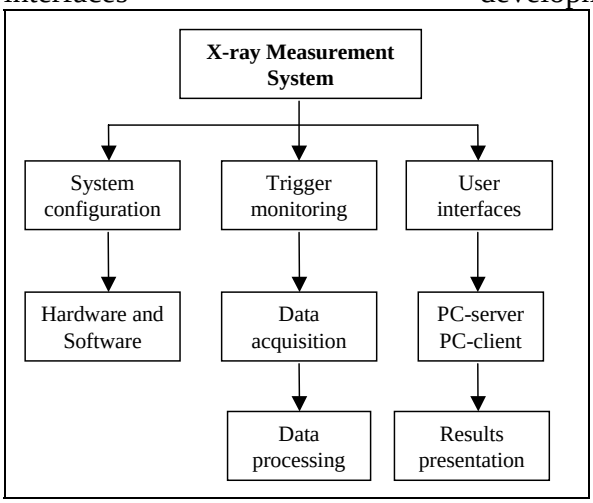

Figure 8. Partition of the X-ray measurement system.

3.4 Domain Analysis

Figure 9 presents the domain analysis modeling. The X-ray measurement system works in the following application domains:

- **Data acquisition**: The data are acquired with the acquisition card PCI-6024 and its connection terminal, with a virtual instrument developed with graphical programming.
- **Signal processing:** The acquired data are processed with mathematics graphical objects supplied by the graphical language (LabVIEW).
- **User interfaces:** The signals acquired and processed are presented in graphical indicators of the user interface.
- **Client / server communication:** The network to the Internet exists because software residing on client computer requests services from server computer.

3.5 Software prototyping.

An evolutionary prototype for the X-ray measurement system was developed with the graphical programming language LabVIEW. This prototype was employed as the first part of an analysis activity that will be continued into design and system construction. Figure 10 shows the prototype for the X-ray measurement system. It presents a graphical user interface with controls for data acquisition and indicators for X-ray signals, and data record of the measurements in the X-ray equipment.

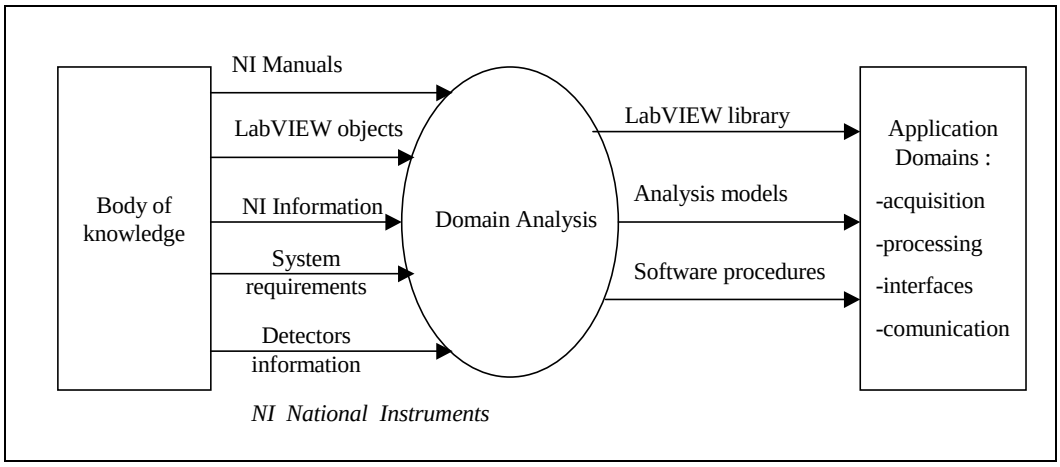

Figure 9. Domain analysis modeling of the X-ray measurement system.

