Un Modelo de Interfaz Usuario para un Ambiente de Resolución de Sistemas de Ecuaciones Lineales

M. González[1,2]
e-mail: mgonzale@asci.fr
http://www.erablis.com/martha

and
S. Petiton[2,3]
e-mail: petiton@lifl.fr, petiton@asci.fr
http://www.lifl.fr/~petiton

[1] Escuela de Computación. Universidad Central de Venezuela. Caracas - Venezuela
[2] Laboratoire d'Applications Scientifiques du Calcul Intensif (ASCI) CNRS-CPR 9029.
Université Paris-Sud. Orsay - France
Laboratoire d'Informatique Fondamentale de Lille (LIFL).
Université des Sciences et Technologies de Lille. Lille - France

Resumen

El propósito de este artículo es presentar un modelo de interfaz hombre-máquina para un ambiente integrado e interactivo que permite la resolución de sistemas de ecuaciones lineales. Este modelo ha sido desarrollado utilizando un proceso que combina los conceptos del enfoque OOSE y el modelo PAC-Amodeus para la estructuración de la arquitectura de un sistema interactivo.

1 *Introducción*

En cálculo científico, se ha incrementado la necesidad de implementar, en computadores de gran rendimiento, técnicas eficientes para la manipulación de grandes cantidades de datos. Para satisfacer estos requerimientos ha sido desarrollado un número importante de métodos y bibliotecas de propósito general, como ejemplo podemos mencionar: **BLAS**: **B***asic* **L***inear* **A***lgebra* **S***ubprograms*, un conjunto de programas para realizar operaciones sobre matrices y vectores densos [2], *PBLAS* una versión de BLAS para arquitecturas paralelas [5], *Sparskit* un conjunto de programas para la manipulación de matrices esparcidas [10]. Existen también algunos intentos de incorporar, la tecnología objetos en el ámbito del cálculo científico intensivo, entre estos se pueden mencionar: **LAPACK++**: **L***inear* **A***lgebra* **P***ackage* [6], una extensión, escrita en C++ de la biblioteca LAPACK, la cual permite resolver problemas clásicos de álgebra lineal, como la resolución de sistemas de ecuaciones lineales y el cálculo de valores y vectores propios. *POOMA* una biblioteca de clases escritas en C++ desarrollada con el objetivo de proveer un ambiente para la programación en paralelo, bajo el modelo de paralelismo de datos [1]. Recientemente se están haciendo algunos desarrollos utilizando el lenguaje Java [13].

Aunque estas bibliotecas han sido optimizadas para un cierto número de arquitecturas, carecen de una interfaz usuario que facilite su utilización para lo cual actualmente es necesario involucrarse en los detalles de programación de las mismas, además estas bibliotecas carecen de un mecanismo que permita lograr su integración e interoperabilidad.

Por las razones antes mencionadas, hemos propuesto el desarrollo de un ambiente interactivo que permita la integración de las diversas herramientas usadas en cálculo científico. En primer lugar se ha considerado el subproblema de resolución de sistemas de ecuaciones lineales, pero se piensa extender el ambiente con funcionalidades para el cálculo de autovalores y autovectores.

La tecnología orientada a objetos ha sido considerada en el proceso de desarrollo, en particular se puede mencionar: el estándar *CORBA: Common Object Request Broker Architecture* definido por el *OMG (Object Management Group)* [9] para la interoperabilidad y portabilidad de aplicaciones distribuidas orientadas a objeto. También se definió un proceso de desarrollo el cual integra el modelo *PAC-Amodeus* [4] y el enfoque *OOSE (Object-Oriented Software Engineering)* [8]. PAC-Amodeus es un modelo para definir arquitecturas de sistemas interactivos el cual está basado en la noción de multiagentes. OOSE es utilizado para la modelación de las funcionalidades del dominio de la aplicación. Es importante mencionar que la utilización de estas técnicas en este dominio particular permitirá evaluar la conveniencia de la orientación a objetos en aplicaciones de cálculo científico intensivo, debido a los requerimientos de eficiencia que estas aplicaciones presentan.

El estándar *CORBA* ha sido utilizado para definir la arquitectura general del ambiente. Esta arquitectura está basada en la noción de componentes distribuidos. Estos componentes, llamados objetos CORBA, poseen interfaces bien definidas que permiten su uso sin que los clientes tengan que conocer detalles de implementación ni donde están localizados físicamente. Los requerimientos son resueltos gracias a la participación del ORB (Object Request Broker), quien intercepta la llamada y es responsable de encontrar el objeto referenciado, enviarle los parámetros, invocar el método y retornar los resultados.

