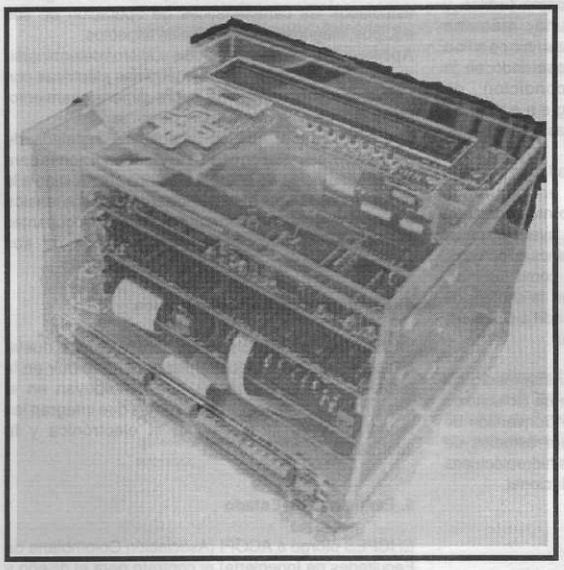
Control de dispositivos físicos a través del paradigma Modelo-Vista-Controlador

Proyecto de investigación EST



❑ **William Joseph Giraldo, M.Sc.**
Profesor asistente,
Universidad del Quindío.
Colombia.

❑ **María Lili Villegas, Ing.**
Profesor catedrático,
Universidad del Quindío
Colombia.

❑ **Diego Fernando Marín, Ing.**
Profesor auxiliar,
Universidad del Quindío
Colombia.

Grupo de Sistemas de Información y Control Industrial SINFOCI
Universidad del Quindío
Colombia

* Recibido: Mayo 2.002
* Aceptado: Junio 2.002

RESUMEN

En este artículo se exponen los resultados obtenidos durante la construcción del soporte a los dispositivos físicos de la herramienta para la automatización de procesos (EST) en la cual se modela un mecanismo para la abstracción de los dispositivos de entrada – salida, buscando independizar su representación física de su representación lógica. Este proceso se logra a través de la implantación del paradigma Modelo – Vista – Controlador por medio de un modelamiento orientado a objetos utilizando el lenguaje UML.

PALABRAS CLAVES

Modelo – Vista - Controlador, EST, ESC, Sistema Embebido.

ABSTRACT

This article shows the results obtained during the construction of the physic device support for the Process Automatization Tool (EST). In this, a mechanism for the abstraction of the Input/Output devices is modelled. Its physical and logical representation were separated. This process is achieved by using the Model-View-Controller paradigm implantation through object oriented modeling using the UML language.

KEYWORDS

Model-View-Controller, EST, ESC.

1. INTRODUCCIÓN

El grupo de Sistemas de Información y Control Industrial "SINFOCI" de la Universidad del Quindío busca el fortalecimiento de sus líneas de investigación "Tiempo Real y Sistemas Embebidos" creando infraestructura en el área de la automatización industrial. Dentro de este contexto se creó el sistema de desarrollo "EMBEDDED SYSTEM TOOLS" (EST) el cual se basa en un lenguaje propietario multiproceso orientado a objetos y cuyo soporte hardware es un sistema embebido.
Uno de los componentes más importantes de EST es su lenguaje de programación donde se definen formatos de datos como bit, nibble, bus y byte para el manejo de entradas/salidas análogas y digitales, permitiendo así una abstracción sobre su administración y una independización de los dispositivos físicos con su modelo lógico.
El respaldo software dado al control de los dispositivos físicos es definido con el uso del paradigma Modelo – Vista – Controlador y el apoyo de un kernel multiproceso en tiempo real "ESK" además de la definición de un modelo para la atención de los eventos.
El objetivo de este artículo es mostrar el proceso de respaldo dado a la definición de los tipos de datos del lenguaje ESC, bajo el paradigma Modelo – Vista – Controlador y su asociación con los demás componentes de la herramienta EST.

2. ESTRUCTURA GENERAL DE EST

EST es una completa herramienta para la automatización de procesos la cual respalda el proceso de desarrollo de aplicaciones embebidas desde la concepción del código de la aplicación hasta la puesta en marcha en su arquitectura hardware basada en sistemas embebidos.
Su estructura está compuesta por la especificación de un lenguaje y la implementación de componentes software y hardware.
EST tiene una interfaz de usuario (compatible IBM PC) compuesta por un display inteligente y varios interruptores para brindar al usuario una interacción con la aplicación la cual es soportada por funciones estándar en lenguaje C. Consta además, de entradas/salidas análogas y digitales que pueden ser configuradas en varios formatos de datos (figura 1).
Todo el soporte a las características de EST está dado por la definición de su lenguaje propietario.

