Universidad Autónoma de Madrid

Escuela Politécnica Superior

Tesis Doctoral

SISTEMA EVOLUTIVO DE GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

Jorge de las Peñas Plana

jorge.delaspennas@uam.es

Directora: Miren Idoia Alarcón Rodríguez

A mi mujer Josefina

A nuestros hijos

Agradecimientos

Quiero dedicar las primeras líneas de agradecimiento en este trabajo a quien más las merece. Mucho más que yo mismo. Es de justicia, de orgullo y de necesidad dar las gracias infinitamente a la verdadera alma de este trabajo. A quien ha hecho que sea realidad. No exagero si digo que es la autora de al menos el 51% de este trabajo. Gracias, Josefina, con todo mi corazón. Esta es tu tesis, por la que tanto has luchado. Tus esfuerzos enormes en dejarme trabajar e investigar, incluso a veces obligarme a ello, tienen aquí su fruto. Esta es mi mayor alegría: por ti.

Un tierno agradecimiento quiero dar a nuestros hijos, Josefina y Jorge. Ellos han nacido y han comenzado a dar sus primeros pasos y a forjar sus personalidades viendo como su papá estaba mañana tras mañana sentado en la misma silla con un ordenador delante en el que hacía un extraño trabajo difícil de explicar hasta para los mayores. Bajo la supervisión incansable de su madre, han sido una gran motivación y orgullo para ayudarme a ver que todo es posible.

Por terminar el capitulo familiar, un cariñoso recuerdo y agradecimiento a toda mi familia, tanto paterna como política. Especialmente mi madre, como buena madre, se pondrá muy ancha cuando enseñe este trabajo a todo el mundo, siempre valorando enormemente cada cosa que hace su pequeño.

Doy las gracias enormemente a la directora de este trabajo de investigación, Doctora Miren Idoia Alarcón Rodríguez. Su papel en este largo proceso ha sido determinante en todos los aspectos. Sin perjuicio de su enorme sabiduría y de sus consejos que han sido guía clave hacia el éxito, me quedo con el factor que más me ha ayudado: la confianza. Desde el primer minuto que le pedí que asumiera la responsabilidad de la dirección, ni pestañeó. Sus empujones han sido constantes, incuso en momentos largos y oscuros donde he caminado en la sombra del fracaso. No sé por qué pero ella siempre supo que iba a finalizarlo. Aquí la tienes Idoia. Lo hemos conseguido. Enhorabuena. Muchísimas gracias.

Debo agradecer también al equipo de dirección de la Escuela Politécnica Superior, y al personal de Administración del departamento de Ingeniería Informática de la Universidad Autónoma de Madrid, su soporte y ayuda en este largo proceso. Es clave sentirse apoyado por un equipo sólido y dinámico. Agradezco especialmente al actual director de la escuela, Doctor Xavier Alamán Roldán, que en su momento cuando era Subdirector de Profesorado, confió en mí para impartir docencia en la escuela. Este hecho me acercó mucho a la senda del éxito en esta investigación.

Agradezco la disposición y el interés del Doctor Gonzalo Martínez, designado como Lector del Departamento. Ha aportado ideas interesantes y una visión muy detallada del trabajo. Igualmente agradecer a los miembros titulares y suplentes del tribunal el interés mostrado en la evaluación de este trabajo de investigación.

Recuerdo y agradezco el apoyo de los directores que he tenido en mi trabajo fuera de la universidad. No dudaron un momento en apoyarme cuando les propuse una idea aparentemente incompatible con un exigente trabajo de cara a clientes y orientado a proyectos de sistemas de información. Una vez más son personas que han estado muy presentes en este camino con esa confianza. Yo creo modestamente que no les he fallado en mi dedicación, cumpliendo mi parte del trato.

Y finalmente acordarme de tantos amigos y tan buenos que tengo de diferentes círculos que siempre me preguntan por mi tesis. Y si no me preguntan se lo digo yo con largos discursos. Agradezco su escucha y paciencia. No sé si pensaron que esto llegaría a buen puerto, pero así ha sido.

Índice.

1 INTRODUCCIÓN 19

1.1 CAMPOS DE INVESTIGACIÓN 19

1.2 PROBLEMA ABORDADO 20

1.3 APROXIMACIÓN A LA SOLUCIÓN 21

1.4 ORGANIZACIÓN DE LA MEMORIA 22

2 ESTADO DE LA CUESTIÓN 23

2.1 GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE 23

2.1.1 Definiciones 23

2.1.1.1 Sistemas de Gestión de la Configuración del Software 24

2.1.1.2 Conceptos 26

2.1.2 Herramientas de Gestión de la Configuración del Software 28

2.1.2.1 Concurrent Versions System (CVS) 29

2.1.2.1.1 Versionado, Revisiones 29

2.1.2.1.1.1 Asignación de revisiones 30

2.1.2.1.1.2 Ramas de Versiones 30

2.1.2.1.2 Añadir, eliminar y renombrar archivos y directorios. 31

2.1.2.1.2.1 Añadir archivos a un directorio 32

2.1.2.1.2.2 Actualización 32

2.1.2.1.2.3 Eliminar archivos 34

2.1.2.1.3 Histórico 35

2.1.2.1.4 Aplicaciones CVS 35

2.1.2.2 Otros modelos de Sistemas y Arquitecturas de Gestión de la Configuración del Software 36

2.1.2.2.1 Herramientas de Control de Versiones 36

2.1.2.2.2 Herramientas para construcción y despliegue 37

2.1.2.2.3 Herramientas Comerciales 38

2.1.2.2.4 Herramientas de Proyectos de Investigación 39

2.2 DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS. MINERÍA DE DATOS. APRENDIZAJE AUTOMÁTICO. REDES NEURONALES ARTIFICIALES. 40

2.2.1 Conceptos 40

2.2.2 Aprendizaje Automático 44

2.2.3 Otras técnicas: Algoritmos Genéticos 50

2.2.3.1 Perspectiva histórica 51

2.2.3.2 Aspectos de Diseño 54

2.2.4 Redes Neuronales Artificiales (ANN) 55

2.2.4.1 Perspectiva histórica. 55

2.2.4.2 Aspectos de Diseño 63

2.2.4.3 Combinaciones de Redes Neuronales y Algoritmos Genéticos 64

2.2.4.4 Librerías de Redes Neuronales 66

2.2.4.4.1 Librerías TORCH 66

2.2.4.4.2 Personalizaciones de TORCH 68

2.3 CONCLUSIONES DEL ESTADO DE LA CUESTIÓN 70

2.3.1 Espacio de Investigación 70

2.3.2 Investigaciones Relacionadas 71

3 HIPÓTESIS 73

4 PLANTEAMIENTO DEL PROBLEMA 75

4.1 EL PROBLEMA DE LA EVOLUCIÓN EN LA GESTIÓN DE LA CONFIGURACIÓN 75

4.2 SOLUCIÓN BASADA EN EL DESCUBRIMIENTO DE CONOCIMIENTO. 76

5 SOLUCIÓN PROPUESTA 79

5.1 VISIÓN GLOBAL DE LA SOLUCIÓN 79

5.2 DESCRIPCIÓN GENERAL DE PROCESOS DE MOVIMIENTO DE SOFTWARE DEL ÁREA DE SISTEMAS DE UNA COMPAÑÍA 82

5.2.1 Esquema general 82

5.2.2 Roles y Responsabilidades 85

5.2.3 Procesos de Instalación, Pruebas y Explotación y sus fuentes de errores 86

5.2.3.1 Proceso de Instalación 87

5.2.3.2 Proceso de Pruebas 88

5.2.3.3 Proceso de Explotación 89

5.3 ARQUITECTURA FUNCIONAL 91

5.3.1 Esquema general 91

5.3.2 Módulo de Usuarios 92

5.3.3 Módulo de Reporting 92

5.3.4 Módulo de Gestión de la Configuración 93

5.3.5 Módulo de Descubrimiento de Datos 93

5.4 ARQUITECTURA TECNOLÓGICA 94

5.4.1 Esquema general 94

5.4.2 Módulo Cliente 96

5.4.3 Módulo Servidor 97

5.4.4 Módulo de Base de Datos 98

5.4.5 Módulo de Repositorio 99

5.5 ANÁLISIS Y DISEÑO DE LA SOLUCIÓN 100

5.5.1 Entornos de Trabajo 100

5.5.2 Modelización del Proceso de Gestión de Cambios 103

5.5.2.1 Procesos de Gestión de Cambios. Unidades de Control 103

5.5.2.2 Jerarquía de Unidades de Control 104

5.5.2.3 Ciclos de Vida de Unidades de Control 106

5.5.2.4 Atributos de Unidades de Control 109

5.5.3 Integración entre Componentes 109

5.5.3.1 Comunicación entre Cliente y Servidor 109

5.5.3.2 Comunicación entre Cliente y Base de Datos 111

5.5.3.3 Comunicación entre Cliente y Repositorio 112

5.5.3.4 Comunicación entre Servidor y Base de Datos 114

5.5.3.5 Comunicación entre Servidor y Repositorio 114

5.5.3.6 Comunicación off-line 115

5.5.4 Cliente 116

5.5.4.1 Clases 117

5.5.4.1.1 Clase de conexión al servidor 117

5.5.4.1.2 Clase de Comunicación con la Base de Datos 119

5.5.4.2 Módulos 119

5.5.4.3 Formularios 120

5.5.4.3.1 Formulario de Autenticación. Módulo de Seguridad 121

5.5.4.3.2 Formulario de Gestión de Informes 122

5.5.4.3.3 Formulario de Instalación 122

5.5.4.3.4 Formulario de Control de Cambios 123

5.5.4.3.5 Formulario de Acciones sobre Repositorio 123

5.5.4.3.6 Formulario de Comunicación "Offline" 124

5.5.5 Servidor 124

5.5.5.1 Estructura 124

5.5.5.2 Atención de Peticiones 125

5.5.5.2.1 Funcionamiento 125

5.5.5.2.2 Fuentes y Librerías 126

5.5.5.3 Ejecución de Peticiones 127

5.5.5.3.1 Librerías 127

5.5.5.3.2 Atención de Peticiones 128

5.5.6 Base de Datos 130

5.5.6.1 Modelo de Datos 130

5.5.6.2 Modelo de Entornos de Trabajo: Tablas ECFM 132

5.5.6.3 Modelo de Informes: Tablas ECFM_I 133

5.5.6.4 Modelo de Usuarios: Tablas ECFM_U 134

5.5.6.5 Modelo de Unidades de Control: Tablas ECFM_C 136

5.5.6.6 Modelo de Descubrimiento de Datos: Tablas ECFM_GA 139

5.5.7 Repositorio 142

5.6 ANÁLISIS Y DISEÑO DE LA PARTE DE DESCUBRIMIENTO DE DATOS 144

5.6.1 Arquitectura de Redes Neuronales del Sistema y Minería de Datos. 144

5.6.1.1 Arquitectura Funcional 144

5.6.1.2 Definición de Componentes de Redes Neuronales. 146

5.6.1.3 Integración con la Solución 148

5.6.1.3.1 Cliente. 149

5.6.1.3.2 Servidor 150

5.6.1.3.3 Base de Datos 150

5.6.1.3.4 Repositorio 151

5.6.2 Red Neuronal de Clasificación de Usuarios basándose en sus Errores 151

5.6.2.1 Definición de Red Neuronal de Errores de Usuario 151

5.6.2.1.1 Número de Nodos de la Capa de Entrada. 151

5.6.2.1.2 Número de Nodos de la Capa Oculta. 153

5.6.2.1.3 Número de Nodos de la Capa Oculta 153

5.6.2.1.4 Iteraciones necesarias para el entrenamiento. 153

5.6.2.1.5 Fichero de Entrenamiento. 154

5.6.2.1.6 Fichero de Modelo. 154

5.6.2.2 Componentes de Red Neuronal de Errores de Usuario 154

5.6.2.2.1 Cliente 154

5.6.2.2.1.1 Formulario de manejo de la Red Neuronal de Errores de Usuario. 154

5.6.2.2.1.2 Informes de la Red Neuronal de Errores de Usuario. 156

5.6.2.2.2 Servidor 158

5.6.2.2.2.1 Preparación de Datos 158

5.6.2.2.2.2 Utilización de la librería de Redes neuronales 159

5.6.2.2.2.3 Resultados de Entrenamiento y de Clasificación. Matrices de Confusión. 160

5.6.3 Minería de Datos sobre Análisis de Causas Origen de los Errores 162

5.6.3.1 Definición de Funciones de Minería de Datos para Análisis de Causas 163

5.6.3.1.1 Entidades Implicadas 163

5.6.3.1.2 Proceso de Registro de la información para detección de causas y posterior análisis. 164

5.6.3.2 Componentes para la Minería de Datos sobre Análisis de Causas Origen de los Errores. 165

5.6.3.2.1 Cliente 165

5.6.3.2.1.1 Formulario de Registro de Causas de Errores. 165

5.6.3.2.1.2 Formulario de Análisis de Causas, y Relaciones con otras entidades del proceso de Gestión de Errores. 166

5.6.3.2.1.3 Visualización de Análisis de Causas. Informes. 167

5.6.3.2.2 Servidor. Librería de Consulta y Obtención de Informes de Causas. 170

6 ANÁLISIS DE VIABILIDAD 171

6.1 VALIDACIONES BÁSICAS DEL SISTEMA 172

6.2 VALIDACIÓN DE MODELO PREDICTIVO DEL SISTEMA 173

6.2.1 Definición y Datos del Modelo de uso. 173

6.2.1.1 Estructura de la Compañía 173

6.2.1.2 Estructura de Usuarios del Sistema 175

6.2.1.2.1 Grupos y Permisos (Roles y Responsabilidades) 175

6.2.1.2.2 Usuarios 176

6.2.1.3 Entornos de Trabajo 177

6.2.1.3.1 Aplicaciones 177

6.2.1.3.2 Entornos Definidos 178

6.2.1.3.3 Máquinas 179

6.2.1.3.4 Información de cada entorno. Variables propias de entorno. 179

6.2.1.4 Definición de Procesos de Gestión de Cambios 180

6.2.1.4.1 Definición de Unidades de Control. 180

6.2.1.4.2 Ciclos de Vida 180

6.2.1.4.3 Atributos propios de las Unidades de Control 182

6.2.1.4.3.1 Unidad de Control Errores 182

6.2.1.4.3.2 Unidad de Control Requerimientos 182

6.2.1.4.3.3 Unidad de Control Paquetes de Instalación 183

6.2.1.5 Tipificación de las Causas de Errores 184

6.2.1.5.1 Causas. 184

6.2.1.5.2 Relación de las Causas con los estados de error, aplicaciones y unidades de control. 185

6.2.1.6 Red Neuronal de Clasificación de Usuarios basándose en sus Errores 185

6.2.1.7 Información Histórica 186

6.2.1.7.1 Periodos ... 186

6.2.1.7.2 Evaluaciones de Usuarios Previas ... 187

6.2.1.7.3 Listado de Unidades de Control disponibles ... 189

6.2.1.7.4 Histórico de estados de Unidades de Control. ... 190

6.2.1.7.4.1 Generación de Datos Históricos Aleatorios ... 190

6.2.1.7.4.2 Información para la Red Neuronal ... 195

6.2.2 Pruebas Realizadas sobre el Modelo Predictivo ... 196

6.2.2.1 Precondiciones y Bloques de Pruebas ... 196

6.2.2.1.1 Predicción de un ciclo aleatorio ... 197

6.2.2.1.2 Predicción de un ciclo similar a otro anterior ... 198

6.2.2.2 Análisis de Resultados ... 198

6.2.2.2.1 Distribución de Resultados ... 199

6.2.2.2.2 Predicción de un ciclo aleatorio ... 201

6.2.2.2.2.1 Análisis de Causas y Errores ... 202

6.2.2.2.3 Predicción de un ciclo similar a otro anterior ... 206

6.2.2.2.3.1 Análisis de Causas y Errores ... 207

6.3 CONCLUSIÓN A LAS PRUEBAS REALIZADAS ... 210

7 CONCLUSIONES. LÍNEAS FUTURAS DE TRABAJO ... 211

7.1 CONCLUSIONES ... 211

7.2 LÍNEAS DE INVESTIGACIÓN ... 213

7.2.1 Líneas futuras de Trabajo ... 213

7.2.2 Mejoras del Sistema ... 216

8 APÉNDICES ... 218

8.1 MANUAL DE USUARIO DE LA APLICACIÓN ECFM ... 218

8.1.1 Cliente ECFM ... 218

8.1.1.1 Login en el Sistema....218
8.1.1.2 Pantalla Principal de la Aplicación....219
8.1.1.3 Pantalla de Instalación....221
8.1.1.4 Pantalla de Acciones de Actualización de Repositorio....223
8.1.1.5 Pantalla de Control de Ciclo de Vida....225
8.1.1.6 Pantalla de Análisis de Causas de Error....229
8.1.1.7 Pantalla de Generación de Informes....230
8.1.1.8 Pantalla de Recuperación de Mensajes....232
8.1.2 Repositorio ECFM....233
8.2 MANUAL DE ADMINISTRACIÓN DE LA APLICACIÓN ECFM....234
8.2.1 Software Base Relacionado....234
8.2.2 Cliente ECFM....236
8.2.2.1 Estructura de Directorios y Archivos....236
8.2.2.1.1 Archivos de Configuración....237
8.2.3 Servidor ECFM....238
8.2.3.1 Estructura de Directorios y Archivos....239
8.2.3.1.1 Archivos de Configuración....240
8.2.3.2 Arranque y Parada....241
8.2.4 Repositorio ECFM....242
8.2.4.1 Configuración....242
8.2.5 Base de Datos ECFM....243
8.3 DATOS PARA REALIZAR LA VALIDACIÓN DEL SISTEMA....244
8.4 DETALLE DE PRUEBAS DE SISTEMA REALIZADAS....245
8.4.1 Módulo de Instalación....245

8.4.2 Módulo de Acciones sobre Repositorio *246*

8.4.3 Módulo de Control del Ciclo de Vida *246*

8.4.4 Módulo de Generación de Informes *248*

8.4.5 Módulo de Comunicación Off-line *248*

8.4.6 Módulo de Red Neuronal de Clasificación de Errores *249*

8.4.7 Módulo de Análisis de Causas de Error *249*

9 BIBLIOGRAFÍA 251

Índice de Ilustraciones

ILUSTRACIÓN 1 – ELEMENTOS DE UN SISTEMA DE GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE 25
ILUSTRACIÓN 2 – ESPECTRO DE CONCEPTOS DE SISTEMAS DE GCS 27
ILUSTRACIÓN 3 – RAMAS DE VERSIONES EN CVS 31
ILUSTRACIÓN 4 – EL PROCESO DE KDD 41
ILUSTRACIÓN 5 – ÁRBOL DE DECISIÓN PARA LA CONCESIÓN DE CRÉDITOS EN UNA ENTIDAD FINANCIERA 46
ILUSTRACIÓN 6 – FUNCIÓN OR SEGÚN EL MODELO MCCULLOCH-PITTS 56
ILUSTRACIÓN 7 – ARQUITECTURA MADALINE 58
ILUSTRACIÓN 8 – RED BACKPROPAGATION 59
ILUSTRACIÓN 9 – CLASES PRINCIPALES Y FUNCIONAMIENTO BÁSICO LIBRERÍAS TORCH 67
ILUSTRACIÓN 10 – VISIÓN GLOBAL DE LA SOLUCIÓN 80
ILUSTRACIÓN 11 - ESQUEMA DEL ÁREA DE SISTEMAS DE UNA ORGANIZACIÓN 83
ILUSTRACIÓN 12 – ERROR EN PROCESO DE INSTALACIÓN 88
ILUSTRACIÓN 13 - ERROR EN PROCESO DE PRUEBAS 89
ILUSTRACIÓN 14 – ERROR EN PROCESO DE EXPLOTACIÓN 90
ILUSTRACIÓN 15 – ARQUITECTURA FUNCIONAL 91
ILUSTRACIÓN 16 – ARQUITECTURA TECNOLÓGICA 94
ILUSTRACIÓN 17 - ENTORNOS DE TRABAJO 102
ILUSTRACIÓN 18 - JERARQUÍA DE UNIDADES DE CONTROL 106
ILUSTRACIÓN 19 - CICLO DE VIDA DE UNIDADES REQ, ERR Y PACK 107
ILUSTRACIÓN 20 - PROTOCOLO DE COMUNICACIÓN CLIENTE-SERVIDOR 110
ILUSTRACIÓN 21 – COMUNICACIÓN CON REPOSITORIO CVS A TRAVÉS DE SSH 113
ILUSTRACIÓN 22 – MÓDULO CLIENTE 117
ILUSTRACIÓN 23 – SERVIDOR CONCURRENTE 126
ILUSTRACIÓN 24 – MODELO DE DATOS 131
ILUSTRACIÓN 25 – ARQUITECTURA FUNCIONAL DE REDES NEURONALES Y MINERÍA DE DATOS DEL SISTEMA. 145
ILUSTRACIÓN 26 – COMPONENTES DE REDES NEURONALES DEL SISTEMA 148
ILUSTRACIÓN 27 – INTEGRACIÓN DE REDES NEURONALES EN LA SOLUCIÓN 149
ILUSTRACIÓN 28 – FORMULARIO DE MANEJO DE RED NEURONAL DE ERRORES DE USUARIO 155
ILUSTRACIÓN 29 – PANTALLA CON EL INFORME DE RED NEURONAL DE ERRORES DE USUARIO 157
ILUSTRACIÓN 30 – GRÁFICO GENERADO PARA INFORME DE RED NEURONAL DE ERRORES DE USUARIO 158
ILUSTRACIÓN 31 – PANTALLA DE OPERACIONES CON UNIDAD DE CONTROL – CAMBIO DE ESTADO 166
ILUSTRACIÓN 32 – CAMBIO DE ESTADO A ERROR. DETECCIÓN DE LAS CAUSAS QUE LO HAN PROVOCADO. 166
ILUSTRACIÓN 33 – FORMULARIO DE ANÁLISIS DE CAUSAS DE ERRORES. 167
ILUSTRACIÓN 34 – FORMATO DE LOS GRÁFICOS DE ANÁLISIS DE CAUSAS. 169

ILUSTRACIÓN 35 – GRÁFICO DE MINERÍA DE DATOS DE ANÁLISIS DE CAUSAS..... 170
ILUSTRACIÓN 36 – ESTRUCTURA DE LA COMPAÑÍA MODELO..... 174
ILUSTRACIÓN 37 – GRÁFICO DE MINERÍA DE DATOS DE ANÁLISIS DE CAUSAS..... 181
ILUSTRACIÓN 38 – GENERADOR DE DATOS HISTÓRICOS ALEATORIOS. 194
ILUSTRACIÓN 39 – GRAFICO GENERAL DE ERRORES EVALUACIONES DE TODOS LOS USUARIOS..... 200
ILUSTRACIÓN 40 – GRAFICO GENERAL DE ERRORES QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS 202
ILUSTRACIÓN 41 – GRAFICO DEL PERIODO 2006-2007 DE ERRORES QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS..... 203
ILUSTRACIÓN 42 – GRAFICO GENERAL DE CAUSAS QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS 204
ILUSTRACIÓN 43 – GRAFICO DEL PERIODO 2006-2007 DE CAUSAS QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS 205
ILUSTRACIÓN 44 – GRAFICO DEL PERIODO 2000-2001 DE ERRORES QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS..... 207
ILUSTRACIÓN 45 – GRAFICO DEL PERIODO 2007-2008 DE ERRORES QUE PROVOCAN LAS EVALUACIONES MÁS BAJAS..... 208
ILUSTRACIÓN 46 – GRAFICO DEL PERIODO 2000-2001 DE USUARIOS CON LAS EVALUACIONES MÁS BAJAS 209
ILUSTRACIÓN 47 – GRAFICO DEL PERIODO 2007-2008 DE USUARIOS CON LAS EVALUACIONES MÁS BAJAS 209

Resumen

La Gestión de la Configuración del Software se ocupa de la identificación, definición y control de los objetos de un sistema. Dentro del Aprendizaje Automático, las Redes Neuronales Artificiales son estructuras para la detección de patrones y tendencias en conjuntos de datos, basando su funcionamiento en el comportamiento neuronal biológico.

En este trabajo de investigación, se elabora un modelo para combinar las disciplinas anteriormente citadas, de manera que, utilizando la información histórica de los procesos de Gestión de la Configuración del Software, se pueda predecir conocimiento útil para una organización, utilizando técnicas de Aprendizaje Automático como las Redes Neuronales Artificiales. Dicho conocimiento se concreta en este trabajo en la identificación de comportamientos de los diferentes usuarios de un sistema, así como la mejora en la detección de errores en el ciclo de vida del software y de las causas que los motivan.

Para completar el trabajo de modelización, se implementa un completo sistema con diferentes componentes integrados entre sí en diferentes plataformas software y hardware. Se realiza asimismo una simulación con un modelo de elevado volumen de datos para validar las predicciones realizadas por el sistema.

Se muestra por tanto que el establecimiento de un modelo predictivo basado en los datos derivados de los procesos de la Gestión de la Configuración del Software puede contribuir a la mejora futura de las actividades relacionadas con el ciclo de vida de las aplicaciones de una organización. Por consiguiente, se añade un importante valor a dichos procesos que podrá influir en decisiones de relevancia dentro de dicha organización.

Abstract

Software Configuration Management looks after identification, definition and control of system objects. In the Machine Learning field, Artificial Neural Networks are structures for patterns and trends detection in data sources, basing their operation in biological neural behavior.

In this research job, a model for combining both subjects quoted is elaborated, in such a way that, using Software Configuration Management processes historical information, useful knowledge for an organization can be predicted, using Machine Learning Techniques and Artificial Neural Networks. This knowledge is concreted in this job in the identification of the behavior of different users of a system and in the improvement of failure detection in lifecycle processes and their origin causes.

To complete the modeling job, a full system is implemented with some integrated components in different hardware and software platforms. A simulation with high volume of data is also made, for validating the predictions made by the system.

It's shown that the establishment of a predictive model based on information derived of Software Configuration Management Processes can contribute to future improvement of activities related to the lifecycle of applications in an organization. So, it's added an important value to these processes, which could have influence in significant decisions inside this organization.

1 Introducción

En este trabajo se va a realizar una investigación basada en dos grandes disciplinas de la Informática: La Ingeniería del Software y la Inteligencia Artificial. Dichas disciplinas han experimentado grandes avances en las últimas décadas y tienen a su vez multitud de materias de estudio. De todas ellas se va a trabajar esencialmente en la Gestión de la Configuración del Software y en la Minería de Datos.

En esta primera sección se van a ubicar los diferentes aspectos de trabajo que cubre este documento. Se señalarán los Campos de Investigación abordados, mencionando las diferentes áreas de conocimiento exploradas. Asimismo se dará una idea del problema que se va a abordar, así como las líneas de la solución a dicho problema. Finalmente se describirá la estructura y la organización de este trabajo.

1.1 Campos de investigación

La *Ingeniería del Software* (Software Engineering, SE) es la rama de la Ingeniería que se encarga de crear y mantener las aplicaciones de software, aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, el propio ámbito de dichas aplicaciones así como otros campos y técnicas relacionados.

Dentro de la Ingeniería del Software se centrarán las investigaciones de este trabajo en la *Gestión de la Configuración de Software* (Software Configuration Management, SCM). La Gestión de la Configuración del Software es la disciplina que se ocupa del mantenimiento de la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan [MAP-01].

El proceso de Gestión de la Configuración del Software abarca todas las etapas del ciclo de vida de las aplicaciones y afecta a todas las áreas de los sistemas de información de las

organizaciones, así como a todos los elementos susceptibles de configuración. Se tratará en este trabajo de elaborar un modelo nuevo para dichos elementos, áreas y aplicaciones.

La otra gran materia sobre la que se va a centrar esta investigación es la *Inteligencia Artificial* (Artificial Intelligence, AI). Se define como la disciplina que se ocupa de la automatización de tareas que requieran un comportamiento inteligente. Dichas tareas pretenden solucionar diferentes tipos de problemas en multitud de campos tales como la economía, la medicina, la ingeniería, la defensa o la propia construcción de aplicaciones de software, entre otros.

El *Aprendizaje Automático* (Machine Learning, ML) engloba una serie de paradigmas, algoritmos, técnicas y teorías que permiten a un programa aprender [Mit-97]. Dichas técnicas se ubican en un proceso más amplio llamado *Minería de Datos* (Data Mining, DM). La Minería de Datos engloba los diferentes mecanismos utilizados para descubrir información que no están representados explícitamente en un conjunto de datos [Gro-00][Per-99].

Dentro de las técnicas de Minería de Datos, y Aprendizaje Automático, se investigará en este trabajo utilizando *Redes Neuronales Artificiales* (Artificial Neural Networks, ANN). Estas son estructuras de aprendizaje fundamentadas en el funcionamiento biológico de las neuronas, que se utilizan para la detección de tendencias y patrones en conjuntos de datos.

Los campos aquí mencionados, así como las particularidades investigadas de ellos, se detallarán en la sección “Estado de la Cuestión”.

1.2 Problema abordado

En una compañía que mantenga desarrollo de software, entendiendo por compañía cualquier organismo colectivo independientemente de su dimensión, normalmente existen una serie de procesos para el control de dicho software. Estos procesos son los procesos de Gestión de la Configuración del Software de dicha Compañía.

El Software como tal evoluciona para adaptarse a las necesidades de dicha compañía, o bien para corregir los propios errores derivados de diferentes fuentes. Esta evolución se reflejará en nuevas Versiones de los elementos cuya configuración se controla.

Pues bien, *el Problema que se aborda* en este trabajo es la posibilidad de detectar tendencias o patrones en los procesos de Gestión de la Configuración de una compañía que representen

conocimiento útil para el futuro de la misma. Se investigará sobre si es posible modelizar los elementos pertenecientes a los procesos de Gestión de la Configuración del Software, de manera que se al aplicar técnicas de detección de tendencias o patrones en los datos, se pueda encontrar información que pudiera resultar de valor para la compañía.

En la sección "Planteamiento del Problema" de este documento se detalla a fondo el problema que se ha planteado.

1.3 Aproximación a la solución

Para tratar de encontrar solución al problema que se plantea, las líneas de actuación deben de ser las siguientes:

- *Modelizar el área de sistemas de una compañía*: Se debe tratar de dar una estructura de organización genérica en la que se definan diferentes roles, junto con sus responsabilidades. En concreto se focalizará el modelado en el área de sistemas de dicha organización o compañía.

- *Modelizar los procesos de Gestión de la Configuración del software de dicha compañía*: Se deben definir cuidadosamente y de forma flexible el ciclo de vida de las aplicaciones, así como los elementos de Gestión de la Configuración, susceptibles de ser controlados y de aportar información al sistema.

- *Definir los procesos de Descubrimiento de Datos*: Aplicando técnicas de detección de tendencias y patrones. Definir los conjuntos de datos fuente y los datos que se quieren obtener.

- *Construir los diferentes componentes de la solución global*: La solución al problema planteado consistirá en un sistema o herramienta con diferentes componentes. Se deberán analizar dichos componentes y construirlos posteriormente.

- *Elaborar un modelo para validar el sistema*: Se debe construir un modelo que permita validar la solución construida y determinar si el sistema es capaz de predecir la información buscada.

En la sección “Solución Propuesta” de este trabajo se detalla en análisis y la implementación solución al problema planteado. Asimismo en la sección “Análisis de Viabilidad” se analizan todas las pruebas realizadasal sistema.

1.4 Organización de la memoria

En primer lugar se hará un recorrido por el *Estado de la Cuestión*, donde se revisará con detalle cada uno de los campos de investigación y se mencionarán diferentes referencias documentales de dichos campos.

Posteriormente se analizará la *Hipótesis* sobre el trabajo de investigación, es decir, los objetivos y líneas de dicho trabajo. Posteriormente se detallará el *Planteamiento del Problema* concreto sobre el que trata este trabajo de investigación, indicando los aspectos que se van a incluir en la solución.

Tras plantear el problema, se detallará la *Solución Propuesta* para el mismo. Se revisará toda la modelización planteada, los análisis realizados, las arquitecturas propuestas y finalmente los aspectos de construcción y tecnología asociados a la implantación real de la solución. Será la sección con más detalles y la más compleja a priori.

Se mostrarán después las pruebas realizadas a la solución propuesta en un *Análisis de Viabilidad*. Estas pruebas constarán de dos tipos: pruebas de sistema y pruebas con volumen de datos. Nótese que los modelos predictivos y de detección de patrones, necesitan cierto volumen de información con lo que se incorporará un modelo completo con cierta cantidad de datos.

Finalmente se detallarán las *Conclusiones* obtenidas del trabajo, así como las principales áreas de investigaciones futuras. Igualmente, se describirán en apéndices algunos aspectos de detalle del trabajo de investigación no incorporados en secciones anteriores por diferentes motivos. Asimismo se hará una descripción bibliográfica detallada de los documentos, libros y artículos que se han utilizado como soporte para el trabajo.

2 Estado de la Cuestión

En esta sección se va a documentar el estado de la cuestión en los temas de referencia de este trabajo de investigación. Estos son principalmente.

- Gestión de la Configuración del Software.
- Descubrimiento de Conocimiento en Base de Datos: Minería de Datos y Redes Neuronales.

2.1 Gestión de la Configuración del Software

Se habla a continuación de la Gestión de la Configuración del Software (GCS, Software Configuration Management, SCM). Se mencionarán los principales conceptos en estos sistemas. También se hablará de algunas herramientas, así como modelos y arquitecturas de sistemas de GCS. Se detallará especialmente el sistema CVS, que ha sido el elegido para este trabajo de investigación.

2.1.1 Definiciones

La *Gestión de la Configuración del Software* es la disciplina que se encarga del control de la evolución de los sistemas Software [Dar-93-1]. La IEEE da también una definición en el estándar 729-1983, como el proceso de identificar y definir los objetos de un sistema, controlando sus cambios a lo largo de su ciclo de vida, registrando e informando de los estados de dichos objetos y de los cambios y verificando su corrección y su completitud [Fei-90][IEE-87].

En estas definiciones se resaltan los siguientes *Aspectos operativos* [Dar-93-1]:

- Identificación: Reflejar la estructura de un producto, identificar sus componentes y sus tipos de una forma única y accesible.

- Control: Controlar las diferentes releases y cambios de un producto a lo largo de su ciclo de vida.

- Registro de Estados: Registrar e informar sobre los estados de los componentes con el fin de poder tener estadísticas de los componentes del producto.

- Auditoria y Revisión: Poder validar que la configuración de un determinado producto es una colección de componentes correcta y completa.

En esta definición están contenidos conceptos básicos como *revisión, línea base, versión y objeto de configuración*. No obstante, en algunos sistemas de GCS se incorporan conceptos de *roles y responsabilidades, entornos de trabajo*, así como mecanismos para que trabajen en grandes proyectos muchas personas de forma eficiente minimizando conflictos. Para ello, se introducen los siguientes aspectos:

- Fabricación: Actividades relacionadas con la Construcción y Compilación de un producto.

- Gestión de Procesos: Implementación de las estructuras de roles y responsabilidades de una compañía y de los ciclos de vida.

- Trabajo de Equipo: Control de las interacciones de múltiples usuarios sobre un mismo producto.

Nótese que en el sistema que se va a implementar se contemplarán los aspectos anteriormente mencionados.

2.1.1.1 Sistemas de Gestión de la Configuración del Software

A continuación, una vez se han definido lo que es Gestión de la Configuración del Software y sus aspectos operativos se va a hablar de Sistemas de Gestión de la Configuración del

Software (en adelante Sistemas de GCS). Nótese que el título de este trabajo es "Sistema Evolutivo de Gestión de la Configuración del Software", por tanto se pretende implementar un sistema de GCS.