El objetivo de este artículo es presentar un modelo para la interfaz usuario del ambiente, así como las etapas más importantes que permitieron su desarrollo.

Además de esta introducción y las conclusiones, este artículo contiene las siguientes secciones: la sección 2, presenta la definición del problema, el modelo de los usuarios y la definición de los requerimientos. La sección 3 presenta el modelo de la interfaz para el ambiente y una breve descripción de la técnica utilizada para su construcción.

2 *Análisis de Requerimientos*

El objetivo de esta sección es presentar los modelos desarrollados durante la fase del análisis de los requerimientos que debe satisfacer el ambiente. Este análisis ha sido realizado siguiendo un proceso que combina varias técnicas orientadas a objeto las cuales permites separar los aspectos relacionados con el dominio del problema, de los aspectos referentes a la interfaz con los usuarios. Siguiendo este proceso, la primera etapa consiste en definir claramente el problema a resolver para tener una mayor comprensión de los conceptos en el involucrados. Seguidamente se realizó la definición de las funcionalidades que los usuarios desean encontrar en el ambiente. En esta etapa se desarrolló un Modelo de Análisis que presenta la arquitectura de los objetos asociados a los conceptos propios del dominio del problema.

Paralelamente se realizó un estudio de los usuarios para identificar sus características y así poder decidir el nivel de soporte que debe brindar el ambiente. Este estudio permitió el desarrollo de un modelo de la interfaz en el cual se define la interacción física con el usuario que consiste de la presentación de la interfaz y de la definición y estructuración de los objetos encargados de establecer la interacción.

A continuación se describe cada una de las etapas seguidas en este proceso de análisis.

2.1 *Definición del Problema*

El problema a resolver es la construcción de un ambiente integrado e interactivo que permita la utilización de un conjunto de herramientas para la resolución de problemas de álgebra lineal que involucran la manipulación de matrices esparcidas de gran tamaño. En una primera etapa se han considerado las herramientas para la resolución de sistemas de ecuaciones lineales, el cual es uno de los problemas más importantes en el dominio de álgebra lineal.

Un sistema de ecuaciones lineales puede ser definido mediante la ecuación:

$$Ax = b \tag{1}$$

Donde:

$$A \in \mathbb{R}^{n \times n} \ (or \ \mathbb{C}^{n \times n})$$
$$x \in \mathbb{R}^{n} \ (or \ \mathbb{C}^{n})$$
$$b \in \mathbb{R}^{n} \ (or \ \mathbb{C}^{n})$$

El sistema de ecuaciones (1) puede ser resuelto utilizando métodos directos o métodos iterativos. En el caso del ambiente que se desarrolla se han considerado los métodos iterativos basados en las técnicas de proyección en subespacios de Krylov. El objetivo de estas técnicas es extraer de un subespacio de $\mathbb{R}^n$ o $\mathbb{C}^n$ un vector que represente una solución aproximada del sistema de ecuaciones [11].

En el futuro se piensa incorporar un subsistema para el cálculo de autovalores y autovectores.

2.2 *Modelo de los Usuarios*

Los usuarios son investigadores pertenecientes a diversas áreas como:

- Análisis Numérico.
- Física.
- Química.
- Matemáticas.

Todos ellos tienen la necesidad común de procesar matrices esparcidas de gran tamaño de una manera eficiente, tanto en el uso de la memoria del computador como en el tiempo de procesamiento, ellos también plantearon algunas necesidades particulares acerca de los resultados que esperan del ambiente, los cuales serán descritos en la sección 2.3.

Los usuarios poseen algunas limitaciones que permitieron determinar el nivel de soporte que debe ser provisto en el ambiente, entre las cuales se pueden mencionar:

1. No todos tienen un conocimiento profundo acerca de los métodos de Krylov.
2. No todos poseen conocimiento acerca de los modelos de paralelismo.
3. Ellos pueden conocer, a priori, las propiedades numéricas de las matrices: simetría, positivo definición, etc.
4. Los usuarios pueden no conocer las ventajas de los distintos formatos de almacenamiento de las matrices.
5. Los usuarios pueden no conocer las distintas técnicas de precondicionamiento y su influencia en la eficiencia de la ejecución de los métodos.

Del estudio anterior se concluye que se debe implementar un proceso que ayude en la utilización de las herramientas provistas por el ambiente para el cálculo de la solución de un sistema de ecuaciones lineales, también se debe proveer ayuda sensitiva al contexto y un tutorial que instruya a los usuarios acerca de las características de los métodos de Krylov y de las diversas formas de implementación dependiendo de los formatos de almacenamiento y de las formas de precondicionamiento y de la arquitectura donde serán ejecutados los procesos.

