Análisis Primario de Requerimientos
$B^{I}C_{O}TI$-II

Marcelo Alarcón
Claudio Risso
Carlos Soderguit

12 de Setiembre 1999

1 Introducción

Algunos años atrás el *Grupo de Tratamiento de Imágenes del Instituto de Ingeniería Eléctrica*, comenzó el desarrollo de una biblioteca sobre la cual implementar aplicaciones en el tema. Si bien existían herramientas de prototipación para tratamiento de imágenes (e.g. Khoros), tener una biblioteca propia añadía importantes ventajas. Entre las mas obvias estaban la eficiencia de los algoritmos y la portabilidad, pero sin duda la principal ventaja potencial tenía que ver con el re-uso. Como estrategia a largo plazo la biblioteca de imágenes era sin duda una importante apuesta al futuro.

En esos tiempos la orientación a objetos como técnica de programación (relativamente joven en nuestro país) prometía ser una buena opción para cumplir con ese fin y se comenzó el desarrollo en C++. Lamentablemente en los hechos el éxito de la biblioteca de imágenes fue muy limitado, sobre todo en lo que tuvo que ver con su meta principal. No nos extenderemos en las causas del fracaso, estas fueron estudiadas en detalle en[RRV99]

$B^{I}C_{O}TI$[1] como proyecto, surge en 1997 como un segundo intento para desarrollar la biblioteca de tratamiento de imágenes del instituto. Impulsado

[1] BIblioteca de COmponentes de Tratamiento de Imágenes

principalmente por la formación en este de investigadores especializados en el desarrollo de software, por el *know-how* adquirido por el grupo tanto en tratamiento de imágenes como en orientación a objetos durante los años en que se trabajo con la primera biblioteca y sin duda también por el fracaso ya a esa altura evidente de esta última.

Este segundo proyecto globalmente es mas ambicioso que el primero, ya que no solo intenta ser lo que el anterior no pudo en cuanto se refiere a "Representación y Procesamiento de Imágenes Digitales" sino que además incorpora algunos aspectos nuevos como ser: "Mecanismos de Visualización e Interacción Simultánea", "Ambiente de Prototipación de Alto Nivel" y "Posibilidades para Importar y Exportar Objetos en Diversos Formatos". Todo esto además dentro de requerimientos de portabilidad a diversas plataformas.

En el proyecto BICOTI-I [RRV99], entre otras cosas, se definió la arquitectura a un nivel macro[2] para la biblioteca en general y se llegó a un nivel de detalle[3] en la parte de "Representación y Procesamiento de Imágenes Digitales".

BICOTI-II tiene como objetivo primario resolver dos de los puntos establecidos en los objetivos generales del proyecto que BICOTI-I dejó planteados. Estos son en particular: "Mecanismos de Visualización e Interacción Simultánea" y "Posibilidades para Importar y Exportar Objetos en Diversos Formatos".

[2]Modelo en capas.

[3]Especificación precisa, diseño e implementación

2 Presentación de la Arquitectura

A continuación se presentará la arquitectura tal como se definió en la primer parte del proyecto[RRV99].

2.1 Modelo en Capas

Un esquema del modelo en capas de BICOTI-I es el presentado en la figura 1.