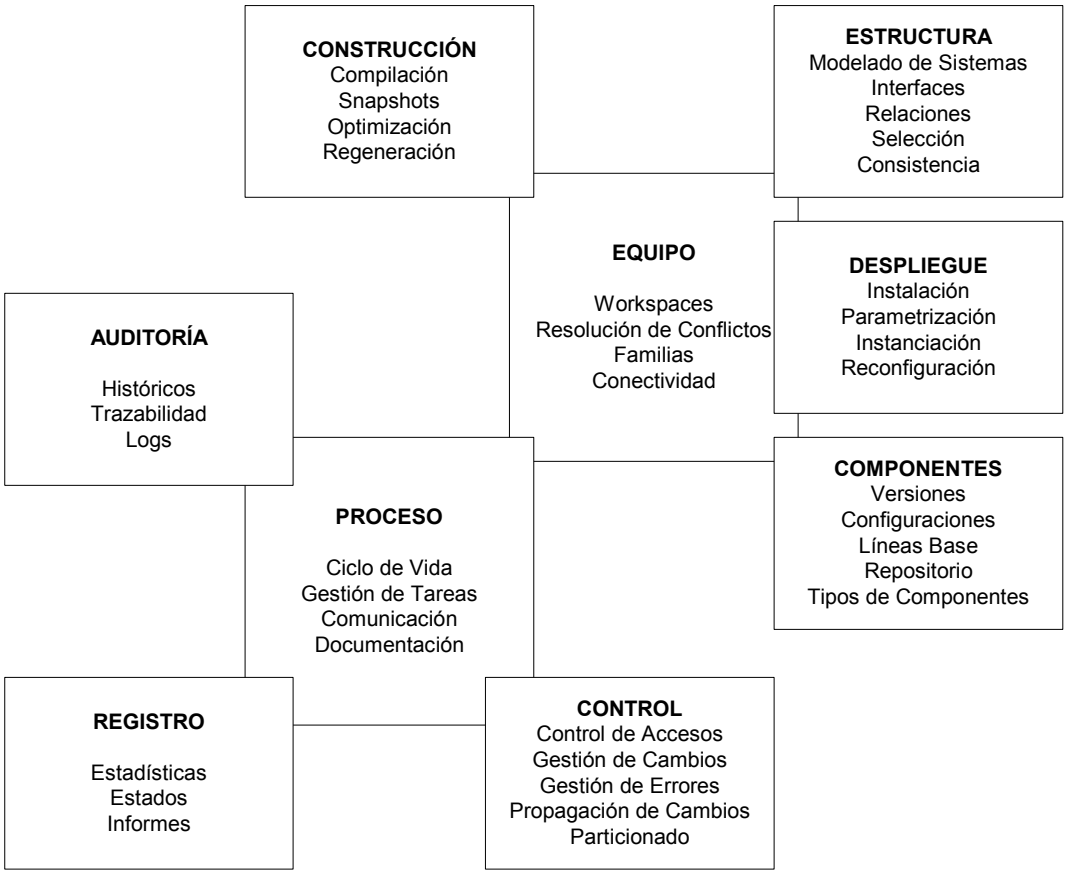


Ilustración 1 – Elementos de un Sistema de Gestión de la Configuración del Software

En el diagrama anterior [Fru-99] se describen los diferentes componentes funcionales que puede cubrir un sistema de GCS. Nótese que no todos los sistemas cubren todos los aspectos anteriores. Asimismo hay sistemas de GCS que cubren otras características no relacionadas con configuraciones. Los elementos descritos en el diagrama se toman como parámetros para evaluar funcionalmente los diferentes sistemas de GCS.

Estos son [Dar-93-2]:

- Procesos: Componentes de soporte a la evolución del producto. Relacionadas con esta área se incluyen:
    - *Auditoria*: Elementos que se utilizan para el registro las acciones que ocurren en un producto y su proceso.
    - *Registro*: Elementos del sistema para generar y obtener las estadísticas de un producto y su proceso.
    - *Control*: Elementos para saber cómo, quién y cuándo hacen cambios en el sistema.
- Equipo: Componentes que permiten a un equipo trabajar en un conjunto de productos. Dentro de este grupo se relacionan:
    - *Construcción*: Elementos para el soporte a la compilación de un producto.
    - *Estructura*: Elementos para representar controladamente la arquitectura de un sistema.
    - *Despliegue*: Elementos para controlar la instalación de un determinado producto.
    - *Componentes*: Elementos para la identificación, almacenamiento y acceso a los componentes de un producto.

2.1.1.2 Conceptos

Una vez descritos los aspectos principales de los Sistemas de GCS, se describen de forma breve diferentes conceptos manejados cuando se trabaja con ellos.

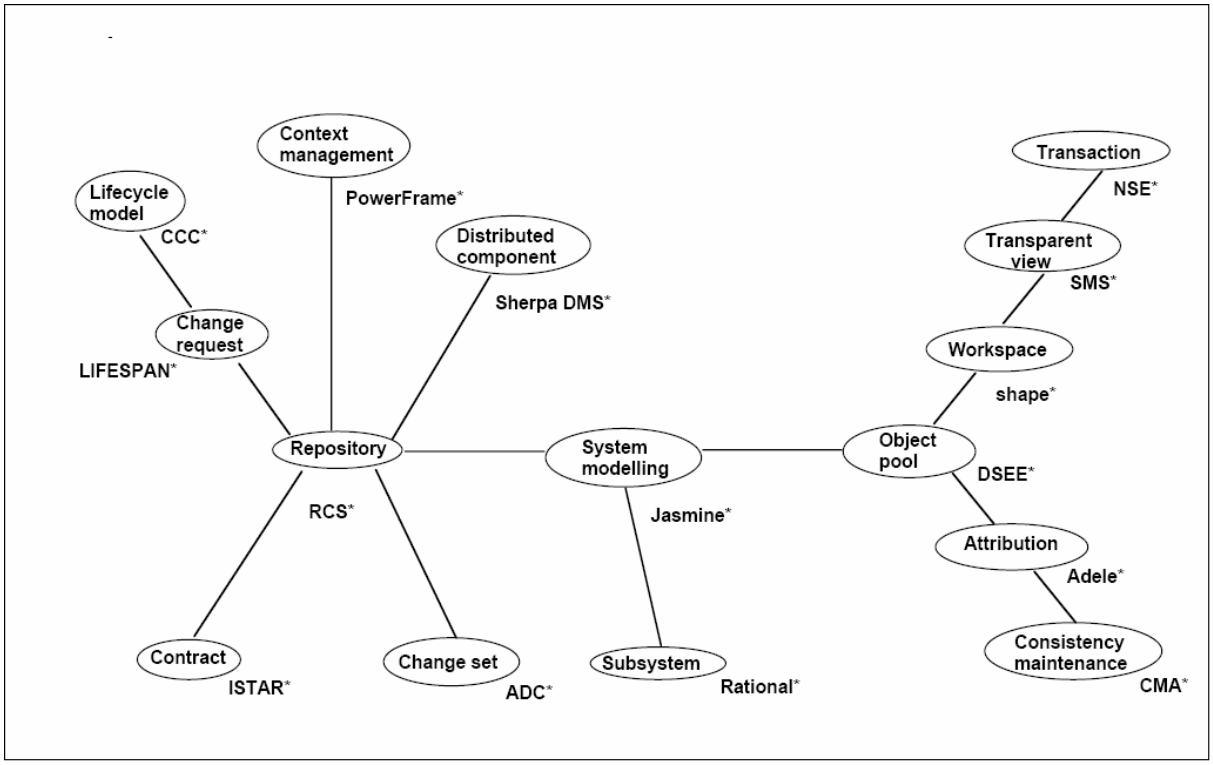


Ilustración 2 – Espectro de Conceptos de Sistemas de GCS

El diagrama anterior muestra un conjunto de *conceptos relacionados con GCS* junto con ejemplos de herramientas de GCS relacionadas con ellos. En este diagrama no aparecen explicitados muchos sistemas de GCS, si bien están relacionados con algunos de estos conceptos [Fei-90] [Dar-92] [Dar-93-1] [Dar-93-2] [Est-00] [MAP-01] [Whi-01]. De todos ellos, se describen a continuación los más directamente relacionados con este trabajo de investigación

- Repositorio: El repositorio es una *librería centralizada de ficheros* y proporciona el *control de versiones* y la forma de acceder a dichas versiones. Los cambios en el repositorio no se hacen sobre el fichero almacenado como tal sino generando una nueva versión del objeto en el repositorio. Dicha generación se hace a través del proceso de "Check out", que implica obtener el objeto para modificar, y "Check in", que implica registrar la nueva versión. Los usuarios pueden por tanto recuperar cualquier revisión del objeto. Un ejemplo claro es el sistema CVS, que se verá en detalle posteriormente.

- Solicitud de Cambio: El modelado de Solicitud de Cambio (Change Request) aparece en el sistema LIFESPAN [Ced-05] y ha sido posteriormente implementado en múltiples herramientas de GCS. Se trata de implementar *un flujo de estados*, y una serie de roles y responsabilidades que actúan sobre un formulario de Solicitud de Cambio, que puede ser una solicitud de evolución del sistema o bien la necesidad de una corrección. Según va pasando por determinados estados actúan los diferentes usuarios según sus roles.

- Modelo de Ciclo de Vida: El modelado de ciclos de vida consiste en *separar el ciclo de desarrollo en diferentes fases*, tales como desarrollo, pruebas, preproducción y producción, por ejemplo. De esta manera según llega a cada fase las actividades sobre el software así como los usuarios que las realizan son los responsables de cada una de las etapas.

- Modelado de Sistemas: Consiste en la *descripción de un producto, con su estructura, sus componentes* y la forma de compilarlo o instalarlo. Asociado a un producto se incluyen las reglas que guían su compilación para la realización del posterior despliegue en los diferentes entornos de trabajo.

2.1.2 Herramientas de Gestión de la Configuración del Software

Como se ha mencionado en la sección anterior, existen multitud de Sistemas de GCS. En este trabajo de investigación se va a utilizar Concurrent Versions System (CVS), por ello se va a describir en detalle el análisis y la motivación de la elección de dicha herramienta. Nótese que el sistema de GCS que se pretende implementar va a tener múltiples componentes, y CVS será uno de ellos integrado con el resto, en concreto el que hará las funciones de repositorio de versiones, ya que las funciones de modelado serán implementadas con otras herramientas y desarrollos.

También se revisarán en esta sección brevemente algunas otras herramientas de GCS.

2.1.2.1 Concurrent Versions System (CVS)

CVS (http://ximbiot.com/cvs/cvshome/) es una *herramienta para el control de versiones* [Ced-05] [Ber-90]. Se trata de un sistema que permite realizar los cambios sobre el software en un *repositorio* centralizado. Dicho repositorio físicamente es un directorio donde el sistema realiza la gestión de las versiones. El acceso a este repositorio no se hace directamente a través de dicho directorio sino a través de los comandos que proporciona CVS para la generación, obtención y gestión de versiones.

Hay que dejar claro que CVS no tiene implementado ningún mecanismo de instalación y despliegue de software, no tiene ningún proceso de ciclo de vida implementado o mecanismos de notificación a usuarios. Esta precisamente ha sido una de las motivaciones de elegir la herramienta CVS para este trabajo, ya que dichos procesos se deben implementar a medida del problema en cuestión que se va a tratar.

Asimismo hay que mencionar que se trata de un software de libre distribución, lo que también ha favorecido su elección para esta investigación.

En la sección “Repositorio ECFM” del Manual de Administración de la aplicación se puede ver en detalle la configuración del repositorio para este trabajo. Veamos a continuación las principales acciones que se realizan con CVS y que se utilizarán en la implementación de este trabajo.

2.1.2.1.1 Versionado, Revisiones

El Repositorio es un depósito de código. CVS gestiona el *acceso de múltiples usuarios a las versiones* alojadas en un determinado repositorio. También ofrece la potencialidad de gestionar los conflictos entre usuarios concurrentes, de manera que si dos usuarios modifican al tiempo una misma versión comprueba las posibles interferencias de las modificaciones y notifica a los usuarios.

Los usuarios trabajan con una copia de trabajo local y acceden a través de comandos al repositorio de CVS. Veamos a continuación los mecanismos de asignación de revisiones de CVS.

2.1.2.1.1.1 Asignación de revisiones

CVS asigna número a las revisiones de los objetos de un modo secuencial, empezando por la 1.1 y continuando por la 1.2, 1.3 y así sucesivamente. Es posible para el usuario forzar el nombre de la revisión y poner otra numeración con la que esté más cómodo. CVS incrementará el último número por defecto.

Para *crear una nueva revisión* de un archivo, subiendo las modificaciones realizadas en la copia de trabajo al repositorio, se utiliza el comando '**commit**':

```
$ cvs commit –m "Información de log" [Archivos …]
```

La opción '**-m**' especifica la descripción para el archivo. Esta descripción aparece en el log histórico. Si se quiere *forzar el número de revisión*, se puede hacer con la opción '**-r**' del comando '**cvs commit**'.

```
$ cvs commit –r 4.0 [Archivos …]
```

El número que se especifique con '**-r**' debe ser mayor que cualquier número de revisión de ese archivo. Cuando se añada un nuevo archivo, el segundo número será siempre uno y el primero será igual al mayor primer número de cualquier archivo en el directorio.

En todo momento se puede *consultar información* relativa a cada archivo, como el número de revisión o la fecha de la última modificación. Para hacer esto se utiliza el comando '**status**'.

```
$ cvs status Archivo
```

2.1.2.1.1.2 Ramas de Versiones

CVS permite realizar *versionado en ramas de versiones* (Branch). Esto facilita identificar diferentes *líneas paralelas* de desarrollo. Cada rama tiene un nombre que el usuario le puede dar e internamente CVS le asocia un número de rama. Este número está compuesto por el

número de la revisión de la que parte, seguido del primer número no usado empezando por un número par (2, 4, 6…). Para crear una rama sobre una determinada versión se ejecuta un comando como el siguiente:

```
$ cvs tag –b nombre_rama Archivo
```

El número de revisión de cada elemento de la rama está compuesto por el número de la rama, seguido de otro número empezando por 1, que irá cambiando de forma sucesiva en cada revisión. El siguiente gráfico indica un ejemplo de cómo CVS gestionaría múltiples ramas en un objeto.

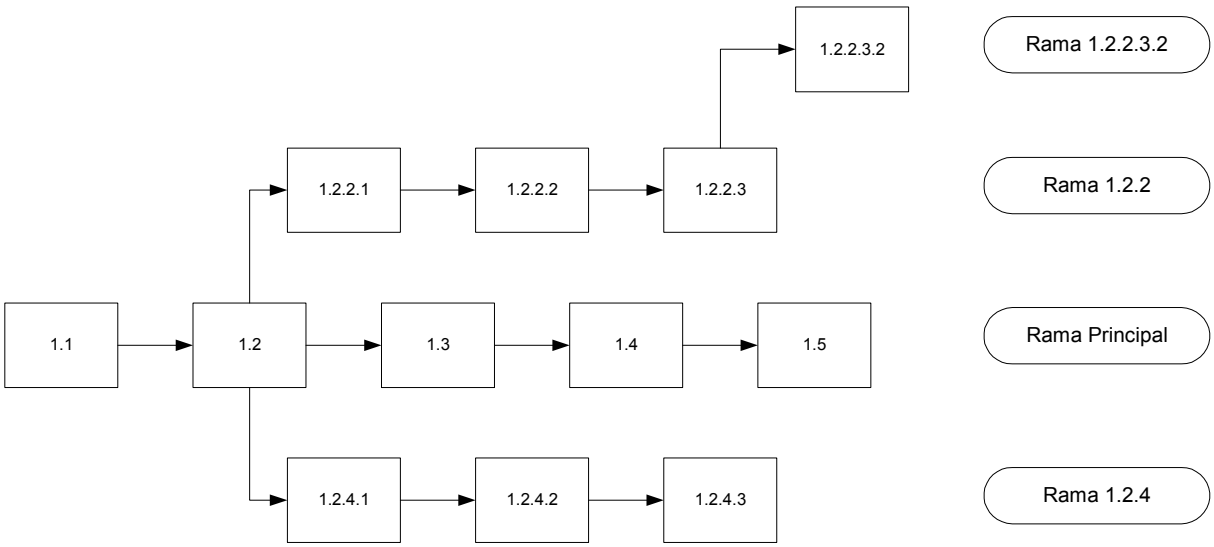


Ilustración 3 – Ramas de Versiones en CVS

2.1.2.1.2 Añadir, eliminar y renombrar archivos y directorios.

En el transcurso de un proyecto se añadirán, eliminarán y renombrarán archivos y directorios. Al realizar cualquiera de estas acciones, CVS las registra para poder acceder en cualquier momento a cualquier modificación.

2.1.2.1.2.1 Añadir archivos a un directorio

Para *añadir un nuevo archivo al repositorio* se deben seguir los siguientes pasos:

- Tener una copia de trabajo del directorio.
- Crear el nuevo archivo en la copia de trabajo.
- Usar '`cvs add [-k kflag] [-m mensaje] Archivos …`' para indicar que se quiere un control de versionado del archivo. Si el archivo contiene datos en binario es necesario la opción "-kb". La opción "-m" sirve para introducir un comentario explicativo.
- Realizar un commit con '`cvs commit Archivo`', y entonces se verá reflejado el nuevo archivo en el repositorio. El resto de usuario no podrán ver este archivo hasta que no sea ejecutado el commit.

También se puede añadir un nuevo directorio. A diferencia de la mayoría de los comandos de CVS el comando '**add**' no es recursivo, por lo que no se podrá hacer '`cvs add directorio/nuevo_directorio`', sino:

```
$ cd directorio
$ cvs add nuevo_directorio
```

Cuando se añade un archivo, este es añadido a la rama en la que se está trabajando.

2.1.2.1.2.2 Actualización

Para obtener las últimas versiones del repositorio en una copia de trabajo, se ejecutará el comando '**cvs update**'

```
$ cvs update [-D date] [Archivos…]
```

La opción '**-D**' actualiza un archivo con la versión más reciente, no posterior a la fecha indicada.

```
$ cvs –q –n update [Archivo]
```

Con la opción –q se muestra la situación del archivo respecto del repositorio. Es decir, si está disponible en la copia de trabajo la última copia disponible del repositorio o no.

La Salida del comando "update" puede ser la siguiente:

U El archivo del repositorio no existía en la copia de trabajo, o tenía cambios que no estaban en la copia de trabajo, y ha sido actualizada.

P Igual que la salida U, pero el servidor CVS envía un parche en vez del archivo completo.

A El archivo ha sido añadido a la copia de trabajo, y será añadida al repositorio cuando se realice un commit.

R El archivo ha sido borrado de la copia de trabajo y será borrado del repositorio cuando se realice un commit.

M El archivo de la copia de trabajo contiene información que no está incluida en el repositorio.

C Indica que existe un **conflicto** entre el cambio del repositorio y la copia de trabajo, a diferencia de U, indica que CVS no puede fusionar los cambios de una manera segura.

? El archivo está en la carpeta de trabajo, pero no corresponde con ninguno del repositorio.

Cuando se genera un *conflicto* al actualizar un archivo de repositorio, el archivo original es salvado sin modificaciones de la forma **.#nombre_archivo**, mientras que el archivo con nombre '*nombre_archivo'* contiene todas las líneas comunes y no comunes, estas últimas marcadas con "<<<<<<<", "=======", y ">>>>>>>". Para resolver el conflicto se edita el

archivo eliminando las marcas y las líneas erróneas y realizando un commit, con lo que se generará la versión definitiva.

Para poder ver las diferencias que hay entre dos revisiones de un mismo archivo se utiliza el comando '**diff**'.

```
$ cvs diff Archivo
```

Con el comando anterior, se pueden ver las diferencias existentes entre una revisión determinada y la última revisión del repositorio.

```
$ cvs diff –c –r rev –r rev2 Archivo
```

Se utilizaría este comando si se quieren ver las diferencias existentes entre dos revisiones del repositorio.

2.1.2.1.2.3 Eliminar archivos

Cuando se elimine un archivo, CVS dará la posibilidad de poder recuperarlo en cualquier momento. Es una eliminación lógica, no física. Marcará el archivo en el repositorio como eliminado.

Para *eliminar un archivo del repositorio* se deben seguir los siguientes pasos:

- Eliminar el archivo de la copia de trabajo.
- Usar '`cvs remove Archivo`' para decirle a CVS que realmente se quiere borrar.
- Usar '`cvs commit Archivo`' para actualizar los cambios en el repositorio.

Nótese por tanto que para *cambiar de nombre un archivo*, las operaciones a realizar serán crear una copia con otro nombre del archivo en cuestión y ejecutar un borrado del antiguo.

Si lo que se pretende es eliminar una copia de trabajo, se utilizará el comando '`cvs release –d directorio_base_de_copia`'.

2.1.2.1.3 Histórico

Existen en CVS varios mecanismos para obtener información a través del histórico. Nótese que cada vez que se realiza un commit a un archivo se le puede especificar un mensaje de log. En este trabajo de investigación, resulta de especial relevancia la información histórica como se verá más adelante.

Para *visualizar la información registrada de los cambios* se utiliza el comando '`cvs log`', el cual dará, además del mensaje mencionado, cada número de revisión, autor, fecha y número de líneas añadidas y eliminadas en caso de ser archivo de texto.

```
$ cvs log [opciones] Archivo
```

Las opciones para este comando se utilizarán para obtener diferentes filtros en su salida.

Se puede obtener también información histórica utilizando el comando '`cvs history`', el cual proporciona *información sobre operaciones* como el commit, checkout, update, rtag o release realizadas sobre los archivos.

```
$ cvs history [-report] [-flags] [-option args] [Archivos …]
```

2.1.2.1.4 Aplicaciones CVS

Debido principalmente a su adaptabilidad y a la facilidad de acceso, ya que se trata de Software de libre distribución, CVS ha sido utilizado en múltiples investigaciones y en diferentes proyectos de investigación, como es el caso de este trabajo.

Así por ejemplo, la *herramienta CVSSearch* [Che-01] viene dada por un proyecto de investigación que consiste en la realización búsquedas en los comentarios que genera CVS en su versión nativa, sobre las diferentes versiones de código, típicamente al hacer "Check in" de un elemento. A partir de estos comentarios, la herramienta aporta información al usuario que facilitaría la comprensión de los diferentes objetos de configuración y sus cambios.

También se han realizado *mejoras y extensiones sobre el propio software de CVS* ya que no sólo es de libre distribución sino que el código fuente está disponible para su modificación y

adaptación. Este es el caso de estudio de una completa mejora realizada sobre el servidor de CVS para formalizar una arquitectura de seguridad [Bru-03]. El propósito de este trabajo ha sido crear un completo módulo de seguridad con roles y responsabilidades de usuario sobre el propio repositorio, así como distribuir los componentes del repositorio entre diferentes sistemas.

Nótese que en este trabajo de investigación los cambios sobre el repositorio se regularán a través de una capa de aplicación superior al mismo y no modificando el código nativo como es el caso comentado.

2.1.2.2 Otros modelos de Sistemas y Arquitecturas de Gestión de la Configuración del Software

Existen multitud de herramientas y sistemas de GCS, tanto comerciales como de libre distribución. A continuación se listan algunos de ellos, indicando brevemente sus características.

2.1.2.2.1 Herramientas de Control de Versiones

Los sistemas de control de versiones son aquellos que, como CVS, ofrecen potencialidades para la *gestión de las versiones de los objetos* en repositorios. Estos sistemas no tienen implementados procesos de ciclo de vida o de registro de cambios. Se va a mencionar a continuación un conjunto de ellas que son de libre distribución.

- *SCCS*: El sistema SCCS (Source Code Control System) viene implementado con la mayoría de distribuciones de UNIX. Sus capacidades son limitadas pero puede servir para un control básico de versionado.

- *BCS*: Baseline Configuration System, se creó para sistemas UNIX quedando posteriormente obsoleto. Su funcionamiento es similar al de SCCS.

- *RCS*: Es similar a SCCS (ftp://prep.ai.mit.edu/pub/gnu/rcs/).

- *CVS*: Vista en detalle en la sección anterior.

- *Aegis*: Contiene una diferencia esencial con CVS, a la hora de hacer Check in ya que esta herramienta requiere antes de ello que un objeto suba de versión se deben hacer múltiples chequeos sobre él: que esté probado, que esté revisado y que se haya comprobado después de hacer el despliegue (http://www.canb.auug.org.au/~millerp/)

- *Subversion*: Una herramienta que extiende algunas capacidades de CVS (http://subversion.tigris.org/)

2.1.2.2.2 Herramientas para construcción y despliegue

Existen también herramientas que sirven para la *automatización de tareas asociadas con la construcción y el despliegue* de software en diferentes entornos. A continuación se da una breve reseña se da a continuación.

- *Make*: La herramienta que viene con múltiples distribuciones de UNIX. Se basa en incluir una serie de reglas de construcción y los archivos que intervienen en ellas. La herramienta tiene en cuenta las fechas de los elementos para detectar si es necesario o no construirlos.

- *Ant*: Esta herramienta se basa en la ejecución de una serie de tareas implementadas en Java formadas en un árbol XML para diferentes automatizaciones asociadas al despliegue de aplicaciones (http://jakarta.apache.org/ant/)

- *Cons*: Construido en Perl aporta capacidades no implementadas en Make (http://www.dsmit.com/cons/). Funciona tanto en entornos UNIX como Windows.

- *SCons*: Como “Cons” solo que está escrito en Phyton (http://www.scons.org/)

- *Jam*: Es una reducción de Make, para compilaciones y construcciones sencillas (http://www.perforce.com/jam/jam.html)

2.1.2.2.3 Herramientas Comerciales

Existen en el mercado cada vez más herramientas y con mayores capacidades dado que cada vez son más las compañías que dan importancia a los procesos de GCS. Dichas herramientas asimismo cubren cada vez más capacidades asociadas con dichos procesos y con otros de la compañía como las fases de diseño de código o de soporte a los procesos de explotación de los sistemas. Veamos una breve relación de ellas.

- *PVCS*: Quizá la herramienta comercial más distribuida en las empresas. Se trata de una suite de productos integrados entre sí para realizar tanto el control de las versiones, como la gestión de procesos de errores y cambios así como la construcción de software o la gestión de informes. (http://www.synergex.com/solutions/pvcs/)

- *Rational ClearCase*: Además de proveer control sobre las versiones, tiene integración con herramientas end-to-end en el ciclo de vida de las aplicaciones, desde el diseño hasta el despliegue. Se integra con múltiples entornos integrados de desarrollo como IBM Websphere Studio, Microsoft Visual Studio o Eclipse. Está integrado con Rational ClearQuest para ofrecer una solución también a la gestión de cambios y errores. (http://www-306.ibm.com/software/awdtools/clearcase/)

- *CA Harvest Change Manager*: Esta herramienta proporciona control de versiones y de releases, así como los procesos de gestión de errores y de código necesarios. También integra mecanismos de construcción de ejecutables y despliegue automáticos. (http://www3.ca.com/solutions/Product.aspx?ID=255)

- *Accurev*: Empezó como una herramienta de versionado con características que mejoraban las existentes como SCCS o RCS. Sin embargo, en la actualidad las nuevas versiones incorporan implementación de procesos, reporting o implementaciones de seguridad. (www.accurev.com)

- *VSS*: Visual Source Safe es la herramienta de Microsoft para la gestión del código y en general del software. Se trata de un repositorio de código con integración con diferentes interfaces de desarrollo de Microsoft, tales como Visual Basic, Visual C++ o Visual Fox Pro. Se incluye de hecho con el producto Microsoft Visual Studio. (http://msdn.microsoft.com/vstudio/products/vssafe/default.aspx)

2.1.2.2.4 Herramientas de Proyectos de Investigación

Existen también una serie de herramientas que se han lanzado a través de proyectos *de investigación* para diferentes fines. Veamos algunos de ellos, que aportan alguna idea a las anteriormente descritas:

- *ADELE* [Est-94]: ADELE propone un modelo de GCS orientado a describir un producto como un modelo de datos. Las acciones sobre un producto son operaciones sobre ese modelo de datos. Los componentes de un producto se representan como objetos de bases de datos con sus atributos y relaciones. Los atributos de cada objeto dan las características del mismo. Ejemplos de atributos pueden ser el tipo, que determinaría la entidad a la que pertenece o si es ASCII o Binario para la forma en que gestionar las versiones.

- *JASMINE* [Mar-86]: Esta Herramienta aporta la idea del modelado de sistemas software. Un producto viene descrito por las relaciones entre sus componentes, las reglas de construcción, las de versionado y las de verificación. Un producto viene a ser descrito por una plantilla que define su estructura de forma flexible. El objetivo es mantener la integridad de todos los componentes de un sistema Software.

- *Sherpa Design Management System* [Dei-90]: Este sistema propuso la idea de repositorios distribuidos en diferentes plataformas Hardware. Los usuarios ven un único repositorio lógico si bien el almacenamiento físico ocurre en diferentes plataformas. Es la herramienta la que se encarga de hacer las correspondientes traducciones y conversiones entre sistemas. Se aborda el tener físicamente separadas las copias de trabajo y dejar al sistema el control de cuál es la copia más reciente y advertir a los usuarios sobre posibles conflictos. Posteriormente esta idea se ha implementado en sistemas comerciales.

Se han descrito diferentes tipos de herramientas relacionadas con la GCS que cubren diferentes partes de los procesos relacionados con dicha disciplina. En este trabajo de

investigación, se utilizará CVS para la parte de control de versiones y el resto se implementará a medida por ser un problema diferente a los procesos estándar el que se plantea.

2.2 Descubrimiento de Conocimiento en Bases de Datos. Minería de Datos. Aprendizaje Automático. Redes Neuronales Artificiales.

2.2.1 Conceptos

Minería de Datos (Data Mining) es el proceso de encontrar tendencias y patrones en los datos [Gro-00]. Se trata de aplicar una serie de técnicas para la obtención de información que no está representada explícitamente en esos datos. Normalmente la utilidad de la Minería de Datos reside en encontrar dichas tendencias en grandes volúmenes de información [del-01].

Se puede considerar este proceso de Minería de Datos como un subproceso del Descubrimiento de Conocimiento en base de datos (Knowledge Discovery in Databases o KDD) [Per-99]. Según el autor que las considere, estas disciplinas se pueden identificar, pero en este trabajo se realizará la diferenciación entre ambas. Nos referiremos a *Data Mining* cuando se hable del hecho de descubrir conocimiento, lo que se enmarca en un ciclo de desarrollo más complejo (*KDD*), que comprende desde la creación del Data Warehouse para almacenar los datos, pasando por la depuración y el enriquecimiento de esos datos hasta llegar a hacer la minería de datos al final.

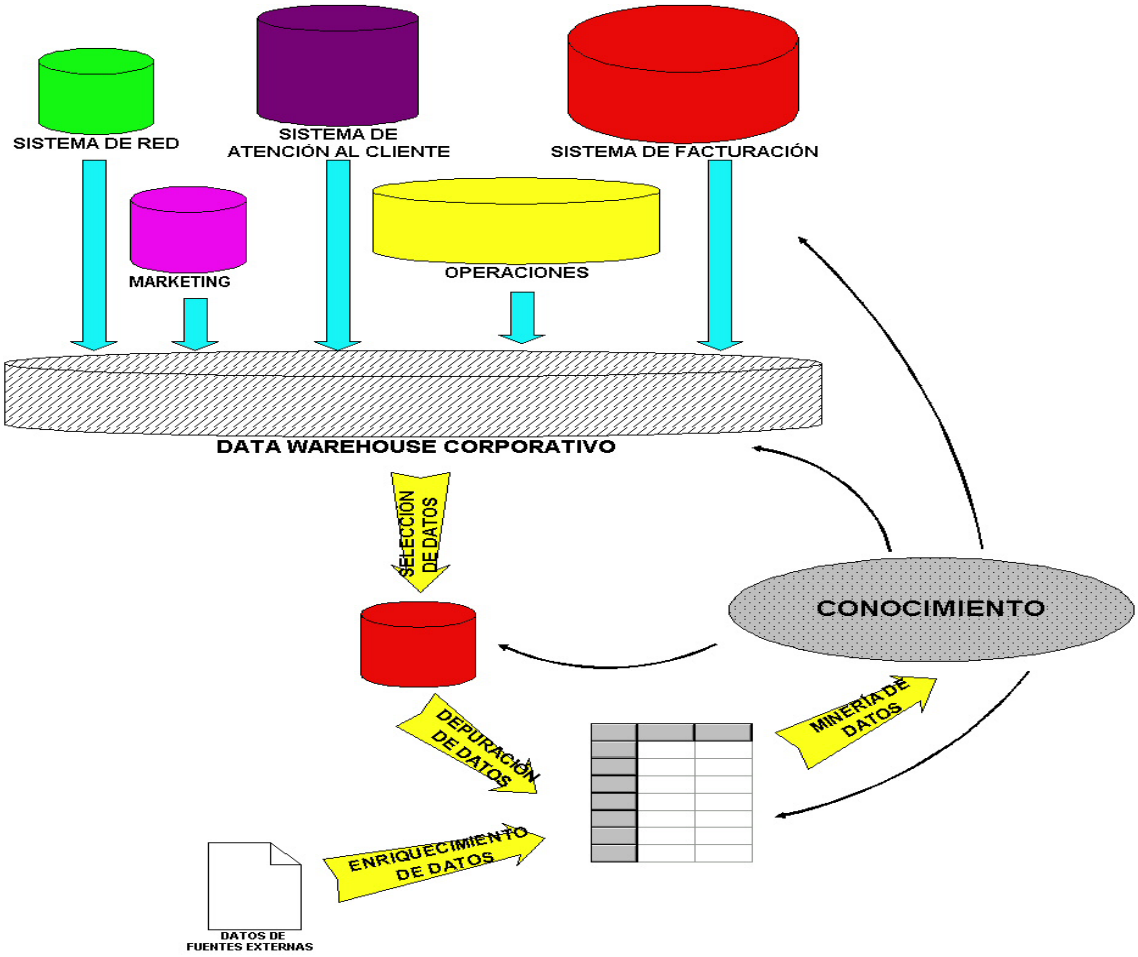


Ilustración 4 – El Proceso de KDD

Como se puede ver en la figura, en una organización normalmente se pueden tener datos de diferentes sistemas, que posiblemente se alojan en diferentes plataformas y con diferentes formatos. En el proceso de KDD es altamente recomendable aunar los datos de interés para la compañía en un *Data Warehouse Corporativo*. Se trata de un almacén de datos en el que se va a extraer la información relevante para el proceso del KDD de las plataformas dispares de la empresa. La alimentación del Data Warehouse suele ser diaria ya que conviene tener la información lo más actualizada posible para realizar el KDD con datos fiables. Se puede encontrar gran información sobre la construcción de este almacén en la industria de las telecomunicaciones en el libro de Mattison [Mat-97].

Se debe indicar que seleccionar los datos que se necesitan para el KDD es un proceso que requiere de especialistas de todas las áreas del negocio y que tiene un peso enorme en la obtención de resultados finales.

Una vez se ha definido el Data Warehouse se realiza la *Selección de Datos*. Consiste en escoger los datos necesarios para un problema concreto. Nótese que existe una enorme cantidad de información en el Data Warehouse para múltiples fines y que se tendrá que descartar aquella que no afecte para nada al problema que nos ocupa. Atención: Al desechar datos se debe tener en cuenta que puede haber información relevante para un problema que a priori no nos lo parezca.

Si en los datos seleccionados hay registros con valores que no son correctos, como por ejemplo que haya una cadena alfanumérica donde debe haber un número, se realizará una *Depuración de Datos* para eliminarlos. A veces se liga este proceso a la propia construcción del Data Warehouse.