2.3 *Definición de los Requerimientos*

El requerimiento principal del ambiente es que permita la integración de diferentes herramientas usadas en cálculo científico, las cuales pueden estar distribuidas a través de una red heterogénea. Esta integración debe lograrse de una forma transparente para los usuarios.

Además debe proveer:

- Facilidad de uso mediante una interfaz gráfica.
- Interoperabilidad entre sus diversos componentes y con sistemas existentes. El objetivo de interoperar con sistemas ya existentes es aprovechar el esfuerzo que se ha hecho en la optimización de los mismos. Esta interoperabilidad debe lograrse a través de un mecanismo transparente de comunicaciones.
- Habilidad de poder manipular diversas estructuras para la representación de los datos y soportar diferentes mecanismos de implementación de las operaciones.
- Independencia de las plataformas de hardware y de software.
- Integración con el Web con el propósito de poder ejecutar la interfaz del ambiente desde cualquier lugar.
- Capacidad de crecimiento incremental. Debe ser posible en cualquier momento agregar nuevos recursos computacionales.

A continuación se describen algunos de los requerimientos presentados por los usuarios para la resolución de sistemas de ecuaciones lineales usando los métodos de Krylov.

- Manipulación básica de matrices esparcidas: entre estas facilidades se pueden mencionar
  1. Recuperar la matriz a partir de un archivo o a partir de una dirección remota.
  2. Visualizar la estructura de la matriz: esto puede hacerse de varias formas, entre las cuales se pueden mencionar:
     - Structure Plots: permite visualizar el patrón de distribución de ceros de la matriz. Un *Structure Plot* es un arreglo rectangular cuyos elementos son puntos. Un punto es negro si el elemento correspondiente en la matriz es no nulo, en caso contrario es blanco. En la figura 1 se presenta un ejemplo del *Structure Plot* de una matriz de 42x42 con 520 entradas de las cuales hay 505 entradas distintas de cero.

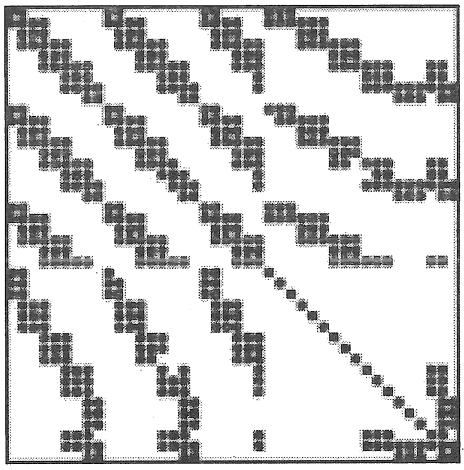

Figura1. *Structure Plot* de una matriz esparcida

     - Cityplot: permite visualizar el patrón de distribución de ceros de la matriz así como la magnitud relativa de los elementos de la matriz. Por cada entrada de la matriz se dibuja una barra tridimensional cuyo color y altura son linealmente dependientes de la magnitud del valor. La primera fila de la matriz es colocada en el tope del *cityplot* y la primera columna está a la izquierda. El *cityplot* es dibujado en perspectiva tal que la primera fila parece mas estrecha que la última fila. Un esquema de la matriz es dibujado en el plano de una entrada con el valor cero.
       Para las matrices complejas se gráfica el valor absoluto de cada elemento.
       En la figura 2 se presenta un ejemplo del *CityPlot* de una matriz esparcida de 42x42 con 520 entradas de las cuales hay 505 entradas distintas de cero.
       La barra a la derecha presenta la escala de colores asociada a las diferentes magnitudes y una etiqueta con la máxima y la mínima magnitud.

