Transporte de Programación por Emulación de Bibliotecas

Parte 2: Caso GLIDER a Plataforma WINDOWS 95/98/NT

Marta Sananes
Fernando Sayago
Instituto de Estadística Aplicada y Computación (IEAC), FACES
Centro de Simulación y Modelos (CESIMO), FAI,
Universidad de Los Andes
sananes@faces.ula.ve

Mérida, Abril 1999

Abstract

En esta segunda parte se reporta el caso de transporte del Lenguaje de Simulación GLIDER y su ambiente de desarrollo originalmente para plataforma PC-DOS-Borland TurboPascal/GRAPH a la a plataforma PC-WINDOWS 95/98/NT-Delphi. Estrategias de portabilidad: (1) Construcción de Módulos de Emulación de las Librerías standard de soporte de aplicaciones en la plataforma de partida sobre Librerías disponibles en la de llegada. (API gráfica de Windows embebida en la herramienta de desarrollo Delphi en forma de una jerarquía de Clases de Objetos y recursos de desarrollo propios de Delphi). (2) Diseño de programación: criterio de separación en componentes Núcleo-Control-Presentación. **Objetivos**: (1) Conservar al máximo el código original. (2) Proporcionar a los usuarios estilo visual y operacional que conserve la familiaridad con la versión original. (3) Aprovechar con eficiencia los recursos propios de las plataformas destino (4) Producir Librerías de uso general que permitan el fácil transporte de otras aplicaciones en la misma plataforma de origen o similares. **Metodología**: Desarrollo por etapas: (1) Estudio de factibilidad (2) Estudio de Clases y Librerías de soporte en el destino (3) Preparación de interfaces de Control a Librerías en el destino (4) Diseño, construcción, prueba y ajuste de cada elemento transportado. (5) Ajustes necesarios en el código original. **Resultados**: Librería de Emulación: Comunicación en modo texto, graficación, gestión de archivos y presentación. Versión transportada del sistema original. Manuales de usuario. Conjunto de pruebas de verificación de consistencia. **Conclusiones**: La conservación del código original facilita el mantenimiento y la generación de nuevas versiones. La estrategia es aplicable a diversas plataformas de destino.

Reconocimientos

Al Consejo de Desarrollo Científico y Humanístico de la ULA por el financiamiento al Proyecto del cual es parte este trabajo.

Introducción

Este trabajo se realizó como parte del cumplimiento del objetivo de construír versiones multiplataforma dentro del proyecto de desarrollo del Lenguaje de Simulación GLIDER [1] y de su ambiente integrado.

El Lenguaje de Simulación GLIDER se diseñó tomando el lenguaje de propósito general TurboPascal [2], una extensión popular a la definición Pascal standard, como lenguaje subyacente. La implementación del Compilador y Módulos para la versión original en plataforma PC-DOS se programó con el mismo TurboPascal (versión 6.0) para PC-DOS.

Para realizar el trabajo de transporte o conversión a nuevas plataformas se ha adoptado la estrategia de sustituír las Librerías de soporte o Librerías de la plataforma de partida, MSDOS-BORLAND TURBO Pascal en este caso, por Librerías que las *emulen* en las nuevas plataformas, cuando sea necesario.

El trabajo de transporte a distintas plataformas se facilita si en la programación se adopta un criterio de separación de componentes o módulos en tres categorías [3]:

- **Núcleo**: independiente de plataforma
- **Presentación**: totalmente dependiente de plataforma
- **Control o Enlace**: interfaz para Presentación

[1] Diseño original: C. Domingo y M. Hernandez. Diseño extendido: C.Domingo, J.G. Silva, G.Tonella. Grupo de Desarrollo GLIDER: C.Domingo, G. Tonella, J.G. Silva, H. Hoeger, T. Jiménez, S. Quiroz, M. Sananes, O. Terán, K. Tucci. (ULA, Venezuela). Consultores: R. del Canto (ULA), C. Zoltan (USB). ©CESIMO-IEAC, Universidad de Los Andes, Mérida, Venezuela

[2] Turbo Pascal y todos los productos de Borland son marcas registradas de Borland International, Inc.

[3] Metodología delineada en solicitud de financiamiento al programa BID/CONICIT por Marta Sananes y Giorgio Tonella

Este enfoque aplicado al desarrollo de nuevos productos de programación, facilita conversión a otras plataformas.

