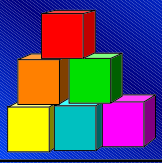
Proyecto Taller 5 - 1997

Ingeniería en Computación
InCo - Fac. Ing. - UDELAR

Manejador de Versiones de Esquemas Entidad Relación



Carla Demarchi
Valeria Irazábal
Alicia Revello

Supervisor: R. Ruggia

Contenido

- Introducción
- Diseño e Implementación
- Conclusiones

- ¿ Qué es el manejo de versiones?
- Manejador de Versiones de Esquemas ER
- Ejemplo
- Aportes

Administración de Versiones

- Conjunto de programas y procedimientos para gestionar ordenadamente el cambio de circuitos, programas, etc.

- Características:
  - Método para mantener información
  - Permite realizar cambios controlados

Administración de Versiones

- Características:
  - Método para rastrear modificaciones
  - Comparación entre versiones
  - Histórico de los cambios
  - Restauración de un mal diseño, sencillamente
  - Permite construir nuevos objetos a partir de otros ya existentes

Manejador de Versiones de Esquemas ER

- Introduce conceptos específicos de los esquemas ER -> relación de inclusión
- Basado en:
  - Conjunto de versiones
  - Relaciones entre ellas: derivación, inclusión y construcción -> Grafos

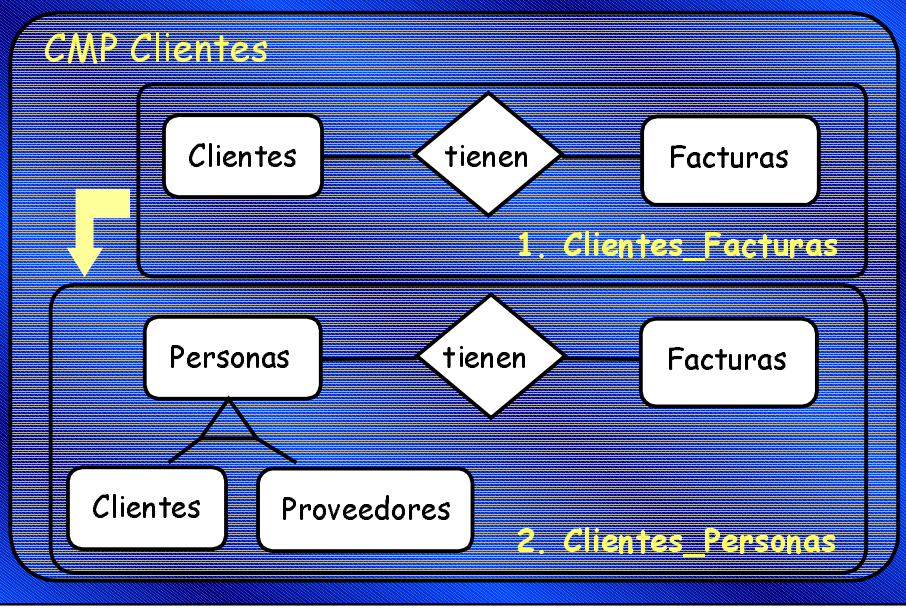
Manejador de Versiones de Esquemas ER

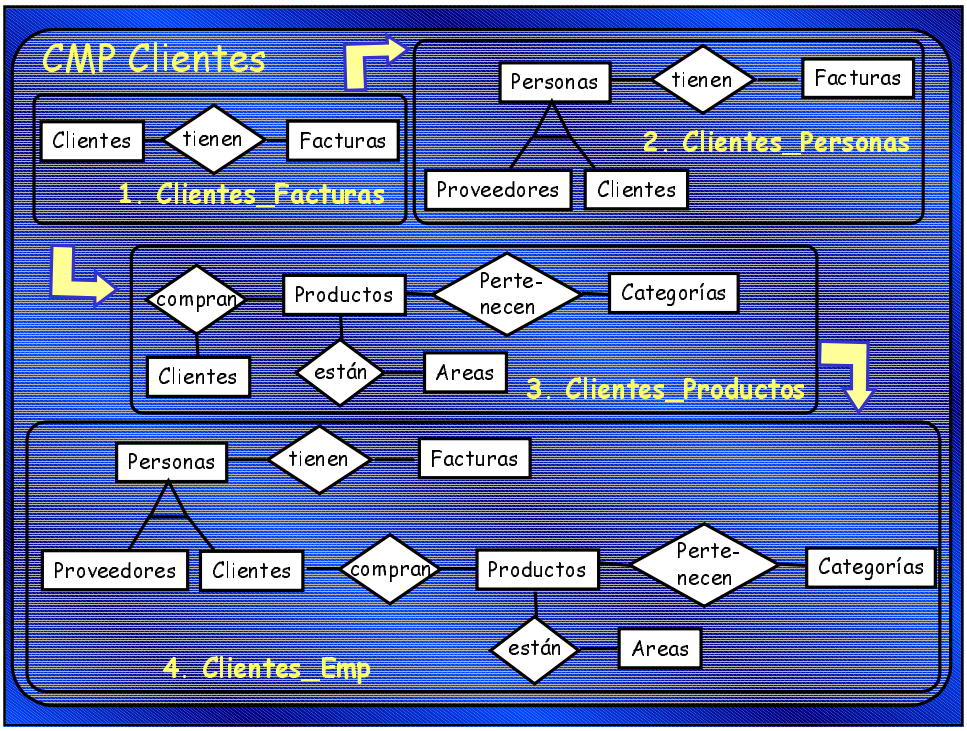
- Implementado en C++
- Arquitectura C/S/S
  - Servidor de Versiones
  - Servidor para el manejo de esquemas ER
  - Cliente
- Integración con ambiente CASE

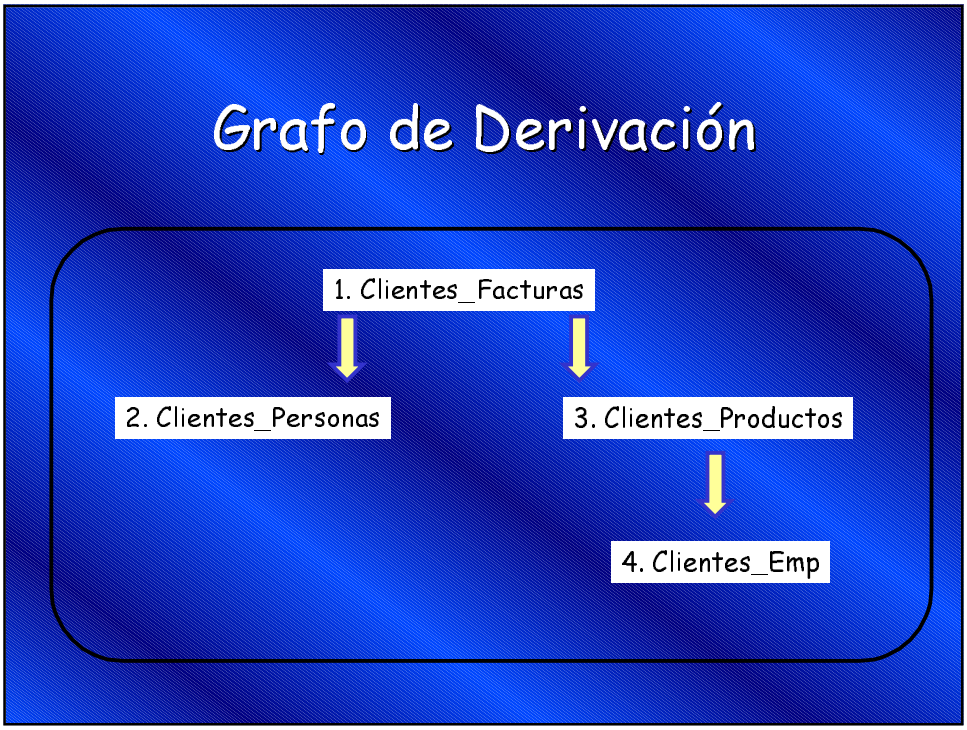
Evolución

- 1ª generación: basada en archivos.
- 2ª generación: basada en repositorios de proyectos.
- 3ª generación: basada en transparencia de archivos.