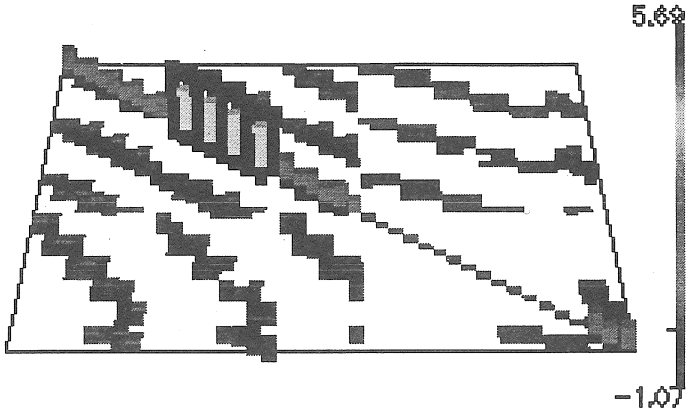

Figura2. *Cityplot* de una matriz esparcida

3. Seleccionar el elemento que ocupa una posición determinada.
4. Seleccionar una región de la matriz (fila, columna, diagonal, etc.).
5. Visualizar las propiedades numéricas de una matriz (simetría, positivo-definición, singularidad, etc).
6. Visualizar las estadísticas de una matriz.
   - Tamaño: indica el número de filas (M) y el número de columnas (N) de una matriz. También se debe indicar el número de entradas en la matriz.
   - Tipo de los valores de la matriz.
   - Cantidad de elementos no nulos: se debe indicar la cantidad de elementos no nulos en la matriz, la consulta puede hacerse para la diagonal, para las triangulares sobre o debajo de la diagonal, para una fila o una columna particular. También se puede consultar el número total de elementos no nulos de la matriz.
   - Estadísticas de las filas/columnas: se debe proveer el promedio y la desviación estándar del número de elementos no nulos por filas/columnas.
7. operaciones algebraicas: el usuario puede ejecutar operaciones algebraicas sobre matrices y vectores (suma, producto escalar, producto de dos matrices, producto entre una matriz y un vector, triangularización, etc.)

– Para la resolución de sistemas de ecuaciones lineales: la interfaz debe realizar la interacción con el usuario a fin de solicitar toda la información necesaria para la ejecución de un métodos y para mostrarle los resultados una vez que la ejecución ha finalizado. La interfaz debe permitir:
  - Seleccionar un método a partir de una lista de métodos.
  - Suministrar los parámetros requeridos por el método seleccionado: tamaño del subespacio, iterado inicial, número máximo de iteraciones, etc.
  - Seleccionar el formato para la representación de la matriz.
  - Seleccionar la forma de ejecución del método (secuencial, paralelo).
  - Seleccionar la técnica de precondicionamiento.
  - Seleccionar un test de convergencia.
  - Visualizar los resultados:
    - vector solución.
    - vector residual y norma del vector residual.
    - número de iteraciones efectuadas.
    - velocidad de convergencia: en este caso, si el usuario así lo requiere, se puede presentar la curva de convergencia del método. Esta curva muestra la norma del residual en cada iteración.
    - nro. de operaciones punto flotante ejecutadas y tiempo total de procesamiento.
    - En el caso de que se haya seleccionado el modo de ejecución en paralelo sería interesante mostrar la cantidad de datos transmitidos y la cantidad de tiempo utilizada para la transmisión de datos.

- También se debe proveer facilidades que permitan la construcción de tareas a partir de operaciones elementales, durante la construcción, el usuario puede expresar el flujo de ejecución de las subtareas, el cual puede ser secuencial o en paralelo. Por esta razón se debe definir un lenguaje gráfico que permita expresar el modelo de ejecución.
- El usuario debe poder seleccionar una máquina en particular donde quiere que se realicen los cálculos, para ayudar en la selección se debe proveer la facilidad de consultar la lista de recursos disponibles en los servidores pertenecientes al ambiente.

Le definición de requerimientos presentada anteriormente permitió la construcción del Modelo de Análisis cuyo diagrama de clases se presenta en la figura 3, este diagrama ha sido especificado usando la notación UML [12], y será brevemente descrito en la sección 3.

El estudio de los usuarios permitió el desarrollo de un modelo de la interfaz el cual se presenta en la siguiente sección.

3 Modelo de la Interfaz

Para la organización de los componentes que conforman la arquitectura del ambiente se ha seleccionado el modelo PAC-Amodeus para sistemas interactivos, este modelo está basado en la noción de multiagentes e integra los modelos Arch/Slinky y PAC [4], [3]. El modelo Arch/Slinky considera que un sistema interactivo está compuesto por: el Núcleo Funcional (NF), la Interfaz con el Núcleo Funcional (INF), el Controlador de Diálogo, el Componente de Técnicas de Presentación y el Componente de Técnicas de Interacción de Bajo Nivel. El modelo PAC permite expresar la arquitectura de un sistema interactivo como una jerarquía de de agentes especializados en la interacción con el usuario. La integración de estos dos modelos es lograda mediante la reutilización de los componentes definidos por el modelo Arch/Slinky y el refinamiento del Controlador de Diálogo en términos de agentes PAC.

A continuación se describen los componentes del ambiente en términos del modelo PAC-Amodeus.

