Desarrollo de una herramienta de simulación para estudio y análisis del protocolo IEEE488

Heriberto I. Hernández Martínez[α], Angel F. González Hernández[β]
Instituto de Electrónica y Computación
Universidad Tecnológica de la Mixteca
69000, Huajuapan de León, Oaxaca
Tel. +52 (953) 53 20214 Fax +52 (953) 53 20399
email: [α]hhdez@nuyoo.utm.mx, [β]angelf@mixteco.utm.mx

RESUMEN

Los autores presentan el desarrollo de una herramienta software para el estudio y análisis del protocolo IEEE 488 mediante la utilización de herramientas que proporciona la tecnología orientada a objetos y de los conceptos de la instrumentación electrónica programable. El desarrollo de la herramienta de simulación SepiGPIB se describe en base a la metodología RUP, el modelado en UML y la realización en el lenguaje de programación de alto nivel C++ Builder. Asimismo, se describe el funcionamiento del software final.

La herramienta de simulación SepiGPIB intenta ser una introducción a las nuevas herramientas software de metodología y modelado para el desarrollo de sistemas de simulación de protocolos de comunicaciones industriales, de instrumentación y de redes de computadoras.

Palabras clave:

GPIB, IEEE 488, Instrumentación electrónica, Instrumentación virtual, Modelado, RUP, UML.

I. INTRODUCCIÓN

El desarrollo de sistemas software se ha visto afectado por la falta de estandarización en sus procesos de metodología de desarrollo y modelado. Los desarrolladores de software han creado sistemas que carecen de una planeación y como consecuencia son difíciles de modificar para actualizarlos a las necesidades emergentes. En la actualidad, la ingeniería de software y la tecnología orientada a objetos ofrecen herramientas de metodologías de desarrollo, de modelado y lenguajes de programación de alto nivel para lograr sistemas robustos, de alta calidad y fuertemente respaldados por documentación mediante modelos que facilitan su comprensión y actualización en modificaciones futuras.

Por otro lado, la instrumentación electrónica ha dejado de ser un campo de estudio basado en el conocimiento y manipulación de instrumentos modulares sencillos. En la actualidad los términos instrumentación electrónica programable e instrumentación virtual han dado origen a sistemas automatizados de medida complejos que interconectan instrumentos programables y/o instrumentos virtuales para análisis, procesamiento y presentación de resultados, todo ello con la finalidad de controlar procesos, verificar productos, explorar servicios, analizar la calidad del producto, etc.

II. TECNOLOGÍA ORIENTADA A OBJETOS

El desarrollo de sistemas de software es una industria relativamente joven que aún no ha alcanzado un nivel de madurez, consecuentemente, los productos desarrollados a menudo carecen de la estabilidad requerida para ser explotados como productos comerciales. Por lo tanto, uno de los aspectos más importantes de la ingeniería del software orientada a objetos (OOSE, *Object Oriented Software Engineering*) es proveer de alternativas para mejorar el proceso de desarrollo de software.

Dentro de las alternativas propuestas existen dos aspectos vitales, por un lado, plantear una metodología de desarrollo del software que permita realizar un modelo del sistema a construir y por otro lado, brindar una documentación adecuada para presentar los aspectos más relevantes del propio desarrollo.

La construcción de un sistema debe iniciar por conocer las demandas de los usuarios finales y los requerimientos que el sistema debe cubrir, para ello es necesario visualizar un bosquejo preliminar de cómo será el sistema mediante modelos.

La importancia de los modelos ha sido evidente en todas las disciplinas de ingeniería a lo largo de la historia [1, 2]. Los dibujos, simples o complejos, dan pauta a la especificación de lo que será el producto final y en ellos se planean los costos, tiempos y estimaciones de distribución y de los recursos para el desarrollo total del producto o sistema bajo desarrollo. En el desarrollo de software un modelado asegura un software final de calidad.

Por otra parte, el contar con una buena documentación permite tener la habilidad de reutilizar la tecnología, construyendo bloques de un sistema que son plenamente identificados y explotados en base a su documentación para ser aplicados a nuevos proyectos.