Objetivos

Para la realización de trabajos de conversión o transporte hacia distintas plataformas, nos fijamos los siguientes criterios:

1. Conservar al máximo posible el código original de la plataforma de partida, para reducir al mínimo la necesidad de reprogramar, ahorrar tiempo y evitar introducir errores.

2. Proporcionar a los usuarios similar estilo visual y operacional en las distintas plataformas

3. Aprovechar con eficiencia los recursos propios de las plataformas destino

4. Producir Librerías de uso general que permitan el fácil transporte de otras aplicaciones en la misma plataforma de origen o similares.

Metodología

Para cada conversión se pasa por las siguientes etapas:

1. Estudio de factibilidad

2. Estudio detallado de las Librerías de soporte y lenguajes disponibles en el destino

3. Preparación de interfaces de enlace hacia las Librerías en el destino

4. Diseño, construcción, prueba y ajuste de cada elemento emulado.

5. Ajustes necesarios en el código original para adaptación a la plataforma de destino, cuando -como es frecuente- no hay compatibilidad total entre los lenguajes de desarrollo

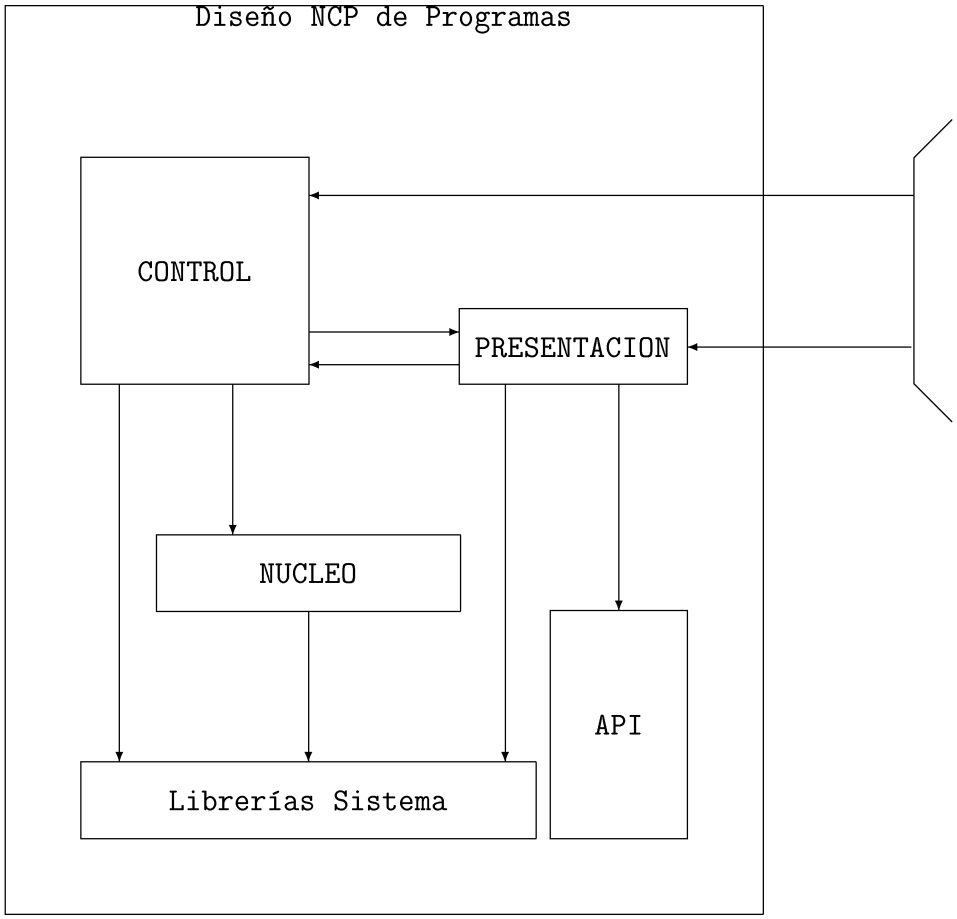

Figure 1: Estructuración de Programas NCP

Factibilidad

Se inició con exploración de los recursos disponibles en las plataformas de destino. Para el transporte del GLIDER a plataforma PC-WINDWS se estaba considerando usar los lenguajes C/C++ o TurboPascal for Windows. Para usar la alternativa C/C++ se exploró sobre la existencia de sistemas automáticos de conversión, que no consideramos satisfactorias. Por consiguiente, el transporte a C/C++ requiere reprogramación total, contradiciendo el primer objetivo propuesto.