1. **Núcleo Funcional**: implementa los conceptos específicos al dominio de la aplicación. El modelo de este componente es presentado en el diagrama de clases de la figura 3. Este diagrama está formado por las clases y las relaciones que representan los conceptos del dominio de álgebra lineal, tales como matriz, vector, etc. También representa conceptos relacionados con la manipulación computacional de matrices esparcidas, tales como, métodos numéricos y formatos de almacenamiento. Entre las principales clases pertenecientes al diagrama se puede mencionar:
   (a) *Matrix:* esta clase representa el concepto de matriz. Contiene atributos que definen las características de una matriz, tales como: dimensión de las filas, dimensión de las columnas y el número de elementos no nulos. Ha sido definida como una clase genérica en función del formato en el cual pueden ser almacenados sus valores, lo cual involucra diferentes implementaciones de los métodos. Se han identificado también operaciones básicas, como suma, cálculo de normas, producto, etc. Esta es la raíz de una jerarquía de clases donde la especialización depende dependen de la estructura de la matriz.
   (b) *Format:* representa el concepto de formato de almacenamiento. Existen diversos formatos en función de las diferentes arquitecturas utilizadas para el cálculo y en función de las características estructurales de la matriz. El objetivo de estos formatos es aprovechar la estructura de la matriz para optimizar el espacio de almacenamiento y el tiempo de cálculo de las operaciones. Es una clase genérica que depende del tipo base de los elementos que conforman la matriz. En esta clase se han identificado principalmente operaciones para el acceso a los elementos.
   (c) *LAMethod*: representa el concepto de método numérico. Sus instancias son los distintos métodos para resolver sistemas de ecuaciones lineales. Estos métodos pueden ser directos o iterativos y pueden ser escritos utilizando un enfoque secuencial o paralelo.
2. **Controlador de Diálogo**: tiene la responsabilidad de secuenciamiento a nivel de las tareas. Una tarea puede ser definida como un requerimiento del usuario. El Controlador de Diálogo recibe eventos desde el Núcleo Funcional vía la interfaz INF y desde el usuario vía el componente de Técnicas de Presentación.

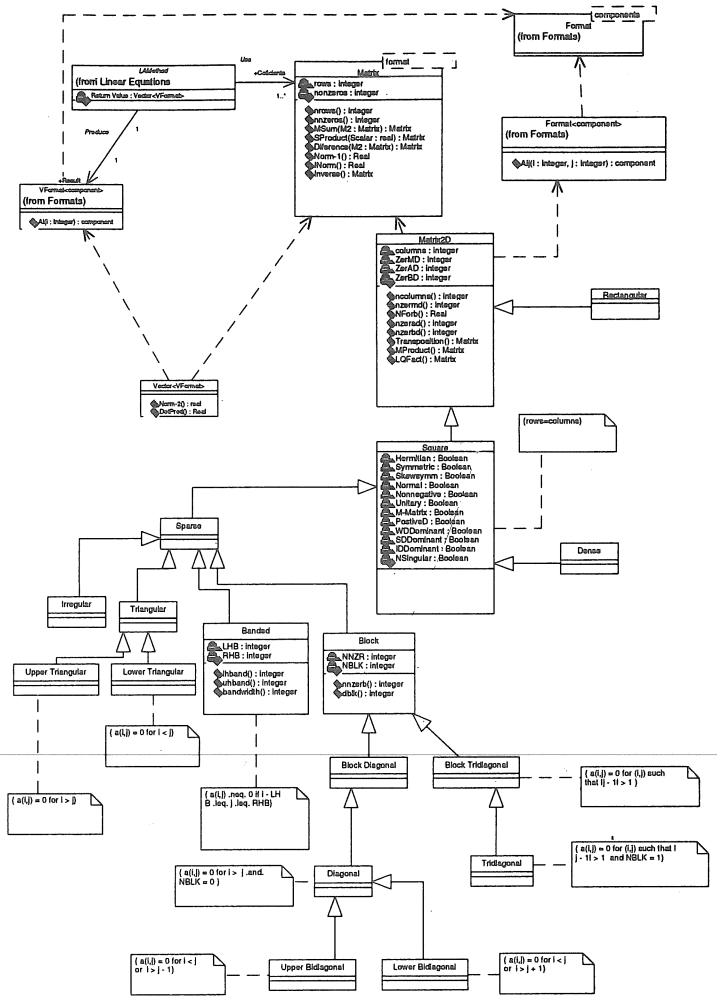

Figura3. Diagrama de Clases del Componente "Dominio de la Aplicación"

El Controlador de Diálogo está compuesto por una jerarquía de agentes PAC. Un agente es una unidad de procesamiento compuesta de tres perspectivas:

- *Presentación*: define la imagen del agente, es el comportamiento perceptible por el usuario. Está relacionado con el componente de técnicas de presentación. Un agente puede no tener perspectiva de presentación.
- *Abstracción*: es el núcleo de las funcionalidades del agente. Esta perspectiva mantiene el estado abstracto del agente y puede estar relacionada con algunos objetos conceptuales del componente Núcleo Funcional.
- *Control*: permite establecer la comunicación entre las perspectivas de Abstracción y Presentación así como con los otros agentes de la jerarquía.