Finalmente, se utilizarán las herramientas de *Minería de Datos* para extraer conocimiento que no está explícitamente representado en nuestra información inicial. Normalmente las empresas utilizan herramientas comerciales que se pueden englobar en una serie de grupos [Mat-97]:

- *Agentes:* Se trata de programas que, dadas una serie de preferencias del usuario sobre la información que se requiere, hacen que el propio sistema busque, detecte, analice y reporte lo referente a esa información.
- *Herramientas de Consultas e Informes:* Permiten el acceso a la información de los negocios con un mínimo conocimiento de SQL, con lo que ponen al alcance de cualquier empleado la información de las bases de datos de los sistemas, lo cual acelera la cadena de valor de una empresa.
- *OLAP (Online Analytical Processing):* Son herramientas que permiten navegar a través de los datos de un negocio con mecanismos como hojas de cálculo o bases de datos tridimensionales.
- *Visualización:* Permiten analizar los datos de forma sencilla a través de diagramas, mapas o gráficos.

- *Herramientas de Análisis Estadístico:* Realizan análisis estadísticos en la información para poder predecir, valorar o evaluar comportamientos en los negocios. De nuevo ponen al alcance estos métodos a personas que no tienen grandes conocimientos en estadística.
- *Descubrimiento de Datos:* Se englobarán en este grupo todas las herramientas que son capaces de realizar predicciones y estimaciones, lo cual está relacionado con la inteligencia artificial y las técnicas estadísticas. Aquí se incluyen técnicas como los árboles de decisión, técnicas de regresión, el aprendizaje bayesiano o los Algoritmos Genéticos, y las que se van a emplear en este trabajo: Redes Neuronales Artificiales.

Los principales tipos de problemas que se suelen afrontar con la minería de datos [Gro-00], son:

- *Aprendizaje Supervisado (Clasificación):* Son problemas en los que se trata de asignar una categoría a una serie de patrones entre una lista de posibles categorías predeterminadas. Por ejemplo, a este tipo de problemas pertenecerían preguntas tales como: ¿Cuáles son *los tipos de cliente* más propensos a abandonar mi compañía? ¿Cuáles son los *tipos de productos* que no aportan el valor necesario a nuestro negocio? En este problema existe una variable dependiente (tipo de producto o cliente) y basándose en ella se darán los resultados. Este es el tipo de problema que se va a tratar en este trabajo de investigación.
- *Aprendizaje No Supervisado (Agrupación o Segmentación):* En este caso no existe una pregunta concreta que contestar. Se agrupan en diferentes conjuntos los datos que sigan tendencias similares o que se identifiquen con patrones parecidos. Problemas tales como hacer grupos de clientes diferenciados a los que vender productos podrían ubicarse en este tipo
- *Visualización:* Es la representación gráfica de los datos. En la representación gráfica de los datos se puede identificar conocimiento que no se obtendría con los tipos de aprendizaje observados anteriormente. Por ejemplo, la observación de mapas, gráficos multidimensionales, etc... En este trabajo se van a dar soluciones basadas en la visualicación de datos.

- *Predicción:* La predicción es también uno de los objetivos en este trabajo de investigación. Son los problemas en los que se trata de anticiparse al comportamiento de una industria o de cualquier otro fenómeno o proceso. Por ejemplo predecir los clientes que van a tratar de hacer fraude a la hora de concederles un crédito, o los pacientes de un hospital que van a tener problemas cardio-respiratorios.

A continuación se van a analizar cuáles son las técnicas que hay detrás de la Minería de Datos. Para ello las siguientes secciones se centrarán en los algoritmos y técnicas usadas en las herramientas que se han denominado de "*Descubrimiento de Datos",* que son las técnicas de Aprendizaje Automático

2.2.2 Aprendizaje Automático

El *Aprendizaje Automático* (Machine Learning o ML) [Mit-97] engloba una serie de paradigmas, algoritmos, técnicas y teorías que permiten a un programa aprender. ¿Qué significa en este contexto aprender? Dada una Tarea un programa *aprende* de una Experiencia evaluada con una Medida de Rendimiento si este rendimiento en las tareas mejora con dicha experiencia. Ejemplos de Tarea podrían ser jugar al ajedrez o reconocer palabras escritas a mano. Ejemplos de Experiencia pueden ser un conjunto de partidas ganadas o una serie de palabras ya clasificadas, y ejemplos de medidas de rendimiento para estas tareas son el porcentaje de partidas ganadas o el de palabras bien clasificadas.

Para diseñar un sistema de aprendizaje automático, además de definir el problema como se ha hecho en el párrafo anterior, se debe escoger un *objetivo* que representa el tipo de conocimiento a aprender y cómo debe ser usado por la medida de rendimiento. También se elegirá una *representación* de dicho objetivo como por ejemplo a través de unos pesos, o de unos coeficientes. Además, es necesario un *algoritmo para aproximar el objetivo* ya que de lo que se dispone es de una serie de ejemplos con los que únicamente se podrá aproximar la

función objetivo. Posteriormente se pondría en marcha el sistema de aprendizaje automático, una vez completado el diseño.

A continuación se va a hacer un recorrido rápido por algunas de las diferentes técnicas de aprendizaje automático, que se describen con mayor detalle en el libro de Mitchell [Mit-97]. Entre estas técnicas se encuentran las Redes Neuronales Artificiales, que se verán posteriormente con más detalle al aplicarlas en este trabajo y los Algoritmos Genéticos, que podrían utilizarse como una de las líneas futuras de investigación de la propuesta de este documento.

Aprendizaje de Conceptos Conjuntivos (Version Spaces)

Este tipo de aprendizaje consiste en inferir una función a partir de una serie de ejemplos. Se puede decir que el problema es buscar en un espacio de hipótesis, las que satisfagan los ejemplos suministrados. Para ello se organiza el espacio de búsqueda para tener una forma ordenada de buscar entre las hipótesis: Desde la hipótesis más general, que es la que comprende todos los conceptos, normalmente representada por <?, ?, ?...> indicando "?" el valor "cualquiera de entre los posibles", a la más específica, que es la que no contiene ninguno, representada por <0, 0, 0...> siendo "0" el valor "ninguno de los posibles".

Dos conceptos se manejan para esta técnica:

- Una hipótesis h es *Consistente* con un conjunto de ejemplos si para cada ejemplo <x,c(x)>, siendo c(x) el valor de clasificación, se obtiene que h(x)=c(x)
- Los *Espacios de Versión* (Version Spaces) denotados por $VS_{H,D}$ sobre un espacio de hipótesis H y un conjunto D de ejemplos, son el conjunto de hipótesis de H que son consistentes con D.

El algoritmo asociado a esta técnica es el de *Eliminación de Candidatos* que precisamente calcula, dado un conjunto H de hipótesis, todas las hipótesis consistentes con los ejemplos de entrada D. Para ello irá acotando desde la hipótesis más general G=<?, ?, ?...> y la más específica S=<0, 0, 0...>. Con cada ejemplo se irán acotando ambos espacios hasta obtener finalmente el Espacio de Versión asociado al problema.

Se debe mencionar que no es una técnica robusta para *Datos con Ruido*. Esto quiere decir que si se introduce algún ejemplo con algún valor erróneo, el espacio de versión calculado por el algoritmo contendrá ese error y afectará a posteriores clasificaciones. Nótese que en la práctica no se utiliza apenas, pero sirve como fundamento teórico para otras técnicas de aprendizaje.

Esta técnica, al igual que las que se van a mencionar en esta sección y en el resto del documento, es de *Aprendizaje Inductivo* ya que se incluye un Sesgo o *"Bias"*, restricción o preferencia de elegir unas hipótesis sobre otras, para que el aprendizaje tenga sentido. En este caso este Bias consiste en la afirmación de que "el concepto se encuentra en el conjunto de hipótesis inicial". Sin esta afirmación, el aprendizaje sería inútil.

Árboles de decisión

En la figura se muestra un sencillo ejemplo de árbol de decisión para la concesión de créditos en una entidad financiera que ilustrará esta estructura de aprendizaje automático.

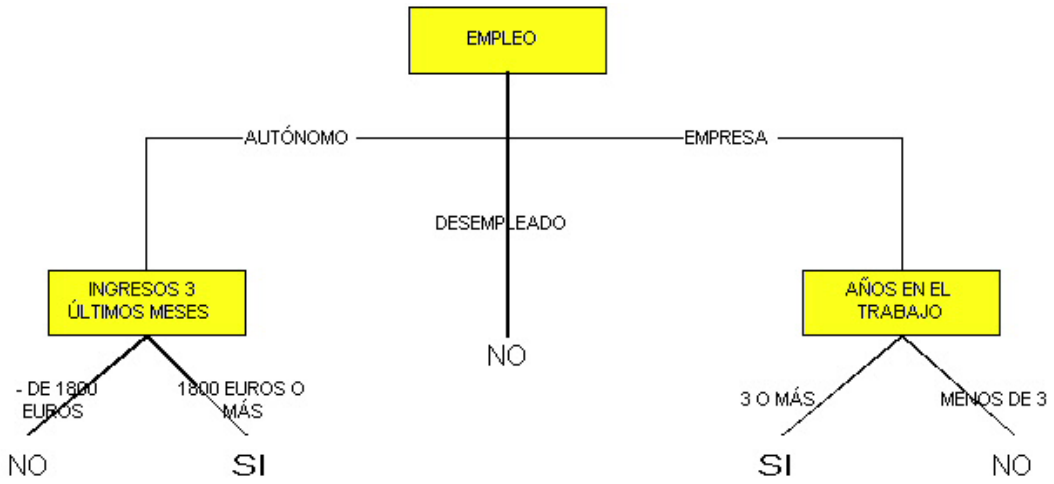


Ilustración 5 – Árbol de Decisión para la concesión de créditos en una entidad financiera

La estructura de un árbol de decisión consta de:

- *Nodos*: Son los atributos sobre los que se pregunta la clasificación de un patrón determinado.
- *Ramas*: Son los posibles valores de ese atributo.
- *Hojas*: Son los valores de clasificación.

Los árboles de decisión se suelen utilizar en problemas de Clasificación para atributos de valores discretos (o discretizables) así como en problemas de Regresión o aproximación de funciones. Para clasificar un patrón determinado, basta con realizar en cada nodo la pregunta sobre el atributo que corresponda hasta llegar a una hoja. Por ejemplo, Juan López de 65 años, Autónomo con una renta mensual los últimos 3 meses mayor a 1800 euros, será clasificado positivamente en el anterior árbol.

Véase brevemente el algoritmo básico de construcción de un árbol de decisión (ID3) [Qui-86]. De este se han sacado múltiples variantes:

- Seleccionar el nodo mejor para raíz. En concreto se selecciona el atributo con mayor "Ganancia de Información".
- Crear un nodo descendiente para cada posible valor del atributo.
- Clasificar los ejemplos basándose en ese atributo.
- Repetir el proceso recursivamente en cada una de las ramas hasta que todos los ejemplos sean clasificados con el mismo valor

La obtención del nodo raíz se hace calculando la "*Ganancia de Información*" de todos los atributos posibles. Este cálculo se hace en función del valor de la Entropía que supone la elección de un atributo en un sistema, favoreciéndose la elección de los atributos con mayor Ganancia de Información, lo cual es el Bias inductivo de los árboles de decisión.

Estos árboles soportan los ejemplos con "ruido", pero hay que prestar atención al problema del *"Sobreajuste" (o Overfitting)* que consiste en la pérdida de capacidad de generalización de la estructura debido a un entrenamiento excesivo. Este problema consiste en que si se entrena el árbol con un conjunto grande de patrones de ejemplo, cuando se trate de clasificar nuevos patrones no pertenecientes a dicho conjunto, perderá precisión al haberse construido una estructura demasiado específica para los ejemplos de entrada.

¿Cómo atajar el problema del "Overfitting"? Normalmente se realiza una "*poda*" del árbol posterior al aprendizaje, en la que se eliminan las ramas que quitan capacidad de generalización. Para realizar esta eliminación normalmente se eligen los nodos que separan los ejemplos de una forma muy desigual. Por ejemplo, si después del entrenamiento de 1000 patrones ejemplo se obtiene un nodo que clasifica 999 de ellos positivamente y 1 negativamente, será eliminado ya que no está aportando información útil para la separación.

Asimismo, es muy común el pasar el árbol a reglas [Qui-93] tales como "Si es un autónomo y cobra menos de 1800 euros, entonces NO", y posteriormente hacer procesos de poda sobre dichas reglas.

También se suele dividir el conjunto de patrones de ejemplo en *el Conjunto de Prueba y el Conjunto de Validación* realizando el entrenamiento con el primero y validando la capacidad de generalizar en el segundo. Si no es excesivamente grande el conjunto de patrones de ejemplo se ha demostrado empíricamente [Esp-97] que es preferible entrenar con todos ellos al árbol de decisión.

Mecanismos Bayesianos

Los algoritmos de aprendizaje bayesianos [Pea-98] [Jen-96] calculan las probabilidades explícitas para las hipótesis que se manejan en un sistema. Se fundamentan en el teorema de Bayes, que indica que para calcular la probabilidad "a posteriori" de una hipótesis h, hay que conocer las probabilidades "a priori" de la evidencia D, de la hipótesis h, y de la evidencia D dado que sucedió h:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

En esencia lo que se hace es que dado un conocimiento a priori de una serie de hipótesis, y un conjunto de evidencias observadas, se va a obtener la probabilidad de las diferentes hipótesis

a posteriori. A menudo lo que se busca es la más probable dados unos datos (MAP o Maximum a posteriori).

En la actualidad están siendo muy usadas las "Redes de Creencia Bayesiana" (Bayesian Belief Networks), en las que se representan las dependencias entre las evidencias para poder calcular las probabilidades a posteriori. Esta técnica se utiliza además para explicar otros algoritmos que no tienen un enfoque probabilístico.

Métodos de Aprendizaje basado en ejemplos

Otro enfoque de las técnicas de Aprendizaje Automático es el del Aprendizaje basado en Ejemplos [Aha-91]. En él se tiene que para cada una de las nuevas consultas se realiza una aproximación de la función objetivo. Esto es diferente a lo que se ha visto hasta ahora, ya que en las anteriores técnicas se realizaban a priori un proceso de entrenamiento y las consultas eran simples ejecuciones con unos datos de entrada para dar un valor de salida. La idea es justamente la contraria: no se tendrá proceso de entrenamiento, y por cada ejemplo que venga nuevo se va a realizar una aproximación a su valor de salida basándose en los otros ejemplos que se tienen.

Entre estas técnicas está la de los **k-vecinos** más cercanos que consiste en coger los k vecinos más cercanos, mediante la distancia euclídea, al de la entrada y devolver la media, o el valor más común de todos los valores de salida de ellos como valor de clasificación del patrón de entrada. Se le puede añadir pesos, según la distancia al vecino. Es decir, que a mayor distancia del vecino, menor será la influencia en la salida de la clasificación.

También se pueden destacar los métodos basados en **Regresión** que consisten en aproximar la salida de un determinado patrón de entrada mediante una función, que se construye incluyendo los ejemplos previamente clasificados.

Aprendizaje basado en reglas

Otra técnica comúnmente usada es el Aprendizaje Basado en Reglas. En este caso no se utiliza un árbol de decisión para formar posteriormente conjuntos de reglas, sino que *a partir de los ejemplos se generan directamente las reglas*.

Para ello se comienza con una regla sencilla, se clasifican los ejemplos según esa regla, se quitan los clasificados positivamente, y se itera para el resto de los ejemplos creando nuevas reglas cada vez más complejas. La selección de los atributos que intervienen en estas reglas se hace de manera similar a ID3. Un ejemplo de estos algoritmos está en el llamado CN2 [Cla-89] donde se buscan las reglas desde la más general a la más específica.

Asimismo existen algoritmos como C4.5 [Qui-96-1][Qui-96-2] para pasar un árbol de decisión a un conjunto de reglas sobre las que se pueden clasificar diferentes ejemplos o hacer posteriores optimizaciones.

Además de las mencionadas, otras muchas técnicas que no se han comentado aquí se usan en la disciplina del aprendizaje automático actualmente en expansión en el campo de la investigación y la aplicación.

2.2.3 Otras técnicas: Algoritmos Genéticos

Los Algoritmos Genéticos, como se verá posteriormente, podrían utilizarse como línea futura de investigación de este trabajo. Por ello, se va a hacer una introducción más detallada a los algoritmos genéticos viendo:

- Su *evolución histórica* subrayando las aportaciones más significativas a los GA
- Y los *aspectos de diseño* a la hora de implementarlos.

2.2.3.1 Perspectiva histórica

La base de los algoritmos genéticos (en adelante GA) está en el concepto de evolución de las especies. Se define *Evolución* como *el cambio en los genes de una población en el tiempo.* Biológicamente hablando, cuando se menciona "tiempo" se refiere a grandes periodos de tiempo.

En concreto, en este campo ha tenido mucha influencia la teoría de la evolución de *Darwin.* En ella se clasifican los mecanismos evolutivos en dos grupos: los que aumentan la variación genética en una población y los que la disminuyen. Los que aumentan la variación se identifican con las *mutaciones* de los genes que suceden en la reproducción y los que la disminuyen con *la selección natural* que sostiene que de una población sobreviven los que mejor se adaptan a los entornos, que son los individuos que mejor descendencia generan. También se enmarca en la selección natural el simple hecho de que los organismos que más se reproducen tienden a imponer sus alelos, o atributos genéticos, en la población.

En la selección natural no se garantiza que se llegue a una población óptima. Es más, se tiende a caer en poblaciones localmente óptimas (mínimos locales) que difícilmente salen de ese máximo local. La única forma de salir de esa situación es mediante una variación afortunada, o mutación, o bien mediante la combinación de individuos de alta adaptación.

En los años 70 John Holland diseña los primeros algoritmos genéticos [Hol-75]. Se basan en las teorías Darwinianas de la evolución en las que se afirma que las mejores soluciones de los problemas, que son individuos de una determinada población de posibles soluciones, se obtienen combinando individuos de alta adaptación, así como mediante mutaciones afortunadas sobre esa población.

Cada individuo en una población representa una posible solución al problema a resolver que puede ser por ejemplo, optimizar una función. Reciben el nombre de *cromosomas* que a su vez están formados por genes, que pueden representarse de muy diversas maneras como números reales, enteros, bits o estructuras más complejas. El *genotipo* es el conjunto de cromosomas de un organismo y su estructura, mientras que el *fenotipo* es la manifestación externa de dichos cromosomas, es decir, la representación de esos cromosomas en su entorno.

En algoritmos genéticos normalmente se identifican ambos conceptos, aunque no siempre es así, y es común hablar de individuos de un solo cromosoma. Se define *fitness* (rendimiento, adaptación, fortaleza...) de un cromosoma, como una función que dado un cromosoma devuelve el rendimiento para un problema determinado del mismo. Para cada problema se tendrá definida esta función, que ponderará la validez de una solución o cromosoma.

Definidos estos parámetros se revisa el **algoritmo** que definió Holland, que sirve como punto de partida para los estudios basados en algoritmos genéticos.

1. Inicialmente *se crea una población* de cromosomas aleatoriamente

   Se trata de inicializar aleatoriamente una población. Además de definir la función de generación de números aleatorios, se debe determinar el tamaño de la población, que es un aspecto determinante a la hora de converger a un máximo global, en vez de converger rápidamente a un máximo local.

2. Se calcula el *fitness* de cada miembro de la población.

   Para ello se tendrá que haber definido la función fitness previamente.

3. Si alguno de los miembros ha dado un resultado satisfactorio al calcular su fitness, *devolverlo como solución* y detener el problema. En otro caso seguir.

   Para lo que se deberá determinar la condición de parada. En caso de no llegar a una solución definitiva, se debe establecer un máximo de iteraciones.

4. Crear una población intermedia usando el operador de *selección* designado.

   Consiste en hacer una selección probabilística en función del valor del fitness de cada cromosoma. Es decir, que las soluciones con mayor valor del fitness, tendrán más probabilidad que las otras de ser seleccionadas.

5. Generar una nueva población aplicando los *operadores de cruce* (crossover) y de *mutación* (mutation)

   Aplicar con probabilidad “Pc” el cruce. El valor “Pc” es la tasa de cruce. Una vez escogidos aleatoriamente dos cromosomas de la nueva población intermedia, se aplicará el cruce con una probabilidad “Pc”. Es decir, que a mayor valor de “Pc”, mayor será el número de cruces en la descendencia.

Aplicar con probabilidad "Pm" la mutación. Después de aplicar el cruce, cada gen de cada cromosoma resultante esta sujeto a una posible mutación, que se aplicará con probabilidad "Pm". Nótese, que en la mayoría de los problemas se tiene un valor de "Pc" mucho mayor que el de "Pm".

6. Volver al paso 2

Solamente se menciona que la aproximación a los fundamentos matemáticos de los algoritmos genéticos está en el *Teorema del "Schema"* [Hol-75] de Holland, en el que se define un "schema" como un conjunto de strings o cromosomas con una serie de genes iguales en las mismas posiciones, es decir que siguen un patrón. Este teorema da una idea matemática del funcionamiento de los operadores de cruce y mutación, basándose en dichos "schema" (en plural "schemata"). El teorema concluye que los "schemata" con mayor adaptación (o fitness) aparecerán con mayor frecuencia en posteriores generaciones.

La hipótesis de los *"Building Blocks"* o ladrillos [Gol-89], en la que Goldberg extiende el teorema de Holland, definiendo "Building Blocks" como "schemata" de alto fitness, afirma que la combinación de dos "Building blocks" tiende a producir un descendiente de mayor fitness. No se analizará la demostración teórica de esta conclusión, pero se debe tener en cuenta a la hora de realizar cruces en cromosomas. Si se eligen múltiples puntos de combinación, como es el caso del operador de cruce uniforme, se corre el peligro de estar desbaratando (disruption) los bloques de construcción del sistema, con lo que se actuaría en contra de esta hipótesis.

Por último se debe mencionar la aportación de Koza [Koz-92] en la que define la "Programación Genética" (*"Genetic Programming" o GP).* En este enfoque la población que se combina son programas informáticos en vez de strings de bits. La representación de estos programas se hace mediante árboles, en los que las funciones son los nodos del árbol, y sus hijos son los argumentos con los que se llama a esa función, de tal forma que las hojas de un árbol son siempre símbolos terminales.

El fitness de uno de estos programas, es el resultado de ejecutarlo con los datos de ejemplo. Los operadores que se definen para dar una descendencia son los típicos de cruce y de mutación. En este caso el cruce consiste en escoger dos nodos al azar de cada uno de los

padres, e intercambiar los subárboles que cuelgan de esos nodos, dando lugar a dos hijos nuevos. La mutación pasa por hacer cambios en algún nodo con alguna probabilidad como por ejemplo cambiar una función, o alterar el valor de una hoja del árbol.

2.2.3.2 Aspectos de Diseño

Se trata a continuación de señalar cuáles son los aspectos principales a revisar a la hora de implementar un GA. Esta sección es fundamental a la hora de realizar posibles combinaciones con ANN, ya que se tienen que ajustar estos aspectos en cada uno de los problemas en los que se utilicen GA.

Cuando se ejecuta un algoritmo genético sobre una población, la cuestión primera y determinante para el resto es de qué población hablamos. La pregunta que se hace es: ¿Qué población está evolucionando? o bien *¿Cómo representar las solucuones al problema en cromosomas?*

En primer lugar se debe pensar a alto nivel qué es lo que representa un cromosoma, es decir, la solución del problema. Puede ser un string de bits, o bien una serie de números reales, o bien, como se va a ver más adelante, una red neuronal. Se elegirá en este momento si es necesario representar el genotipo y el fenotipo, o bien si van unidos.

Una vez elegida la representación de la población, otra cuestión crucial es determinar los *operadores* de cruce, mutación, selección y otros a utilizar, así como las tasas de aplicación de los mismos. En este momento se determinarán las restricciones y casos especiales de aplicación de estos operadores.

Los *operadores de cruce* pueden ser:

- "1-point" lo que significa que se seleccionará un punto de cruce en los cromosomas padre, y los cromosomas hijos tendrán la primera mitad de uno de los padres y la segunda del otro.
- "2-point" con lo que habrá dos puntos seleccionados en los cromosomas padre para el intercambio de información

- En general "n-point" o "uniforme" en los que se seleccionan n puntos de intercambio de información.

Por otro lado los *operadores de mutación* suelen consistir en cambios en un cromosoma en algún gen aleatorio como por ejemplo la variación de un bit de un string o bien la suma de un valor a un número real.

También se debe definir la *función fitness* que va a medir la calidad de las soluciones en cada momento del algoritmo. Igualmente se determinarán las condiciones de finalización, basadas en dicha función fitness.

2.2.4 Redes Neuronales Artificiales (ANN)

Las Redes Neuronales Artificiales, como se verá posteriormente, son las que van a utilizarse para la tarea de descubrimiento de conocimiento en este trabajo de investigación. Para estudiar las ANN se van a analizar:

- La *perspectiva histórica* de las mismas, es decir, de dónde surge la idea, cuáles son las primeras aportaciones y cuáles son los *enfoques* que se han dado a las ANN.
- *Los aspectos de diseño*, o qué factores se deben tener en cuenta para diseñar una ANN.

2.2.4.1 Perspectiva histórica.

Las ANN se puede decir que nacen del intento de simular el funcionamiento neuronal del cuerpo humano. En la actualidad cuando se habla de ANN se está hablando de *modelos computacionales compuestos de unidades de proceso paralelo ("neuronas") densamente interconectadas*. Esta estructura conserva algunas de las propiedades del aspecto biológico

como la capacidad de aprendizaje, la interconexión masiva, el procesamiento paralelo o la evolución. No obstante el funcionamiento cerebral es mucho más complejo.

En 1943, **McCulloch y Pitts** [McC-43] realizan un estudio en el que fijan un *modelo de neurona* con una serie de características:

- Elementos de detección binaria con valores 0 o 1
- Los enlaces entre las conexiones están ponderados por pesos.
- Se fijan pesos excitadores, con valor positivo, e inhibidores con valor negativo, que impiden el disparo de la neurona sobre la que se conecta, produciéndose dicho disparo únicamente en caso de superar un umbral fijo de dicha neurona. En la salida, se define una función que normaliza el valor de entrada (si es mayor que el umbral da 1 y en caso contrario 0). Esta se conoce como **función de activación.**
- Además se tiene que dar la concurrencia temporal entre las conexiones, de modo que para saber lo que sucede en el instante "t" debo primero conocer lo que sucedió en "t-1"

En la figura 3 se representa un ejemplo de la función OR según este modelo. Si cualquiera de los dos valores es 1, se supera el umbral de salida, y la función de activación lo normaliza a 1. La única forma de no superarlo es poniendo ambos valores a 0 en la entrada.

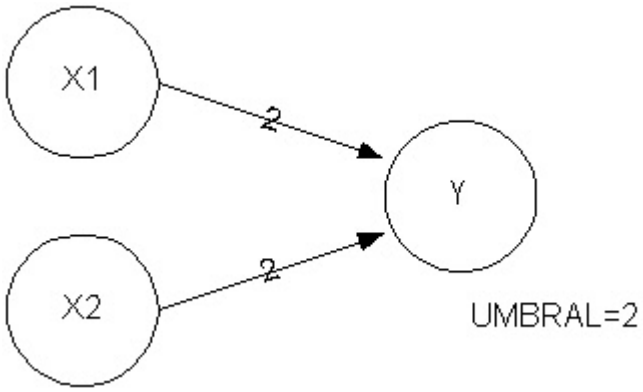


Ilustración 6 – Función OR según el modelo McCulloch-Pitts

Este modelo fue utilizado para simular funciones lógicas así como para algunas aplicaciones biológicas de simulación.

Posteriormente se utilizan estas estructuras para la clasificación de patrones. Este problema es conocido también como el de la *separabilidad lineal* de patrones. Se podría enunciar el problema de la siguiente forma: "Dado un conjunto de patrones de entrada ya clasificados, quiero clasificar los que me vengan después en diferentes grupos basados en estos patrones". Para ello se dispone de una arquitectura formada por varias capas de neuronas: La de entrada, la de salida y capas ocultas entre ellas. El objetivo es separar los patrones mediante una línea (separabilidad lineal).

Para la solución de este problema se usa la **regla de Hebb** [Heb-49] que consiste en actualizar los pesos como sigue:

$$w(t+1) = w(t) + xy$$

El peso en el instante "t+1" es el mismo que en "t" más el valor de la salida "y" por el valor en la entrada "x". Esta regla la usaría el **PERCEPTRÓN** [Ros-58] (sistema usado para simular la retina consistente en un vector de unidades de entrada "x" y una unidad de salida "y") en un *factor de aprendizaje "α"* entre 0 y 1, que regula lo que puede aprender de un paso a otro una neurona.

$$w(t+1) = w(t) + \alpha xy$$

En 1960, Widrow y Hoff [Wid-60] dan un paso más con su **ADALINE** (Neurona lineal adaptativa) de la que se derivan arquitecturas más complejas. La arquitectura es igual que la del PERCEPTRON. La innovación viene cuando enlazan n elementos ADALINE para obtener MADALINE (Multiple ADALINE).

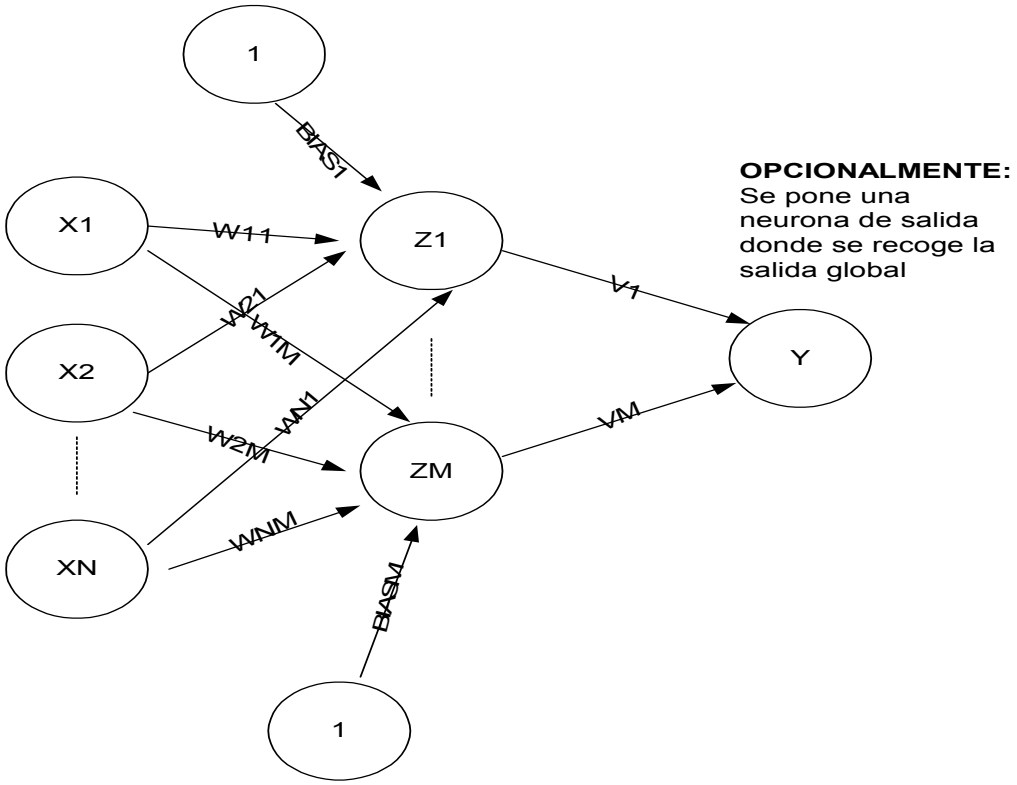


Ilustración 7 – Arquitectura MADALINE

La regla de entrenamiento es la que cambia respecto a la del PERCEPTRON:

$$w(t+1) = w(t) + \alpha x(y_{entrada} - y_{objetivo})$$

En este caso se calcula la distancia al valor de salida objetivo, para evaluar el aprendizaje y se propagan los errores hacia atrás en función de la comparación entre el valor objetivo y el resultante de la red. Este es un claro precursor del algoritmo más comúnmente usado que es BACKPROPAGATION, cuya esencia reside en la propagación de los errores hacia atrás.

Es en 1969, cuando la irrupción de un libro de Minsky y Papert [Min-69] sumiría la investigación en Redes Neuronales en el desuso durante un período de casi veinte años, ya que demostraron muchas de las limitaciones del perceptrón. Entre estas limitaciones se encontraba la de no ser capaz de aprender una operación lógica tan básica como el Or-

exclusivo o XOR, que es el operador de comparación. Así pues, un perceptrón no era capaz de indicar si sus entradas eran iguales o distintas.

Es en el año 1986, cuando Rumelhart y McLelland [McL-86] dan un empuje definitivo a un entonces apagado uso de las ANN. Es con el nacimiento de **BACKPROPAGATION,** posiblemente el algoritmo de aprendizaje más usado en las ANN. Se trata de un método que reduce el error cuadrático total de las unidades de salida en una red. La arquitectura es idéntica a la de MADALINE solo que con "p" unidades de salida. Se muestra un diagrama de una red Backpropagation, de tipo "Hacia delante" o "Feed-Forward". También la red mostrada en el diagrama se conoce como "Perceptrón Multicapa" que se utilizará en este trabajo de Investigación.

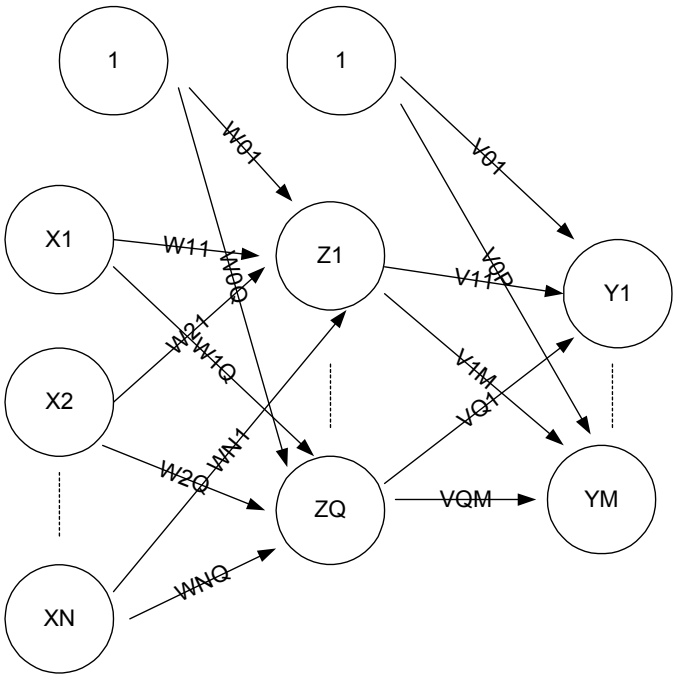


Ilustración 8 – Red Backpropagation

El algoritmo de Entrenamiento Backpropagation (o Retropropagación) tiene 3 fases:

Entrenamiento con los patrones de entrada:

Cada unidad Xi propaga hacia delante el valor de entrada a las unidades Zj. Estas a su vez propagan su valor a las unidades Ym de salida. En cada una de las neuronas se calcula la

función de activación que se explicó anteriormente. La más común es la sigmoidal, que para el caso de codificación binaria, 0 y 1, es:

$$f(x) = \frac{1}{1 + e^{-\sigma x}}$$

NOTA: σ determina la suavidad de la curva en los valores cercanos a 0

Esta función es parecida a la función "*escalón*", con la diferencia de que se trata de una función derivable en todos sus puntos y su derivada se puede calcular partiendo del valor de f(x) lo cual es muy importante a efectos de rendimiento a la hora de realizar cálculos en el algoritmo:

$$f'(x) = \sigma f(x)(1 - f(x))$$

Cuando se calculan los valores de activación, también se guardan los valores origen o valores de entrada, para posteriores cálculos.