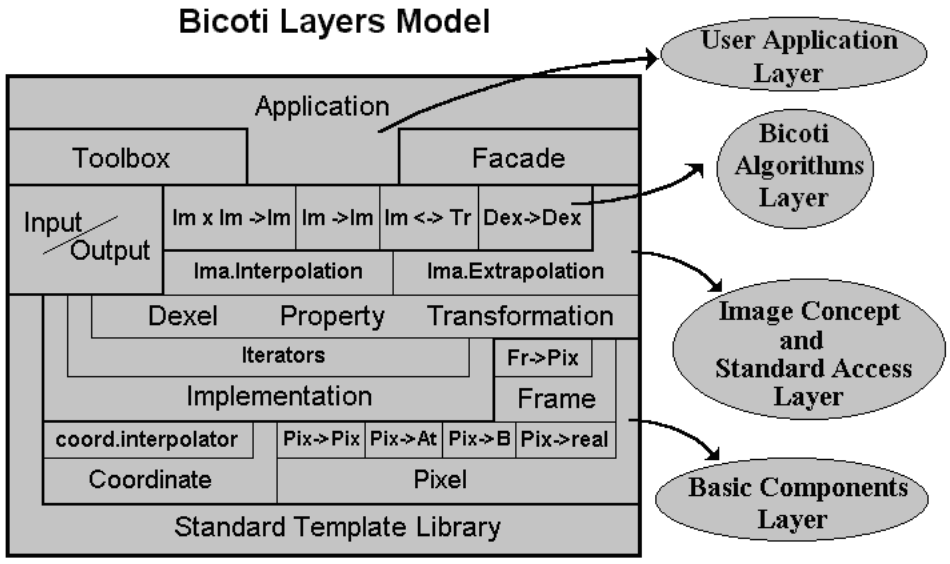

Figura 1

En el esquema se aprecia claramente que el módulo *INPUT/OUTPUT*[4], existe casi como una entidad autónoma dentro del sistema. Desde este módulo solo es importante conocer las diversas implementaciones para realizar las conversiones de formato desde y hacia esta; el procesamiento en sí, se da exclusivamente en las capas de algoritmos.

[4]Observar que de acuerdo a lo expuesto en la introducción, este sería exactamente el punto a resolver en esta etapa del proyecto

2.2 Macro-Arquitectura del Módulo

Es conveniente pensar en este módulo a dos niveles:

- A *Manejo de Archivos* le compete todo lo referente a recuperación y Almacenamiento de Objetos Gráficos[5]. Es esperable poder leer y grabar desde y hacia los formatos estándar de archivos(e.g. BMP, GIF, JPEG, TIFF, etc.), pero también es probable que aparezca en el futuro la necesidad de incluir algún formato propietario[6].

- A la *Visualización e Interacción Gráfica* por otra parte, le compete la tarea de proveer de mecanismos para visualización e interacción con las imágenes de BICOTI. A modo breve sería deseable poder desplegar en pantalla imágenes de 2 o 3 dimensiones, color o BW, e incluso lograr a través del mouse o el teclado por ejemplo, interactuar con cierto grado de procesamiento previamente establecido.

Cabe observar que ambos sub-módulos son básicamente independientes. Un esquema de la interacción con el resto del sistema, se presenta en la figura 2.

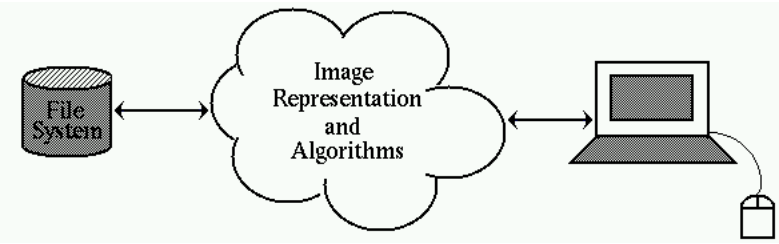

Figura 2

Ya ubicados los módulos en el sistema y esquematizada su interacción, procederemos a fijar las cualidades esperables de estos.

[5]Como se desprende de la documentación de BICOTI[RRV99] los objetos gráficos o sea aquellos usados para representar la imagen, son mas generales que el mapa de *bits*

[6]Cabe aclarar que esto último, escapa al alcance de nuestro proyecto. Aún así, por conveniencia práctica, incluiremos soporte para algunos formatos básicos. Es de suponer que esto sirva de ejemplo para en una etapa posterior redondear el módulo.