Se continuó se estudiando la versión TurboPascal for Windows de Borland para Windows 3.x. En 1995, mientras se trabajaba en la versión para Unix Solaris-XWindow, se anunció la introducción en el mercado de herramientas de desarrollo de software para plataforma PC-WINDOWS del nuevo producto de BORLAND, Delphi, definido como herramienta tipo RAD (Rapid Application Development) como competidor de Microsoft VisualBasic y otros en la misma categoría. El lenguaje de programación soportado por Delphi es una versión de BORLAND del ObjectPascal [4] [6] , originalmente introducido por Apple Co. Posteriormente BORLAND se transformó en la empresa Inprise Corporation (Integrating the Enterprise), manteniendo el nombre BORLAND como marca comercial.

Se decidió adoptar Delphi/ObjectPascal como herramienta y lenguaje de desarrollo para la plataforma PC-WINDOWS. ObjectPascal es una extensión Orientada a Objetos de Pascal en cuyo diseño participó también el Prof. Wirth y que ha sido revisada y extendida por BORLAND. Es altamente compatible con TURBO Pascal. El ambiente DELPHI proporciona herramientas de programación visual, de manejo de recursos de computación, de desarrollo con SMDB en modelo Cliente/Servidor y de acceso a recursos de Internet, todo envuelto en una impresionante jerarquía de Clases de Objetos. Actualmente está disponible la versión 4 para WINDOWS 95/98/NT, con la cual estamos trabajando actualmente.

Para el mantenimiento de versiones para UNIX y LINUX un grupo de desarrolladores asociados al GNU, liderado por Peter Gerwinski, en Alemania, están desarrollando versiones de ObjectPascal para diversas plataformas UNIX, entre ellas Solaris y LINUX (GNU Pascal: gpc [3]). Esto permitirá adoptar también ObjectPascal como lenguaje embebido y de desarrollo para las versiones UNIX, lográndose mejor grado de compatibilidad con las

[4]Originalmente diseñado y realizado por Apple Co. para su serie Macintosh

Concepto	**TURBO Pascal**	**Object Pascal**
Modularidad	Sí: Unit 's	Sí: Unit 's y DLL 's
Interfaz de módulo y referencia	Constructos del lenguaje	Idem
Mezcla con otros lenguajes	Sólo PC-Macroassembler	Acceso a librerías de C, C++
Vinculación con módulos compilados de otros lenguajes	PC-Macroassembler; C: limitado	C, C++ Otros vía DLL's
Programación de Objetos	Sí	Sí
Jerarquía de Clases	Sí	Sí, muy extensa
Programación Visual	No	Sí: Visual Components
Programación por Eventos	Algo: TurboVisión	Sí: fácil, poderoso
TAD cadena	Tipo `string:` variable hasta 255 cars.	Tipos `string:` variable hasta 255/65535; `PChar: estilo C`
Intercepción de Errores	Directivas al compilador; Sustitución de procesamiento standard Variable de estado `IOResult`	+ Clases de Excepciones + Instrucción `try`
Multi-hilado (Threads)	No	Sí
Acceso a Internet	No	Sí

Figure 2: Cuadro Comparativo TURBO Pascal v6/7 con Object Pascal

versiones para PC.

En el cuadro 2 se comparan algunas características de TurboPascal y ObjectPascal.

Por la compatibilidad con el lenguaje de desarrollo original y la relativa facilidad del ambiente para generar aplicaciones para WINDOWS, la tarea de conversión es relativamente sencilla y agradable. Para proporcionar a los usuarios las facilidades de operación propias de Windows, se prepararon módulos de interfaces adicionales.

También fué necesario *emular* la Librería GRAPH, para aprovechar la API de WINDOWS envuelta por DELPHI en la subclase `TCanvas` en forma

compatible con la programación original.

WDgraph es el nombre de la Unit ObjectPascal de emulación. Aparte de su uso en el desarrollo de la versión WINDOWS, está disponible para otros usuarios como Librería de Enlace Dinámico (DLL), con el nombre DlGraph.

Resultados

- Librería conformada con Módulo de Emulación de la Librería GRAPH.
- Librerías adicionales de presentación e interface operativa para WINDOWS.
- Conversión del Precompilador GLIDER.
- Conversión de la Librería GLIDER.
- Conversión del ambiente del Editor Gráfico y de Programas GLIDER.