Propagación hacia atrás de los errores

A partir del valor objetivo de la Unidad "k" de la Capa de Salida (T_k) se calcula la información del error, para posteriormente corregir los pesos:

$$\delta_k = (T_k - Y_k) f'(Y_{in\_k})$$

Donde Y_{in_k} es el valor de Y_k antes de aplicarle la función de activación.

Igualmente se calculará el valor de corrección para las unidades Z de la capa oculta, para ello se obtiene el valor de entrada del error a la unidad "j" (δ_{in_j}) a partir de la suma de los δ_k calculados anteriormente y los pesos W_{jk}. El valor δ_j se obtendrá de:

$$\delta_j = \delta_{in\_j} f'(Z_{in\_j})$$

Y servirá para el cálculo de un término de corrección de los pesos.

Ajuste de Pesos

Una vez se ha propagado el error y se tienen los términos de corrección de los pesos:

$$\Delta W_{jk} = \alpha \delta_k Z_j \qquad \Delta V_{ij} = \alpha \delta_j X_i$$

Nótese que "α" es el factor de aprendizaje.

Se pueden actualizar los pesos y bias, simplemente sumando estos términos:

$$W_{jk}(nuevo) = W_{jk}(antiguo) + \Delta W_{jk}$$

$$V_{ij}(nuevo) = V_{ij}(antiguo) + \Delta V_{ij}$$

Este procedimiento se repetirá hasta que se clasifiquen correctamente los patrones de entrada, o bien hasta que se cumpla la condición de terminación impuesta.

Una típica Red Backpropagation son los **PERCEPTRONES MULTICAPA** [Roh-95] que son de utilización generalizada y flexible. Pueden tener más de una capa oculta. Tendrán tantas variables de entrada como tenga el problema y una variable de salida por clase o

variable de clasificación del problema. En este trabajo de investigación se utilizarán estas estructuras para el aprendizaje

Esencialmente existen dos *enfoques* en la actualidad en las ANN. Estos son el de la *Neurocomputación Biológica* y el de las *Redes Neuronales Artificiales como Estructura de Aprendizaje.*

- La *Neurocomputación biológica* [Sig-93] consiste en la simulación del comportamiento del sistema nervioso. La definición de *Neurona* coincide con la común en biología: Estructura física cuyos límites vienen delimitados por una membrana que separa lo intracelular de lo demás, con una forma más "sofisticada" que otras células, al tener ramificaciones como las dendritas y el axon. Se trata de simular el estudio de la conducta, tanto en lo genético que son los reflejos o lo innato, como en el aprendizaje que es lo modificable. Se estudia en detalle el proceso de la *Sinapsis* que es el proceso de paso de información de una a otra neurona (mediante los potenciales de acción que cuando se disparan se produce la transmisión de una información de una a otra neurona), así como las propiedades químicas de la misma. Se utiliza también para modelar circuitos neuronales como los visuales, en los que se han hecho grandes avances en el campo de la visión artificial.

- Las *Redes Neuronales Artificiales como Estructura de Aprendizaje* no tienen ni mucho menos la complejidad en el análisis de las neuronas y sus conexiones que se hace en la neurocomputación biológica. Asimismo, no son un mecanismo fiable para simular la conducta porque su representación es mucho más sencilla. Algunos autores han afirmado que es exclusivamente por razones históricas el hecho de que se llamen "neuronales". Sin embargo sí que guardan una relación de alto nivel con el aspecto biológico. Por ejemplo, las conexiones entre ellas se pueden inhibir o potenciar dependiendo de la evolución del peso, evolucionan con el tiempo para aprender un comportamiento o tienen un estado de activación determinado a partir del cual "disparan" la información, lo que se identifica, en sentido amplio, con un potencial de acción. Esta es la estructura que se usa comúnmente cuando se habla de aprendizaje automático y la que se va a aplicar en este estudio.

2.2.4.2 Aspectos de Diseño

En una red de tipo Perceptrón Multicapa, hay varios *aspectos que se deben elegir a priori*. Se comentan a continuación:

- Elección de los pesos iniciales: Bien aleatoriamente o bien con otro procedimiento.
- Condición de finalización ya mencionada.
- División de los patrones de entrada: Se suele dividir el conjunto de patrones de entrada en los patrones de entrenamiento (para entrenar la red) y los patrones de validación (para validar los resultados obtenidos).
- Representación de los datos. Ya que determinará la arquitectura de la red.
- Número de capas ocultas: El algoritmo que se ha presentado contempla una sola, pero es fácilmente generalizable a varias.
- Uso de momento: Se mejora la eficiencia de Backpropagation (BP) utilizando los valores calculados por BP en el paso anterior para variar los pesos.
- Cambio de regla de aprendizaje: Se ha presentado la regla de descenso de gradiente o Delta, llamada así porque BP tiende a buscar la pendiente máxima de descenso del error para modificar sus pesos. No en todos los problemas es la mejor opción.

Si nos fijamos en Redes Neuronales que no necesariamente son de tipo Perceptrón Multicapa deberemos considerar los siguientes aspectos:

- *Evolución de los pesos*

Lo que se hace en la ejecución de un algoritmo de aprendizaje basado en redes neuronales es calcular los pesos de toda la red para obtener una correcta clasificación de una serie de patrones de entrada. Uno de los algoritmos más comunes es Backpropagation (BP) que se ha descrito anteriormente.

Por tanto, un primer aspecto de diseño en una red neuronal es el método elegido para evolucionar los pesos. Este aspecto está directamente relacionado con la representación de los pesos que puede variar entre diferentes técnicas.

- *Elección de las reglas de aprendizaje*

Una vez ejecutado Backpropagation (BP), el paso decisivo es el de la actualización de los pesos. Aquí se deben recordar otros factores como el factor de aprendizaje o el uso de momentos, pero sobre todo lo que más influye en este paso es la regla de aprendizaje. Se ha visto en la versión comentada anteriormente en la sección la regla de descenso por gradiente o Delta, que ha aportado muy buenos resultados en multitud de problemas.

Cada problema tiene una regla óptima de aprendizaje, que será un aspecto de diseño a tener en cuenta en el manejo de ANN.

- *Diseño de las arquitecturas*

Por último, se está hablando de BP y de reglas de aprendizaje sobre arquitecturas fijas. Resulta evidente que hay que fijar una arquitectura previamente a la ejecución de BP, y que esta elección influirá decisivamente en el buen o mal resultado de la red implementada.

Se utiliza mucho el conocimiento de expertos en la materia, así como mecanismos de prueba y error, para fijar la arquitectura de la ANN. Existen también mecanismos constructivos y evolutivos para automatizar esta tarea.

2.2.4.3 Combinaciones de Redes Neuronales y Algoritmos Genéticos

Se han comentado los aspectos de diseño de una red neuronal. Su elección viene muchas veces determinada por el conocimiento de expertos y por prueba y error. Si bien para problemas dinámicos en el tiempo, como el de este trabajo de investigación, o bien para buscar soluciones alternativas en casos de bloqueo por mecanismos tradicionales se puede tratar de automatizar dicho diseño. Se muestra en esta sección una revisión de los principales mecanismos para esta combinación.

La *combinación entre redes neuronales y algoritmos genéticos* es un campo de investigación de relevancia en la automatización del diseño de redes y en la búsqueda de mejores algoritmos de entrenamiento [del-02]. Se han hecho múltiples investigaciones y aplicaciones en esta disciplina [Yao-99] si bien resumimos a continuación las principales líneas de dichas investigaciones:

- *Evolución de los Pesos*: Combinar las redes neuronales con los algoritmos genéticos para actualizar los pesos de una red consiste en representar como posibles cromosomas una sucesión de pesos de la red y posteriormente combinarlos a través de un algoritmo genético, para tratar de obtener la mejor solución al problema. Sería dejar de utilizar completamente el algoritmo de entrenamiento tradicional para basarse en algoritmos genéticos. La mayoría de los experimentos realizados en esta línea dan peores resultados que los algoritmos tradicionales [Mon-89] tanto en rendimiento como en capacidad de solución, si bien para algún problema concreto pueden ser mejor solución.

- *Evolución de las Reglas de Aprendizaje*: La obtención de nuevas reglas de aprendizaje utilizando algoritmos genéticos consiste en codificar cada regla de aprendizaje como un individuo de una población o cromosoma y probar cada individuo contra un conjunto de redes neuronales para las que se pretende que sea óptima. Posteriormente a través de los algoritmos genéticos se irán combinando hasta obtener una regla óptima. Esto permitirá descubrir automáticamente reglas que maximicen el rendimiento en diferentes tipos de redes [Rad-98].

- *Evolución de las Arquitecturas*: El campo de más éxito en la combinación entre redes neuronales y algoritmos genéticos es el descubrimiento de arquitecturas óptimas de red para diferentes problemas. Preguntas tales como el número de capas ocultas o el número de unidades en cada una de ellas son determinantes en el rendimiento y la capacidad de generalización de la red. El uso de algoritmos genéticos consiste en la codificación como cromosomas de la arquitectura de la red. Posteriormente se combinarán dichas arquitecturas entre sí hasta obtener una óptima para la solución del problema. La *perspectiva constructiva* consiste en progresivamente ir añadiendo unidades y capas a la red hasta tener la arquitectura óptima [Fah-90]. La *perspectiva evolutiva* consiste en codificar, bien toda la estructura como un bloque [Puj-99] o bien

partes de ella [Kit-94] para obtener la red que mejor se adapte al problema. Nótese que la evolución de las arquitecturas es la línea de investigación que se acoplaría mejor al problema planteado en este trabajo de investigación.

2.2.4.4 Librerías de Redes Neuronales

Para implementar las redes neuronales en un sistema, se pueden escoger varias maneras. Una sería implementar los algoritmos y mecanismos de uso de redes neuronales desde el principio en la estructura de la aplicación.

Siendo esta la forma más limpia, realmente no es la más óptima ya que existen multitud de implementaciones disponibles tanto comerciales como de libre distribución. Otra posibilidad es la de coger un código de uso de las redes neuronales y acoplarlo en el sistema. Así se aprovechan trabajos ya realizados que son potencialmente grandes consumidores de tiempo, como por ejemplo todo lo que se refiere al tratamiento de ficheros.

Finalmente, se pueden integrar en el código librerías o APIs de acceso directo a funciones de manejo de redes neuronales con diferentes parámetros. Esta opción únicamente implica incluir las librerías en tiempo de compilación y conocer la parametrización de las funciones de la librería para su encaje en el programa.

Esta última ha sido la opción elegida en este trabajo. Se mostrará a continuación el detalle de la librería escogida.

2.2.4.4.1 Librerías TORCH

Las librerías TORCH [Col-02] surgen de la necesidad de aunar en un Framework *una serie de algoritmos de Aprendizaje Automático* para hacerlos fáciles de utilizar y de integrar en diferentes aplicaciones. Están bajo una licencia BSD (Berkeley Software Distribution), lo que quiere decir que el uso del código fuente es libre en las diferentes aplicaciones e investigaciones que se requiera.

Esta librería está implementada con C++ y puede ser modificada o adaptada para las investigaciones que se requieran. Hay cuatro clases principales:

- *Dataset*: Es la clase que gestiona el manejo de datos desde disco o su gestión en la memoria.

- *Machine*: Se trata de una "caja negra" que dada una entrada realiza operaciones sobre ella y devuelve una salida. Puede ser una red neuronal, como en este trabajo, así como otras técnicas de aprendizaje automático.

- *Trainer*: Se encarga de buscar los parámetros adecuados para la entidad "Machine" dados unos criterios, como por ejemplo elegir que sea una red neuronal con descenso de gradiente, y unos Dataset de entrenamiento y de prueba.

- *Measurer*: Obtiene diferentes medidas de interés de las operaciones realizadas como las desviaciones típicas, los tiempos medios o los errores de clasificación obtenidos.

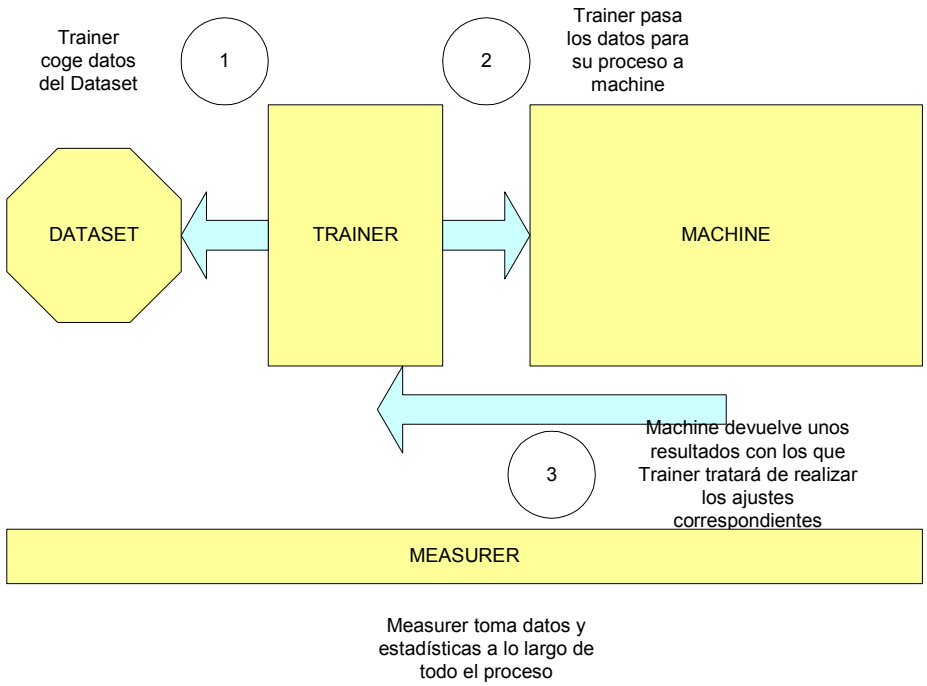


Ilustración 9 – Clases principales y funcionamiento Básico Librerías Torch

En el diagrama se puede ver el encaje de las diferentes clases de Torch. Hay diferentes tipos de Machine implementados. Estos son:

- *Gradient Machines*: Son entrenadas con algoritmos de descenso de gradiente. Aquí están incluidos los Perceptrones Multicapa y en general las redes tipo Backpropagation. En términos de código, la clase asociada es "GradientMachine" y el Trainer asociado es un "StochasticTrainer". Nótese que en este trabajo de investigación se utilizará este tipo de Machine en la librería Torch.

- *Support Vector Machines*: Para problemas de clasificación o de regresión. Las clases asociadas son "SVMClassification" o "SVMRegression" y se entrenan a través del Trainer "QCTrainer".

- *Distribution*: Como distribuciones Gaussianas o modelos ocultos de Markov, se entrenan a través de un maximizado de expectativas ("EMTrainer").

Se han realizado comparativas de TORCH con otras herramientas de Redes Neuronales utilizadas para problemas similares siendo los resultados obtenidos muy similares en cuanto a tiempos de ejecución, uso de recursos o rendimiento. Este factor unido a su facilidad de encaje en aplicaciones ha determinado su elección para este trabajo.

2.2.4.4.2 Personalizaciones de TORCH

Como se ha visto anteriormente, se puede adaptar TORCH [Col-02] para el problema que se pretenda. En un caso como el que nos ocupa se utilizará una ANN con algoritmos de descenso por gradiente (Tipo Backpropagation o Perceptrón Multicapa).

Las personalizaciones que se deben hacer son las siguientes.

- *Compilación*: En tiempo de compilación es necesario explicitar los paquetes de toda la librería Torch que va a ejecutar la aplicación en cuestión. En nuestro caso utilizará para un descenso de gradiente los paquetes "distributions" y "gradients".

- *Main y Librería de Uso*: Para un determinado problema hay que generar un Main y opcionalmente una librería de funciones para el uso de la API de Torch de forma

personalizada. Existen una serie de ejemplos de ayuda para su creación dentro de la distribución de Torch.

- *Parámetros*: Hay que fijar los datos del mecanismo elegido. Entre ellos, para la arquitectura de la red se utilizan los siguientes:
    - "n_inputs" entradas
    - "n_outputs" posibles valores de salida
    - "n_hu" unidades de capa oculta
- Salidas: Asimismo hay que determinar cuáles de las salidas del Measurer van a ser de utilidad. En nuestro problema utilizaremos las matrices de confusión.

Dichas *matrices de confusión* indican en las filas el valor obtenido y en las columnas el valor que se esperaba. Por ejemplo:

	COLUMNA 0	COLUMNA 1	COLUMNA 2
FILA 0	0	0	0
FILA 1	0	0	0
FILA 2	1	0	0

Es una matriz de confusión en la que se ha obtenido un valor 2 determinado por la fila cuando se preveía haber obtenido un valor 0 dado por la columna.

Por tanto, una red neuronal bien entrenada sería aquella que tras el entrenamiento devuelve para la clasificación una matriz de confusión cuyos valores están concentrados mayoritariamente en la diagonal. Para conseguir este efecto podrían ser necesarios varios entrenamientos de la red. Los principales factores a definir son:

- Tener un buen *fichero de entrenamiento*, con ejemplos suficientemente representativosl problema.
- Determinar una *topología adecuada al problema*, variando los factores que pueden determinar un mejor ajuste.

2.3 Conclusiones del estado de la cuestión

En esta sección se han repasado los aspectos y el estado de la cuestión en el campo de la Gestión de la Configuración y de la Minería de Datos, que van a ser las materias objeto de este trabajo.

En lo que se refiere a *Gestión de la Configuración del Software* se han repasado los principales conceptos y definiciones. Asimismo se han visto diferentes herramientas y sistemas junto con sus características principales. Se ha analizado en detalle la herramienta CVS, que es la que se utilizará en este trabajo. Se trata de un campo en expansión, al ser una disciplina de éxito dentro de la Ingeniería del Software.

Posteriormente se ha repasado el estado de la cuestión en el campo de la *Minería de Datos*. Se han revisado diferentes técnicas de aprendizaje automático, para realizar el descubrimiento de conocimiento en bases de datos. Finalmente se han analizado las *Redes Neuronales Artificiales* que serán utilizadas en esta investigación. Se ha analizado también la librería Torch de funciones de aprendizaje automático que se empleará posteriormente.

2.3.1 Espacio de Investigación

Las disciplinas que se han visto por separado y que tienen múltiples sistemas y aplicaciones asociadas, son aparentemente disjuntas en lo que a investigaciones se refiere. La GCS busca tener controlada la información del presente y del pasado de los elementos de configuración, mientras que el Aprendizaje Automático busca detectar tendencias y patrones en el conocimiento pasado para aplicarlas al futuro.

Con este escenario, sería de gran *utilidad* poder predecir comportamientos en la GCS para ser capaces de identificar tendencias o patrones que guíen los comportamientos de los usuarios que gestionan los diferentes elementos de configuración. Para esta tarea, además de ser necesario un modelo de GCS flexible y general, las estructuras asociadas con el Aprendizaje Automático pueden completar dicho descubrimiento de conocimiento futuro.

Por tanto, el cruce entre la GCS y el Aprendizaje Automático es el espacio que se va a investigar con este trabajo para tratar de elaborar un modelo de GCS y de mecanismos de

descubrimiento de conocimiento como las Redes Neuronales, con el fin de aportar a una organización conocimiento de utilidad para su futuro.

Las investigaciones realizadas en estos campos *pueden tener gran utilidad*, ya que los procesos de GCS ayudan por si mismos a las compañías. Si sobre estos mismos procesos se añade un nuevo valor de descubrimiento de conocimiento, serán todavía de más utilidad para las organizaciones. *No se trata por tanto de una innovación en el campo de las Redes Neuronales y el Aprendizaje Automático, si no de la aplicación de estas estructuras para añadir valor a procesos*.

2.3.2 Investigaciones Relacionadas

En el campo de la Minería de Datos, se han hecho algunas investigaciones relacionadas de algún modo con este espacio de investigación, que tiene por objetivo la detección de conocimiento de utilidad a partir de la información de los procesos de GCS. Son, no obstante, pocas y su relación con este trabajo no es muy fuerte. Se mencionan a continuación.

El trabajo realizado por Graves et al. [Gra-99], muestra intentos de *predecir comportamientos a través de la medición estadística de los errores producidos en un sistema.* Es decir, busca la información histórica de los errores que se producen en los sistemas estudiando las posibles decadencias en el código. La idea es que cuanto mayor es el uso y la corrección de errores de un determinado código, más se pueden corromper sus estructuras hasta llegar a un punto en el que se vuelva problemático su mantenimiento. Si se pueden anticipar estas tendencias a la corrupción se podrá tratar de poner solución antes a problemas futuros lo que facilita el mantenimiento.

Sayjad et al. [Say-01] han realizado experimentos para intentar detectar *relaciones entre los elementos de los repositorios de código y los registros del mantenimiento de software*. El objetivo es detectar las relaciones que pudiera haber entre objetos del sistema que no estuvieran explicitadas. La comprensión de dichas relaciones puede ser de utilidad en el *mantenimiento* de un sistema. Así por ejemplo, se podría detectar que cierto componente “A” cada vez que se corrige, genera al “X” tiempo un fallo en un componente “B” cuya relación no es trivial.

Hassan et al. [Has-05] investigan en la línea de la *detección de errores* a partir del estudio de los cambios en el código fuente. Proponen un marco de trabajo para modelizar los cambios y su propagación por los sistemas. Buscan como objetivo saber en qué medida los cambios en una entidad de código fuente se propagan a otras. Estas relaciones se visualizan para facilitar su comprensión y se dan una serie de recomendaciones extraídas de la literatura según el patrón que sigan los cambios producidos.

Una vez repasado el estado de la cuestión y sus conclusiones, se formula a continuación la Hipótesis de este trabajo.

3 Hipótesis

La hipótesis que se pretende validar en este trabajo se planteará en el marco de la Gestión de la Configuración del Software y del Descubrimiento de Conocimiento en Bases de Datos. A continuación se habla de *las premisas necesarias* para la formulación de la Hipótesis de trabajo. Son las siguientes:

- *Se trabaja en el marco de los sistemas de información de una compañía*. Se entiende por sistemas los sistemas informáticos que dan soporte a diferentes procesos. Se entiende por compañía un grupo de personas que trabajan para una organización común.

- Los *usuarios* de la organización asumen diferentes roles y se encuadran en diferentes etapas de los *ciclos de vida del software* que están bien definidos y delimitados para cada uno de los sistemas software.

- Cada sistema software será identificado como un Producto. Toda la *información histórica de cambios* en un producto, tanto de código como de controles de requerimientos y errores está disponible.

Sobre el marco de trabajo descrito existen una serie de procesos de Gestión de la Configuración del Software utilizados para el control de la evolución de dicho software. Es decir, para tener en cada momento las fotos presentes y pasadas de los diferentes elementos sobre los que se controla la configuración.

La **Hipótesis Principal** se enunciará como sigue:

H: *La Gestión de la Configuración del Software, aplicando Técnicas de Aprendizaje Automático, puede ayudar, no sólo a controlar el presente y el pasado de la evolución de los sistemas software, sino también a detectar tendencias y patrones que impliquen conocimiento de utilidad para ayudar a una organización en la toma de decisiones futuras*.

Se pretende por tanto modelar y descubrir conocimiento en la Gestión de la Configuración que permita extraer información de utilidad para mejorar diferentes aspectos de la organización.

Se pretende validar la Hipótesis principal atendiendo a las siguientes subhipótesis o **Hipótesis Secundarias**:

H_1: *La Gestión de la Configuración del Software, aplicando Técnicas de Aprendizaje Automático, puede ayudar en los procesos de evaluación de los diferentes usuarios de los diferentes productos. Al mantener la información histórica de todas las acciones que han realizado los diferentes usuarios de los diferentes sistemas de la compañía y la información de las evaluaciones que han obtenido, se puede tratar de inferir cuál será la evaluación en un periodo*.

Es decir, se pretende establecer un mecanismo que sugiera las evaluaciones del desempeño de los diferentes empleados de una organización en un periodo de tiempo a partir de la información histórica mantenida en los sistemas de la compañía. Puede ser una ayuda para la identificación de acciones concretas sobre diferentes áreas de la compañía y sobre diferentes usuarios de la misma.

H_2: *La Gestión de la Configuración del Software, aplicando Técnicas de Aprendizaje Automático, puede ayudar en la detección de acciones correctoras a nivel general, con el fin de trazar acciones de mejora general de los sistemas a partir de la detección de tendencias y patrones en las causas que provocan los errores de una compañía*.

Es decir, se pretende establecer un mecanismo para poder detectar cuales son las causas principales de los diferentes errores dentro del marco de productos, roles y responsabilidades de una compañía con el fin de identificar acciones de formación o de mejora de la calidad más enfocadas y concretas en las diferentes áreas de los sistemas de información

4 Planteamiento del Problema

En esta sección se muestra el planteamiento del problema, una vez formulada la Hipótesis base de trabajo. También se apuntarán las líneas en las que se basará la solución.

4.1 El Problema de la Evolución en la Gestión de la Configuración

En el proceso de Gestión de la Configuración del Software intervienen múltiples agentes y abarca una serie de tareas de control sobre diferentes elementos. El fin que tiene es garantizar la coherencia de dichos elementos y poder rescatar las situaciones del pasado en un momento del tiempo.

Existen, como se ha analizado, una serie de técnicas, algoritmos y paradigmas que se utilizan para el descubrimiento de conocimiento no explícito en conjuntos de datos. Este conocimiento se obtiene en la detección de diferentes tendencias o patrones en dichos datos que pueden ayudar a predecir comportamientos.

El problema que se plantea en este trabajo es si la aplicación de las técnicas de descubrimiento de conocimiento a los procesos de Gestión de la Configuración del Software puede ayudar a predecir determinados comportamientos o tendencias en los diferentes elementos que componen dichos procesos.

Se entiende como una ventaja el hecho de que se pueda descubrir conocimiento de utilidad para una organización derivada de un proceso que de por sí está implantado en ella, como es la GCS. Se deberán precisar los siguientes aspectos para acotar el problema:

- *Descripción de la Organización sobre la que se aplica el proceso*. Se deben describir los roles y responsabilidades de los diferentes actores de la compañía.

- *Descripción de los procesos de la compañía*. Se deben determinar los diferentes procesos de GCS sobre los que se va a tratar de inferir el conocimiento, es decir, procesos que pueden aportar datos de interés en el modelo.

- *Determinar los diferentes elementos de la arquitectura funcional*. Identificar cuáles serían los módulos funcionales de la aplicación, identificando los diferentes objetivos de la aplicación en lo que se refiere a operaciones a realizar, informes a generar y procesos de GCS que se implementarán.

- *Identificar el conocimiento a descubrir, los patrones que se quieren detectar, sus fuentes y los mecanismos de descubrimiento de conocimiento* que se van a aplicar. Diseñar los procesos de Descubrimiento de Conocimiento de principio a fin.

Este trabajo propone una solución para la detección de la Evolución en la Gestión de la Configuración del Software. No se pretende mejorar las herramientas de GCS que existen actualmente. Tampoco se pretende ahondar en la identificación de nuevos mecanismos de Descubrimiento de Conocimiento. El objetivo es obtener frutos de la combinación de ambas técnicas con el fin de añadir información de utilidad a la compañía a partir de procesos ya implantados en la misma.

4.2 Solución basada en el Descubrimiento de Conocimiento.

La solución al problema que se plantea implica la generación de una herramienta en la que confluyan todas las técnicas que se han mencionado anteriormente, y que permita establecer un modelo predictivo sobre los procesos de GCS.

Se detallan a continuación las líneas básicas de la implementación de dicha herramienta. Nótese que se buscará en todo momento una solución que tenga como máximas la *utilidad y la generalización*, de manera que pueda obtener conocimiento útil y sea adaptable a escenarios de organizaciones dinámicas.

En primer lugar la herramienta debe contener el *modelo de la compañía*. Para ello se debe modelizar de forma flexible el esquema de la misma. En este modelo deben intervenir:

- Usuarios y responsabilidades*:* Determinar cuáles son los posibles usuarios de la compañía y cuáles son las tareas que desempeñan en el proceso de GCS.

- Elementos de control de los procesos de GCS: Es necesario determinar cuáles serán los diferentes elementos de configuración que controlará la herramienta. Asimismo se deben determinan sus jerarquías y relaciones, así como sus ciclos de vida. En este punto figurarán:
    - *Proceso de Gestión de Versiones*
    - *Proceso de Gestión de Requerimientos de usuario*.
    - *Proceso de Gestión de Errores*.
    - *Procesos de Gestión de Despliegue de Software en los diferentes entornos*.

Por otro lado, se deben definir todas las actividades que los usuarios van a realizar desde la herramienta y cómo las realizarán. Se trata de una *descripción de procesos y de arquitectura funcional derivada de esos procesos*. Las actividades a realizar desde la herramienta deberán cubrir los procesos anteriormente mencionados y estar integradas en una misma plataforma.

De la arquitectura funcional se derivarán diferentes componentes, cuyas integraciones habrá que determinar y cuya *arquitectura tecnológica* habrá que diseñar. Entre estos componentes figurarán las pantallas de usuario, los repositorios de componentes, las bases de datos o las propias máquinas sobre las que se controla la configuración.

Asimismo se debe incorporar la capa de *Descubrimiento de Conocimiento* al sistema y establecer los objetivos que se pretenden con dicho nivel. Esta capa estará integrada con el resto de componentes del sistema. Habrá que definir los mecanismos de aprendizaje automático a utilizar así como las entradas de las que se van a alimentar dichos mecanismos y sus posibles salidas.

En este módulo es en el que se deberá dar cobertura a las Hipótesis de este trabajo validando la idea de que se puede establecer un modelo de predicción sirviéndose de la información de los procesos de GCS.

Finalmente, se debe completar un *modelo de compañía completo* con los datos de GCS históricos con el fin de validar este sistema. Los mecanismos de Descubrimiento de

Conocimiento normalmente funcionan bien con conjuntos de datos grandes que son los que permiten detectar tendencias y patrones más precisos.

En la siguiente sección se detalla la solución propuesta a este problema consistente en la elaboración de un modelo predictivo en los procesos de GCS, con el fin de validar las Hipótesis planteadas en este trabajo.

5 Solución Propuesta

En este bloque se van a tratar los diferentes aspectos asociados a la solución propuesta. Se van a describir desde el modelo de procesos de sistemas de una compañía, hasta las técnicas de Inteligencia Artificial que se han empleado. Se explicará asimismo de forma detallada la arquitectura y el análisis de la solución.

También se revisarán ejemplos de los diferentes aspectos comentados, sin embargo hay que mencionar que la sección donde se van a describir los mecanismos de validación y resultados obtenidos, es la del "Análisis de Viabilidad".

El esquema seguido en esta sección es:

- Describir los procesos de control que ocurren en las organizaciones.
- Determinar la Arquitectura Funcional derivada de dichos procesos.
- Identificar los componentes de la Arquitectura Tecnológica.
- Analizar y Diseñar la solución para cubrir esos procesos.
- Analizar y Diseñar la funcionalidad de Descubrimiento de Conocimiento.

5.1 Visión Global de la Solución

Para facilitar la comprensión de esta sección y de la solución al problema planteado se va a exponer un esquema de la misma a modo de visión global.

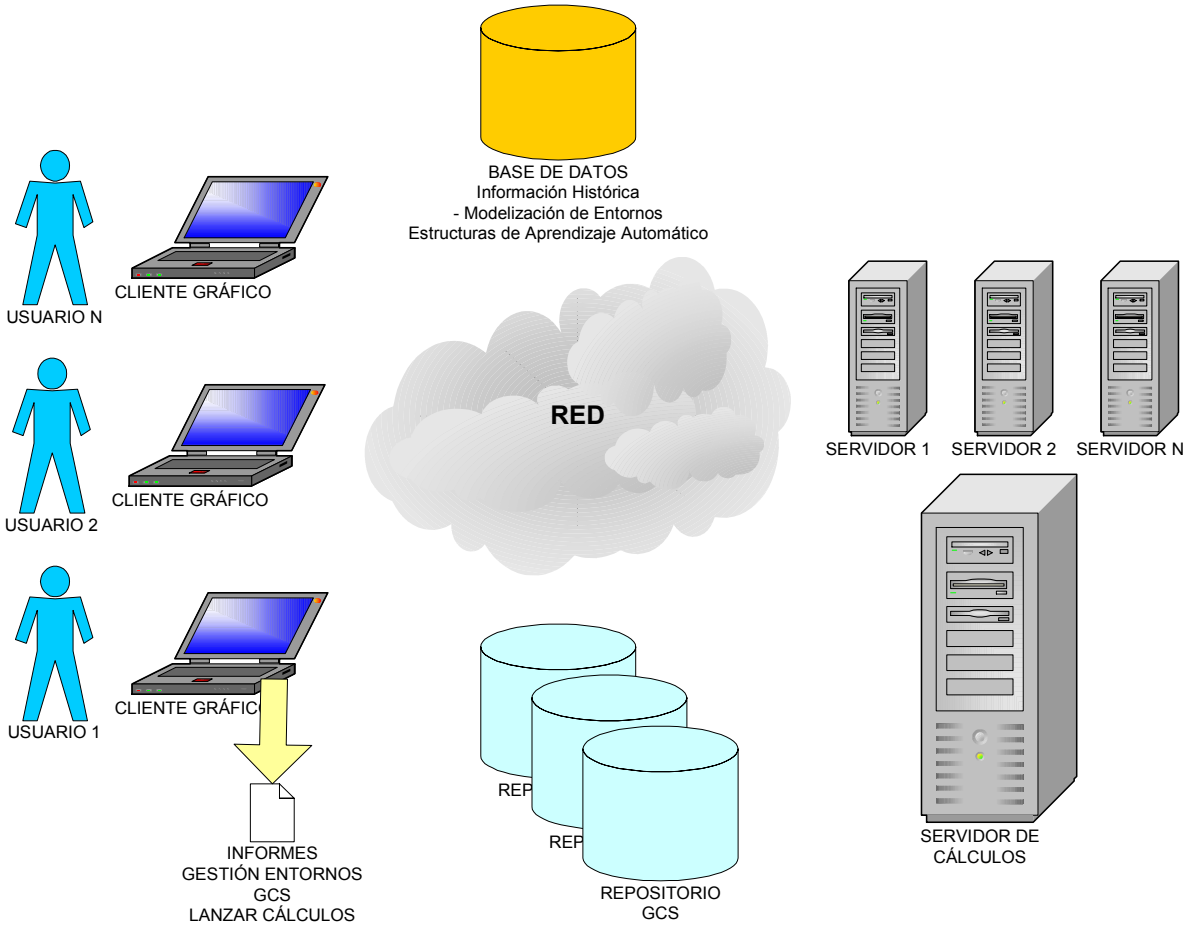


Ilustración 10 – Visión Global de la Solución

En el diagrama que se expone se pueden ver los diferentes actores y componentes que van a intervenir en la solución y que se describen a continuación.

- *Usuarios*: Habrá una serie de usuarios destinatarios del sistema. Básicamente serán los usuarios del área de sistemas de la organización que intervengan en los procesos de GCS. Otros usuarios potenciales del sistema son los destinatarios de las conclusiones e informes generados que deben contribuir a ayudar a la dirección en la toma de decisiones.

- *Clientes Gráficos*: En cada uno de los puestos de los usuarios habrá un cliente gráfico. Desde estos clientes se dirigirán las operaciones que haga cada usuario con el sistema final, de manera que visualice el progreso y resultado de sus diferentes tareas. Desde este

cliente se podrán realizar principalmente las siguientes acciones (todas ellas se explicarán en detalle posteriormente):

  - Generar *informes*, de forma que se centralice la gestión de la información.
  - *Gestionar los diferentes entornos* de trabajo de forma remota.
  - Centralizar los *procesos de Gestión de Configuración del Software* .
  - Lanzar los *procesos de realización de cálculos y operaciones de servidor*, sobre todo las relacionadas con estructuras de Aprendizaje Automático.