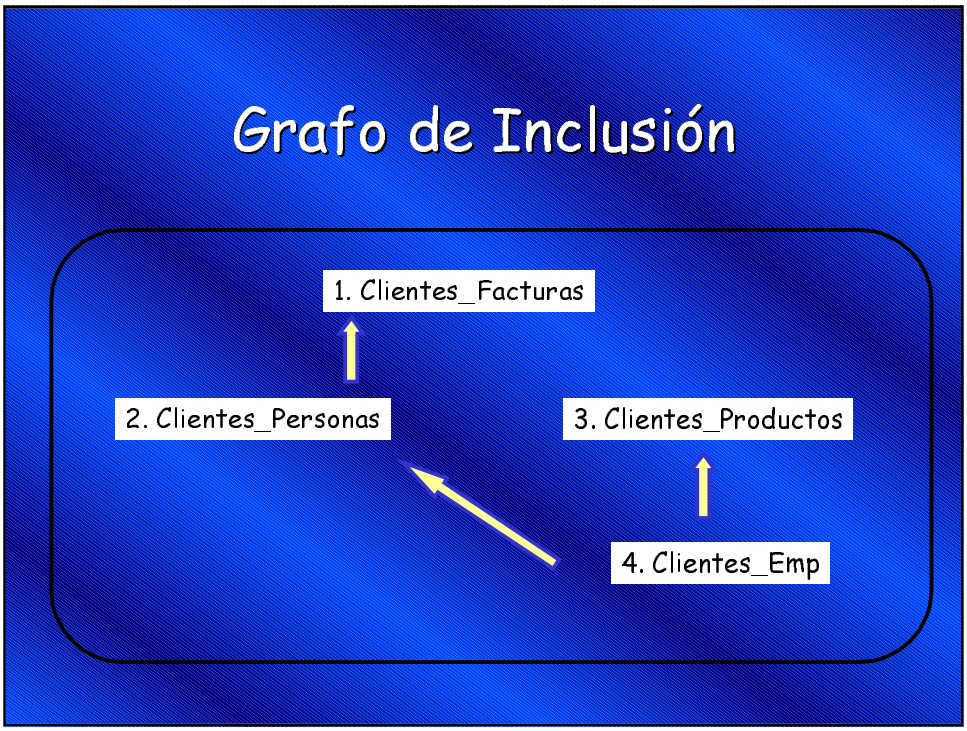
Ejemplo



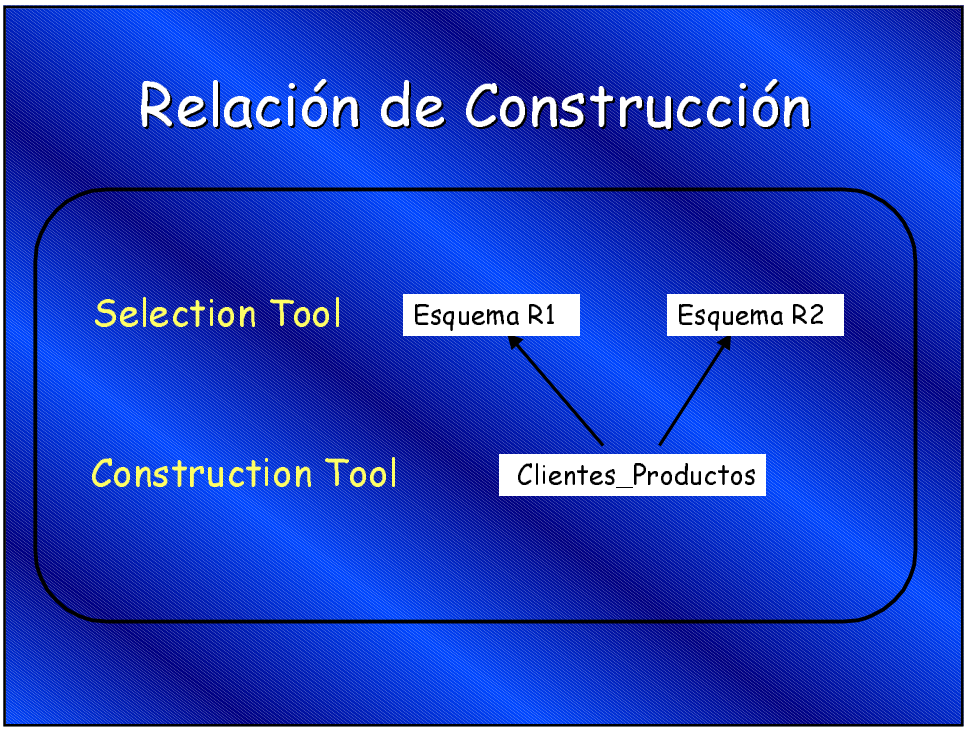

CMP Clientes
Clientes
tienen
Facturas
1. Clientes_Facturas
Personas
tienen
Facturas
Proveedores
Clientes
2. Clientes_Personas
compran
Productos
Perte-
necen
Categorías
Clientes
están
Areas
3. Clientes_Productos
Personas
tienen
Facturas
Proveedores
Clientes
compran
Productos
Perte-
necen
Categorías
están
Areas
4. Clientes_Emp