Cabe destacar que desde la aparición de la tecnología orientada a objetos han surgido serios problemas para modelar sistemas en base a esta nueva tendencia, debido principalmente, a que cada modelo presenta sus propias herramientas, las cuales contienen símbolos y terminologías propias que resultan en frustraciones para quienes intentan modelar en base

J. Díaz de León, G. González, J. Figueroa (Eds.): Avances en Ciencias de la Computación, pp. 319-323, 2003.

a la tecnología orientada a objetos [3]. Sin embargo, los procesos de estandarización han dado origen a herramientas de modelado como es el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) y de metodología como es el Proceso Unificado de Rational (RUP, *Rational Unified Process*).

El UML utiliza un modelado visual con notaciones gráficas para analizar y diseñar las aplicaciones, las cuales distinguen entre los dominios del negocio y los dominios de la computadora. UML tiene como objetivo el describir cualquier tipo de sistema en términos de diagramas orientados a objetos, para ello UML especifica elementos, diagramas y símbolos basados en el paradigma OO y se emplea en las diferentes fases del desarrollo de un sistema, desde la fase de requerimientos hasta la fase final de pruebas, independientemente de la tecnología de implementación [4, 5].

Grady Booch, Ivar Jacobson y James Rumbaugh han propuesto y desarrollado un proceso unificado llamado Objectory, que en la actualidad se conoce como Proceso Unificado de Rational (RUP, *Process Unified Rational*), el cual describe una serie de pasos para llegar al resultado final (*quién* está haciendo *qué*, *cuándo* lo hace y *cómo* alcanzar el objetivo) [6].

III. INSTRUMENTACIÓN ELECTRONICA

En la actualidad, la instrumentación electrónica afronta constantes cambios y se ha convertido en una herramienta indispensable para ingenieros, científicos y técnicos que requieren de sistemas electrónicos de medida de gran exactitud y precisión [7]. Por un lado, el continuo avance de la microelectrónica, las prestaciones de los paquetes informáticos y el desarrollo de nuevas tecnologías en el diseño de sistemas de medida para el control de procesos, verificación de productos, explotación de servicios, análisis de calidad, etc., han permitido el desarrollo de potentes sistemas de medida automatizados (ATE, *Automated Test Equipment*) [8, 9, 10], mientras que por el otro lado, es cada vez más común la utilización de las computadoras personales (PC, *Personal Computer*) como el principal recurso en diversas áreas de aplicación como son laboratorios, entornos industriales, sistemas de instrumentación, etc.

Los sistemas ATE guardan una gran dependencia respecto a los sistemas de adquisición de datos (DAQ, *Data Acquisition*) y para lograr la interconexión de los diversos sistemas electrónicos de medida, existe un amplio número de protocolos de comunicaciones dedicados a dicha tarea, como son IEEE 488 o GPIB (*General Purpose Instrumentation Bus*), VXI (VME *Bus eXtension for Instrumentation*), PXI/CompactPCI (PCI *eXtension For Instrumentation*), MXI (*Multisystem Instrument Interface*), etc.

El estudio del GPIB ha alcanzado una enorme expansión permitiendo el diseño de complejos sistemas ATE implementados en diversas plataformas de computadoras bajo diferentes sistemas operativos, lo cual ha dado lugar al concepto de instrumentación virtual que, de forma paulatina, ha venido a reemplazar al concepto de instrumentación clásico [11].

Las aplicaciones en un entorno de instrumentación programable se realizan comúnmente mediante herramientas software propietarias con un gran soporte matemático y con funciones destinadas al control y programación de los instrumentos. La libertad con la que cuenta el programador de sistemas se ha obtenido de la propia estandarización de los protocolos, resultado de ello son las librerías, controladores (*drivers*), etc., que los fabricantes proporcionan al usuario para utilizarse junto a lenguajes de programación de alto nivel, como C++, Visual C++, etc., para el desarrollo de aplicaciones de usuario.

IV. IEEE 488

La interfaz IEEE 488, conocida ampliamente como GPIB, fue diseñada para integrar uno o más instrumentos a una computadora o controlador [12, 13].

En el bus GPIB se pueden conectar hasta 15 instrumentos o dispositivos, los cuales se comunican unos con otros bajo una configuración maestro/esclavo mediante cables y conectores requeridos por el bus. El control del sistema lo realiza un dispositivo maestro, llamado controlador, el cual generalmente es una computadora personal o un controlador del bus dedicado basado en un microcontrolador. Hoy en día, el bus GPIB contempla un protocolo que, mediante órdenes, permite a los usuarios diseñar sistemas simples para realizar pruebas de medición complejas.