- *Base de Datos:* Habrá una base de datos donde se guarde toda la información asociada al sistema. En concreto se gestionará:
  - *Información Histórica* de los procesos de GCS.
  - Modelización de *sistemas y de arquitectura*
  - Información relacionada con las *estructuras de aprendizaje automático*.
- *Repositorio GCS:* Igualmente existirá una serie de repositorios en los que se guardarán las diferentes versiones de los objetos cuya configuración es susceptible de control.
- *Servidor de Cálculos*: Habrá un servidor que realice los Cálculos asociados con las estructuras de aprendizaje automático del sistema.
- *Servidores de la Organización*: En la organización existirán diferentes servidores donde residirán las diferentes aplicaciones.

Todos los componentes anteriormente mencionados estarán conectados a través de una Red de Datos y habrá que definir sus interfaces.

Se debe destacar que *la solución está orientada a buscar en todo momento la máxima Generalización*, ya que se persigue un modelo aplicable en diferentes organizaciones con diferentes procesos de GCS y diferentes arquitecturas de sistemas.

Por último, la Herramienta a Generar se llamará "Sistema de Gestión de la Configuración Evolutivo". Las siglas identificativas de la misma serán "ECFM" de "Evolutive Configuration Manager".

En las siguientes secciones se va a detallar cada una de las partes de la solución de forma separada.

5.2 Descripción General de Procesos de Movimiento de Software del Área de Sistemas de una Compañía

En primer lugar se debe establecer un *esquema general* que describa el área de sistemas de una compañía cualquiera. Este esquema debe ser flexible, ya que no todas las compañías tienen esta área organizada de igual manera. Se debe tratar de identificar aquellas funciones de alto nivel que se ejecutan en la mayoría de las compañías para establecer el modelo de procesos basado en estas funciones. Se describirán asimismo los roles implicados y las responsabilidades de los mismos. Finalmente se describirán las diferentes fuentes de errores que posteriormente se analizarán en este trabajo.

5.2.1 Esquema general

El Área de Sistemas de una Organización es la encargada de implantar y mantener los Sistemas de Información de la misma. Estos dan soporte a las diferentes áreas de la organización, que en adelante se denominarán Áreas Usuarias. Dichas áreas usuarias son por ejemplo los directivos, los agentes de centros de clientes, los trabajadores en marketing, los investigadores o los distribuidores del producto o aplicación. En el siguiente diagrama se muestra un esquema general que según el tipo o tamaño de compañía, puede agrupar determinadas funciones o bien dividirlas.

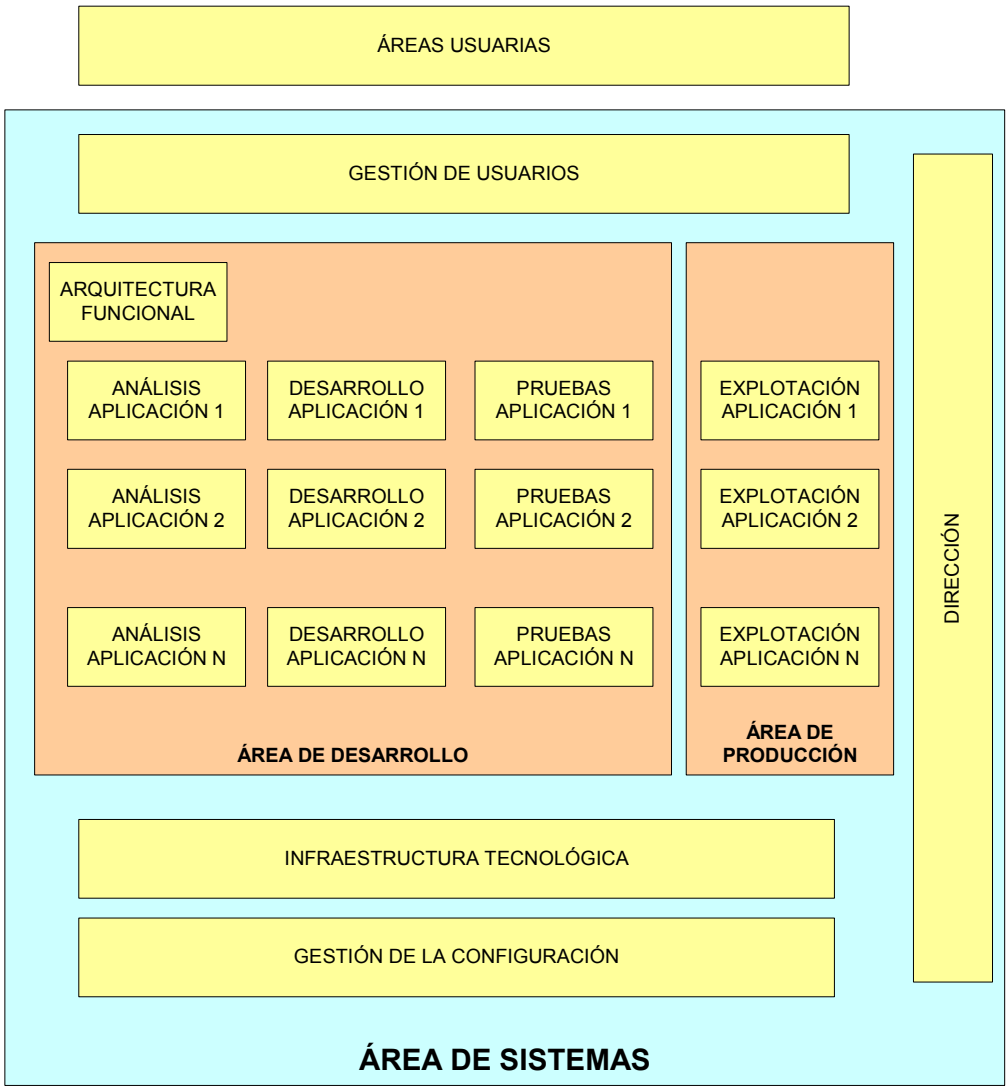


Ilustración 11 - Esquema del Área de Sistemas de una Organización

El Área de Sistemas de una Compañía se divide como podemos ver en diferentes partes o secciones según las tareas que desempeñan. Estas son:

- *Gestión de Usuarios*: Es el área encargada de llevar la interlocución con las áreas usuarias. Se encargarán de realizar la toma de requerimientos, de revisar los procesos de usuario o de validar la aceptación de los sistemas junto a los usuarios.

- *Arquitectura Funcional*: Se encargan de trasladar los requerimientos de los usuarios a sistemas de información. Tiene la visión de la organización de forma global, como la unión de muchos sistemas en un mapa.

- *Análisis*: Para cada uno de los sistemas se encargará de analizar el impacto de una determinada funcionalidad que viene determinada por arquitectura funcional. Con alto conocimiento de cada sistema, el área de análisis trasladará unos requerimientos concretos a los desarrolladores.

- *Desarrollo*: Es el equipo que hará los desarrollos software a partir de unos análisis determinados. Son especialistas en un determinado sistema de la compañía.

- *Pruebas*: Es el área encargada de validar que los desarrollos software funcionan correctamente. Esto se debe garantizar antes de realizar el paso a producción de dichos desarrollos. Estas pruebas pueden ser unitarias, de integración o de volumen.

- *Explotación*: Son los encargados de realizar la operación de los sistemas en producción. Los sistemas se hacen para que produzcan unos resultados que son extraídos y gestionados por el área de producción.

- *Gestión de la Configuración*: Es el área encargada de gestionar la correcta configuración de los diferentes sistemas. En la sección Gestión de la Configuración de esta memoria se detallan sus funciones.

- *Infraestructura Tecnológica*: Son los encargados de dotar a los sistemas de la compañía de un soporte tecnológico adecuado. Cubren los requerimientos tanto de Software base como los de Hardware.

- *Dirección*: La dirección del Área de Sistemas es la encargada de la supervisión de las diferentes áreas y del cumplimiento de sus respectivas tareas.

Una vez revisado el esquema general de el área de sistemas de una organización, se van a analizar con mayor detalle algunos de los roles que intervienen en la misma.

5.2.2 Roles y Responsabilidades

En el diagrama explicativo del Área de Sistemas de una Compañía que se ha propuesto en la sección anterior, se han identificado una serie de áreas con sus determinadas funciones. A partir de estas se pueden definir una serie de roles y responsabilidades de los que hablaremos en adelante en este trabajo.

- *Project Manager*: Pertenece a la dirección del área de sistemas. Se encarga de supervisar determinados trabajos a partir de una visión global de los mismos. Maneja los presupuestos asignados a proyectos y controla el progreso de las tareas.

- *Analista*: Tiene amplio conocimiento de un determinado sistema o bien de un conjunto de ellos. Tiene la capacidad de trasladar los requerimientos del negocio a la particularidad de los sistemas. Asimismo puede identificar determinados errores o anomalías y determinar el impacto que tienen sobre los sistemas.

- *Desarrollador*: Conoce el detalle de un determinado producto software, aplicación o lenguaje de programación. Implementa en código los análisis que se le han pasado. Es el encargado de corregir también errores determinados que no requieran de la intervención de un analista, como por ejemplo los errores de codificación incorrecta.

- *Pruebas*: Perfil encargado de probar la funcionalidad implementada por un desarrollador. Es el encargado de ejecutar planes de pruebas que pueden elaborar ellos mismos o bien otros miembros del equipo de trabajo. Puede haber diferentes especialistas en pruebas tales como el que prueba un sistema en solitario, la integración con otros sistemas o bien el comportamiento de un sistema con gran volumen de datos. Asimismo informarán a los desarrolladores de los errores que pudieran encontrar.

- *Operador*: Es el encargado de operar los procesos en explotación. Realizará las ejecuciones según los manuales de operaciones de los sistemas. Los posibles errores que detecte y que no pueda corregir él mismo, se los trasladará al equipo de Desarrollo.

- *Configuration Manager*: Es el encargado de llevar el control de la configuración de los sistemas. Lleva la traza de lo que está sucediendo en cada momento en cada entorno de cada sistema. Tiene el registro de los errores que suceden y del estado en el que se encuentran sus soluciones en el área de gestión de la configuración.

A partir de este esquema de organización y de roles, se van a detallar las diferentes fuentes de errores que serán susceptibles de análisis en el problema que se plantea en este trabajo.

5.2.3 Procesos de Instalación, Pruebas y Explotación y sus fuentes de errores

En esta sección se van a analizar los diferentes procesos de control de cambios en la Gestión de la Configuración que se pueden dar en el modelo de Área de Sistemas que se ha propuesto. Nótese que esta tipificación es muy general, ya que los errores o cambios pueden surgir en las circunstancias más diversas.

Es importante reseñar que los cambios en el software estarán determinados por dos razones:

- *Por nuevos requerimientos o mejoras.*
- *Por corrección de errores*.

Nótese que el objetivo que se pretende en este trabajo es buscar en el sistema de Gestión de la Configuración las tendencias o patrones que siguen los errores, para intentar que cada vez ocurran menos en el sistema. Por tanto se centrarán los procesos en el descubrimiento de errores.

A partir de esta definición de procesos, se podrán derivar y modelizar en fase de análisis las estructuras de control y ciclos de vida.

5.2.3.1 Proceso de Instalación

El primer proceso que se define es el de Instalación. Una vez el Desarrollador termina de implementar su Software, prepara la entrega a realizar a los diferentes entornos de pruebas para finalmente ser instalada en el entorno de producción.

El Desarrollador liberará un *Paquete de Instalación* consistente en:

- Los objetos de Software desarrollados con sus correspondientes versiones.
- El listado de Requerimientos y Errores que soluciona ese Software.
- La Documentación asociada al software. Esta principalmente será:
    - Documentación de Instalación del Software entregado.
    - Documentación de Explotación de dicho Software.

El Configuration Manager va a recibir el paquete de instalación del desarrollador y tendrá que instalarlo en el siguiente entorno de pruebas.

Si durante el proceso de Instalación se produce un error, se determinará que ha habido un *Error de Instalación* que se reportará al Desarrollador. El Desarrollador, basándose en el error que le sea reportado deberá aportar una solución que podrá llevar cambios en los objetos software entregados así como en la documentación de instalación asociada.

En el siguiente diagrama se puede ver el *proceso de un Error de Instalación*, en el que intervienen el Desarrollador y el Configuration Manager. Es importante señalar que el proceso de detección de un error se cerrará documentándo dicho error y comunicándolo al desarrollador.

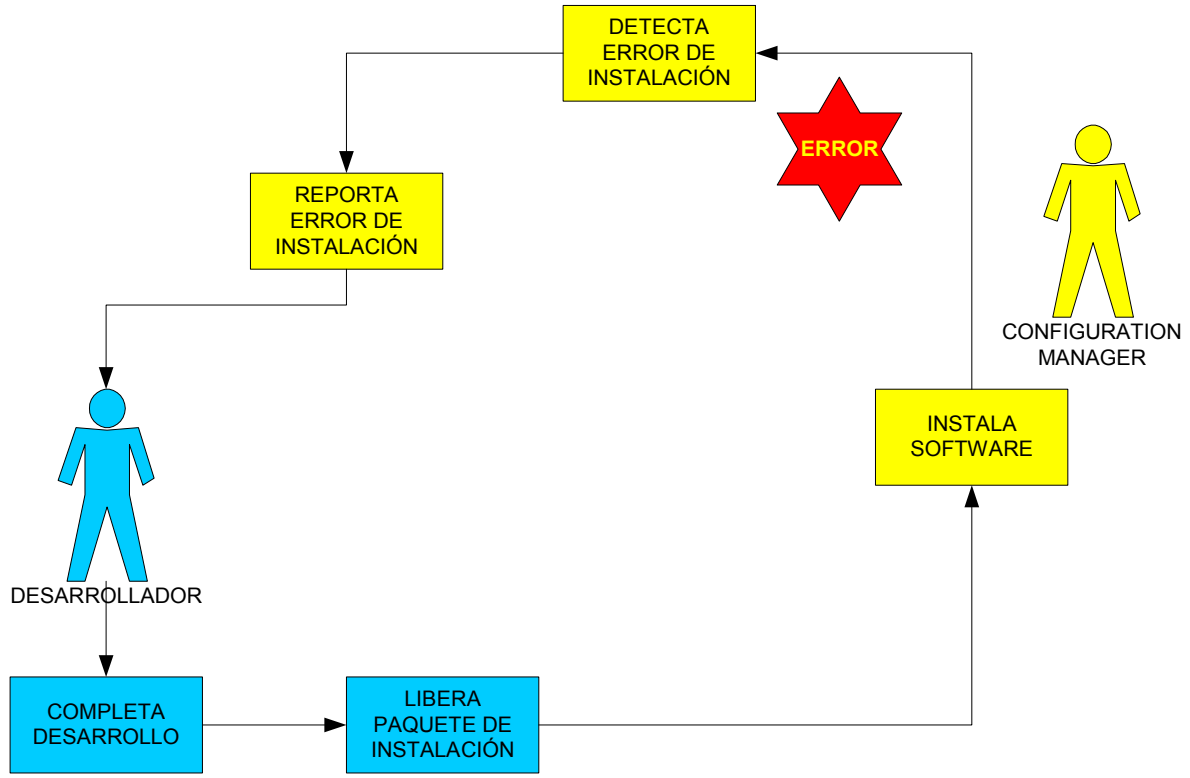


Ilustración 12 – Error en Proceso de Instalación

5.2.3.2 Proceso de Pruebas

En el caso de que la instalación del paquete de software enviado por el Desarrollador se haga satisfactoriamente, podrán comenzar las *pruebas* sobre la funcionalidad. El equipo de Pruebas necesita, además del Software, un documento que refleje el Plan de Pruebas a seguir. Este documento reflejará cada una de las pruebas a realizar, así como los resultados esperados de las mismas.

Si en alguno de los casos se produce un error, el equipo de Pruebas lo reportará al de Desarrollo, dándoles el detalle de lo que ha sucedido, comparando los resultados obtenidos con los resultados esperados. El equipo de Desarrollo tendrá que solucionar este error y dependiendo de la complejidad del mismo, puede necesitar de la ayuda del equipo de análisis.

En el siguiente diagrama se puede ver el proceso de detección y comunicación de un error en las pruebas.

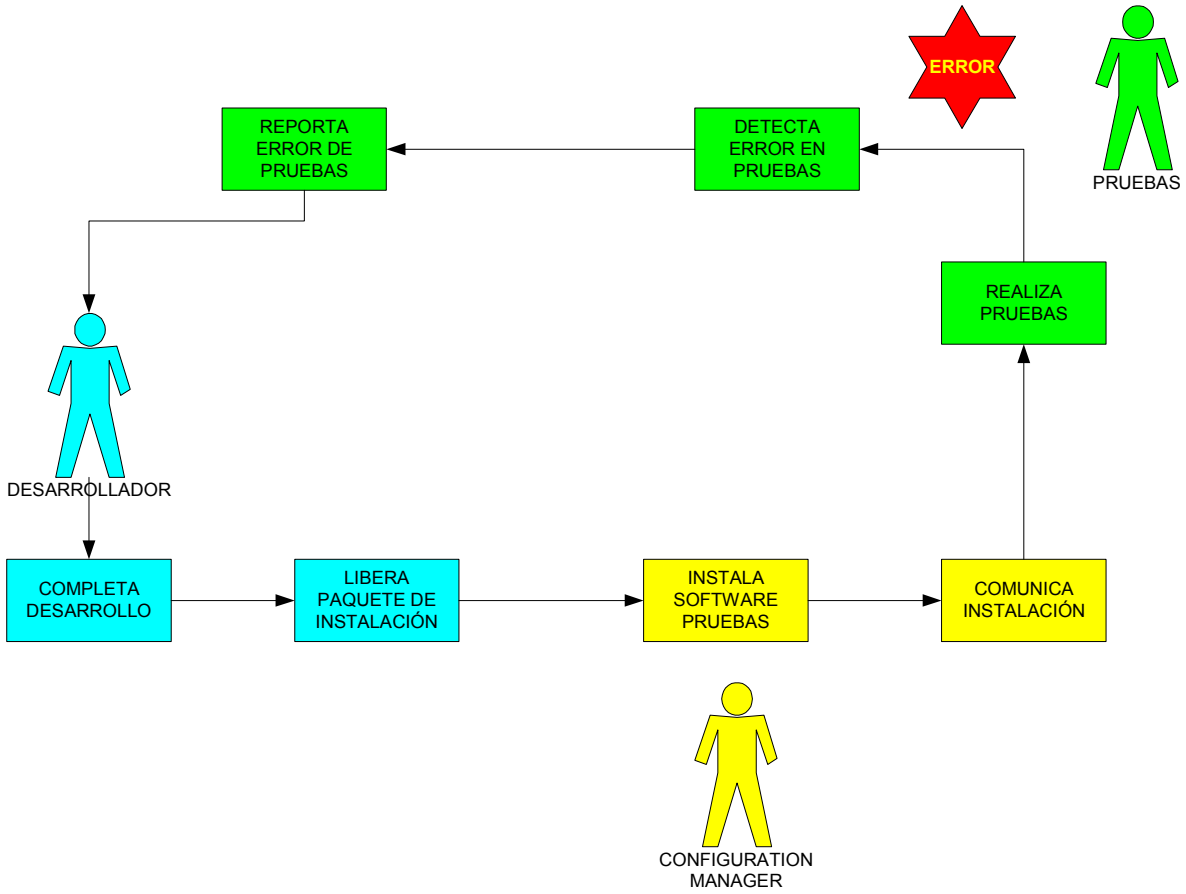


Ilustración 13 - Error en Proceso de Pruebas

5.2.3.3 Proceso de Explotación

Al finalizar correctamente las pruebas sobre un determinado paquete de Software enviado por el equipo de Desarrollo, este software queda validado para su *paso al entorno de Producción*. Este paso se debe hacer de forma coordinada con las áreas usuarias, que deben ser conscientes de la funcionalidad que se incorpora, de los cambios en sus operativas, así como de posibles indisponibilidades de servicio.

Una vez instalado el Software en Producción, este software comienza su Explotación, que es en definitiva para lo que se ha implementado. Dicho software dará una serie de resultados que

pueden ser mejoras en los sistemas, correcciones de errores, nuevas funcionalidades o cambios en el reporte a los usuarios.

En el caso de que estos resultados no sean los esperados por las áreas usuarias o los que fueron pactados por ellos, estos documentarán y comunicarán un Error de Explotación al equipo de desarrollo. El equipo de desarrollo determinará el error que se ha producido y su posible solución. En caso de errores de elevada complejidad puede precisar de la labor del equipo de Análisis.

En el siguiente diagrama se puede ver el proceso de un error de explotación.

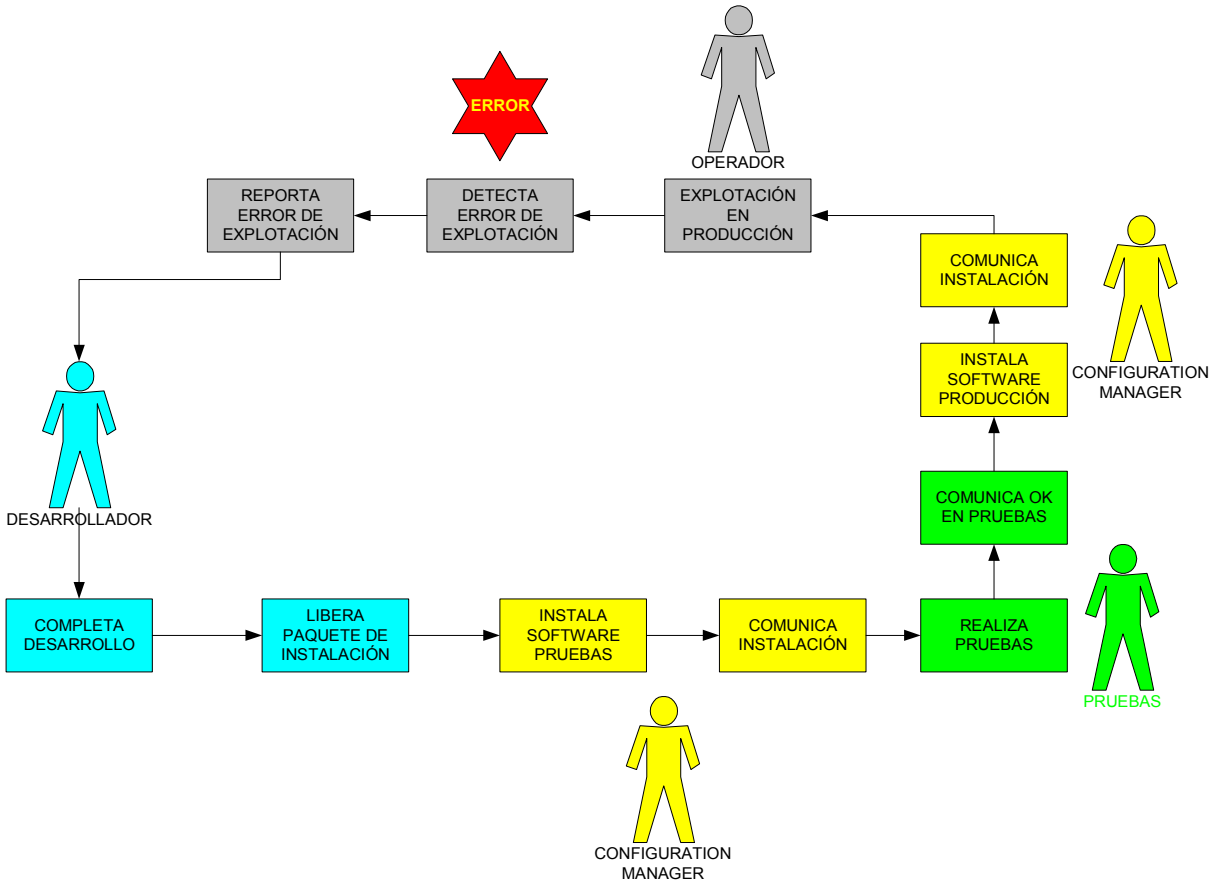


Ilustración 14 – Error en Proceso de Explotación

Una vez se ha descrito el área de sistemas de una compañía y los procesos de Instalación, Pruebas y Explotación junto con sus puntos de error, se va a definir la arquitectura de la solución, tanto desde el aspecto funcional como tecnológico.

5.3 Arquitectura Funcional

Se describen en este apartado los diferentes módulos de la solución desde el punto de vista funcional. Se van a definir las diferentes funcionalidades a cubrir y sus relaciones.

Se verá inicialmente un esquema general de la arquitectura y posteriormente se analizarán las relaciones entre los diferentes módulos.

5.3.1 Esquema general

Se incluye a continuación un esquema general de la arquitectura funcional de la solución.

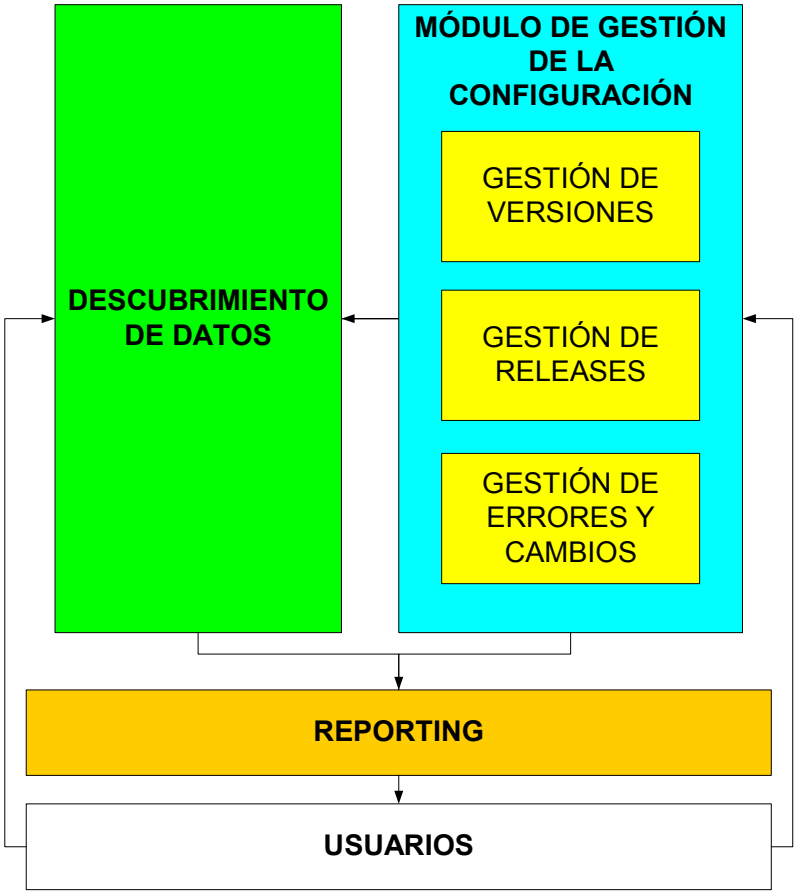


Ilustración 15 – Arquitectura Funcional

La solución consta de dos grandes módulos funcionales:

- *Módulo de Gestión de la Configuración*: Módulo para las tareas de Gestión de la Configuración. Tendrá a su vez un módulo de Gestión de Versiones, un módulo de Gestión de Releases y uno de Gestión de Cambios y Errores.

- *Módulo de Descubrimiento de Datos*: Módulo para la búsqueda de nuevo conocimiento a partir de la información obtenida del módulo de usuarios y del módulo de Gestión de la Configuración. Este módulo se tratará en detalle en la sección "Análisis y Diseño de la parte de Descubrimiento de Datos".

Se debe asimismo tener la capacidad de extraer y mostrar la información a los diferentes usuarios del sistema según sus características, por ello se deben cubrir los siguientes módulos:

- *Módulo de Gestión de Usuarios*: Módulo para la identificación y gestión de los diferentes usuarios y perfiles de la aplicación.

- *Módulo de Reporting*: Módulo para dar la capacidad de visualizar información a los usuarios.

Se describe a continuación brevemente la funcionalidad de cada uno de los módulos.

5.3.2 Módulo de Usuarios

El sistema debe ser capaz de identificar en cada momento las operaciones que puede realizar un determinado usuario. El objetivo del sistema es poder establecer puntos de mejora en la gestión de la configuración y en el ciclo de vida del software, para lo cual debe separar los papeles de los diferentes tipos de usuarios, tal y como se comentó en la sección "Roles y Responsabilidades".

5.3.3 Módulo de Reporting

El Módulo de Reporting debe dar la posibilidad de presentar a los diferentes usuarios la información que necesitan. De este modo se podrán identificar una serie de pantallas e

informes que aportarán dicha información. Se trata de un aspecto fundamental en este trabajo la visualización de resultados obtenidos.

5.3.4 Módulo de Gestión de la Configuración

El núcleo del sistema es el módulo de Gestión de la Configuración. Se tiene que llevar el control de lo que sucede con cada versión, con cada release, con cada error o con cada requerimiento. En la sección del análisis "Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos" se revisará con detalle este módulo.

5.3.5 Módulo de Descubrimiento de Datos

Una vez se tiene determinada información en el Módulo de Gestión de la Configuración, el sistema debe tratar de encontrar tendencias o patrones que permitan identificar mejoras para la empresa. Esta función se hace en el Módulo de Descubrimiento de Datos utilizando técnicas de Aprendizaje Automático como Redes Neuronales.

5.4 Arquitectura Tecnológica

En esta sección se va a proponer una primera aproximación a la arquitectura tecnológica que se va a adoptar en la solución. Al igual que en la sección dedicada a la Arquitectura Funcional, se va a mostrar un esquema general para posteriormente mostrar un mayor detalle de cada una de las partes.

5.4.1 Esquema general

En el siguiente esquema se pueden ver las partes principales de la arquitectura tecnológica de la solución.

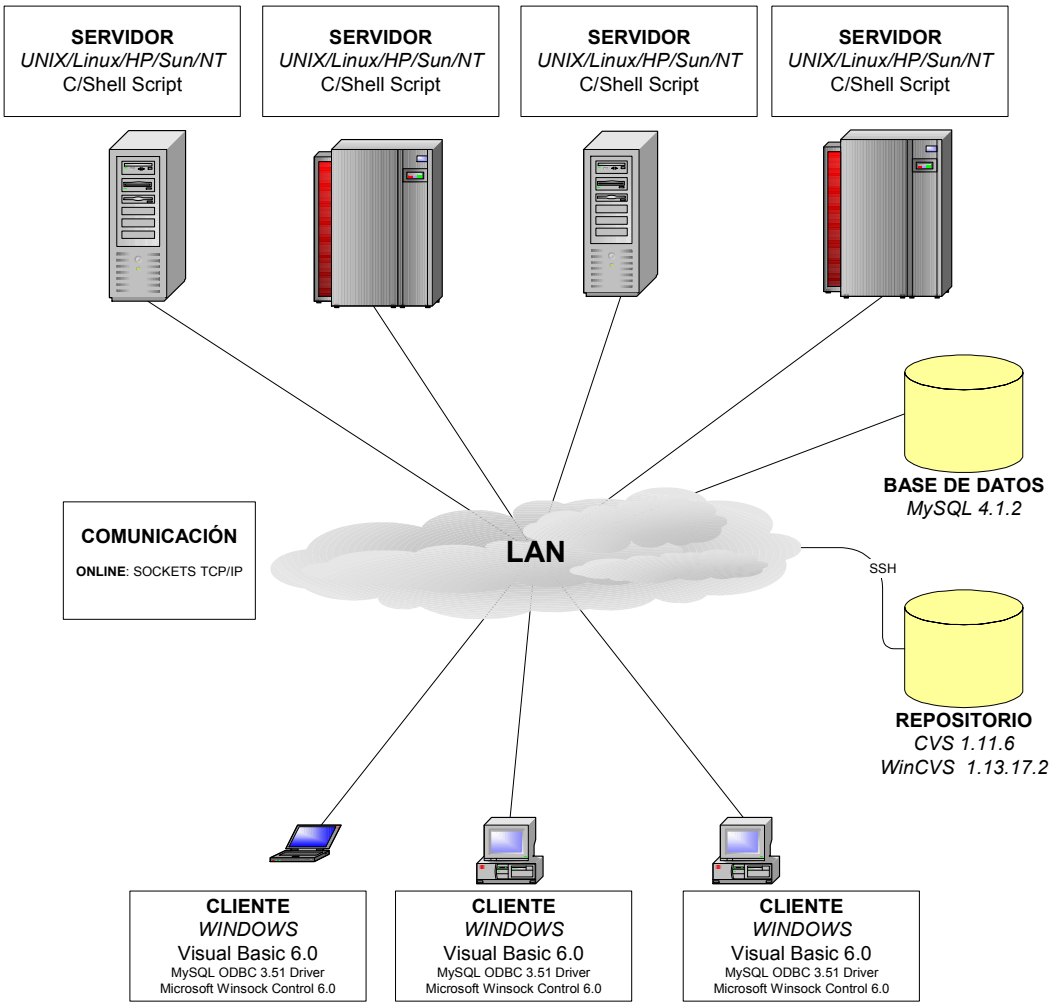


Ilustración 16 – Arquitectura Tecnológica

Existen varios módulos diferenciados unidos por una red. Estos son:

- *Módulo Cliente*: Es el módulo desde el que se lanzarán todas las operaciones y se harán todas las peticiones. Asimismo es donde se presentará la información a los diferentes usuarios, bien por pantalla o bien a través de informes generados por el sistema. Habrá un cliente instalado en cada uno de los usuarios del área de sistemas que vayan a utilizar la herramienta.

- *Módulo Servidor*: Es el encargado del proceso de las peticiones del cliente. En cada una de las máquinas de la compañía cuya configuración se quiera controlar habrá un programa servidor escuchando en un determinado puerto. Se realiza una comunicación con los clientes a través de una comunicación Online, cuando se establece un socket entre ambos. Los procesos servidores serán los encargados de realizar las operaciones de cálculo, tales como las asociadas al Aprendizaje Automático.

- *Módulo de Base de Datos*: Es la Base de Datos en la que se recoge la información del sistema. En dicha base de datos habrá una parte del modelo donde se recoja la información de los entornos del área de sistemas, y en otra parte la relativa a usuarios, relacionada con la información que se pueda descubrir del sistema de Gestión de la Configuración. Asimismo se tendrá otra parte del modelo donde se recogerán los datos relativos al control de la configuración, tales como los errores, requerimientos o releases. Con toda la información anterior se podrán hacer las tareas de descubrimiento de datos.

- *Módulo de Repositorio*: El módulo central de información que se va a manejar es el de Repositorio. En él se guardará toda la información de las versiones generadas por los diferentes usuarios del sistema, así como las propias versiones. Se utilizará una herramienta de uso extendido (CVS) para soportar estas funciones.

Se verán a continuación más detalladamente cada una de las partes de la Arquitectura Tecnológica.

5.4.2 Módulo Cliente

El Módulo Cliente es desde el que se pueden lanzar las diferentes operaciones de usuarios de la aplicación. Deberá ser accesible desde todos los puestos de usuarios finales de la aplicación.

Se va a utilizar una herramienta de uso generalizado para aplicaciones gráficas: *Microsoft Visual Basic 6.0*. Esto implica que los clientes de la aplicación estarán en sistemas operativos *Microsoft Windows*. Nótese que el cliente únicamente lanza peticiones a un servidor, con lo que se podría construir un cliente para otros sistemas, como por ejemplo Linux.