Estas perspectivas son usadas para expresar funciones diferentes pero muy bien complementadas.

La noción de jerarquía introducida por el modelo PAC puede ser explotada ampliamente para definir niveles de abstracción. El agente que se encuentra en la raíz de la jerarquía representa la interfaz con funcionalidades del sistema.

En los niveles intermedios los agentes son usados para representar combinaciones de relaciones y de niveles abstracción. Una relación típica es la relación de composición, esto significa que un agente PAC puede estar compuesto por otros agentes PAC, definiendo un nuevo nivel de abstracción.

En el nivel más bajo de la jerarquía, se encuentran agentes PAC elementales los cuales constituyen las mínimas unidades de interacción, como por ejemplo un botón, un menú, etc.

Una de las principales ventajas del modelo PAC, con respecto a otros modelos de arquitecturas de sistemas interactivos, es la separación de las funcionalidades entre las distintas perspectivas de cada agente lo cual facilita la modificabilidad de una perspectiva sin afectar las otras. Además en PAC se expresa explícitamente cómo se realizan las comunicaciones entre los distintos agentes.

3. **Interfaz con el Núcleo Funcional**: este componente mantiene la asociación entre los conceptos del dominio de la aplicación y los conceptos propios del controlador de diálogo, implementando las funciones de comunicación y las eventuales transformaciones de los datos entre los distintos formalismos utilizados, como ejemplo se puede mencionar el caso particular de la matriz esparcidas; es necesario contar con una interfaz que permita consultar la información acerca de los valores de las instancias de la clase *Matrix* y transformarlos en información que pueda presentada al usuario.
4. **Técnicas de Presentación**: este componente implementa la interacción física con el usuario vía hardware y software. Está compuesto por los objetos que permiten desplegar información, por ejemplo tablas, gráficos, etc. En nuestro caso se ubican en este nivel las técnicas empleadas para la representación gráfica de los tareas que se construyen, de las matrices (structure plot, cityplot), etc.
5. **Técnicas de Interacción de Bajo Nivel**: este componente esta representado por las facilidades provistas por la plataforma subyacente. Este componente incluye las facilidades básicas de interacción tales como el sistema de ventanas, el *toolkit*, etc. En el caso de nuestra aplicación particular se consideraran para la implementación las técnicas provistas por $Java^{TM} Swing$ [7].

A continuación se presenta una descripción detallada de Controlador de Diálogo.

3.1 Controlador de Diálogo

Antes de describir los agentes PAC que conforman este componente, se presentará un ejemplo de la presentación de la interfaz usuario. En particular consideraremos la ventana para la construcción de un programa, debido a que constituye una de los principales puntos de interacción con el usuario, es de hacer notar que los eventos enviados por el usuario generan el despliegue de ventanas particulares según la naturaleza del evento.

Durante el diseño de la presentación de esta ventana se identificaron los componentes que se describen a continuación.

- Marco: contiene el nombre del sistema de ecuaciones lineales que se está resolviendo, un botón que despliega un menú para realizar operaciones sobre la ventana, tales como: inconificar, maximizar, cerrar, mover, etc. También posee botones que permiten realizar directamente, sin necesidad de navegar en el menú, las operaciones mencionadas anteriormente para la manipulación de las ventanas.
- Menú: este componente presenta la lista de las opciones que permiten utilizar las funcionalidades ofrecidas por el subsistema de resolución de ecuaciones lineales.
- Barra de Herramientas: presenta un conjunto de botones que permiten utilizar las funcionalidades ofrecidas por el subsistema de resolución de ecuaciones lineales. Este componente es otra forma de representación del menú mencionado en el párrafo anterior. El contenido de la barra de herramientas cambia dinámicamente según el subsistema con el que interactúa el usuario, debido a que las facilidades provistas depende del contexto de trabajo.
- Contenido: el aspecto visual de este componente depende del procesamiento que se realice. Por ejemplo durante la construcción de un programa, esta área contendrá un gráfico que representa el modelo de ejecución de cada subtarea que compone el programa. En el caso de este ambiente se ha seleccionado representar los programas como un árbol. (ver parte superior izquierda de la figura 4). También se presenta información referente a cada una de las subtareas que componen el gráfico (ver parte inferior izquierda de la figura 4).
  Al final de la ejecución de una operación, este componente contendrá información referente a los resultados que se deben mostrar al usuario. Por ejemplo, el gráfico que representa el vector solución del sistema de ecuaciones lineales o el cityplot de una matriz esparcida (ejemplo ver figura 2).
- Barra de Estado: contiene información diversa acerca del estado de ejecución de los procesos que se realizan en el sistema.