En 1987, el IEEE propuso una modificación al estándar original IEEE 488, conocido como el estándar IEEE 488.2, para crear sistemas compatibles y desarrollar programas flexibles mediante la estandarización de formatos y código de datos.

Años después, se formó un consorcio de empresas fabricantes de instrumentos programables cuya principal aportación ha sido la especificación de órdenes estándares para instrumentos programables (SCPI, *Standard Commands for Programmable Instruments*), la cual define un conjunto de órdenes de programación para instrumentos de diferentes fabricantes. La Fig. 1 muestra la relación entre los estándares IEEE 488.

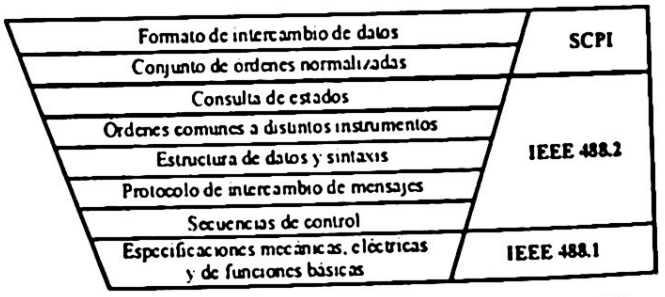

Figura 1. Relación entre los estándares IEEE 488.

V. MODELADO DEL SEPIGPIB

El modelado del SepiGPIB (Sistema educativo para el estudio del protocolo de instrumentación GPIB) aplica el paradigma orientado a objetos (sección II) al diseño e implementación de un software para simular el funcionamiento del protocolo de instrumentación GPIB.

La metodología utilizada durante el desarrollo del sistema fue RUP y algunos aspectos de la orientación a objetos, las herramientas que contribuyeron al desarrollo de este sistema fueron Rational Rose y UML como lenguajes de modelado y Borland C++ Builder como lenguaje de implementación del modelo obtenido. El SepiGPIB utiliza los formatos del RUP para documentar, organizar y explicar los procesos desarrollados.

El SepiGPIB consideró las etapas más relevantes de la metodología RUP y se desarrolló en forma interactiva mediante un desarrollo en cascada, es decir, se realizaron las etapas de requerimientos, análisis y diseño de manera secuencial hasta llegar a una versión ejecutable del sistema (Fig. 2).

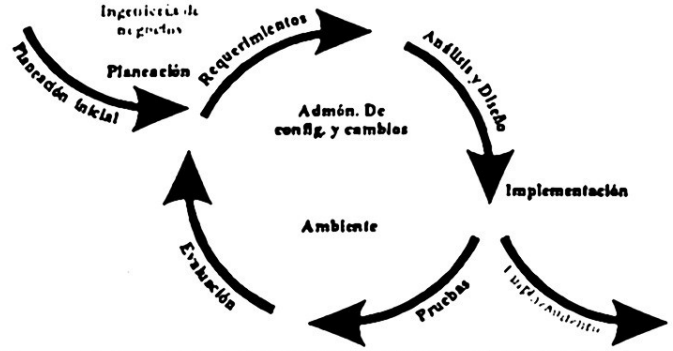

Figura 2. Etapas del desarrollo interactivo del RUP.

Cada fase de desarrollo genera una documentación cuya finalidad es explicar cómo es que dicha fase se lleva a cabo en el modelado del sistema.

Una vez obtenidos los requerimientos del sistema, se desarrolló el modelado de casos de uso para mejorar la comprensión de los requerimientos del sistema. La identificación de los casos de uso (Fig. 3) se obtuvo a partir del punto de vista del usuario con respecto al comportamiento esperado del sistema.

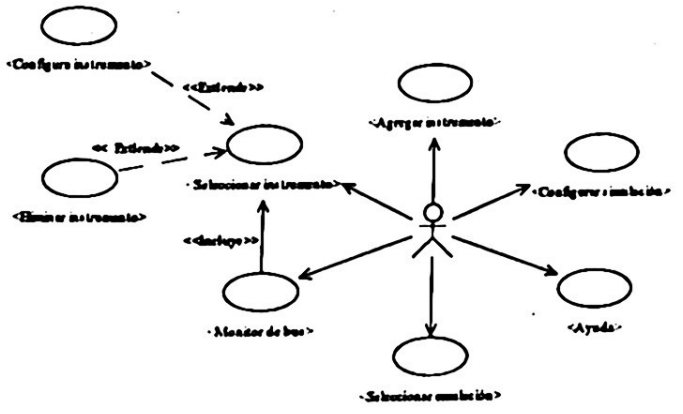

Figura 3. Diagrama de casos de uso.