La comunicación entre el módulo cliente y el módulo de Base de Datos se realizará a través de ODBC. El controlador utilizado es *MySQL ODBC 3.51 Driver.* A través de esta tecnología es muy sencillo realizar la portabilidad entre diferentes Sistemas Gestores de Bases de Datos, ya que simplemente habría que modificar la referencia a dicho controlador desde la aplicación y adaptar la sintaxis de las sentencias SQL. Para simplificar la tarea de gestionar la comunicación con una base de datos, se separarán las sentencias SQL a un único módulo de código que sería el único a modificar en un eventual cambio de Sistema de Bases de Datos.

La comunicación con el Módulo Servidor se realizará a través de Sockets TCP/IP con un protocolo específico del sistema. Por ello, hay que incorporar en la aplicación cliente el control OCX que habilita el uso de Sockets. Se trata del *Microsoft Winsock Control 6.0*, cuyos eventos y métodos permiten dirigir una comunicación por sockets.

El puesto cliente deberá disponer también de un cliente CVS para la comunicación con el repositorio de código. Al igual que en el caso de las sentencias SQL, se deberán aislar las sentencias CVS en un módulo de código independiente para posibles cambios de repositorio de versiones. Para contactar con dicho repositorio, será necesario que tenga también un cliente ssh.

5.4.3 Módulo Servidor

El Módulo Servidor tiene dos partes diferenciadas, para el proceso de atención de peticiones. Se trata de las partes "Demonio" y "Ejecución". Habrá un servidor en cada una de las máquinas desde las que se puedan atender peticiones.

La parte "Demonio" es la que está escuchando en un determinado puerto la llegada de peticiones y genera una rutina de atención a la petición a través de un proceso hijo una vez llega dicha petición. Esta parte se desarrollará en *lenguaje C*, incorporando las librerías que permiten realizar la comunicación a través de sockets. Típicamente:

```
#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>
```

Asimismo se deben incluir las librerías de sistema para el manejo de procesos hijo y mecanismos de comunicación entre procesos. Estas son:

```
#include<sys/ipc.h>

#include<sys/shm.h>

#include<sys/sem.h>

#include<signal.h>

#include <sys/stat.h>
```

Al ser desarrollado en lenguaje C será necesario generar un ejecutable para cada una de las plataformas en las que se instale un servidor. Si bien no será necesario cambiar el código fuente ya que el lenguaje es portable a las diferentes plataformas, siempre que admitan el manejo de procesos hijo, así como el manejo de rutinas de comunicación por sockets.

El proceso hijo, una vez que es generado, lanzará una rutina de "Ejecución" de la petición que se ejecutará en la segunda parte diferenciada del servidor. Las rutinas de atención de peticiones estarán implementadas a través de *Shell Scripts realizados con ksh (Korn Shell)*. Se

pretende ejecutar estos scripts de sistema y redireccionar la salida al socket para que le llegue al cliente que ha disparado la petición.

El servidor se puede establecer en plataformas UNIX, como HP, SUN o AIX, Linux, en sus diferentes distribuciones o bien en distribuciones Windows NT, 2000, XP o posteriores. Para ejecutar el servidor en distribuciones Windows se debe hacer a través de la herramienta Cygwin que es un emulador de un sistema Linux para Windows. En la sección "Servidor ECFM" se comenta detalladamente cómo realizar la configuración en el sistema operativo Windows.

Es necesario asimismo que el servidor disponga de un cliente CVS para establecer la comunicación con el módulo de repositorio, junto con un cliente SSH para gestionar comunicaciones seguras. Igualmente necesitará un cliente MySQL para poder conectar con el Sistema Gestor de Bases de Datos.

5.4.4 Módulo de Base de Datos

La base de datos que se ha elegido es el *Sistema de Gestión de Bases de Datos MySQL 4.1.2*. En todos los puestos clientes y servidores debe haber un cliente MySQL para poder hacer operaciones con la base de datos. Asimismo en los puestos clientes, se debe habilitar el Driver MySQL ODBC 3.51, tal y como se comentó anteriormente en la sección "Módulo Cliente". Debe ser accesible por ambos ya que es el punto donde se va a almacenar toda la información relativa a los diferentes componentes del sistema.

Se trata por tanto de una base de datos relacional. Se va a mantener una integridad referencial real, no únicamente lógica. Para habilitar el manejo de claves extranjeras y la integridad referencial, es necesario declarar los objetos que en ella se creen de tipo INNODB.

5.4.5 Módulo de Repositorio

Para el Módulo de Repositorio de Versiones se ha elegido como sistema *CVS (Concurrent Versions System) 1.11.17*, como ya se ha justificado anteriormente. Se trata de un sistema abierto para manejo de versiones por múltiples desarrolladores.

Existirán varios repositorios centralizados, bien independientes o bien submódulos de un mismo repositorio físico, en los que los desarrolladores van almacenando las diferentes versiones de su software generado. Así disponen de una misma foto de la evolución del sistema todos ellos, eliminándose posibles riesgos derivados de un incorrecto manejo de versiones como puede ser la pérdida de cambios realizados por otros, la duplicidad en esfuerzos, o simplemente la no realización de copias de seguridad de forma centralizada.

En los puestos cliente se puede instalar una interfaz gráfica llamada *WinCVS 1.13.17.2*. Se trata de una herramienta que permite trabajar en modo gráfico con las operaciones más comunes de CVS. Se puede ver con más detalle el uso de dicha herramienta en la sección "Repositorio ECFM".

Para la comunicación entre el resto de componentes del sistema, ya sean clientes o servidores, y el repositorio es necesario el uso de SSH. Este protocolo permite realizar comunicaciones de forma encriptada entre diferentes máquinas. Para ello:

- En el Servidor donde esté instalado cada repositorio central debe haber un demonio SSHD para atender las peticiones que lleguen por SSH.

- En las diferentes máquinas que vayan a acceder se debe tener un cliente SSH debidamente configurado para comunicar con el repositorio.

La configuración del Repositorio de Versiones desde el punto de vista de Administración se detallará en la sección "Repositorio ECFM".

5.5 Análisis y Diseño de la Solución

En este apartado se describe el análisis y el diseño detallado de la solución. Una vez se ha descrito la Arquitectura Funcional del Sistema completo y la Arquitectura Tecnológica sobre la que se van a articular los diferentes componentes del sistema, se van a analizar a un nivel inferior los diferentes componentes del Sistema. Se tratarán en esta sección:

- Orientación del Sistema a Entornos de Trabajo.
- Modelo de Proceso de Gestión de Versiones, Releases, Errores y Requerimientos.
- Comunicación entre componentes del sistema.
- Análisis y Diseño en detalle de los componentes principales:
    - Cliente
    - Servidor
    - Base de Datos
    - Repositorio.

Nótese que la parte de Descubrimiento de Datos se analizará en una sección posterior.

5.5.1 Entornos de Trabajo

En el esta solución cuya arquitectura se ha descrito previamente, se va a hablar de forma continua de Entornos de Trabajo. Se define un *Entorno de Trabajo* como aquel lugar en el que un usuario ubicado en una determinada fase del ciclo de vida de desarrollo puede desempeñar sus funciones.

Para establecer estos entornos de trabajo se va a proponer la siguiente Modelización basada en una serie de entidades implicadas en la descripción de dichos entornos:

- **APLICACIÓN**: Un Entorno de Trabajo está asociado a una determinada Aplicación. Se define Aplicación como una determinada unidad funcional separada del resto dentro de los sistemas de una compañía. Se puede ver un ejemplo en el diagrama de la sección "Descripción General de Procesos de Movimiento de Software del Área de Sistemas de una Compañía".

- **ENTORNO**: En un Entorno de Trabajo también se encuentra el concepto de Entorno Lógico, o simplemente Entorno. Se asociará al lugar que ocupa en el ciclo de vida. Por ejemplo se puede hablar de Entorno de Desarrollo, para el lugar donde se hacen los desarrollos o de Explotación, donde se utiliza el sistema en Producción.

- **MÁQUINA**: Para un determinado Entorno de Trabajo se asocian una o muchas Máquinas. La Máquina es el lugar hardware físico donde se realiza dicho Trabajo. Para simplificar el modelo se supondrá que la relación entorno/máquina es uno a uno, de manera que si se tiene un entorno con varias máquinas se deberá tratar como varios entornos. Por ejemplo, si se tuvieran tres máquinas en un entorno de producción se podría tener los entornos PROD_1, PROD_2 y PROD_3 para referenciar a una u otra máquina.

- **INFORMACIÓN DEL ENTORNO**: Asociado a un determinado Entorno de Trabajo se tiene una información básica asociada que nos permitirá identificarlo unívocamente. Para nuestro sistema y el modelo que se está estableciendo, algunas de las partes básicas de un Entorno de Trabajo son las siguientes:

    - La Aplicación asociada.
    - El Entorno asociado.
    - La Máquina asociada.
    - Ruta Base a partir de la que reside el software instalado en el Entorno.
    - Herramienta de Versionado que se usa en dicho entorno.
    - Información sobre el Servidor ECFM que trabaja en dicho entorno: Puerto en el que escucha el demonio o directorio donde se guardan los logs.

- Nombre identificador del Entorno de Trabajo.

- **VARIABLES DE ENTORNO**: Además de la Información general del Entorno que se acaba de describir y que aplica a todos los entornos, es claro que puede haber información particular para cada Entorno de Trabajo, que haya que definir como variables propias del mismo. Así, por ejemplo, si un determinado Entorno de Trabajo tiene como herramienta de Versionado CVS, se deberá establecer para ese entorno las variables de conexión al servidor de CVS.

A continuación se dispone un sencillo ejemplo de Entornos de Trabajo para una determinada aplicación. Se puede ver en el diagrama como existen tres entornos lógicos, Desarrollo, Pruebas y Producción, y cada uno de ellos tiene su máquina, que en el caso de producción son dos. Para todos ellos se establecen el directorio base y la herramienta de Versionado, así como una serie de variables propias cuyo valor difiere en cada uno de los entornos.

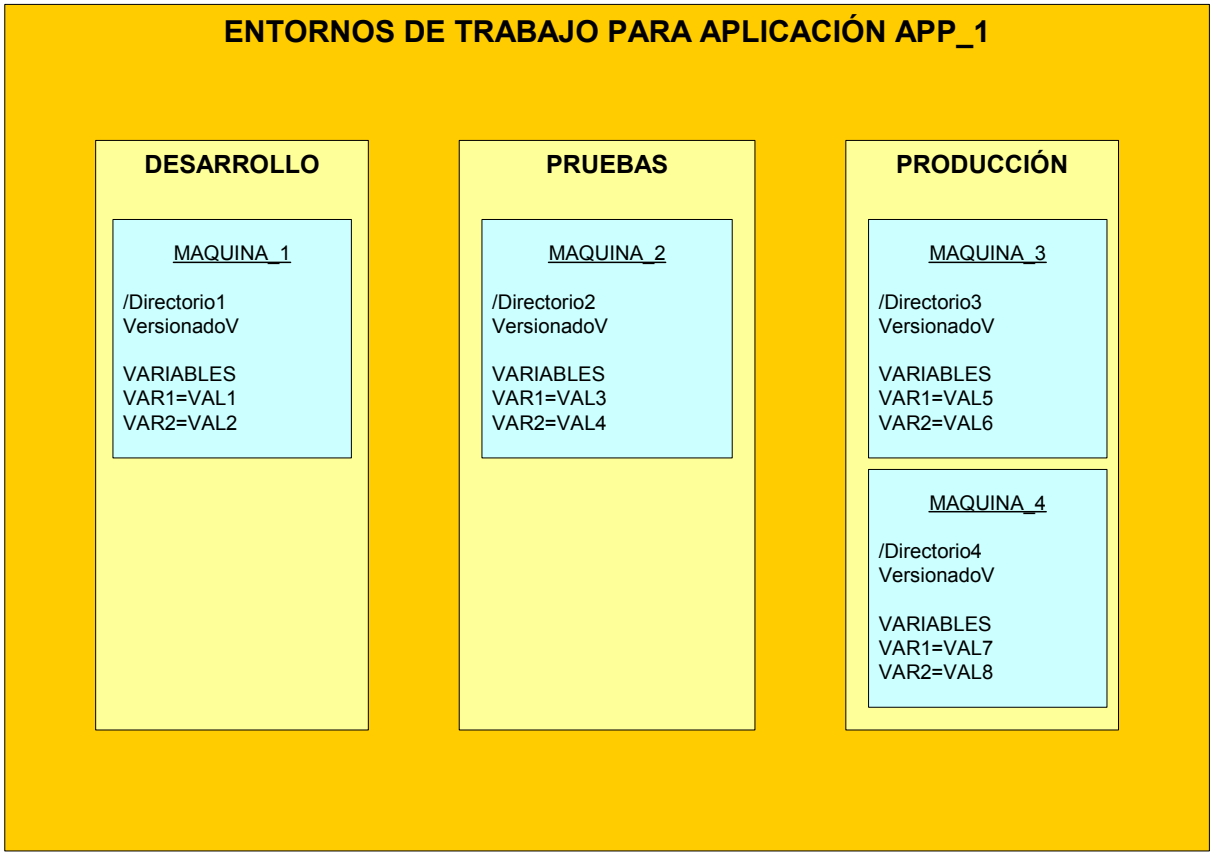


Ilustración 17 - Entornos de Trabajo

En adelante se denominará "Entorno" bien a un Entorno de Trabajo, es decir el conjunto de todos estos componentes que se han descrito, o bien a un Entorno Lógico, como Desarrollo, Pruebas o Producción, de forma indistinta.

5.5.2 Modelización del Proceso de Gestión de Cambios

En esta sección se va a proponer un modelo a utilizar en esta solución para los procesos asociados con el control de cambios en el software que se van a agrupar en Gestión de Versiones, Releases, Errores y Requerimientos. El objetivo es la integración de todos estos procesos para poder establecer relaciones entre ellos de forma flexible.

5.5.2.1 Procesos de Gestión de Cambios. Unidades de Control

En adelante se hablará de *Unidad de Control* para referirnos a las unidades de manejo de los procesos de Gestión de Versiones, Releases, Cambios y Requerimientos en esta solución. Habrá un tipo de unidad de control para cada uno de estos procesos. En la siguiente tabla se describen las Unidades de Control Básicas asociadas.

PROCESO	DESCRIPCIÓN PROCESO	UNIDAD DE CONTROL BÁSICA	DESCRIPCIÓN UNIDAD DE CONTROL BÁSICA
Gestión de Versiones	Es el proceso que describe la forma de realizar los cambios en las versiones individuales de Software	FICHERO	Cualquier objeto Software susceptible de ser versionado. Por ejemplo: Documentos, Fuentes, Binarios o Modelos de Datos.

PROCESO	DESCRIPCIÓN PROCESO	UNIDAD DE CONTROL BÁSICA	DESCRIPCIÓN UNIDAD DE CONTROL BÁSICA
Gestión de Releases	Este proceso define la forma de realizar las entregas de software y su instalación en los diferentes Entornos de Trabajo.	PACK	Un conjunto de unidades FICHERO con sus versiones asociadas que en conjunto forman una unidad funcional y que se puede instalar en uno o varios Entornos de Trabajo.
Gestión de Errores	Proceso que describe las actividades asociadas con el manejo de Errores en el Software.	ERR	Error Software. Normalmente producido por un defecto en un FICHERO, ya sea funcional o de otro tipo, o bien por una incorrecta instalación de un PACK o por cualquier otro motivo.
Gestión de Requerimientos	Este proceso describe las acciones que se llevarán a cabo cuando se traten nuevos Requerimientos Funcionales en el sistema.	REQ	Nuevos Requerimientos. Normalmente surgirán por una nueva funcionalidad requerida o bien por alguna mejora o por un cambio de alcance.

Así por ejemplo, cuando se hable de actualizar un FICHERO se hablará de un objeto asociado al proceso de Gestión de Versiones. O bien, cuando se hable de un PACK concreto, se estará refiriendo a un paquete de instalación preparado.

5.5.2.2 Jerarquía de Unidades de Control

Entre las unidades de control se establece una *jerarquía lógica*. Las unidades FICHERO estarán asociadas a una o varias unidades tipo PACK, ya que se distribuirán con una o varias

Releases. Cada una de las unidades PACK estará asociada a una o muchas unidades de tipo REQ y/o ERR, ya que las Releases entregadas cubrirán o bien nuevos requerimientos o bien solucionarán Errores surgidos en el sistema.

Asimismo se define una relación de cada una de las unidades de Control con las Aplicaciones del Sistema. Es decir, cada una de las Unidades de Control debe manejarse asociada siempre a una determinada Aplicación. Por ejemplo se tendrá FICHERO, PACK, ERR y REQ de la aplicación 1, 2 o N de forma separada.

Se podrán establecer relaciones entre unidades REQ y ERR entre diferentes aplicaciones. Esto se debe a que puede haber un REQ o ERR en una aplicación 1 que afecte a otro producto. La forma de establecer esta jerarquía será a la hora de generar la unidad PACK o Release. Esta unidad llevará asociada los REQ y ERR de su aplicación, así como los REQ y ERR de otras aplicaciones que le pudieran afectar.

De igual manera se podrán establecer relaciones entre REQ y ERR, de manera que un ERR puede estar asociado a una implementación incorrecta de un REQ determinado.

Nótese por último que las unidades REQ y ERR podrían subdividirse a su vez en diferentes unidades. Por ejemplo podrían separarse en diferentes unidades de control los distintos errores según el entorno en el que se produjeran. Por ejemplo, podrían existir unidades de control ERR_PRU, o ERR_INST o ERR_PROD, si se quisieran llevar por separado los errores en pruebas, en procesos de instalación o en producción.

En el siguiente diagrama se muestra la jerarquía entre unidades de Control ERR, REQ y PACK y las diferentes aplicaciones.

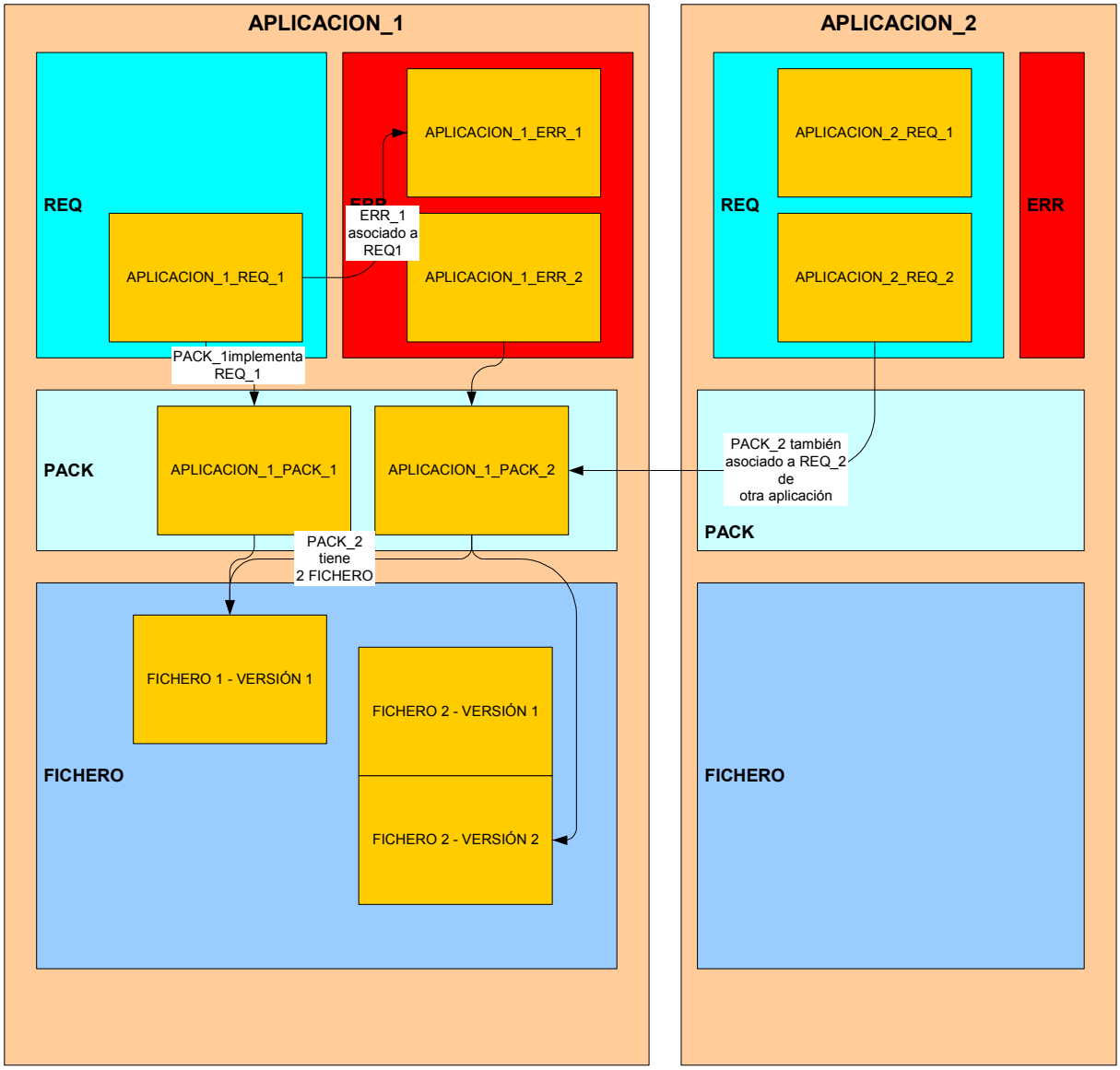


Ilustración 18 - Jerarquía de Unidades de Control

5.5.2.3 Ciclos de Vida de Unidades de Control

Asociado también a cada una de las Unidades de Control se tiene un determinado Ciclo de Vida. El *Ciclo de Vida* de una Unidad de Control es el conjunto de estados por el que pasa dicha Unidad de Control desde su creación hasta que se da por finalizada.

Es necesario poder definir en nuestro modelo un Ciclo de Vida diferente para cada unidad de Control, ya que puede ser un requisito de la compañía en la que se implemente. En el siguiente diagrama se muestra un ciclo de vida que puede ser un modelo para las unidades REQ, ERR y PACK.

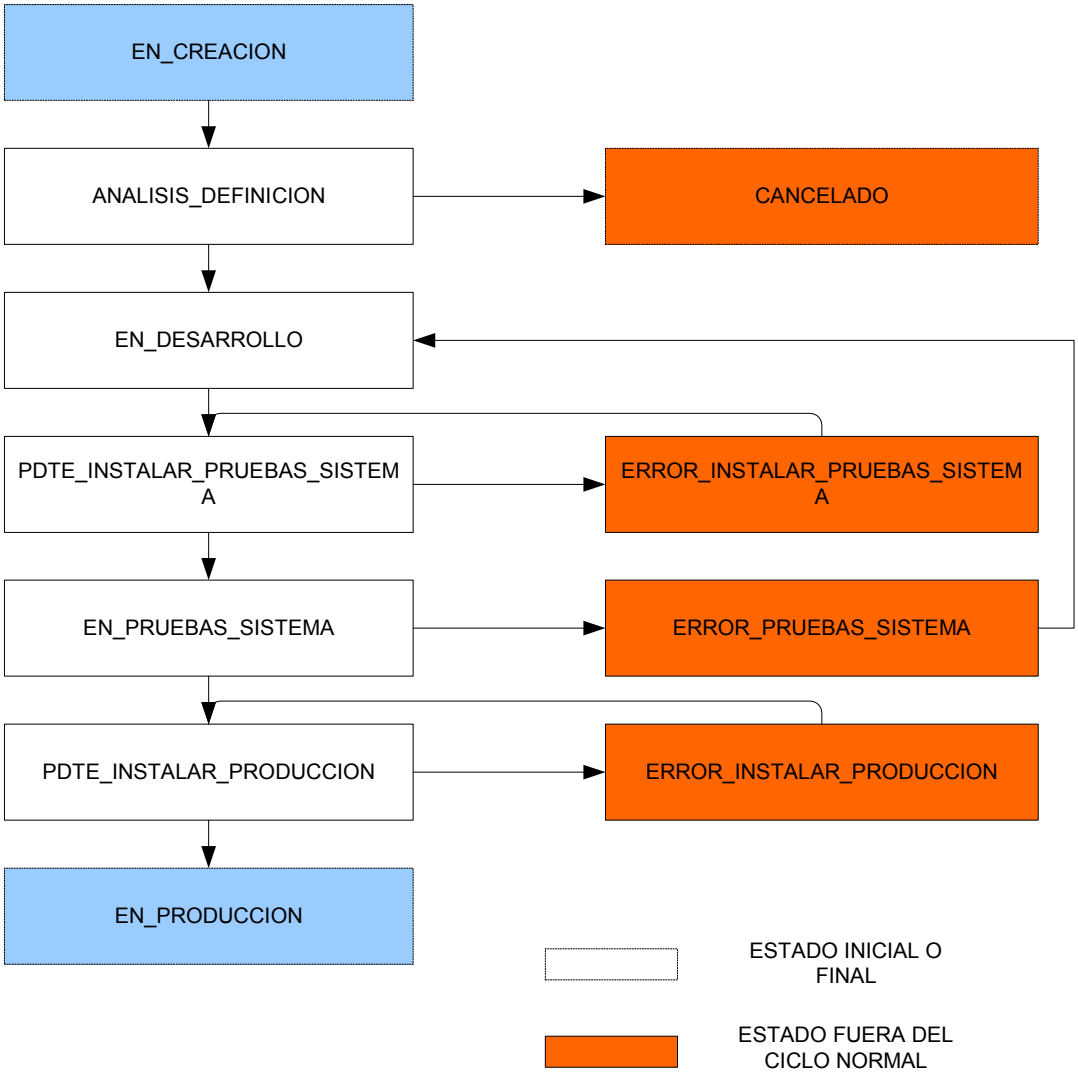


Ilustración 19 - Ciclo de Vida de Unidades REQ, ERR y PACK

A continuación se resumen los estados del Ciclo de Vida de estas unidades de Control:

- **EN_CREACION**: La Unidad de Control está siendo creada y rellenada con la información necesaria para su debida identificación.

- **ANALISIS_DEFINICION**: Se está acotando el impacto que representa la Unidad de Control y definiendo su solución.
- **EN_DESARROLLO**: Se está desarrollando la solución.
- **PDTE_INSTALAR_PRUEBAS_SISTEMA**: El desarrollo ha sido concluido, y está pendiente de instalar en entorno de Pruebas.
- **EN_PRUEBAS_SISTEMA**: Se están realizando pruebas según plan establecido.
- **PDTE_INSTALAR_PRODUCCION**: Las pruebas han finalizado y queda pendiente de instalar en entorno productivo.
- **EN_PRODUCCION**: Se da por cerrada la Unidad de Control una vez se ha implantado en producción.
- **CANCELADO**: La Unidad de Control ha sido Cancelada durante su Ciclo de Vida y deja de tener validez.
- **ERROR_INSTALAR_PRUEBAS_SISTEMA**: Hubo errores en la instalación en los entornos de pruebas.
- **ERROR_PRUEBAS_SISTEMA**: Hubo algún error al realizar las pruebas en el entorno de pruebas.
- **ERROR_INSTALAR_PRODUCCION**: Hubo errores en la instalación en el entorno de explotación.

Para la unidad FICHERO no se establece inicialmente un ciclo de vida ya que se puede asociar al del PACK que tenga asociado. No obstante se podría establecer un sencillo ciclo de vida con unos estados tales como DESARROLLO_EN_CURSO → EN_SERVICIO o DESARROLLO_EN_CURSO → CANCELADO.

5.5.2.4 Atributos de Unidades de Control

Deben existir una serie de informaciones que identifiquen debitamente a las diferentes Unidades de Control. Así por ejemplo puede ser necesario en un PACK establecer la Fecha de Entrega prevista, o para un REQ si se trata de un Cambio de Alcance o bien de un nuevo requerimiento funcional.

Para ello se establecerá una relación 1 a N entre cada una de las unidades de control y los posibles atributos que estas tengan.

5.5.3 Integración entre Componentes

En secciones anteriores, se ha mencionado que el sistema debe funcionar de forma integrada. Para ello es necesario establecer una serie de mecanismos de comunicación entre los diferentes componentes que se van a describir a continuación.

5.5.3.1 Comunicación entre Cliente y Servidor

Para la comunicación entre el Cliente y el Servidor se van a utilizar Sockets. Como se ha descrito en la sección "Arquitectura Tecnológica", el Cliente utilizará el Control Winsock desde la interfaz gráfica y el servidor utilizará las librerías para manejo de sockets desde C.

Se va a establecer un protocolo que permita que:

- El cliente lance una petición.
- El servidor cree un proceso hijo para atender la petición.
- El proceso hijo ejecute un Shell Script mediante una llamada controlada al sistema.
- La salida del Shell Script llegue en tramas al cliente.
- El proceso hijo mande una trama de finalización de comunicación una vez haya terminado la llamada al sistema.

En el siguiente diagrama se muestra el Protocolo de comunicación que se establecerá entre Cliente y Servidor.

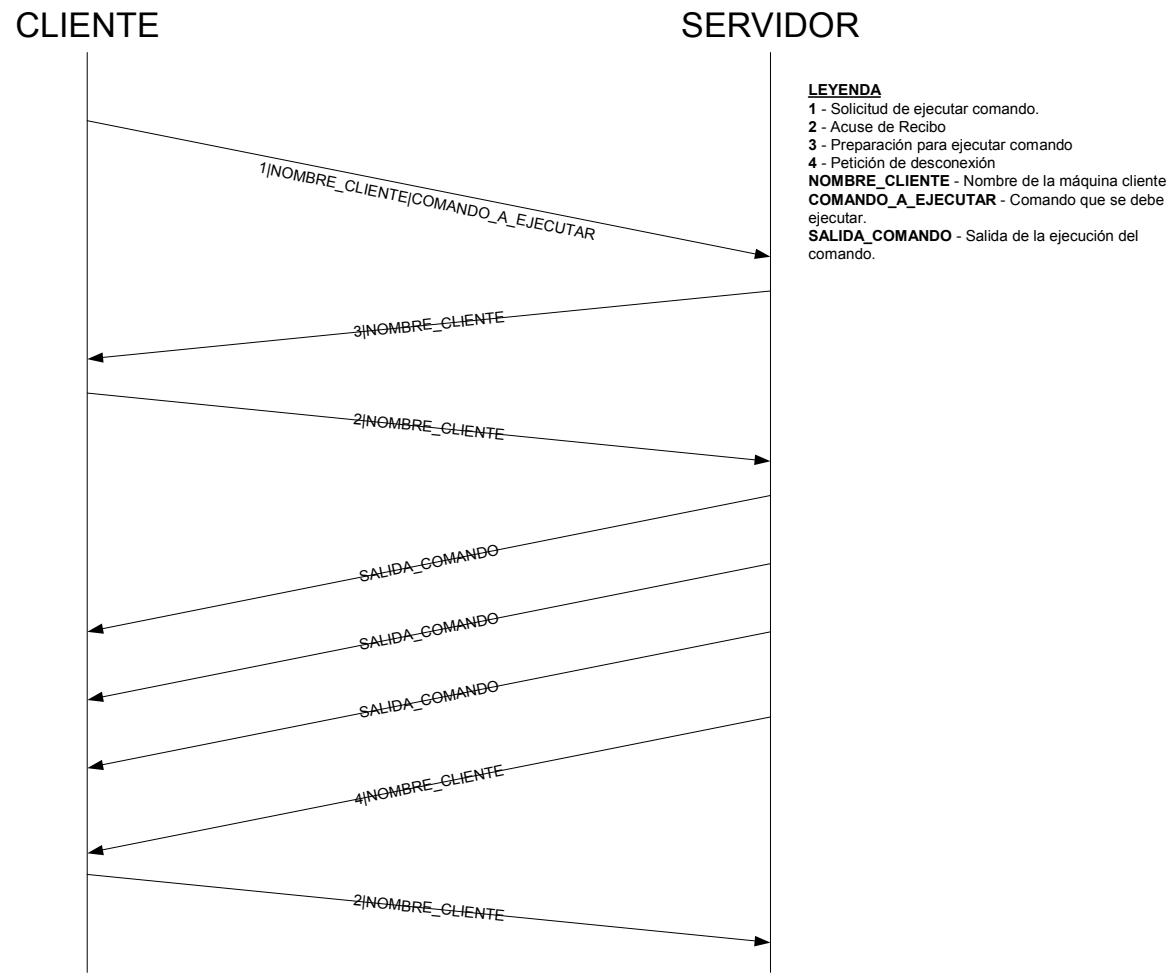


Ilustración 20 - Protocolo de Comunicación Cliente-Servidor

Como se puede observar, se establecerán las tramas de comunicación de la siguiente manera:

```
ID_TRAMA|NOMBRE_HOST|[RESTO]
```

Donde:

- ID_TRAMA es el identificador del tipo de trama que se manda. Podrá ser:
    - 1: Solicitud de ejecutar comando en el servidor

  - 2: Acuse de Recibo
  - 3: Notificación desde el servidor para indicar que la siguiente trama será ya de ejecución
  - 4: Trama de fin de ejecución y solicitud de desconexión.
- NOMBRE_HOST es la máquina cliente implicada en la comunicación.
- RESTO será el comando en caso de que sea una trama de ejecución de comando y vacío en el resto de casos.

Las tramas de resultado de la ejecución que se identifican como SALIDA_COMANDO no llevarán encabezados ya que serán la salida directa de la ejecución del Shell Script. Será el cliente el encargado de interpretarlas y de mostrar posibles errores que existieran en esta comunicación.

En el Cliente se implementará una *Clase de Conexión al Servidor*, que permitirá detectar los eventos de cambios en el socket tales como llegada de nuevos datos, fin de envío de datos o conexión completada. Asimismo dirigirá las comunicaciones a través del protocolo establecido para independizar esta comunicación del resto de módulos del sistema, que únicamente tendrán que referenciar a métodos de la clase para hacer operaciones de comunicación.

5.5.3.2 Comunicación entre Cliente y Base de Datos

El Cliente se comunicará con la Base de Datos a través del Driver ODBC dispuesto por el Sistema Gestor de Bases de Datos, en este caso MySQL, tal y como se comenta en Arquitectura Tecnológica.

Para independizar el desarrollo del Cliente y la base de datos elegidos se deberán adoptar dos medidas:

- Crear una *Clase de Conexión a Base de Datos* independiente: Se generará un módulo de clase cuyos métodos sean las diferentes llamadas que se puedan hacer a las operaciones sobre la base de datos (Por ejemplo Listados de Unidades de Control, Consulta de Usuarios con ciertos permisos o Inserción de nuevos valores en los atributos de un objeto). Las propias sentencias SQL irán únicamente allí de manera que si se cambia de gestor de Bases de Datos, lo único que habrá que cambiar si procede, son algunas sentencias SQL dentro de los métodos de esta clase.

- Extraer los datos de comunicación con el Gestor de Bases de Datos a un fichero de configuración aparte, de manera que se eliminen las dependencias del código con el sistema destino al que conecta. Simplemente se consultará el fichero de configuración.

5.5.3.3 Comunicación entre Cliente y Repositorio

La herramienta cliente realizará una llamada a sistema a través de línea de comandos que permita acceder al repositorio. Se deberá habilitar el protocolo de comunicación necesario para que se puede establecer conexión a través de dicha línea de comandos.

Como se ha definido en la sección “Arquitectura Tecnológica” se va a utilizar la herramienta CVS para el manejo de versiones. La comunicación se establecerá a través del protocolo SSH. En el siguiente diagrama se puede observar el sentido de dicha comunicación y sus componentes.