3 Consideraciones de Diseño

Es imprescindible que el sistema cumpla con las siguientes propiedades:

1. *Transparencia*: Es fundamental que en todo momento, el usuario que desarrolla una aplicación usando nuestras rutinas de interacción gráfica, lo haga con absoluta transparencia. Esto es, en el momento de usar las rutinas, el usuario no debería tener por que conocer, ni lidiar con los detalles de implementación particulares del caso. Es necesario fijar una interface clara para todos los métodos implementados de forma tal, que sea posible usar una aplicación en forma independiente de, por ejemplo la biblioteca de visualización elegida y/o el *hardware* subyacente.

2. *Flexibilidad*: Debe ser posible, integrar cualquier biblioteca tanto de visualización como de almacenamiento de archivos gráficos existente, que cumpla con los requerimientos mínimos necesarios para lograr las prestaciones requeridas. Mas aún, dado el grado de performance requerido para ciertas aplicaciones como interacción con imágenes 3D, es razonable extender este requerimiento para incorporar *hardware* especializado.

3. *Grado de Interacción*: Hay que fijar el grado mínimo de interacción requerido hacia el sistema. Entre otras prestaciones es necesario contar con formas de "Mostrar Imágenes", "Recibir eventos desde el Mouse o Teclado".

4. *Escalabilidad*: En complemento a lo anterior, la interacción no debe limitarse a la fijada en principio. En cambio debe proveerse un mecanismo que permita extender la funcionalidad básica del módulo, y este debe estar perfectamente definido.

5. *Portabilidad*: Si bien no es realista pensar que este módulo logre el grado de portabilidad de BICOTI-I [7], es imprescindible sortear esta dependencia mediante mecanismos abstractos de interacción y manejo de archivos.

[7] Tener en cuenta que no existen bibliotecas estándar, ni para visualización ni para el manejo de archivos gráficos en C++.

4 Consideraciones Generales

A la luz de lo expuesto en los puntos anteriores, parece claro que las biblioteca de visualización y manejo de archivos elegidas, cumpliendo con las prestaciones mínimas, son irrelevantes. De todos modos se integrarán efectivamente bibliotecas en ambos casos, con el fin de ejemplificar la solución propuesta.

Se ha hecho un esfuerzo en seleccionar bibliotecas adecuadas. La factibilidad de la elección, está sujeta claro, al alcance tanto en el grado de interacción, como en la diversidad de los formatos gráficos manejables; ambos inciertos al momento, pero seguramente definidos una vez terminada la etapa de "Análisis y Especificación Formal de Requerimientos".

A pesar de lo anterior y basados en lo sugerido en el punto 3.3, el equipo ha estudiado algunas alternativas posibles entre las que destacamos *VTK*[SML98] y *Java 3D*[BOU99] para visualización e interacción así como *Image Magick*[PONT98] para el manejo de archivos con formatos gráficos estándar.

En principio todos se muestran factibles de ser usados. Aunque el grado de visualización e interacción permitido por ambos paquetes es similar, la opción VTK presenta dos ventajas destacables:

- *Performance.* VTK a diferencia de Java3D es compilado, con la consecuente mejora en el tiempo de ejecución.
- *Integración con el Lenguaje.* VTK esta completamente escrito en C++, el mismo lenguaje de programación elegido para el resto de $B^{I}C_{O}TI$. En consecuencia es esperable surjan menos problemas de integración.

References

[RRV99] Claudio Risso, Rodolfo Rodríguez, Álvaro Valdés.
Documentación de Bicoti-I.
URL http://www.iie.edu.uy/interno/g+i/bicoti.htm

[SML98] Will Schroeder, Ken Martin, Bill Lorensen.
The Visualization Toolkit. Second Edition.
ISBN 0-13-954694-4. Prentice Hall PTR.

[PONT98] E. I. du Pont de Nemours and Company.
Image Magick.
URL http://www.wizards.dupont.com/magick/ImageMagick.html

[BOU99] Denis J. Bouvier.
Getting Started with the Java 3D™ API.
URL http://sun.com/desktop/java3D/collateral