El análisis se realizo mediante el modelado de las clases, miembros y especificaciones que constituyen el problema real y las relaciones estáticas, o de uso, que existen entre ellas, dicho análisis tiene la finalidad de obtener modelos que se ajusten mejor al problema real.

En esta fase se realizaron actividades para la identificación de clases del sistema. La identificación de clases se realizo mediante el análisis de los casos de uso y la adopción de las estrategias identificación de clases por categoría y por frases nominales.

El desarrollo de la fase de diseño se basó en describir los componentes del software a implementar en el sistema, finalmente estos modelos se utilizan para crear un modelo de clases del proyecto.

Se diseñaron las fases de ayuda para definir el comportamiento de las clases y de los objetos manejados por el sistema. El modelado conceptual, los diagramas de clases, de secuencia y colaboración describen el comportamiento dinámico del sistema.

El modelo conceptual (Fig. 4) ilustra la relación existente entre la clase TSimula y cada una de las clases asociadas, indicando sus responsabilidades. La clase TControlador se encarga de manipular los módulos de los instrumentos, archivos y configuraciones de las clases asociadas a la simulación.

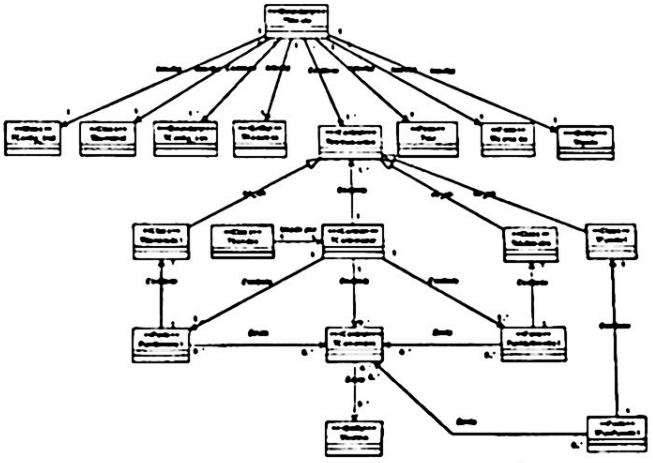

Figura 4. Diagrama del modelo conceptual.

El diagrama de clases (Fig. 5) ilustra los métodos de cada clase, sus atributos, el tipo de información y la visibilidad entre clases, se utiliza, junto al diagrama de colaboración, para agregar asociaciones y atributos a las clases.

El diagrama de secuencia resalta el orden de los eventos entre las clases y los objetos, dichos diagramas ilustran los eventos que ocurren desde el punto de vista del actor que utiliza el sistema. La Fig. 6 ilustra el diagrama de secuencia para el caso de uso Agregar instrumentos.

Los diagramas de colaboración ilustran la interacción de mensajes entre los objetos definidos en el modelo conceptual, dentro de los diagramas de colaboración se asignan responsabilidades a los objetos utilizados.