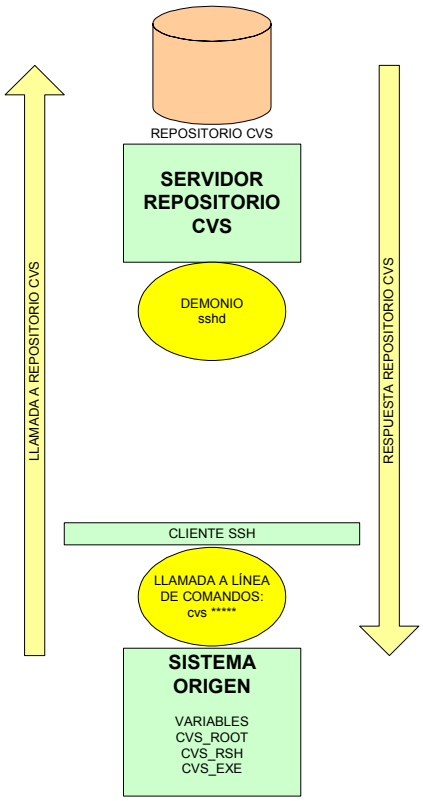


Ilustración 21 – Comunicación con Repositorio CVS a través de SSH

El Sistema origen tendrá establecidas las variables de Entorno:

- CVS_ROOT: Para indicar el repositorio CVS destino.
- CVS_RSH: Para indicar el uso de ssh para la comunicación.
- CVS_EXE: Para indicar dónde está el ejecutable de CVS en el servidor.

Llamará a través de línea de comandos al sistema destino, teniendo debidamente configurado el cliente ssh.

En el Servidor de Repositorio CVS debe haber un demonio sshd para atender peticiones de ssh. Desde este demonio se llamará al servidor de repositorio para contestar a la petición realizada por el cliente en el sentido inverso.

Se deberá tratar de independizar las llamadas al repositorio a través de línea de comandos en un módulo aparte de funciones, de manera que un eventual cambio de herramienta de versionado o el añadir otra nueva al sistema, únicamente se traduciría en añadir nuevos módulos con funciones equivalentes adaptadas a dicha herramienta.

5.5.3.4 Comunicación entre Servidor y Base de Datos

La comunicación entre el servidor y la base de datos se hará a través del cliente en línea de comandos del sistema gestor de bases de datos. Como el servidor trabaja con llamadas al sistema a través de Shell Scripts, las llamadas a la base de datos se harán a través de ese cliente en línea.

Para tratar de mantener independencia del código con el gestor se mantendrán:

- Una librería de funciones de consulta de base de datos, de manera que potenciales cambios del gestor de bases de datos únicamente conlleven cambios en dicha librería.
- Un fichero con la configuración necesaria para conectar a la base de datos.

5.5.3.5 Comunicación entre Servidor y Repositorio

La comunicación entre el Servidor y el Repositorio sigue la misma guía que se ha establecido en la “Comunicación entre Cliente y Repositorio” que consiste en el manejo de llamadas a la línea de comandos CVS. Igualmente se utilizará el protocolo SSH para comunicaciones seguras.

5.5.3.6 Comunicación off-line

Hay que establecer asimismo un protocolo para la comunicación off-line, para gestionar las comunicaciones entre clientes y servidores de forma no en línea, por ejemplo cuando finalice algún cálculo de largo proceso.

Habrá un Servidor conocido por el resto de componentes al que se llamará Servidor de Mensajes. Este tendrá la estructura de directorios:

inbox ← Buzón de entrada

error ← Mensajes erróneos

in ← Mensajes entrantes

in_progress ← Mensajes siendo procesados

processed ← Mensajes leídos

Los mensajes serán archivos de texto que tendrán la siguiente nomenclatura:

USUARIO_DESTINO@ACCIÓN@DESCRIPCIÓN_MENSAJE@FECHA

Donde:

- USUARIO_DESTINO: Es el usuario al que se dirige el mensaje. El valor TODOS indicará que es un mensaje dirigido a cualquiera de los usuarios de la aplicación.
- ACCIÓN: Determinará la acción a realizar por el usuario. Es decir, el tipo de mensaje enviado.
- DESCRIPCIÓN_MENSAJE: Descripción del mensaje. Estará asociada a la acción.
- FECHA: Fecha en formato YYYYMMDD.

Los clientes tendrán un demonio que cada cierto tiempo comprobará el directorio “in”, llamando a una rutina de Atención de Petición. Si hay mensajes nuevos se le mostrará por pantalla un listado de los mismos para que los procese.

Una vez descritas todas las interfaces de comunicación entre componentes, se van a analizar cada uno de dichos componentes por separado con mayor detalle.

5.5.4 Cliente

Se describe en esta sección el *análisis y diseño detallado de la parte Cliente* de la solución, una vez se han descrito la orientación a Entornos de Trabajo, el modelo de procesos y la integración con el resto de componentes. Se parte de una “Arquitectura Funcional” previamente definida y de una “Arquitectura Tecnológica” determinada, como se ha indicado en anteriores secciones.

Se van a describir en detalle los diferentes componentes del cliente, en cuanto a librerías, pantallas o módulosde código:

- *Clases*: Clases de la aplicación que serán instanciadas desde las diferentes partes de la misma.
- *Módulos*: Módulos generales de código.
- *Formularios*: Cada una de las pantallas de la aplicación.

Se muestra a continuación un esquema de las diferentes partes que se van a detallar en esta sección.

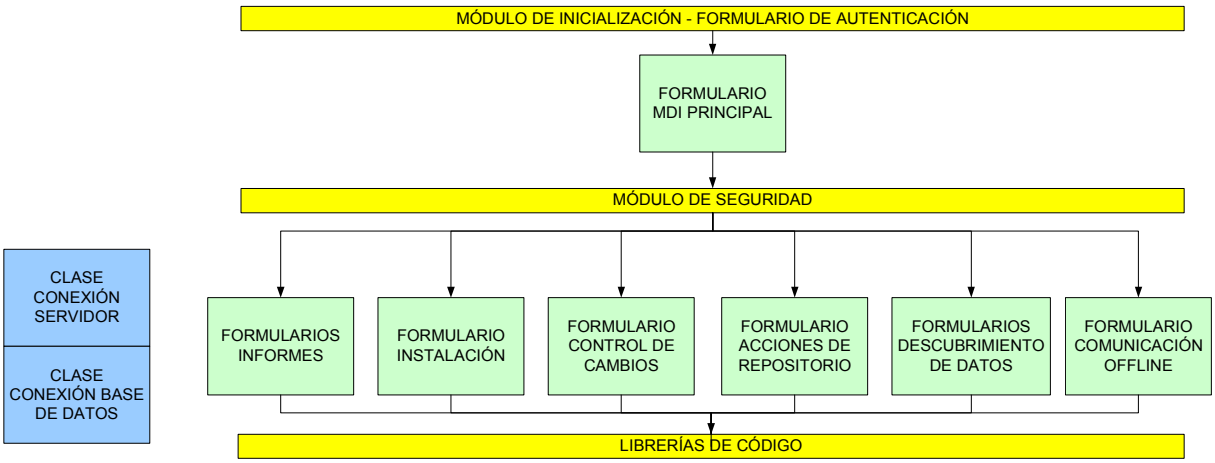


Ilustración 22 – Módulo Cliente

5.5.4.1 Clases

Se definen dos clases para el cliente, que se han comentado anteriormente en las secciones "Comunicación entre Cliente y Servidor" y "Comunicación entre Cliente y Base de Datos". Se trata de las clases de conexión al servidor y de conexión a la base de datos.

5.5.4.1.1 Clase de conexión al servidor

Las llamadas al servidor se harán a través de esta clase. Controlará el manejo de sockets con los eventos de un control WinSock.

Se deberán definir los siguientes **atributos**:

- *Estado_Conexion*: Indicará el estado de una conexión en un momento del tiempo. Podrá tener los siguientes valores, según los posibles valores del atributo "State" de un objeto "WinSock":
  - *"0" - "Conexión cerrada"* – Conexión cerrada. No existe conexión.
  - *"1" - "Conexión abierta"* – Conexión abierta con el servidor.
  - *"2" - "Escuchando"* – Conexión esperando datos

  - *"3" - "Pendiente de establecer conexión"* – Pendiente de realizar conexión con el servidor.
  - "*4" - "Consiguiendo puerto"* – Consiguiendo el puerto donde hacer la comunicación.
  - *"5" - "Puerto conseguido"* – Puerto del servidor obtenido.
  - *"6" - "Conectando"* – Intentando conectar.
  - *"7" - "Conectado"* – Conexión completada.
  - *"8" - "Cerrando la conexión"* – Cerrando la conexión en curso.
  - *"9" - "Error"* – Error realizando la conexión.

- *Flag_Enviados*: Flag que indicará que se ha completado el envío de datos.
- *Flag_Llegada*: Flag que indicará la llegada de datos.
- *wsSocket*: Socket sobre el que se trabaja.
- *bResetear*: Variable booleana que indica que se ha recibido una orden de interrumpir la comunicación desde fuera de la clase.
- *RemoteHost*: Host remoto al que se quiere conectar.
- *RemotePort*: Puerto del Host Remoto.

Los**métodos** de la clase serán:

- *Constructor de la Clase*: Método que se ejecutará cuando se crean objetos de esa clase. Pondrá "Estado_Conexión" a "Conexión Cerrada"
- *Establecer el servidor destino*: Establecer el servidor destino y el puerto de conexión con los atributos "RemoteHost" y "RemotePort" de la clase.

- *Conectar con un Servidor*: Llamará al método "Connect" de winsock.
- *Recibir datos del Socket*: Llamará al método "GetData" de winsock.
- *Enviar datos del Socket*: Llamará al método "SendData" de winsock.
- *Mandar Petición*: Método para realizar la petición completa de ejecución de un comando en el servidor. Se deberán mandar las peticiones al servidor con el formato de mensajes y el protocolo establecido en "Comunicación entre Cliente y Servidor".

5.5.4.1.2 Clase de Comunicación con la Base de Datos

Esta clase permitirá lanzar operaciones contra el sistema gestor de Bases de Datos. Para ello tendrá los siguientes atributos:

- *ConexionBD*: Objeto de conexión de tipo ADODB (Active Data Objects)
- *rsResult*: Recordset para manejo de operaciones.

Los métodos serán:

- *Constructor*: Establecerá la conexión con la base de datos.
- *Cerrar la conexión*: Eliminará la conexión con la base de datos.
- El resto de métodos serán llamadas a la base de datos en forma de consultas, u operaciones de inserción, borrado o actualización a través de Recordsets o cadenas de caracteres que se manejarán desde las diferentes partes de la aplicación.

5.5.4.2 Módulos

Se definirán una serie de Módulos de código a modo de librerías de funciones. La idea es agrupar las funciones asociadas a determinadas funcionalidades en un mismo módulo. Serán necesarios los siguientes módulos:

- *Funciones de Comunicación con el Servidor*: Desde muchas pantallas de la aplicación se llamará al Servidor. Las sentencias de llamadas a scripts ("ksh") de servidor se encapsularán en este módulo de funciones de comunicación.

- *Funciones de Repositorio*: El hecho de que se haya elegido un repositorio CVS no quiere decir que no pudieran añadirse más repositorios. Para poder hacer esto último de la manera más sencilla posible, se aislarán en un mismo módulo las funciones que realizan las llamadas al sistema para la ejecución de comandos CVS.

- *Funciones Generales*: Habrá módulos con funciones de uso general que pudieran afectar a varios formularios, de manera que se evite la duplicación de código en la aplicación en la medida de lo posible.

- *Función de Inicialización*: Asimismo habrá un módulo con el main de la aplicación, de manera que sea la primera función que se ejecute al arrancar el sistema.

- *Funciones de Redes Neuronales*: Grupo de funciones encargadas del manejo de redes neuronales y de la presentación de resultados.

Podrá haber más módulos siguiendo el principio de que se trata de agrupar funciones que tienen un objetivo común.

5.5.4.3 Formularios

Como se ha comentado en anteriores secciones el módulo cliente se trata de una interfaz gráfica. Como tal se definen una serie de formularios o pantallas. Nótese que en la relación que se va a dar a continuación no se incluirán todos los formularios finales, que se determinarán en tiempo de construcción, sino los que deben existir para cubrir las funciones principales descritas anteriormente en "Arquitectura Funcional". Estos son:

- *Formulario de Autenticación o Inicialización*: Permitirá acceder a cada usuario únicamente a las pantallas a las que tenga acceso.

- *Formulario de Informes*: Centralizará la generación de informes de la aplicación.

- *Formulario de Instalación*: Permitirá realizar o dirigir procesos de instalación de software en Entornos de Trabajo, al menos en parte. Se trata del módulo de Gestión de Releases que se presenta al usuario.

- *Formulario de Control de Cambios*: Este formulario permitirá realizar las operaciones básicas sobre las Unidades de Control de la aplicación, tales como crearlas, relacionarlas, actualizarlas, rellenar sus atributos o consultar su información. Esto implementaría el módulo de Gestión de Errores y Cambios.

- *Formulario de Repositorio*: Habrá formularios que permitan interactuar con el repositorio, de manera que se pueda introducir en la base de datos información asociada a las nuevas versiones que se generan, que nos permitan a posteriori poder descubrir datos en dicha información. Complementará al propio repositorio para conformar el módulo funcional de Gestión de Versiones.

- *Formularios de Descubrimiento de Datos*: Realizarán el manejo y Administración de Redes Neuronales y minería de datos de causas de errores. Se verán detalladamente en la sección "Análisis y Diseño de la parte de Descubrimiento de Datos".

- *Formulario de comunicación "offline":* Para indicarle a un usuario algún mensaje recibido desde otro componente de la aplicación en un momento del tiempo.

Se utilizará un formulario MDI (Multiple Document Interface) como padre del resto de formularios para manejar una barra de menús y una botonera común.

A continuación se detalla el análisis de cada uno de los formularios principales.

5.5.4.3.1 Formulario de Autenticación. Módulo de Seguridad

El control de la seguridad de la aplicación se hace a través de usuarios, grupos y pantallas. Un determinado usuario pertenecerá a uno o varios grupos. Cada grupo tendrá acceso a una o

varias pantallas. Por tanto un usuario tendrá permisos sobre las pantallas asociadas a los grupos a los que pertenezca. Esto se deberá comprobar al activar cada formulario de la aplicación.

Se habilitará un formulario inicial en el que se pedirán el usuario y la contraseña de acceso. Estas variables determinarán el perfil del usuario y dicho perfil, las pantallas a las que tiene permisos. Se guardarán como variables globales a todos los formularios para realizar las comprobaciones necesarias en cualquier momento.

5.5.4.3.2 Formulario de Gestión de Informes

Para centralizar los informes generados por la aplicación se creará un Formulario de Informes. La salida de este formulario para el usuario final serán diferentes informes parametrizables y predefinidos.

El usuario seleccionará el informe y se permitirá filtrar por una serie de atributos definidos en la aplicación por cada uno de los informes. En caso necesario se podrá exportar a un fichero de texto o bien a una herramienta externa.

5.5.4.3.3 Formulario de Instalación

Se pretende poder identificar errores de instalación en los diferentes Entornos de Trabajo. Para ello se dotará al sistema de un Formulario de Instalación. Este estará orientado a Unidades de Control de tipo PACK, cuyo detalle se ha visto en la sección "Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos". Se tendrá en el formulario:

- Un listado de Packs disponibles para las instalaciones, obtenido a partir de la aplicación y del Entorno de Trabajo seleccionados.

- Un botón que lanzará la acción de Verificación de la instalación para comprobar la coherencia lógica de la misma.

- Un botón que lanzará la acción de Instalación para iniciar la copia de Software u otras actividades asociadas a la misma.

- Una ventana de texto que muestre la comunicación con el servidor en el que se realiza la acción.

5.5.4.3.4 Formulario de Control de Cambios

Este Formulario servirá para realizar las acciones principales con las Unidades de Control tal y como se indica en la sección "Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos".

Se sacará un listado de Unidades de Control a partir de la Aplicación asociada y el tipo de Unidad de Control elegida. A partir de este listado se permitirá:

- Crear Nuevas Unidades de Control del Tipo Elegido.

- Relacionar las Unidades de Control con otras de otros tipos o bien con objetos de tipo FICHERO.

- Ver el Detalle de las Unidades de Control.

- Modificar los atributos de las Unidades de Control.

5.5.4.3.5 Formulario de Acciones sobre Repositorio

Es necesaria una interfaz que permita introducir información adicional en la base de datos cuando se introducen nuevos cambios en el Repositorio.

Se requiere por tanto en este formulario:

- Un listado de objetos que se extraerá del repositorio, que dependa de la aplicación elegida y de la acción que se quiera realizar. En un primer momento la acción será únicamente añadir nuevos ficheros al repositorio o cambiar una versión.

- Obtener de la base de datos una serie de preguntas a realizar que dependerán de la aplicación elegida y que representan la información que nos interesa de la acción. Sus respuestas se guardarán en la base de datos.

5.5.4.3.6 Formulario de Comunicación "Offline"

Se establecerá un demonio en el cliente que cada cierto tiempo compruebe la recepción de mensajes en un servidor de Mensajes preestablecido. En caso de que haya nuevos mensajes se mostrarán en este Formulario de Comunicación, de manera que podrán ser procesados en ese momento o no, según convenga al usuario.

5.5.5 Servidor

El Servidor será el encargado de atender y procesar las peticiones que le lleguen desde los clientes. En cada una de las máquinas en las que se quiera atender algún tipo de petición deberá existir un servidor instalado y arrancado. Un servidor escuchará por un determinado puerto que debe ser conocido por el cliente para que este pueda lanzar sus peticiones.

5.5.5.1 Estructura

El servidor tendrá dos partes diferenciadas:

- *Demonio de Atención de Peticiones*: Es el proceso que está escuchando constantemente en un determinado puerto de la máquina. Se debe configurar como un servidor concurrente ya que podrán llegar peticiones de diferentes clientes al mismo tiempo.

- *Scripts de Ejecución de Peticiones*: Es el encargado de llamar al sistema, una vez el demonio de atención de peticiones ha recogido la petición. La salida de esta llamada será redirigida al cliente, a través del canal de comunicación previamente establecido.

La comunicación con los clientes se hará a través de sockets. Por ello serán necesarias la IP y el puerto del Servidor y la IP y el puerto del cliente, para poder establecer el canal. El cliente elegirá un puerto que tenga libre en tiempo de ejecución, en concreto al llamar a "Connect", y será comunicado al servidor al realizar la petición.

5.5.5.2 Atención de Peticiones

Se describe a continuación un mayor detalle del módulo de Atención de Peticiones.

5.5.5.2.1 Funcionamiento

Como se ha comentado anteriormente se tratará de un servidor concurrente. Se desarrollará en C, con las extensiones y librerías para implementación de sockets.

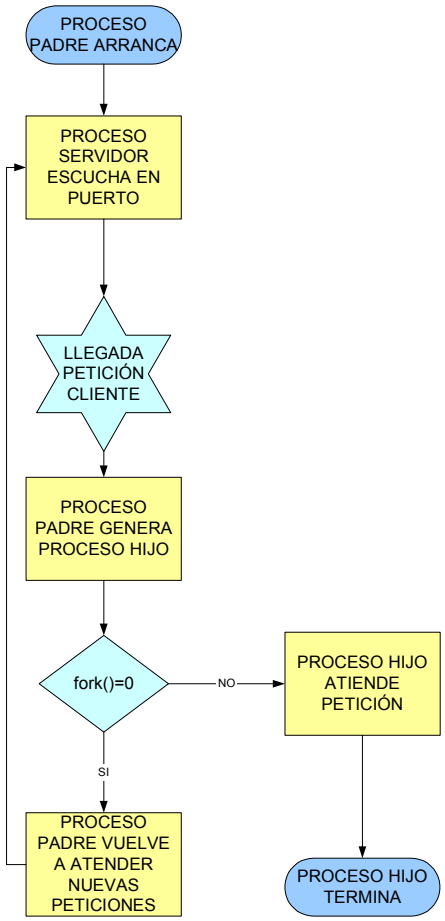


Ilustración 23 – Servidor Concurrente

En este diagrama se muestra el flujo del demonio de atención de peticiones que se debe implementar. La idea es que fork genere un proceso hijo por cada petición que se realice y deje al proceso padre en escucha de nuevas peticiones.

La salida del proceso hijo debe ser dirigida al socket de comunicación.

5.5.5.2.2 Fuentes y Librerías

Los archivos fuente a desarrollar serán:

- *Librería para comunicación con sockets*: Será un módulo que encapsulará las llamadas a las funciones de sockets, tales como conectar, escuchar, aceptar nuevos clientes, mandar mensaje, recibir mensaje y cerrar.

- *Librería para funciones de servidor*: Encapsulará las funciones de llamada al sistema para la ejecución de comandos, lanzando los scripts que correspondan, así como otras funciones necesarias para el servidor.

- *Main*: Rutina principal del proceso, que será la encargada de establecer el bucle en el que se atienden peticiones y se generan los procesos hijo.

5.5.5.3 Ejecución de Peticiones

El módulo de ejecución de peticiones estará desarrollado a través de Shell Scripts. Son llamadas directas al sistema cuya salida es dirigida al cliente. Su desarrollo se dividirá en librerías y rutinas de atención de peticiones.

Las funciones que tienen dependencia con una determinada herramienta de Versionado, se deben separar en un directorio propio de esa herramienta. En nuestro caso se está utilizando CVS, con lo que deberá crearse un subdirectorio "CVS" con los scripts que dependan de acciones con el repositorio. Habría un directorio por herramienta de versionado utilizada.

5.5.5.3.1 Librerías

Al igual que en otros módulos se tratará de independizar el código susceptible de cambiar, como por ejemplo podría suceder si se cambia el Sistema Gestor de Bases de Datos, o bien que sea común a varias rutinas de atención de peticiones.

Por ello se tienen las siguientes librerías:

- *Librería de Consulta a la Base de Datos*: Funciones que hacen llamadas directas al cliente del Sistema Gestor de Bases de Datos. En nuestro caso MySQL.

- *Librería de Herramienta de versionado*: Funciones que hacen llamadas al cliente de la herramienta de gestión de versiones. En nuestro caso CVS.

- *Librería de Informes*: Informes a ejecutar por la aplicación. Normalmente son consultas a la base de datos. Sus resultados serán enviados al cliente.
- *Librería de Funciones Generales*: Funciones de uso general para las diferentes rutinas como funciones de control de Logs.
- *Librería de funciones de Redes Neuronales*: Tendrá las funciones de manejo y cálculo de información de redes neuronales.

5.5.5.3.2 Atención de Peticiones

Para las diferentes peticiones realizadas por el cliente se asociarán diferentes rutinas de Atención a dichas peticiones. En la siguiente tabla se detallan estas rutinas.

RUTINA	**DESCRIPCIÓN**	**SALIDA**
ProcesarMensaje **VARIABLES**: ACCION (Listado, Procesar, Deshacer, Error) MENSAJE o USUARIO (Si pide un listado)	Esta rutina procesará los mensajes según la acción que se pida realizar que puede ser o listar los mensajes de un usuario, o procesarlos, o volver atrás un mensaje procesado o bien mandar a mensajes erróneos.	OK si termina bien o SIENDO_PROCESADO en caso de no encontrar el mensaje,
ListarPack **VARIABLES**: APLICACIÓN	Listado de los PACK disponibles de una determinada aplicación	Listado obtenido

RUTINA	DESCRIPCIÓN	SALIDA
VerificarPack **VARIABLES**: PACK DIRECTORIO_LOG: Directorio donde están los LOG VARIABLES de Herramienta de Versionado	Análisis de un Pack previo a la instalación para detectar si hubiera algún tipo de posible incoherencia como por ejemplo estar relacionado con un fichero que ya no exista en la herramienta de versionado, o bien que existan versiones superiores de ese fichero en el repositorio o bien que se hayan instalado en el Entorno de Trabajo destino versiones superiores de ese mismo fichero.	Posibles Warnings o Errores en la verificación.
InstalarPack **VARIABLES**: PACK DIRECTORIO_DESTINO: Directorio base del entorno DIRECTORIO_LOG: Directorio donde están los LOG VARIABLES de Herramienta de Versionado	Realizará la copia de todos los ficheros de un determinado PACK al directorio destino indicado. Para ello hará llamadas a la herramienta de versionado para obtener los ficheros relacionados con el pack y copiará el fichero al destino detectando posibles errores. Se habilitará la posibilidad de que se realicen una serie de acciones de instalación propias por aplicación como puede ser ejecutar una herramienta propia de una aplicación para desplegar o compilar el software copiado.	Mensajes de la instalación, atendiendo a posibles Warnings o Errores.
ListadoUControl **VARIABLES** TIPO_UNIDAD_CONTROL a listar	Listado de las unidades de control de un determinado tipo disponibles de una determinada aplicación	Listado obtenido
EjecutarInforme **VARIABLES**: Informe a ejecutar Filtros a aplicar	Ejecuta un determinado informe, aplicando los filtros que se pasan como parámetro	Resultado de la ejecución del informe, separando registros y campos.

RUTINA	DESCRIPCIÓN	SALIDA
EjecutarComando **VARIABLES**: Comando a ejecutar	Ejecutará un comando redireccionando la salida estándar y la salida de error a un mismo canal. Es importante que el cliente reciba la salida de error de sistema también. Por ello se llamará a esta rutina desde el módulo de gestión de peticiones, a través del proceso hijo para mostrar su salida completa al cliente.	La del comando que se ejecute.

5.5.6 Base de Datos

La Base de Datos del sistema es donde se va a almacenar toda la información utilizada por los diferentes componentes. Existirán una serie de grupos de tablas diferenciados, no obstante no deben de ser tratados como modelos independientes ya que tienen relaciones con el resto de componentes en alguna de las tablas.

5.5.6.1 Modelo de Datos

Se describe en esta sección el detalle el Modelo de Datos a utilizar por la aplicación. En secciones posteriores se identificará cada uno de los subgrupos de tablas que se podrán dividir en:

- Tablas de Entornos de Trabajo.
- Tablas de Informes
- Tablas de Usuarios
- Tablas de Unidades de Control
- Tablas de Descubrimiento de Datos

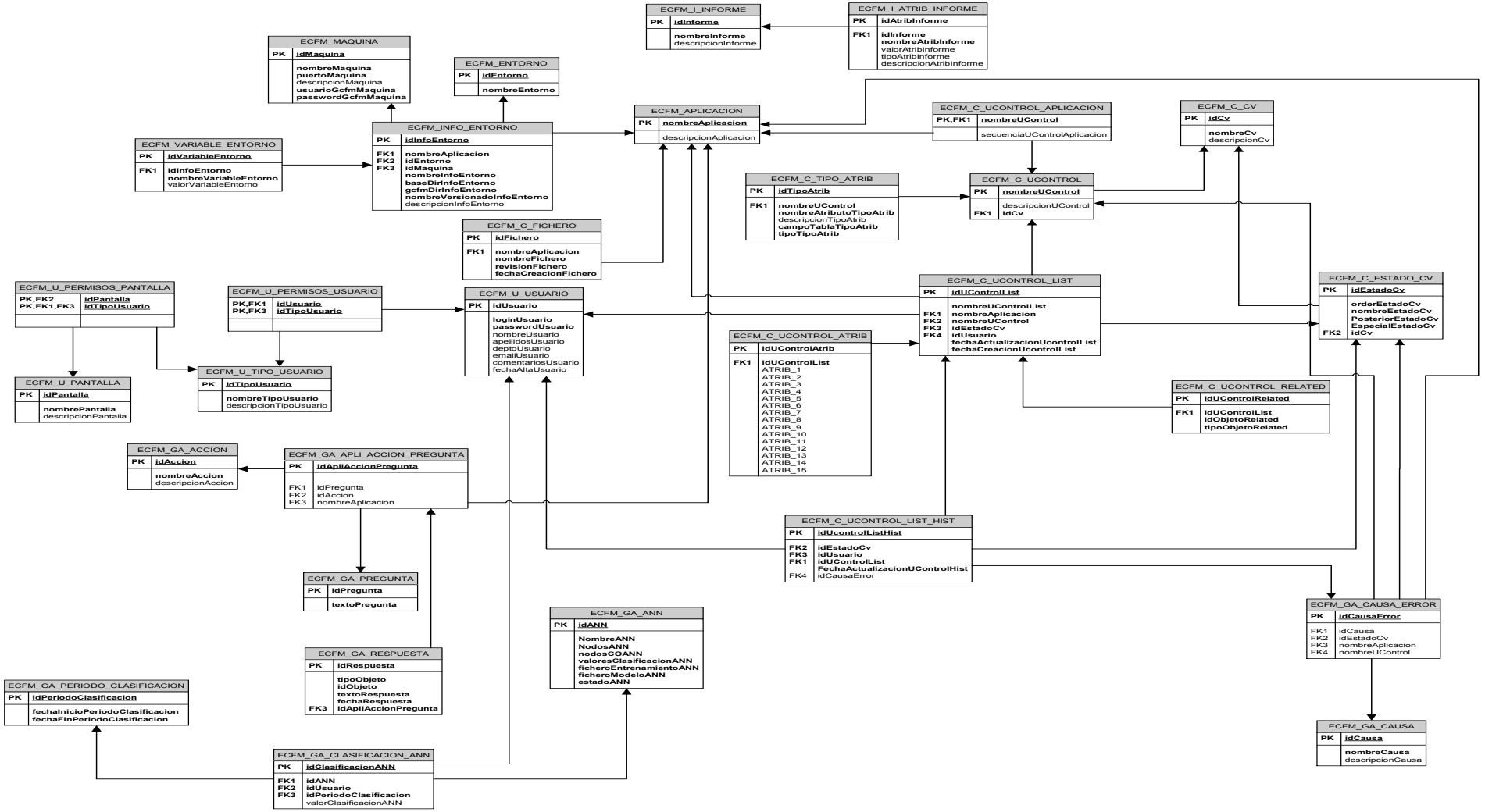


Ilustración 24 – Modelo de Datos

5.5.6.2 Modelo de Entornos de Trabajo: Tablas ECFM

Las tablas ECFM_* representarán la información relativa a Entornos de Trabajo de la Aplicación tal y como se ha descrito en la sección "Entornos de Trabajo" y que se ha utilizado en muchas otras secciones.

Para ello se describen las siguientes entidades:

- Aplicaciones.
- Máquinas.
- Entornos.
- Información básica de entornos.
- Variables propias de cada entorno.

TABLA	**ECFM_APLICACION**
DESCRIPCIÓN:	Tabla donde se guardan las diferentes aplicaciones del sistema.
ATRIBUTOS	**DESCRIPCIÓN**
nombreAplicacion	Nombre de la aplicación. Clave primaria.
descripcionAplicacion	Descripción.

TABLA	**ECFM_MAQUINA**
DESCRIPCIÓN:	Máquinas que se van a controlar desde el sistema.
ATRIBUTOS	**DESCRIPCIÓN**
idMaquina	Identificador de máquina. Clave primaria.
nombreMaquina	Nombre de la Máquina. Requerido.
puertoMaquina	Puerto de la Máquina donde escucha el servidor. Requerido.
descripcionMaquina	Descripción de la Máquina.
usuarioECFMMaquina	Usuario de la máquina que controla el servidor. Requerido.
passwordECFMMaquina	Password del usuario. Requerido.

TABLA	ECFM_ENTORNO
DESCRIPCIÓN:	Posibles Entornos Físicos de Trabajo.
ATRIBUTOS	**DESCRIPCIÓN**
idEntorno	Identificador. Clave Primaria.
nombreEntorno	Nombre del Entorno (Pruebas, Producción…)

TABLA	ECFM_INFO_ENTORNO
DESCRIPCIÓN:	Información básica común a todos los Entornos de Trabajo.
ATRIBUTOS	**DESCRIPCIÓN**
idInfoEntorno	Identificador. Clave Primaria.
nombreAplicacion	Aplicación. Referencia a ECFM_APLICACION.
idEntorno	Entorno Físico. Referencia a ECFM_ENTORNO.
idMaquina	Identificador de Máquina. Referencia a ECFM_MAQUINA.
nombreInfoEntorno	Nombre que se le da al entorno.
baseDirInfoEntorno	Directorio Base de Trabajo.
ECFMDirInfoEntorno	Directorio base donde se dejan los logs y otra información del entorno.
nombreVersionadoInfoEntorno	Nombre de la herramienta de Versionado asociada al entorno.
descripcionInfoEntorno	Descripción.

TABLA	ECFM_VARIABLE_ENTORNO
DESCRIPCIÓN:	Variables propias del entorno.
ATRIBUTOS	**DESCRIPCIÓN**
idVariableEntorno	Identificador. Clave Primaria.
idInfoEntorno	Identificador del entorno al que pertenece la variable. Referencia a ECFM_INFO_ENTORNO.
nombreVariableEntorno	Nombre de Variable.
valorVariableEntorno	Valor de la Variable.

5.5.6.3 Modelo de Informes: Tablas ECFM_I

Las siguientes tablas se utilizan para definir los informes con sus atributos o filtros propios.

TABLA	ECFM_I_INFORME
DESCRIPCIÓN:	Informes definidos.
ATRIBUTOS	**DESCRIPCIÓN**
idInforme	Identificador informe. Clave Primaria.
nombreInforme	Nombre del Informe.
descripcionInforme	Descripción del Informe.

TABLA	ECFM_I_ATRIB_INFORME
DESCRIPCIÓN:	Atributos del Informe.
ATRIBUTOS	**DESCRIPCIÓN**
idAtribInforme	Identificador de Atributos del Informe. Clave Primaria.
idInforme	Identificador del Informe. Referencia a ECFM_INFORME.
nombreAtribInforme	Nombre del Atributo.
valorAtribInforme	Valor por defecto del atributo.
tipoAtribInforme	Tipo de Dato del informe, puede ser CHAR, INT o bien DATE.
descripcionAtribInforme	Descripción del informe.

5.5.6.4 Modelo de Usuarios: Tablas ECFM_U

El Modelo de Usuarios se divide en las siguientes entidades:

- Descripción de los usuarios.
- Descripción de los tipos de usuario o grupos.
- Descripción de las Pantallas.
- Relación entre los usuarios y los grupos.
- Relación entre las pantallas y los grupos.

TABLA	ECFM_U_USUARIO
DESCRIPCIÓN:	Descripción de los usuarios.
ATRIBUTOS	**DESCRIPCIÓN**
idUsuario	Identificador de usuario. Clave Primaria.
loginUsuario	Login de usuario.
passwordUsuario	Contraseña.

TABLA	ECFM_U_USUARIO
nombreUsuario	Nombre del usuario.
apellidosUsuario	Apellidos del usuario.
deptoUsuario	Departamento del usuario.
emailUsuario	Correo Electrónico.
comentariosUsuario	Comentarios.

TABLA	ECFM_U_TIPO_USUARIO
DESCRIPCIÓN:	Tipos de usuario o grupos.
ATRIBUTOS	**DESCRIPCIÓN**
idTipoUsuario	Identificador del tipo.
nombreTipoUsuario	Nombre del tipo de usuario.
descripcionTipoUsuario	Descripción.

TABLA	ECFM_U_PANTALLA
DESCRIPCIÓN:	Pantallas disponibles.
ATRIBUTOS	**DESCRIPCIÓN**
idPantalla	Identificador de la pantalla.
nombrePantalla	Nombre de la pantalla.
descripcionPantalla	Descripción de la pantalla.

TABLA	ECFM_U_PERMISOS_USUARIO
DESCRIPCIÓN:	Grupos a los que pertenece cada Usuario.
ATRIBUTOS	**DESCRIPCIÓN**
idUsuario	Identificador del usuario. Pertenece a ECFM_U_USUARIO.
idTipoUsuario	Identificador del grupo del usuario. Pertenece a ECFM_U_TIPO_USUARIO.