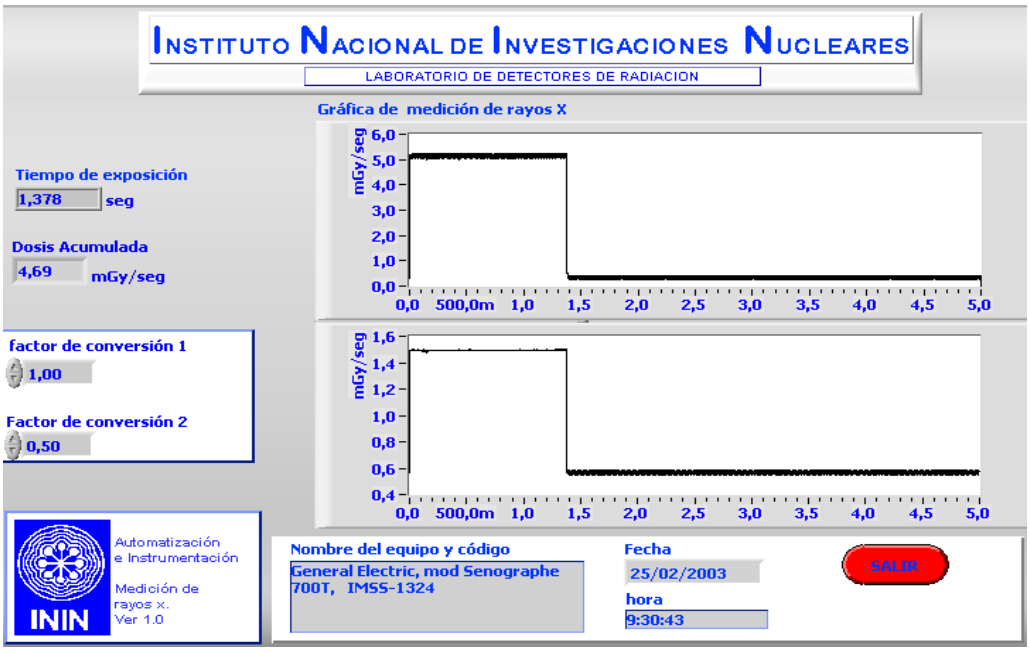

Figure 10. Prototype for the X-ray measurement system.

3.6 The Software Requirements Specification for the X-ray measurement system

The requirements specification was presented in a manner that ultimately leads to successful software implementation, this contains: an introduction, description of the problem that the software must solve (functional and behavioural description), validation criteria, bibliography and appendix. The specification was accompanied by the executable prototype.

3.6.1 Validation Criteria. Specification of validation criteria is an implicit review of all other requirements and was conducted by both the software developer and the customer.

3.7 Documentation

Documentation provides a foundation for successful requirements analysis and, more important, guidance for software support. A general outline for the contents was developed. Representations reveal layers of information, and diagram numbering indicates the level of detail presented.

4. Conclusions

This paper shows that if we apply the requirements analysis in the development of systems with graphical programming language LabVIEW, it helps to improve the quality of the product, and permits to save costs derived from systems without requirements analysis. This is more important in the development of complex systems in which a methodology of design is a must. The requirements analysis helps too, to decide if is convenient to realize a system or not. As a future work we plan to develop methodologies with methods for design, development and probes in systems with graphical programming languages.

5. References

[1] I. Summerville, P. Sawyer, "Requirements Engineering", http://www.comp.lancs.ac.uk/computing/resources/re-gpg/preface.html#contents.

[2] M.L. Griss, R. R. Kessler, "Building Object-Oriented Instrument Kits", Object Magazine, April, 1996.

[3] G. W. Johnson, *LabVIEW Graphical Programming, Practical applications in Instrumentation and Control*, McGraw-Hill, 1994.

[4] E. Gaytán G., "Modelo de Proceso de Software para Programación Gráfica". Memorias del Taller Internacional de Tecnología de Software, CIC, IPN, p. 23-30, Nov. 1999, *ISBN 970-18-3738-x.*

[5] http://www.togethersoft.com/services/practical_guides/umlonlinecourse/index.html.

[6] R. S. Pressman, Software Engineering. Fifth Edition, McGraw-hill international edition, 2001.

[7] E. Gaytán G., F.J. Ramírez J. "Sistema de medición de rayos x (análisis de requisitos)", Informe técnico IT.AU-0206-2002, ININ.

[8] Mercado I., Ramírez J. F. J., Tovar V., Becerril A., "Prototipo para la Medición de Parámetros en una unidad de Mamografía utilizando Fotodiodos", II Conferencia Internacional y XII Congreso Nacional sobre Dosimetría de Estado Sólido. México, D. F., 22-24 Sep. (1999).

[9] National Instruments, 6023E/6024E/6025E, User Manual, 1999, part number 322072B-01,

[10]National Instruments NI-DAQ Function Reference Manual for PC Compatibles, 1995, part number 320499C-0.

[11]B. Selic, J. Rumbaugh, "Using UML for Modeling Complex Real- Time Systems" http://www.rational.com/products/whitepapers/