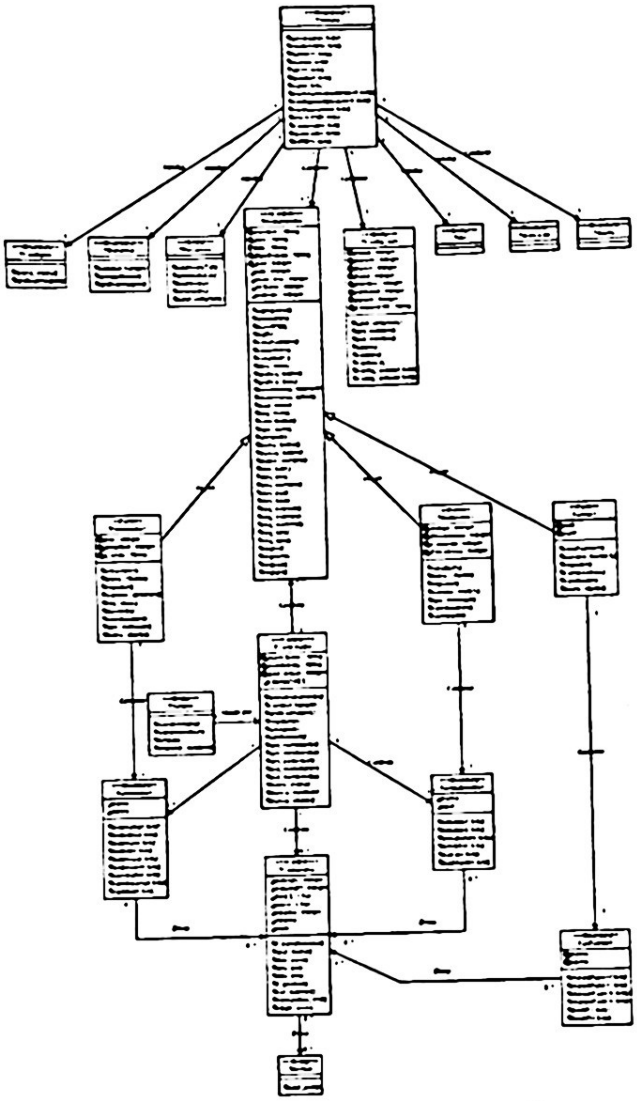

Figura 5. Diagrama de clases del SepiGPIB.

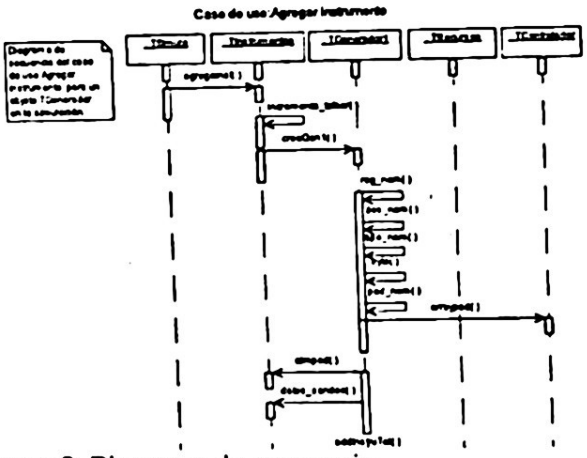

Figura 6. Diagrama de secuencia.

La Fig. 7 muestra el diagrama de colaboración para el caso de usc Agregar instrumento.

Las fases de prueba consistieron en realizar un análisis de los módulos que constituyen la arquitectura del SepiGPIB. Cada uno de los módulos principales que componen el sistema fue sometido a sesiones de pruebas para localizar posibles errores de ejecución o de funcionamiento lógico.

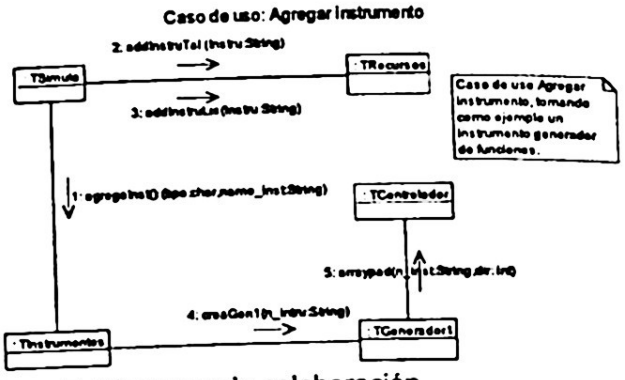

Figura 7. Diagrama de colaboración.

VI. RESULTADOS

Debido a que los instrumentos de medida utilizados en la simulación presentan caracteristicas propias de cada modelo y fabricante, se realizó un análisis funcional de los atributos reales de los dispositivos a incorporar.

Los instrumentos configuran sus funciones y obtienen datos a través de un panel frontal. El panel frontal es la interfaz con el usuario para realizar una petición de medida o de modificación a los datos almacenados en un instrumento.

El SepiGPIB especifica la forma de agregación de los instrumentos al bus y clasifica a los instrumentos en dos tipos, emisor o receptor, de acuerdo a las características propias de cada instrumento. El SepiGPIB implementa la configuración lineal indicada en las especificaciones mecánicas del GPIB (Fig. 8).