El controlador de diálogo contiene otros componentes que representan los elementos de interacción con el usuario para solicitar la información necesaria en cada una de las fases de utilización del ambiente.

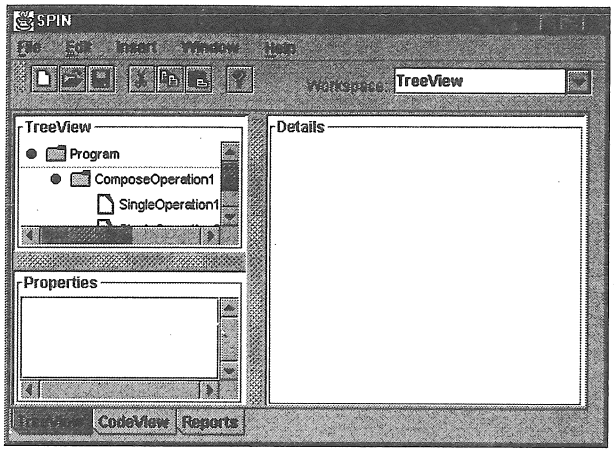

Figura4. Esquema de la interfaz usuario

La figura 4 presenta un esquema de la interfaz usuario del ambiente durante el proceso de construcción de tareas. En este esquema el árbol que representa las tareas que conforman el programa son representadas usando la noción de *carpeta* para las tareas complejas y de *archivo* para representar las tareas elementales.

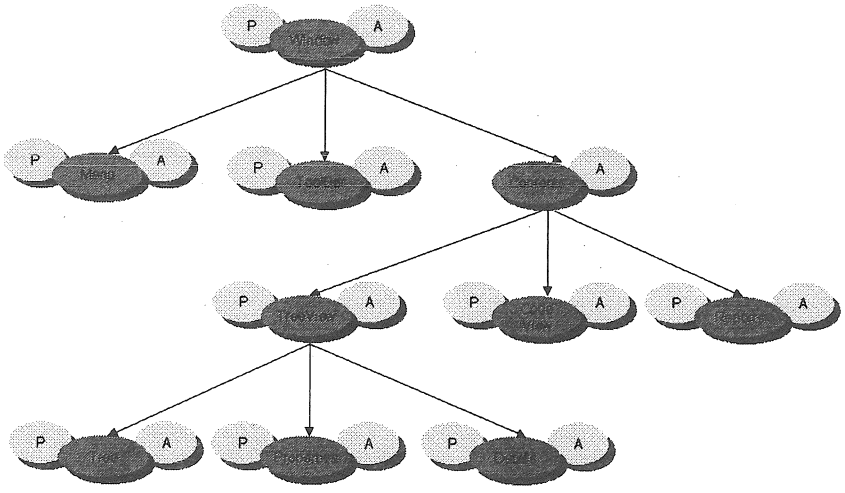

Figura5. Diagrama Pac de la interfaz usuario

Para implementar el comportamiento interactivo de cada uno de los componentes descritos anteriormente se han definido agentes PAC, los cuales son representados en el diagrama que se muestra en la figura 5. En este diagrama cada agente PAC es representado como un trio de óvalos. Cada óvalo corresponde a una de las perspectivas del agente: abstracción, control y presentación. El óvalo central representa la perspectiva de control, el óvalo a la izquierda representa la perspectiva de presentación y el óvalo a la derecha representa la perspectiva de abstracción.

El agente denominado *Window* permite implementar el comportamiento interactivo de los componente gráficos del subsistema. Es un agente compuesto por los agentes *Menu, ToolBar, Contents* y *Status Bar*. Estos agentes implementan, respectivamente, las funcionalidades interactivas de los componentes Menú, Barra de Herramientas, Contenido y Barra de estados descritos anteriormente.