3. LENGUAJE ESC

3.1 Características de ESC

ESC tiene una fuerte conexión con los dominios del multiprocesamiento, soporte de dispositivos físicos y programación orientada a objetos. Llamado ESC por sus siglas en ingles "Embedded System C", tiene como finalidad, soportar el desarrollo de aplicaciones para sistemas embebidos (basadas en conceptos de multiprocesamiento y tiempo real), sobre un modelo de programación que ofrece a los constructores de software un nivel de abstracción más amplio que el dado por lenguajes como C++. Por esta razón, se define como un C++ ampliado que cuenta con estructuras de control propias para el manejo de los procesos de la aplicación y los dispositivos físicos.

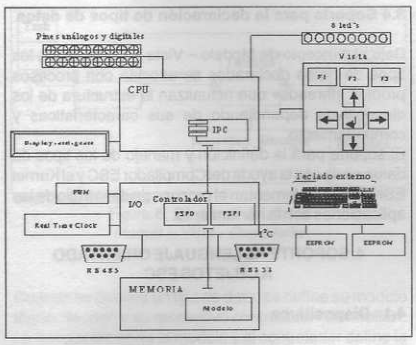


Figura 1. *Módulos de EST*

3.2 Definición de nuevos tipos de datos

Aparte de los tipos de datos soportados por el lenguaje C++, ESC incorpora un conjunto de nuevos tipos especiales con el fin de facilitar la programación y el manejo de aplicaciones embebidas.
Los nuevos tipos de datos se definen para encapsular dentro de modelos lógicos algunos de los dispositivos físicos del sistema como son los diferentes pines de interconexión. Éstos pueden ser manipulados por el programador bajo características y comportamientos diferentes dependiendo del tipo de dato que los contenga. Los relacionados con modelos formados por un pin físico son: analogo y digital. Los relacionados con modelos de agrupamiento (modelo digital) de varios pines físicos son: bus, nibble y byte, el primero de longitud variable.

3.3 Modificadores de tipo

La definición de nuevos tipos de datos está respaldada con el uso de modificadores de tipo que establecen su comportamiento como dispositivos de entrada, salida o bidireccionales. Los modificadores son: in, out y bidirectional, el último relacionado sólo con dispositivos de tipo digital.

3.4 Soporte para la declaración de tipos de datos

Bajo el concepto de Modelo – Vista – Controlador, los tipos de datos declarados se asocian con procesos propios «threads» que actualizan la estructura de los dispositivos dependiendo de sus características y comportamiento.
El soporte para la definición y manejo de los tipos de datos se da con la ayuda del Compilador ESC y el Kernel ESK que complementan el proceso de desarrollo de las aplicaciones con la herramienta.

4. SOPORTE AL LENGUAJE ORIENTADO A OBJETOS ESC

4.1 Dispositivos

Un dispositivo se puede considerar como una clase que modela un dispositivo físico conectado al hardware del sistema embebido. Un dispositivo puede comportarse igual que una clase, sólo que este tipo especial de clase se añade a la aplicación como una tarea y tendrá una ejecución en paralelo junto con las demás clases de tipo dispositivo y los procesos actuales del sistema embebido. Cada recurso de la herramienta (dispositivo físico) se manipula por medio de una vista, un controlador y un modelo. Para el kernel, el manejo de un dispositivo se define a través de su modelo y su manipulación se realiza a través del controlador que es el encargado de actualizar el modelo lógico del sistema así como el dispositivo físico, que es la vista para el usuario. En la figura 2 se puede apreciar el diagrama de Casos de Uso relacionado con la definición de un dispositivo.

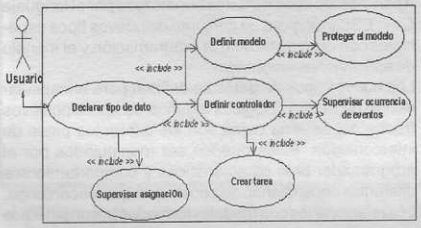


Figura 2. *Diagrama de casos de uso: definición de un dispositivo.*

La *declaración de un tipo de dato* es realizada por el usuario en el programa fuente; como los recursos físicos del sistema son limitados, éste supervisa la asignación en la declaración. El sistema reporta la definición de tipos de datos sobre dispositivos físicos ya asignados; es responsabilidad del usuario controlar la asignación. Cuando se declara un dispositivo se *define el controlador* que se encargará de actualizar tanto la vista como el modelo del tipo de dato. Para lograr su objetivo debe planificarse como un proceso del sistema que se ejecuta dinámicamente, por lo cual se incluye el caso de uso crear tarea.
Así como se define un controlador para la actualización de un tipo de dato, *se define el modelo.* El modelo es la abstracción del tipo de dato y es atacado por el controlador cuando lo actualiza, si el tipo es de entrada o cuando lo supervisa, si el tipo es de salida. El modelo también puede ser atacado por procesos propios de la aplicación, lo que obliga a que sobre él se implemente la exclusión mutua. La exclusión mutua se logra *protegiendo el modelo* como un recurso del sistema.