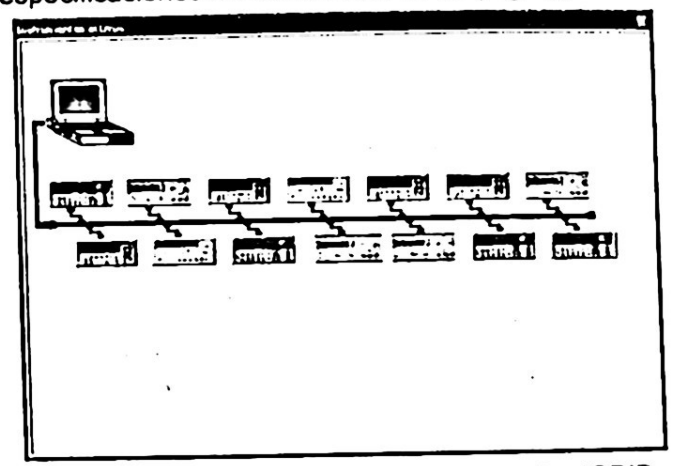

Figura 8. Configuración del bus lineal del SepiGPIB.

El sistema cuenta con filtros de tramas, que pueden ser configurados por el usuario mediante la ventana Configuración de simulación, y determinan el tipo salida que se desea como resultado de la simulación (Fig. 9). Por defecto, el SepiGPIB realiza la configuración con valores predeterminados por el sistema.

La función de transferencia analiza cada uno de los datos en base a la activación de las señales en el bus, una vez activada la comunicación entre los dispositivos. La función de transferencia almacena cada uno de los datos enviados al bus en un archivo de tramas, que puede visualizarse durante la simulación (Fig. 10).

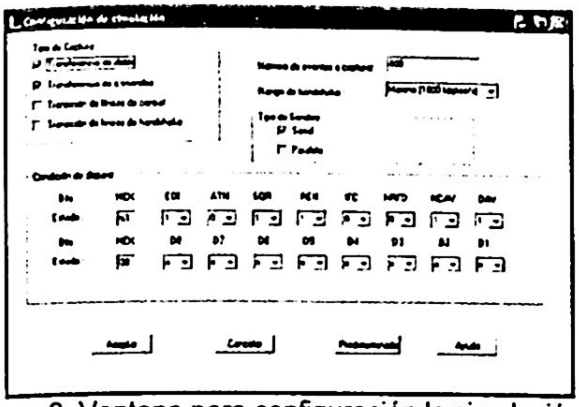

Figura 9. Ventana para configuración la simulación.

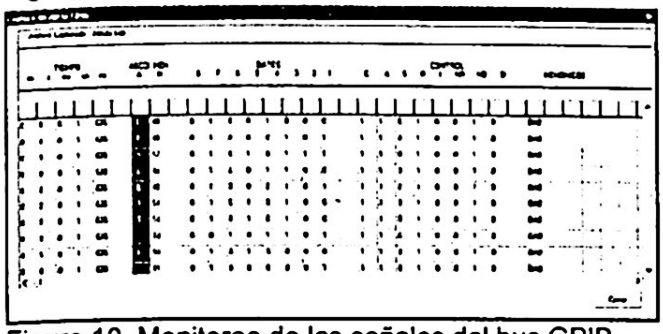

Figura 10. Monitoreo de las señales del bus GPIB.

El SepiGPIB proporciona dos alternativa de comunicación con un instrumento generador de funciones modelo 33120A, mediante la utilización del controlador del GPIB, que hace uso de la dirección primaria del instrumento para direccionar cada petición al dispositivo correspondiente, o mediante el panel frontal del instrumento, que permite al usuario realizar peticiones o configurar el instrumento (Fig. 11).

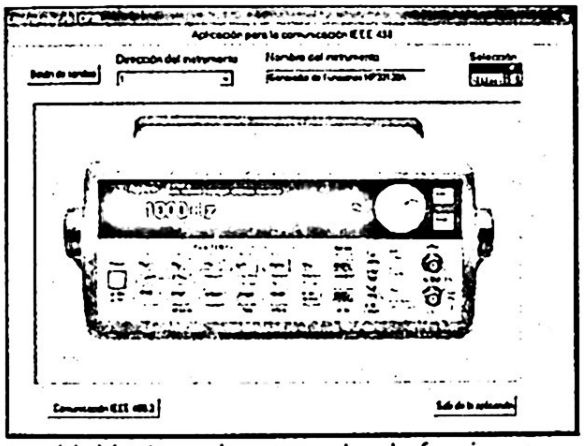

Figura 11. Ventana de generador de funciones.