La perspectiva de *presentación* del agente *Window* contiene entre otros elementos: el título de la ventana y los botones que permiten realizar la manipulación de la misma, tal como se explicó para el componente *Marco*. La perspectiva de *abstracción* contiene la información interna de la ventana tal como su posición en la pantalla y el estado de los botones (seleccionado o no). La perspectiva de *control* permite la comunicación entre las perspectivas de presentación y abstracción, por ejemplo cuando el usuario selecciona un botón la perspectiva de presentación envía un mensaje al control para indicar que el elemento que ocupa la posición determinada ha sido seleccionado el control transfiere la información a la abstracción quien cambia el estado del elemento e inicia las acciones correspondientes a la selección. Siguiendo este modelo han sido diseñados todos los componentes de interacción del ambiente.

Actualmente el controlador de diálogo se encuentra en la fase de implementación, para lo cual se ha seleccionado el lenguaje de programación $Java^{TM}$ y los componentes $Java^{TM}Swing$ [14], [7].

4 Conclusiones y Perspectivas

En este articulo ha sido presentado un modelo para la interfaz usuario de un ambiente para la resolución de sistemas de ecuaciones lineales. La importancia de esta interfaz es que ella permite la integración de un conjunto de herramientas facilitando su utilización. El modelo de interfaz ha sido desarrollador empleando un proceso que combina las técnicas orientadas a objetos con el modelo PAC-Amodeus. El enfoque orientado a objeto ha sido también utilizado durante el proceso completo de desarrollo del ambiente, lo cual permitirá la extensión del ambiente a través de la incorporación de nuevos tipos de matrices, nuevos formatos de almacenamiento y nuevos métodos de cálculo.

La técnica empleada para el modelo de la arquitectura del ambiente permite la implementación independiente de los aspectos de interfaz y del dominio del problema. Los distintos modelos han permitido una comunicación eficiente con los utilizadores.

El ambiente ha sido diseñado siguiendo un enfoque basado en la noción de componentes para cuya integración se sigue el estándar CORBA. Este modelo de componentes facilitará las extensiones del ambiente. Actualmente se trabaja en la incorporación de un subsistema para el cálculo de autovalores y autovectores. También se considerará un conjunto de componentes para servicios de información general tal como un servicio de directorio que permita conocer los recursos disponibles en el conjunto de servidores.

Referencias

1. S. Atlas, S. Banerjee, J. Cummings, P. Hinker, M. Srikant, V. Reynders, M. Tholburn, W. Humphrey, S. Karmesin, and K. Keahey. POOMA : A Framework for Scientific simulation on parallel Architectures. http://www.acl.lanl.gov/PoomaFramework/papers/pooma.ps.
2. R Barret et al. *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods.* SIAM, Philadelphia, 2nd edition, 1994.
3. J. Coutaz. *Interface Homme-Ordinateur.* Dunod, France, 1990.
4. J Coutaz. *Formal Methods in Human-Computer Interaction*, chapter 3. Software Architecture Modelling: Bridging Two Worlds Using Ergonimics and Software Properties, pages 49–73. Formal Approaches to Computing Information Technology. Springer, 1998.
5. J. Dongarra et al. LAPACK Working Note, 1995. http://www.netlib.org/utk/papers/pblas/pblas.html.
6. J. Dongarra, R Pozo, and D Walker. LAPACK : A Design of Object-Oriented Extensions for High Performance Linear Algebra. *Proceedings of the Supercomputer 93 Meeting*, pages 162–171, November 1993.
7. Mageland Institute. Fundamantals of JFC/Swing, Part 1. Short Course. Online Traning. Java Developers Connection. http://java.sun.com, april 1999.

8. I. Jacobson, M. Christenson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering. A Use Case Driven Approach.* Addisson-Wesley Publishing Company, New York, 1992.
9. Object Mangement Group Inc. The Common Object Request Broker : Architecture and Specification. ftp://ftp.omg.org/pub/docs/formal/98-02-01.ps.gz.
10. Y. Saad. SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations. Technical Report 90-20, NASA Ames Research Center, 1991.
11. Y Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, 1996.
12. Rational Software Corporation. UML Notation Guide. http://www.rational.com/-uml/html/notation.
13. G. W. Stewart. JAMPACK: A Java Package for Matrix Computations. Department of Computer Science. Institute for Advanced Computer Studies. University of Meryland and Mathematical and Computations Sciences Division. NIST. ftp://thales.cs.umd.edu/pub/Jampack/Jampack/AboutJampack.html, 1999.
14. Sun Microsystems. The $Java^{TM}$ Language: An Overview. Technical report, 1995.