4.2 Control sobre el dispositivo

La función del controlador es mantener actualizado el dispositivo, evitando una comunicación directa entre el modelo y la vista. Un controlador es el proceso en sí que representa la definición de un dispositivo para el sistema. Si el dispositivo es de entrada (ejemplo: in digital a;), el controlador se encargaría de actualizar el modelo basado en los cambios ocurridos en la vista. Si el dispositivo es de salida (ejemplo: out analog a), el controlador se encargaría de actualizar la vista basado en los cambios ocurridos en el modelo.
El acceso a un modelo por parte de un controlador "proceso" es protegido por un semáforo, mientras que el acceso a la vista es protegido por el propio hardware del sistema.
De esta manera, las características de bajo nivel son sólo conocidas por el controlador del dispositivo y no se relacionan de manera directa con la declaración del tipo de dato. La ventaja es que no es necesario efectuar cambios en el código de la aplicación si la vista cambia.
En la figura 3 se puede apreciar el proceso normal de acceso a la vista donde la aplicación utiliza un conjunto de librerías de servicios para atacar el modelo pero sólo el controlador tiene acceso a la vista y al modelo. El acceso de la aplicación al controlador del dispositivo es restringido; sólo se permite a través del modelo.

Figura 3. *Acceso a un dispositivo por parte de la aplicación.*

4.3 Supervisar la ocurrencia de eventos

El cambio de valor o estado de un tipo de dato depende de los eventos asociados a él y de su calificador de tipo. El controlador debe registrar los cambios en la vista o el modelo para actualizar el tipo de dato. Según el tipo de dato el controlador puede ser: de entrada o de salida.

- Un tipo de dato de entrada es manejado por un controlador que supervisa la vista para actualizar el modelo.
- Un tipo de dato de salida es manejado por un controlador que supervisa el modelo para actualizar la vista.

4.4 Administración de tipos de datos de entrada

Los controladores de entrada tienen como función actualizar el modelo basados en los cambios que registren en su vista "evaluar la vista" a causa de eventos (figura 4).

Figura 4. *Diagrama de casos de uso: controlar dispositivos de entrada*

El acceso al modelo por parte del controlador, depende de que éste no esté protegido (caso de uso solicitar acceso). Es responsabilidad del controlador reportar el evento ocurrido para que el sistema lo clasifique y atienda a los procesos asociados a éste. Esta actividad se representa por el caso de uso reportar evento.

4.5 Administración de tipos de datos de salida

Los controladores de salida actualizan la vista basados en los cambios que registren en el modelo (figura 5).

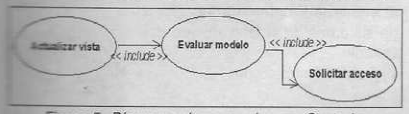


Figura 5. *Diagrama de casos de uso: Controlar dispositivos de salida*

Para actualizar la vista asociada a un tipo de dato, el controlador verifica los cambios ocasionados en el modelo (los cambios son hechos por procesos de la aplicación), para lo cual, debe solicitar el acceso (modelo protegido por un semáforo).

5. MODELAMIENTO DE LOS TIPOS DE DATOS

5.1 Soporte para el lenguaje ESC

La asociación de los objetos de la figura 6, busca la implementación del concepto MVC a través de un modelo orientado a objetos.

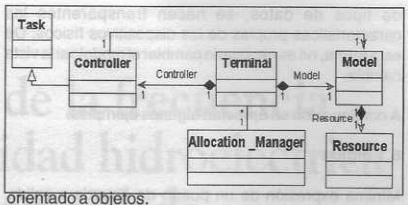


Figura 6. *Diagrama de clases: Modelo – Vista - Controlador*

Cuando se declara un tipo de dato, se define su modelo lógico "Model" y su respectivo controlador "Controller". La asociación entre el modelo y el controlador define la actualización del dispositivo. Es importante que por cada modelo se declare un semáforo para protegerlo, ésto garantiza la exclusión mutua (asociación entre Model y Resource).