VII. CONCLUSIONES

El software fue elaborado en base a ciertos requerimientos definidos por el problema. Al ser un software de simulación, el diseño se basó en la existencia de software de instrumentación electrónica.

Durante el desarrollo del SepiGPIB se realizaron evaluaciones referentes a la funcionalidad, la interfaz, los procedimientos y el comportamiento del software con ayuda de ingenieros en electrónica y computación, mismos que aportaron valiosos comentarios para la realización de un software robusto.

El resultado final es una herramienta software con aplicaciones académicas para la enseñanza de las comunicaciones en entornos industriales y de instrumentación.

El modelado en UML y la utilización de la metodología RUP facilitaron la codificación al lenguaje de programación Builder C++, siendo a su vez herramientas de gran valor en la verificación de errores durante el desarrollo del sistema.

Como líneas futuras, se pretende actualizar el SepiGPIB mediante la implementación de la función de emulación del protocolo GPIB, misma que ayudará a monitorear las tramas del bus en tiempo real, asimismo están en desarrollo herramientas similares para diversos protocolos de comunicaciones industriales.

AGRADECIMIENTOS

El trabajo presentado en este artículo ha sido realizado gracias al proyecto financiado por la Secretaría de Educación Pública, Ref. P/PIFI 2001-41-FO-02.

REFERENCIAS

[1] G. Booch, J. Rumbaugh and I. Jacobson, "The Unified Modeling Language; User Guide," Addison Wesley Longman, 1998.
[2] H. Eriksson and M. Penker, "UML Toolkit," John Wiley & Sons, Inc., 1998.
[3] M. Aksit and L. Bergmans, "Obstacles in Object-Oriented Software Development," Technical report of the Faculty of Computer Science, University of Twente, The Netherlands.
[4] M. Fowler y S. Kendall, "UML gota a gota," Addison Wesley Longman, 1999.
[5] D. Beringer, "Modelling Global Behaviour in Object-Oriented Analysis: Scenarios, Use Cases and Interaction Diagrams," Technical report of the Laboratoire du Génie Logiciel, Lausanne, 1996.
[6] I. Jacobson, G. Booch and J. Rumbaugh, "The Unified Software Development Proccess," Addison Wesley Longman, 1999.
[7] P. Mariño, J. Nogueira and H. Hernández, "Programmable Instrumentation Laboratory for Testing of Electronic Circuits and GPIB's Signal Analysis," Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, CATE'99, pp. 167-171, Philadelphia, Pennsylvania (USA), May 6-8, 1999.
[8] P. Mariño, J. Nogueira and H. Hernández, "Training on Programmable Instrumentation for a curriculum of Electronic Engineering," Proceedings of the 2nd International Conference in Recent Advances in Mechatronics, ICRAM'99, Estambul (Turquía), May 24-26, 1999.
[9] P. Mariño, J. Nogueira and H. Hernández, "Electronics Laboratory Practices based on Virtual Instrumentation," Proceedings of the FIE'99, ASEE/IEEE, vol. 12, pp. 6-10, San Juan (Puerto Rico), November 10-13, 1999.
[10] P. Mariño, J. Nogueira y H. Hernández, "Laboratorio de Instrumentación Programable para prueba de circuitos electrónicos y análisis de señales de bus GPIB," Revista TEMAS, Universidad Tecnológica de la Mixteca, Vol. 4, pp. 29-36, 2000, México.
[11] P. Mariño, J. Nogueira and H. Hernández, "Laboratory of Virtual Instrumentation for Industrial Electronics," Proceedings of the IEEE International Conference on Industrial Technology, ICIT'2000, Goa (India), January 19-22, 2000.
[12] ANSI/IEEE Std 488-1978, "Standard Digital Interface for Programmable Instrumentation. Recommendes Practice for Code and Format Conventions for Use with ANSI/IEEE Std 488-1978, ANSI/IEEE Std 488-1978, IEEE Std 728-1982," The Institute of Electrical and Electronics Engineers, 1983.
[13] A. Caristi, "IEEE-488 General Purpose Instrumentation Bus Manual," Academic Press, 1989.