Grafo de Derivación
1. Clientes_Facturas
2. Clientes_Personas
3. Clientes_Productos
4. Clientes_Emp

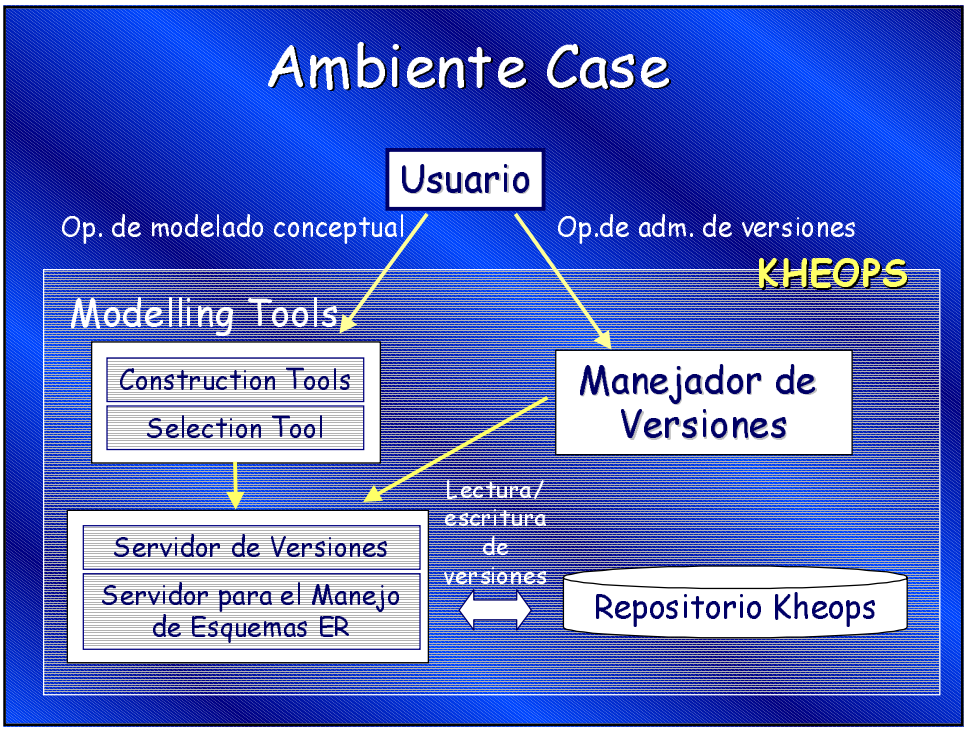
Grafo de Inclusión



Relación de Construcción



Ambiente Case



Aportes

- Permite integrar resultados obtenidos por diferentes herramientas
- Relación de inclusión -> evita la proliferación de versiones
- Posibilita la integración de herramientas con arquitectura C/S
- Arquitectura portable

Aportes

- La organización de los esquemas ER es independiente de la herramienta con que se desarrollaron
- Provee una interfaz amigable

Contenido

Trabajo Realizado

- Análisis de Requerimientos
- Búsqueda de Herramientas
- Diseño de la arquitectura
- Codificación y testeo de módulos
- Integración y testeo de todo el sistema

Requerimientos

- Interfaz Amigable
- Código de los fuentes coherente con la especficación
  - Renombre de objetos
  - Implementación de funciones
- Arquitectura que permita la modularización de los componentes