Las conversiones se iniciaron trabajando con la versión Delphi 2 para Windows 3.1. Actualmente se están completando las versiones para WINDOWS 95/98/NT con Delphi 4.

En la figura 3 se presenta un esquema que muestra algunos de los soportes de Clases y Librerías usadas del ambiente Delphi.

Conclusiones

La estrategia de emulación tiene el costo inicial del proceso de su construcción pero se gana en cuanto facilita la futura conversión de otras aplicaciones sobre la misma plataforma.

La conservación del código original facilita el mantenimiento y la generación de nuevas versiones. Esto facilita que un equipo de desarrollo pueda liberar versiones para varias plataformas casi simultáneamente, sin retrasos significativos, ya que el trabajo de implantación de cambios se homogeiniza.

La estrategia es aplicable a diversas plataformas de destino. Es recomendable la adopción de criterios de diseño de aplicaciones con el concepto NCP o similar, que permite aislar los módulos con dependencias de plataforma.

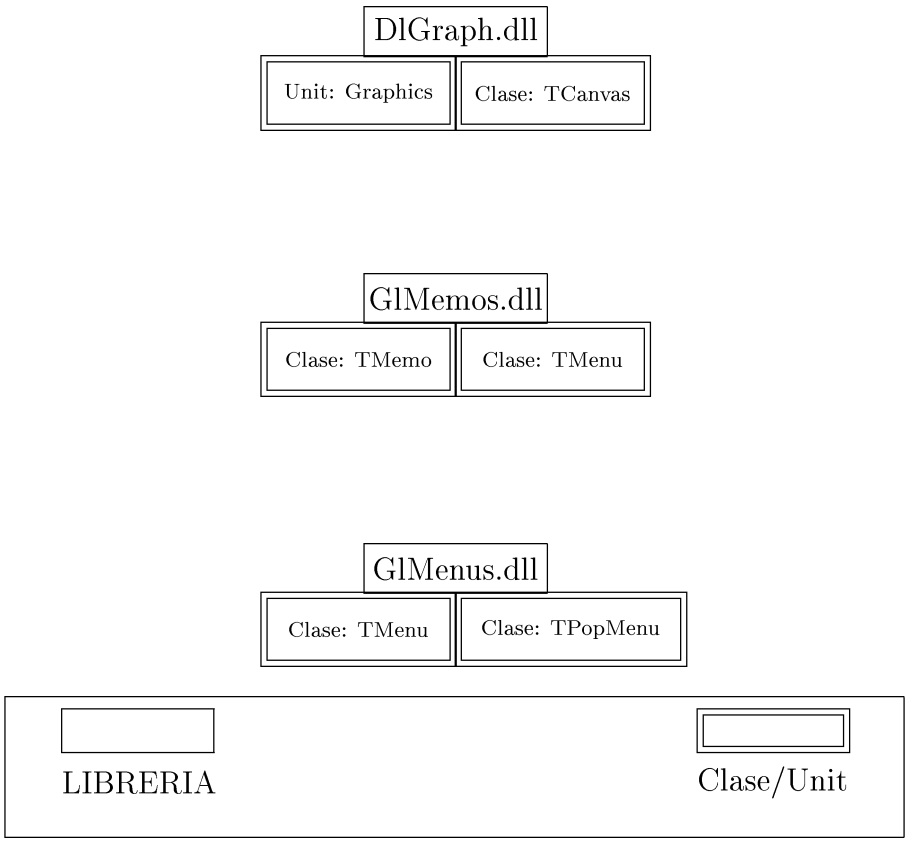

Figure 3: Uso Librerías y Clases Delphi/WINDOWS

Pascal, en la forma de su heredero ObjectPascal, sigue siendo una buena alternativa de lenguaje de desarrollo de software, solo o dentro de una un ambiente de desarrollo enriquecido con Librerías de Clases como Delphi.

References

[1] Borland International, Inc. *Turbo Pascal version 6.0 Reference Manual*

[2] Borland International, Inc. *Turbo Pascal version 6.0 Library Reference*

[3] Gerwinski, Peter. *http://home.pages.de/ GNU-Pascal/*

[4] Inprise Corporation. *Borland Delphi 4 for Windows 95 & Windows NT.* 1998.

[5] Jensen, K. and Wirth, N.: *Pascal User Manual and Report* Springer Verlag.

[6] Shumucker, Kurt J. *Object-Oriented Programming for the Macintosh* Hayden Books, 1986.