La asignación de dispositivos físicos (pines) encapsulados en los tipos de datos declarados en la aplicación, es controlada por el objeto Allocation_Manager, el cual contiene en su estructura un arreglo asociado a pines físicos de EST. Cada vez que se declara un tipo de dato, Allocation_Manager verifica la disponibilidad de los pines que encapsula y define un registro para identificar el tipo de dato al que pertenecen. La redeclaración de pines físicos en diferentes tipos de datos es responsabilidad del usuario.

5.2 Soporte para los tipos de datos del lenguaje ESC

La definición de tipos de datos propios del lenguaje ESC, es controlada bajo el modelamiento de una jerarquía de tipos de datos, basada en las propiedades que éstos poseen. De esta forma, cada dispositivo definido puede clasificarse como de tipo digital o análogo y posteriormente asociarse con su calificador de tipo como de entrada o salida. El concepto de modelo no se relaciona con los calificadores de tipo, estos son aplicados a la vista y al controlador de los dispositivos.

A diferencia de los dispositivos de tipo análogo que están relacionados directamente con un solo pin físico del sistema, los dispositivos de tipo digital, pueden encapsular en sus declaraciones agrupaciones de éstos.

6. IMPLANTACIÓN DEL MODELO PARA EL CONTROL DE HARDWARE

La implantación del modelo de objetos para el control de hardware, permite al desarrollador, la declaración y administración del hardware de EST. Bajo la definición

de tipos de datos, se hacen transparentes las características propias de los dispositivos físicos; De esta forma, no es necesario cambiar el modelo si la vista cambia.

A continuación se aprecian algunos ejemplos.

6.1 digital

Mínima expresión de un puerto de entrada o salida. Ejemplo:

```
in digital sw;  //pin 0
out digital led;  //pin 1
```

6.2 bus

Objeto para la utilización de los pines del sistema en forma de bus. La forma general es la siguiente:
in bus selector(2,5); //4 pines, del 2 al 5
De esta forma, cada vez que se declara un dispositivo de

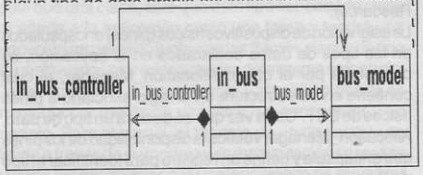


Figura 7. *Diagrama de clases: declaración de un tipo de dato in_bus*

7. CONCLUSIONES

El acceso a dispositivos físicos se convierte en un proceso sencillo "a pesar que éste es muy dependiente de la arquitectura hardware del sistema" que permite a los diseñadores de la aplicación, los cuales pueden ser ingenieros de sistemas que conocen muy poco de hardware, crear aplicaciones más eficientes y de fácil adaptación a la hora de cambiar las especificaciones de los dispositivos periféricos.

La implantación del paradigma Modelo - Vista - Controlador, respaldado por un kernel multitarea en tiempo real orientado a objetos, facilita la planificación del controlador y la protección del modelo para cada dispositivo físico del sistema.

La utilización del paradigma Modelo - Vista - Controlador para la administración de dispositivos hardware, permite a los desarrolladores obtener independencia entre el modelo lógico y el modelo físico de los tipos de datos que los encapsulan en declaraciones de lenguajes de programación como ESC.

8. REFERENCIAS BIBLIOGRAFÍCAS

[1] POWEL DOUGLASS, BRUCE. Real Time UML, Addison Wesley USA, segunda edición (1999).

[2] LIEM CLIFFORD retargeable Compilers for Embedded Core Processors Kluwer academic Publishers (1997)

[3] GREHAN, MOOTE, CYLIAX Real-Time Programming Addison Wesley (1998)

[4] J.E.COOLING, Software Design for Real-Time Systems ED Chapman and Hall (1991)

[5] JEAN-PAUL CALVEZ, Embedded Real Time System a Specification and design Methodology, ED Jhon Wiley and Sons Limited (1991)

[6] P.J. WARD, S.J. MELLOR, Development for Real Time Systems, ED Prentice Hall (1985).

[7] WOLFGANG A. HALANG, KRZYSZTOF M. SACHA, Real-Time System World Scientific (1992)

AUTORES

William Joseph Giraldo Orozco. Ingeniero electricista, y MSc de la Universidad del Valle. Profesor asistente del Programa de ingeniería de sistemas de la Universidad del Quindío.
wjgiraldo@hotmail.com

Diego Fernando Marín Sanabria. Ingeniero de sistemas de la Universidad del Quindío. Profesor auxiliar del Programa de ingeniería de sistemas de la Universidad del Quindío.
dmarin@quimbaya.uniquindio.edu.co

María Lili Villegas Ramírez. Ingeniera de sistemas de la Universidad del Quindío. Profesor catedrático del Programa de ingeniería de sistemas de la Universidad del Quindío.
m_lvillegas@hotmail.com.