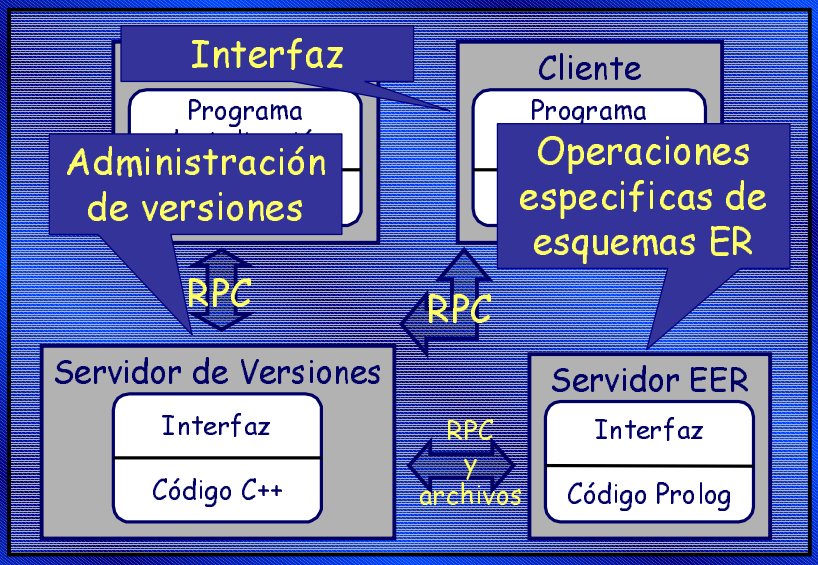
Búsqueda de Herramientas

- Escalable
- Extensible
- Interacciones
- Portabilidad
- Buena Documentación

Criterios de Selección

ILOG VIEWS

Arquitectura



Codificación, Integración y Testeo.

- Se implementó el Servidor de Versiones
- Se implementó un cliente con un menú ASCCI
- Se realizó la conexión C/S en la red con RPC
- Testeo de la conexión
- Conexión del SV con el SEER
- Testeo de la conexión C/S/S
- Implementación de la interfaz gráfica en el cliente
- Testeo de todo el sistema

Operaciones

Operaciones sobre CMP

Operaciones sobre Versiones

Operaciones sobre Grafos de Versiones

Operaciones sobre CMP

- Create_CMP
- Save_CMP
- Close_CMP

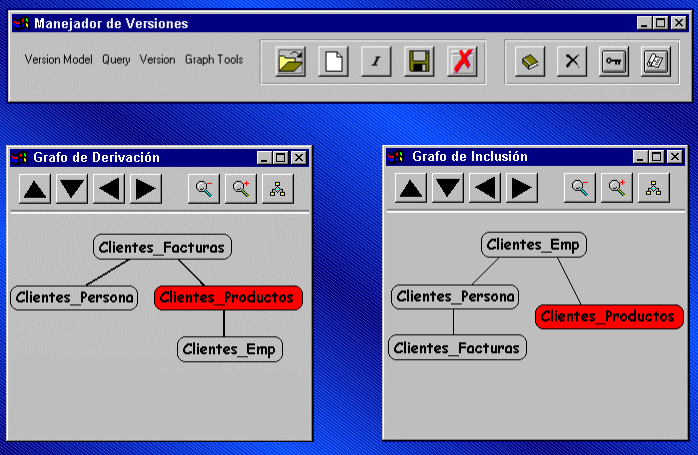
Operaciones sobre Versiones

- Write_version
- Delete_version
- Rename_version
- Set_Current

Operaciones sobre Grafos de Versiones

- Ancestors
- Path
- Del_Sub_Tree

El Cliente - Interfaz



El Cliente - Las clases


```
class Client_Env {
        char    *server_VM;
        CLIENT  *cl_VM;
public:
        int conn_number; // Número de conexión del cliente
        table_vg  vg_t;  // Estructora donde se pasan los nodos
                         desde el servidor
        obj_names[MAX_LENGHT_ID][MAX_QTTY_OBJECTS];
        objs_in_cmp obj_cmp;
        Client_Env(int cn, void (*f)(char *a[], Client_Env *));
// Funciones de acceso al Version Server
        void conn_server_VM(char *server_name);
        int create_module(char *a,char *m,char *w);
        int open_module(char *a,char *m,char *w);
        int close_version_plane();
        int save_version_plane();
        void give_vg_arrays(); //carga en vg_t los dos grafos
}
```

Class Client_Env

```
class Menu1 : public IlvGadgetContainer {
public:
        char* server; // Extensión de la clase
        int conectado;
        . . .
        Menu1(IlvDisplay* display, const char* name, const char*
                title, IlvRect* size = 0, IlvBoolean  useAccelerators
                = IlvFalse, IlvBoolean  visible = IlvFalse, IlvUInt
                properties = 0, IlvSystemView transientFor  = 0,
                char* serv, int conectado);
        . . .
        virtual void open_module(IlvGraphic*);
        virtual void Refresh(IlvGraphic*,IlvAny);
        virtual void Save();
        virtual viod Load();
        . . .
        // Funciones generadas por ILOG Views Studio
        IlvButton* getbtn_open() const
                {return (IlvButton*)getObject("btn_open"); }
        . . .
}
```

Class Menu1

```
class Versiones: public IlvApplication {
public:
Versiones (
        const char* appName,
        const char* displayName = 0,
        int argc = 0,
        char** argv = 0
        );
    Versiones (
        IlvDisplay* display,
        const char* appName
    );
    ~Versiones();
    virtual void makePanels();
    virtual void beforeRunning();
};
```

Class Versiones

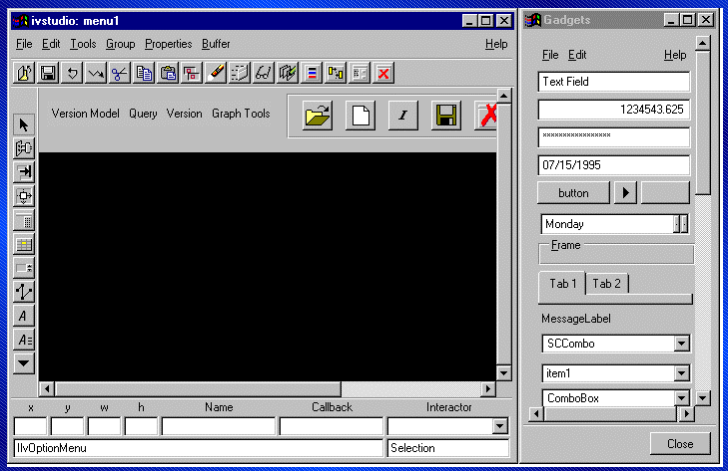
ILOG VIEWS

ILOG VIEWS STUDIO → FUENTES → Interfaz

- Ventanas de los Grafos
- Menú
- Paneles para cada operación

- Refresh de los grafos
  table_vg -> IlvGrapher
- Reprogramas Callbacks
  Panel -> Cl_env

ILOG VIEWS Studio



Contenido

Servidor de Versiones

- Administra el conjunto de grafos (CMP) que son accedidos por los diferentes clientes.
  - Grafos
  - Versiones
  - Relaciones
- Conexión con el servidor para el Manejo de Esquemas ER

Servidor de Versiones - clases

Cliente - Class Cl_Env

Class V_SYSTEM

Class CMProjects

Class Version

Class Arc

Class Inclusion_from

Class Inclusion_to

Class Derivation_from

Class Derivation_to

Servidor de Manejo de Esquemas ER

- Administra los esquemas ER, determinando la relación de inclusión
- Necesidad de acceso a fuente Prolog

Servidor de Versiones

Main

Pipe Unix

Nucleo Prolog

Herramienta C/S

- Remote Procedure Call (RPC). Paradigma de comunicaciones de alto nivel
  - Especificación de archivo de definición protocolo (version.x)
  - RPCGen generador de protocolos
  - Compilar y linkeditar los fuentes generados

Version.x

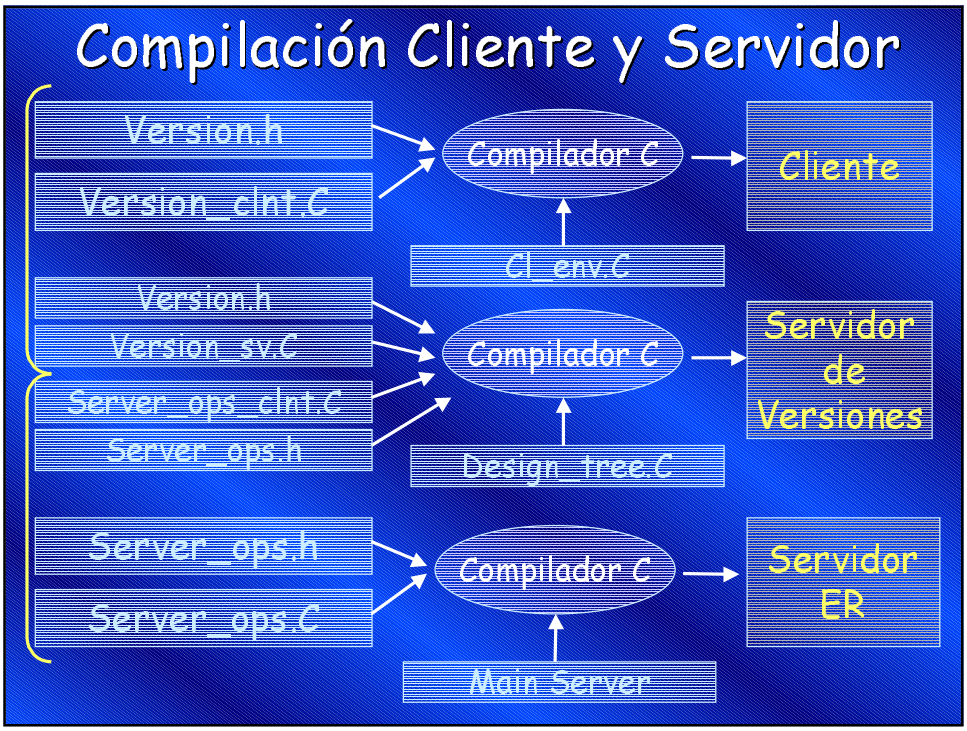
```
program VERSIONS
 {
     version VERSIONS_1
       {
        int SAVE(int) = 1;
        int OPEN(open_rec) = 2;
        int CREATE(open_rec) = 3;
        int CLOSE(int) = 4;
        version_rec GIVE_CURR_VERSION(int) = 5;
        table_vg GIVE_VGA(give_vg_rec) = 6;
        int GIVE_VERSION_ID(give_version_id_rec) = 7;
        int WRITE(version_write_rec) = 8;
            res_read_v_rec READ_VERSION(set_ver_rec) = 9 ;
            int DEL_VERSION(set_version_rec) = 10;
            int DEL_VERSION_TREE(set_version_rec) = 11;
            int RENAME_VERSION(ren_v_rec) = 12;
            int UPDATE_VERSION(upd_v_rec) =  13;
                …
           } = 1;
 } = 0x20000001;
```

RPCGEN Version.x

Generará:

- Version.h
- Version_clnt.C
- Version_svc.C

Compilación Cliente y Servidor




Conexión
Interfaz Gráfica
Cl_envC
Cliente
Proc. remotos
Proc. remotos
Servidor
de Versiones
V_System
CMProjects
Version
Arc
Proc. remotos
Proc. remotos
Servidor EER
Servidor para
el manejo de
Esquemas ER
Unix Pipe
Facts Prolog


Contenido
Introducción
Diseño e
Implementación
Conclusiones
Conclusiones
Trabajo Futuro

Conclusiones

- Manejador de Versiones permite:
  - Administrar los cambios en ambientes de desarrollo multiusuario.
  - Reutilizar componentes (independiente de la herramienta), para el ahorro de tiempo y dinero.
- Integración a una herramienta CASE con otros componentes

Trabajo Futuro

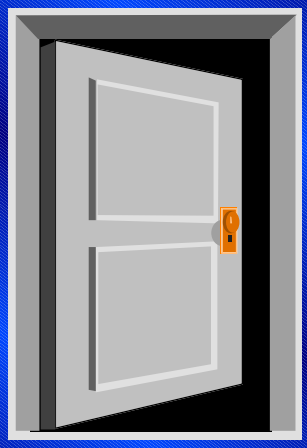
Relación de Construcción :

Diseñar la relación en la clase V_System

Modificar la interfaz gráfica

Concurrencia → Operaciones asincrónicas

The End



Estructura jerárquica de clases.

```
class V_SYSTEM  {
public:
        CMProjects * versions[conn_numbers];
        V_SYSTEM();

        CMProjects *give_graph(int cnn_number);
        int give_id_curr_version( int cnn_nunber );
        int give_cnn_number(char*, char*, char*);
        void save_version_plane(int cnn_number);
        int open_version_plane(char*,char*,char*);
        int write_version(int cnn_n,char *v_n,char
*v_v, char
    *drv_frm_v);
          . . .
};
```

Class V_SYSTEM

Estructura jerárquica de clases.

```
class CMProjects {
public:
char application_name[20];
char module_name[20];
GRAPH<Version,Arc> structure;
node root, current_version;

CMProjects(char*, char*);
~CMProjects();
Version* give_version(node);
GRAPH<Version,Arc>* give_structure();
int give_last_id_version();
node search_version(int st_id);
edge search_edge(int id_arc);
…};
```

Class CMProjects

Marco Teórico

- Versión
  <id, nombre, valor>

- CMP (Proyecto de Modelado Conceptual)
  <nombre, conjunto de versiones, versión corriente, relaciones entre versiones>