TABLA	ECFM_U_PERMISOS_PANTALLA
DESCRIPCIÓN:	Pantallas a las que tiene permiso cada grupo de usuarios.
ATRIBUTOS	**DESCRIPCIÓN**
idPantalla	Identificador de la Pantalla. Pertenece a ECFM_U_PANTALLA.
idTipoUsuario	Identificador del tipo de usuario. Pertenece a ECFM_U_TIPO_USUARIO.

5.5.6.5 Modelo de Unidades de Control: Tablas ECFM_C

El modelo de Unidades de Control define lo asociado a los procesos referenciados en la sección Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos. Se almacenan:

- Los ciclos de vida.
- Los estados del ciclo de vida.
- Los tipos de Unidad de Control.
- Los tipos de Unidad de Control por cada aplicación.
- Los atributos por cada tipo de Unidad de Control.
- El listado de Unidades de Control.
- El listado de estados histórico por los que ha pasado una Determinada Unidad de Control.
- Los atributos de cada Unidad de Control.
- Las unidades de control relacionadas.
- Los ficheros que se pueden relacionar a las Unidades de Control.

TABLA	**ECFM_C_CV**
DESCRIPCIÓN:	Ciclos de Vida de las Unidades de Control.
ATRIBUTOS	**DESCRIPCIÓN**
idCv	Identificador del Ciclo de Vida. Clave primaria.
nombreCv	Nombre del Ciclo de Vida.
descripcionCv	Descripción.

TABLA	ECFM_C_ESTADO_CV
DESCRIPCIÓN:	Estados de un ciclo de vida.
ATRIBUTOS	**DESCRIPCIÓN**
idEstadoCv	Identificador del estado. Clave primaria.
orderEstadoCv	Orden en el ciclo de vida.
nombreEstadoCv	Nombre del Estado del ciclo de vida.
posteriorEstadoCv	Estado al que sigue.
especialEstadoCv	Indica si es o no un estado especial. Es decir, si está fuera del ciclo normal.
idCv	Ciclo de Vida. Referencia a ECFM_C_CV.

TABLA	ECFM_C_UCONTROL
DESCRIPCIÓN:	Tipos de Unidades de Control.
ATRIBUTOS	**DESCRIPCIÓN**
nombreUControl	Nombre del tipo. Clave primaria (PACK, ERR, REQ…).
descripcionUControl	Descripción de la Unidad de Contol.
idCv	Ciclo de vida asociado con el tipo. Referencia a ECFM_C_CV.

TABLA	ECFM_C_UCONTROL_APLICACION
DESCRIPCIÓN:	Relación entre tipo de unidad de Control y la Aplicación.
ATRIBUTOS	**DESCRIPCIÓN**
nombreAplicacion	Nombre de la Aplicación. Pertenece a ECFM_APLICACION.
nombreUControl	Nombre del Tipo de Unidad de Control. Pertenece a ECFM_C_UCONTROL.
secuenciaUControlAplicacion	Indica el número hasta el que se han reservado unidades de control del tipo marcado en la relación con el fin de determinar el siguiente.

TABLA	ECFM_C_TIPO_ATRIB
DESCRIPCIÓN:	Tipo de Atributo asociado a un tipo de unidad de control.
ATRIBUTOS	**DESCRIPCIÓN**
idTipoAtrib	Identificador del Tipo de Atributo. Clave primaria.
nombreUControl	Nombre de la Unidad de Control. Referencia a ECFM_C_UCONTROL.
nombreAtributoTipoAtrib	Nombre del atributo.
descripcionTipoAtrib	Descripción.
campoTablaTipoAtrib	Campo de la tabla de atributos al que se asocia.
tipoTipoAtrib	Tipo de Dato en la tabla de tipos de atributo.

TABLA	ECFM_C_UCONTROL_LIST
DESCRIPCIÓN:	Listado de Unidades de Control.
ATRIBUTOS	**DESCRIPCIÓN**
idUControlList	Identificador. Clave primaria.
nombreUControlList	Nombre en el listado de Unidades de Control.
nombreAplicacion	Nombre de la Aplicación. Referencia a ECFM_APLICACION.
nombreUControl	Nombre de la Unidad de Control. Se obtendrá concatenando aplicación, tipo de unidad de control y secuencia. Referencia a ECFM_C_UCONTROL.
tituloUControlList	Título de la Unidad de Control.
idEstadoCv	Estado del Ciclo de Vida. Referencia a un estado en ECFM_C_ESTADO_CV.
idUsuario	Identificador del usuario. Referencia a ECFM_U_USUARIO.
fechaActualizacionUcontrolList	Fecha de última actualización.
fechaCreacionUcontrolList	Fecha de creación.

TABLA	ECFM_C_UCONTROL_LIST_HIST
DESCRIPCIÓN:	Histórico de Acciones sobre una unidad de Control.
ATRIBUTOS	**DESCRIPCIÓN**
idUcontrolListHist	Identificador. Clave Primaria.
idEstadoCv	Identificador del Estado del Ciclo de Vida.
idUsuario	Identificador del Usuario.
idUControlList	Identificador de la Unidad de Control en el Listado. Referencia a ECFM_C_UCONTROL_LIST.
FechaActualizacionUControlHist	Fecha de actualización.
idCausaError	Causa que ocasionó el error. Referencia a ECFM_GA_CAUSA_ERROR.

TABLA	ECFM_C_UCONTROL_ATRIB
DESCRIPCIÓN:	Atributos de una unidad de control.
ATRIBUTOS	**DESCRIPCIÓN**
idUControlAtrib	Identificador del atributo.
idUControlList	Identificador en el listado de unidades de control. Referencia a ECFM_UCONTROL_LIST.
ATRIB_1	Atributo 1.
ATRIB_2	Atributo 2.
ATRIB_...	Atributo …
ATRIB_15	Atributo 15.

TABLA	ECFM_C_UCONTROL_RELATED
DESCRIPCIÓN:	Relaciones entre las diferentes unidades de control.
ATRIBUTOS	**DESCRIPCIÓN**
idUControlRelated	Identificador de la unidad de control relacionada. Unidad "padre".
idUControlList	Identificador en el Listado de Unidades de Control. Referencia a ECFM_C_UCONTROL_LIST.
idObjetoRelated	Identificador del objeto "hijo".
tipoObjetoRelated	Podrá ser de tipo de Unidad de Control o de tipo Fichero.

TABLA	ECFM_C_FICHERO
DESCRIPCIÓN:	Descripción de los ficheros.
ATRIBUTOS	**DESCRIPCIÓN**
idFichero	Identificador de Fichero. Clave primaria.
nombreAplicacion	Nombre de la Aplicación. Referencia a ECFM_APLICACION.
nombreFichero	Nombre del Fichero.
revisionFichero	Revisión del Fichero.
fechaCreacionFichero	Fecha de Creación del Fichero.

5.5.6.6 Modelo de Descubrimiento de Datos: Tablas ECFM_GA

Se representa en este Modelo las tablas que sirven para realizar el descubrimiento de datos.

- Preguntas que se le hacen a los usuarios.
- Respuestas de los usuarios.
- Acciones a controlar por el sistema.
- Relación entre aplicaciones, acciones y preguntas a realizar.
- Tablas de Descripción y de Clasificación de Datos de Redes Neuronales.
- Tablas de Descripción de Causas y enlace con resto de Arquitectura.

TABLA	ECFM_GA_PREGUNTA
DESCRIPCIÓN:	Preguntas a Realizar al Usuario.
ATRIBUTOS	**DESCRIPCIÓN**
idPregunta	Identificador. Clave Primaria.
textoPregunta	Texto de la Pregunta.

TABLA	ECFM_GA_ACCION
DESCRIPCIÓN:	Acciones a controlar.
ATRIBUTOS	**DESCRIPCIÓN**
idAccion	Identificador de la Acción. Clave Primaria.
nombreAccion	Nombre de la Acción.
descripcionAccion	Descripción de la Acción.

TABLA	ECFM_GA_APLI_ACCION_PREGUNTA
DESCRIPCIÓN:	Relación entre Aplicación, Acción y Pregunta a Realizar.
ATRIBUTOS	**DESCRIPCIÓN**
idApliAccionPregunta	Identificador. Clave Primaria.
idPregunta	Identificador de Pregunta. Referencia a ECFM_GA_PREGUNTA.
idAccion	Identificador de la Acción. Referencia a ECFM_GA_ACCION.
nombreAplicacion	Identificador de la Aplicación. Referencia a ECFM_APLICACION.

TABLA	ECFM_GA_RESPUESTA
DESCRIPCIÓN:	Respuestas dadas a las preguntas.
ATRIBUTOS	**DESCRIPCIÓN**
idRespuesta	Identificador de la Respuesta.
idApliAccionPregunta	Identificador de la Pregunta concreta. Referencia a ECFM_GA_APLI_ACCION_PREGUNTA.
tipoObjeto	Tipo de Objeto.
idObjeto	Identificador del Objeto.
textoRespuesta	Texto de la Respuesta.
fechaRespuesta	Fecha de la Respuesta.

TABLA	ECFM_GA_ANN
DESCRIPCIÓN:	Descripción de Datos para definir Redes Neuronales.
ATRIBUTOS	**DESCRIPCIÓN**
idANN	Identificador de la ANN.
NombreANN	Nombre de la red neuronal.
nodosANN	Nodos de la red neuronal en la capa inicial.
nodosCOANN	Nodos de la capa oculta de la red neuronal.
valoresClasificacionANN	Valores posibles de clasificación de la red.
ficheroEntrenamientoANN	Fichero de Entrenamiento asociado a la red con los ejemplos iniciales.
ficheroModeloANN	Fichero que guarda el modelo de clasificación de la red una vez entrenada.
estadoANN	Estado de la red neuronal.

TABLA	ECFM_GA_PERIODO_CLASIFICACION
DESCRIPCIÓN:	Periodos de Clasificación posibles.
ATRIBUTOS	**DESCRIPCIÓN**
idPeriodoClasificacion	Identificador del Periodo de clasificación.
fechaInicioPeriodoClasificacion	Inicio del Periodo de Clasificación.
fechaFinPeriodoClasificacion	Fin del Periodo de Clasificación.

TABLA	ECFM_GA_CLASIFICACION_ANN
DESCRIPCIÓN:	Tabla para guardar los valores de clasificación generados por ANN.
ATRIBUTOS	**DESCRIPCIÓN**
idClasificacionANN	Identificador de la clasificación realizada en ANN.
idANN	Identificador de ANN para el que se hace la clasificación. Referencia a ECFM_GA_ANN.
idUsuario	Identificador de usuario para el que se hace la clasificación de ECFM_U_USUARIO.
idPeriodoClasificacion	Identificador del periodo clasificado, referencia a ECFM_GA_PERIODO_CLASIFICACION.
valorClasificacionANN	Valor obtenido por la red.

TABLA	ECFM_GA_CAUSA
DESCRIPCIÓN:	Tabla para identificar posibles causas.
ATRIBUTOS	**DESCRIPCIÓN**
idCausa	Identificador de la causa.
nombreCausa	Nombre de la causa.
descripcionCausa	Descripción de las posibles causas de error.

TABLA	ECFM_GA_CAUSA_ERROR
DESCRIPCIÓN:	Enlace entre las causas, las aplicaciones y los estados de Error.
ATRIBUTOS	**DESCRIPCIÓN**
idCausaError	Identificador de la clasificación realizada en ANN.
idCausa	Identificador de la Causa. Referencia a ECFM_GA_CAUSA.
idEstadoCV	Identificador del Estado de Error. Referencia a ECFM_C_ESTADO_CV.
nombreAplicacion	Aplicación. Referencia a ECFM_APLICACION.

5.5.7 Repositorio

El Repositorio de versiones es el módulo en el que se van a guardar cada una de las versiones generadas por los desarrolladores.

Como se ha comentado, la herramienta que se va a utilizar es CVS (Concurrent Versions System), que permite el trabajo de múltiples desarrolladores sobre un mismo repositorio de forma concurrente.

Se diseñará el sistema de manera que se puedan incluir diferentes herramientas de versionado sin ocasionar impactos sobre el resto de componentes, para lo que se independizarán las llamadas a dicha herramienta en librerías de código aisladas.

El objeto sobre el que se operará, según el modelo descrito en la sección "Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos", será el FICHERO. Las acciones principales a controlar sobre el repositorio de versiones serán las siguientes:

- *Creación de nuevos objetos en el repositorio*: Se trata de dar de alta un nuevo fichero en la herramienta de control de versiones. Este hecho significa que nos interesará controlar los cambios en dicho fichero generando sucesivas versiones asociadas a los mismos.

- *Edición para cambio de un objeto o "Check Out":* Consiste en marcar un objeto del repositorio para realizar cambios sobre él. Con esta acción, un desarrollador indica que va a hacer cambios sobre una determinada versión de un objeto, siendo esto visible para el resto del equipo de trabajo. Podrá deshacerse esta acción si finalmente el desarrollador decide no cambiar la versión.

- *Validar versión nueva en el repositorio o "Check In":* Cuando los cambios realizados sobre una nueva versión se aceptan por el desarrollador, se hará la operación de "Check In" sobre la herramienta de control de versiones, introduciendo una nueva versión al repositorio. Para incorporar información del repositorio a la base de datos de nuestro sistema, esta acción se hará desde el Módulo cliente en la pantalla de Acciones sobre el repositorio, como se puede ver en la sección "Formulario de Acciones sobre Repositorio".

- *Consulta del Histórico de Versiones***:** El hecho de realizar los cambios a través de acciones controladas por la herramienta de control de versiones, permite consultar la historia de los cambios realizados en cada uno de los ficheros lo que puede dar información relevante a la hora de descubrir datos.

Se han analizado y diseñado las diferentes partes de la solución. En la siguiente sección se estudiará el análisis y diseño de los componentes de Descubrimiento de Datos.

5.6 Análisis y Diseño de la parte de Descubrimiento de Datos

A continuación se va a detallar la parte de la aplicación asociada con el área de la Inteligencia Artificial, en concreto con las Redes Neuronales y la Minería de datos, que permitirá descubrir conocimiento desde la completa modelización de la Arquitectura de Sistemas y de la Gestión de la Configuración de los mismos que se ha realizado.

5.6.1 Arquitectura de Redes Neuronales del Sistema y Minería de Datos.

5.6.1.1 Arquitectura Funcional

Una vez definida la arquitectura del sistema en cuanto a la Gestión de la Configuración del mismo, se trata de definir una serie de mecanismos para poder descubrir conocimiento dentro de dicho sistema. Se trata del cruce entre el sistema de Gestión de la Configuración y los mecanismos de Aprendizaje Automático, en concreto de Redes Neuronales, para realizar el descubrimiento de conocimiento.

Dicha arquitectura se va a enfocar en la línea de la mejora de la calidad de los trabajos cuya configuración se mantiene en el sistema.

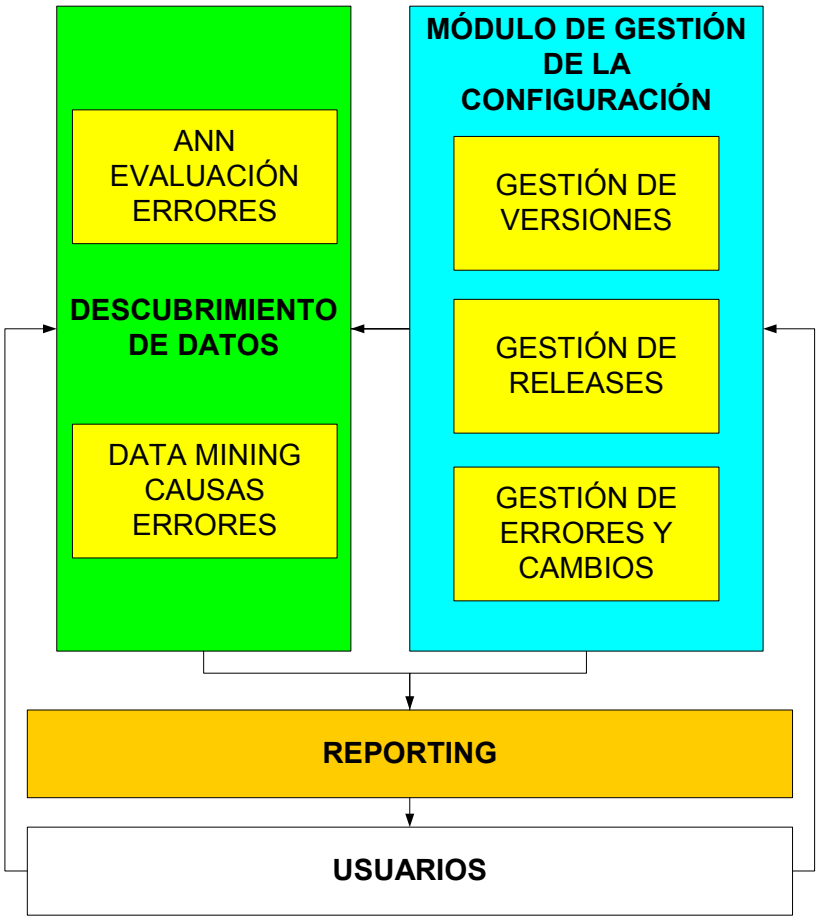


Ilustración 25 – Arquitectura Funcional de Redes Neuronales y Minería de Datos del sistema.

Se van a implementar los siguientes mecanismos de Descubrimiento de Datos:

- Red Neuronal de Clasificación de Usuarios basándose en sus Errores: Esta red será la encargada de establecer las relaciones necesarias y detectar los patrones, entre los errores cometidos por los usuarios y detectados en los procesos de GCS y su evaluación de desempeño, de manera que aprovechando la información histórica de los errores cometidos, se puede determinar el nivel general al que están trabajando los diferentes usuarios y tratar de determinar sus evaluaciones futuras.

- Data Mining de Análisis de Causas origen de los errores: Se trata de detectar cuáles son las causas principales de error que están produciéndose en los sistemas. De esta forma se pueden orientar las políticas correctivas y de formación de la compañía

aprovechando las fuentes de conocimiento históricas y el conocimiento descubierto por la red neuronal.

5.6.1.2 Definición de Componentes de Redes Neuronales.

Para definir una Red Neuronal para el Sistema de Gestión de la Configuración implementado, es necesario partir de la premisa de que esta definición tiene que ser totalmente dinámica y adecuada a las necesidades de cada una de las compañías en las que se personalice el sistema. Por tanto no habrá posiblemente dos redes neuronales iguales entre distintas compañías, al igual que los ciclos de vida o las posibles causas de error varíarán entre entidades.

El tipo de red neuronal que se manejará será un "*Multi Layer Perceptron*" (Perceptrón Multicapa), en adelante MLP explicado anteriormente en la sección "Estado de la Cuestión". Se trata de una red con las siguientes características:

- Una *Capa de Entrada con N unidades*, donde se reciben como input los datos a clasificar.

- Una *Capa Oculta con M unidades* que se utilizan para conseguir un entrenamiento más preciso y efectivo de la red y una mayor capacidad de generalización. En nuestro problema utilizaremos una sola capa oculta.

- Una *Unidad de Salida* por clase que puede tomar P valores de clasificación diferentes, que se corresponden con las posibles clases en las que la red podrá dividir los ejemplos. En nuestra implementación tendremos una única unidad de salida o clase.

La red neuronal elegida es entrenada a través de un algoritmo de Descenso de Gradiente. Es de hecho una red Back-Propagation (o Retropropagación) simple, con una sóla unidad de salida.

Para preparar la Red Neuronal es necesario realizar un *Entrenamiento*. Para realizar este entrenamiento es necesario tener:

- Una serie de *ejemplos previamente clasificados*, a partir de los cuales se podrá entrenar la red, asignando los pesos a las diferentes conexiones entre las unidades de las diferentes capas de la red. Esto permitirá posteriormente realizar la clasificación de nuevos ejemplos. Normalmente tendremos un Fichero de Ejemplos.

- La *definición de la topología de la red,* definiendo las diferentes capas y los posibles parámetros de las mismas. Normalmente se guarda en un Fichero de Modelo, en el que están definidas todas las características de la red. Los datos de definición de la topología son:

    - *Número de nodos de la Capa de Entrada*, o número de inputs de cada ejemplo de clasificación.

    - *Número de nodos de la Capa Oculta*, o número de elementos de las capas intermedias.

    - *Valores de clasificación diferentes de la Capa de Salida*, o número de clases en las que puede la red dividir o agrupar los ejemplos.

- Otros parámetros como las *tasas de aprendizaje*, los *umbrales de error* o el número de *iteraciones* necesario para ajustar el entrenamiento de la red.

En el siguiente diagrama se muestra una Red Neuronal de tipo MLP, en la que se definen las diferentes capas. Se puede observar que existirá un Fichero de Ejemplos, con los ejemplos previamente clasificados para hacer el entrenamiento, y un Fichero de Modelo en el que se almacenará la red ya entrenada y preparada para realizar la evaluación de nuevos ejemplos.

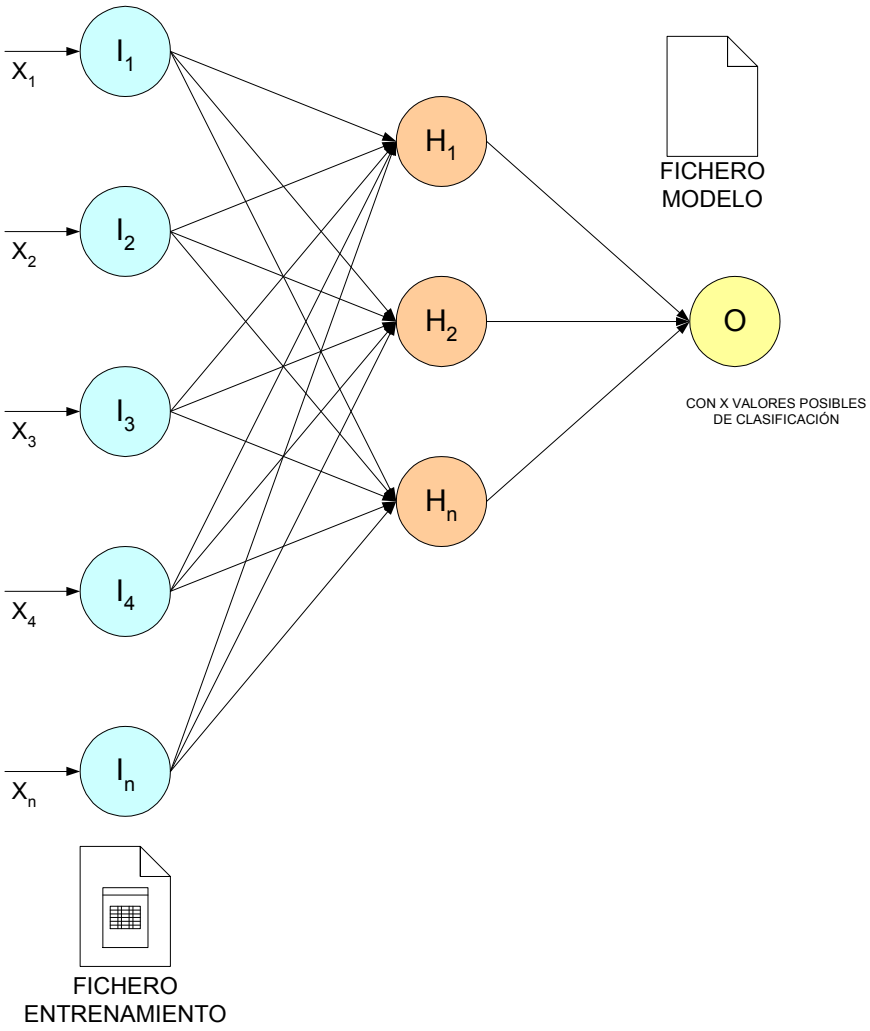


Ilustración 26 – Componentes de Redes Neuronales del Sistema

5.6.1.3 Integración con la Solución

En los diferentes componentes de la aplicación, la inclusión de redes neuronales tiene una serie de impactos que no se han detallado con anterioridad ya que se ha orientado el trabajo como un acoplamiento del Descubrimiento de Datos a la modelización completa del Sistema de Gestión de la Configuración.

En el siguiente diagrama se pueden observar los diferentes componentes de la solución que se ven afectados por la inclusión de Redes Neuronales.

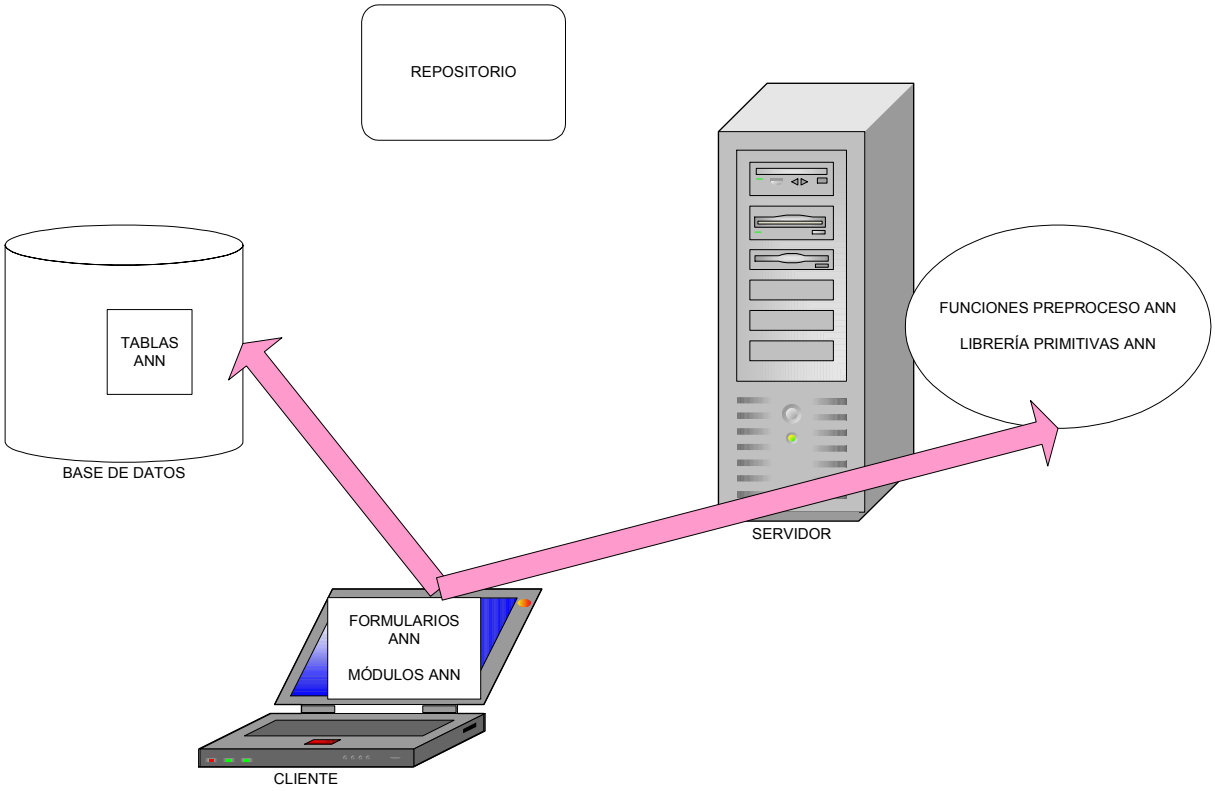


Ilustración 27 – Integración de Redes Neuronales en la Solución

5.6.1.3.1 Cliente.

En el cliente de la aplicación aparecerán *nuevos formularios* para el manejo de las redes neuronales y para utilizarlas, así como para realizar la minería de datos correspondiente. Deberán tener en cuenta los diferentes parámetros de las redes, que se han comentado anteriormente.

Asimismo deberá realizar las comunicaciones debidas con el servidor para que sean los servidores los que realicen el entrenamiento de las redes. Se incluirá por tanto un nuevo módulo específico para el *manejo de redes Neuronales* y *funciones de comunicación* con el Servidor.

También aparecerán *nuevos informes* para facilitar al usuario final el aprendizaje y el nuevo conocimiento que aportan las Redes Neuronales al sistema y la minería de datos a realizar.

5.6.1.3.2 Servidor

En el servidor se incluyen todas las funciones necesarias para el manejo de Redes Neuronales y la realización de cálculos. Nótese que en el problema planteado se trabaja con grandes volúmenes de información y que las capacidades de proceso requeridas son normalmente altas, con lo que conviene trabajar directamente en servidores que realicen las operaciones más pesadas. Asimismo se incluyen las funciones necesarias para la extracción de informes sobre la base de datos para la realización de minería de datos.

A su vez, en el servidor se han creado dos capas para atender a las peticiones asociadas con ANN.

- *Funciones de Preproceso y preparación de datos* (en el diagrama "Funciones Preproceso ANN") que se encargarán de procesar las informaciones que le llegan del cliente y adecuarlas a los formatos necesarios para el manejo de Redes Neuronales.

- *Librería de Funciones de Redes Neuronales* (en el diagrama "Librería Primitivas ANN") que es un API de funciones para el manejo de Redes Neuronales. Se ha adecuado para este problema la librería "Torch" descrita en la sección "Librerías de Uso de Redes Neuronales". Esta librería tiene implementadas primitivas de problemas de aprendizaje automático. Se ha trabajado como base con este conjunto de funciones, adecuándolo a la solución.

5.6.1.3.3 Base de Datos

Se ha incluido un modelo de datos en la sección que permite guardar la información necesaria de la *topología de las redes neuronales* de la aplicación.

Asimismo se permite guardar las *clasificaciones realizadas por la red*, de manera que se puedan generar informes posteriores sobre la adquisición de nuevo conocimiento.

Se han incluido también una serie de tablas para el *modelado de causas y razones que provocan los errores,* sobre las que se fundamentará la minería de datos del sistema. Estas

tablas se han descrito previamente en la sección "Modelo de Descubrimiento de Datos: Tablas ECFM_GA"

5.6.1.3.4 Repositorio

El componente de Repositorio *permanece inalterado*, es decir que no conlleva cambios debidos al Descubrimiento de Datos, ya que su función es el registro de versiones de objetos, si bien la funcionalidad de preguntar las causas de los cambios que se ha incorporado en el cliente está en total relación con la posterior detección de causas que se hará con minería de datos.

5.6.2 Red Neuronal de Clasificación de Usuarios basándose en sus Errores

Se va a analizar a continuación la Red Neuronal de Clasificación de Usuarios que se van a incluir en la aplicación. El objetivo de esta red será obtener algún tipo de clasificación de los usuarios del sistema basándose en los datos que tiene el sistema respecto a errores cometidos por el usuario.

5.6.2.1 Definición de Red Neuronal de Errores de Usuario

Como se ha comentado, es necesario determinar la topología de la red Neuronal como aspecto clave del diseño de la misma. Se va a detallar a continuación en la elección de dicha topología que debe encajar con el modelo de sistema de Gestión de la Configuración, por tanto debe de ser dinámica y diferente para cada compañía.

5.6.2.1.1 Número de Nodos de la Capa de Entrada.

El *número de unidades de la Capa de Entrada vendrá determinado por los ciclos de vida elegidos por cada compañía*, y dentro de esos ciclos de vida, por la *definición de estados*

erróneos. Se define un estado de Error, en términos de definición de ciclo de vida para el Sistema de Gestión de Configuración implementado, como aquel que:

- Es un *estado "Especial",* es decir que tiene el flag "especialEstadoCV" a 1 en la tabla ECFM_C_ESTADO_CV, como se ha indicado en la sección "Modelo de Datos".

- Tiene *"ERROR*"* en su propio nombre, ya que puede haber estados "Especiales" no erróneos.

Para poder controlar los *errores de un usuario* se van a tener en cuenta los siguientes factores por cada error:

- *Número de errores de cada tipo que tiene registrados cada usuario*. Nótese que un usuario pase por un error determinado no necesariamente quiere decir que él lo haya cometido. Por ejemplo, puede ser que sea un usuario encargado de hacer pruebas, y que cuando pase por un error quiere decir que está declarando un error en los desarrollos.

- *Número total de iteraciones en las que se ha pasado por el estado previo al error*. Es decir, el número de veces que el usuario ha podido elegir entre saltar al error y saltar al estado siguiente en el ciclo de vida. Esto nos permitirá obtener la tasa de error:

    - Número de Errores / Número Total de Pasos por iteración de error (*100)

Además de estas unidades de entrada se van a computar los totales por usuario que resultan de sumar el Total de Errores independientemente de su estado y de sumar el Total de Interacciones realizadas por el usuario que podrían resultar de error, de manera que se puede obtener la Tasa global de Error de un determinado usuario.

Por tanto el *total de unidades de entrada de la red* será:

- *2 * Número de Estados ERROR, UNIDAD DE CONTROL y APLICACIÓN* de los diferentes ciclos de vida. Nótese que cuando se referencia el error, se habla de ERROR en una unidad de control. Por ejemplo, a efectos de la red neuronal, serán errores diferentes el ERROR_1 de la unidad de control PACK que el ERROR_1 de la unidad de control REQ.

- *2 adicionales* para contar el total de errores y la tasa global de error.

5.6.2.1.2 Número de Nodos de la Capa Oculta.

El *número de Nodos de la Capa Oculta se determinará por defecto como el mismo número de Nodos de la Capa de Entrada*. Si bien este número debe variarse para buscar una mejor capacidad de generalización de la red. Al ser una red dinámica dependiente de la modelización del sistema para cada compañía, no es una función trivial de maximizar con lo que se tendrá que probar con diferentes valores y estudiar cada caso para determinar el valor final. En la sección "Líneas futuras de Trabajo" se menciona una posible investigación en esta línea.

5.6.2.1.3 Número de Nodos de la Capa de Salida

El número de nodos de la capa de salida será 1, cuyos valores posibles serán los valores en los que *cada compañía quiera evaluar a los usuarios del sistema*. Por defecto se sugiere de 1 a 10, pero cada compañía puede tener su escala de valores. Es necesario que el fichero de entrenamiento inicial de la red, tenga en cuenta estos posibles valores de clasificación para realizar correctamente la separación de los ejemplos. Estos valores serán discretos y redondeados al entero más cercano.

5.6.2.1.4 Iteraciones necesarias para el entrenamiento.

Se *definirán por defecto un número de 1000 iteraciones del fichero de ejemplos* para entrenar la red. Según la dimensión de este fichero se debe ajustar este número de iteraciones a cada problema para no generar problemas de sobreajuste, y evitar así que la capacidad de generalizar de la red quede limitada.

5.6.2.1.5 Fichero de Entrenamiento.

Se debe definir un *fichero de entrenamiento con ejemplos previamente clasificados*. Para dichos ejemplos se incluirán un número de datos igual al número de nodos de la capa de entrada, siguiendo los criterios que se han analizado anteriormente. Se incluirá una columna final en la que se indicará la clasificación del usuario que ha tenido esos datos. Se obtendrá este fichero de históricos de la compañía.

5.6.2.1.6 Fichero de Modelo.

Finalmente, se definirá el *fichero donde se quieren guardar los datos de la red neuronal relativos a la topología y a los resultados de los entrenamientos realizados*, guardando la información de los pesos asignados a cada una de las conexiones. Este fichero será dependiente de la librería final elegida para el trabajo con redes neuronales.

5.6.2.2 Componentes de Red Neuronal de Errores de Usuario

A continuación se detallan los componentes de aplicación que tiene la Red Neuronal de Usuario. Nótese que esta Red Neuronal se encargará de dar una clasificación determinada a cada usuario dependiendo de los datos registrados en el sistema de los errores generados en el ciclo de vida.

5.6.2.2.1 Cliente

En la parte Cliente de la aplicación, la red Neuronal de Errores de usuario implicará los siguientes cambios:

- Realización de un *Formulario* de manejo de la Red Neuronal de Errores de Usuario.
- Realización de *Informe o de Informes* adecuados para la obtención y presentación de resultados.

5.6.2.2.1.1 Formulario de manejo de la Red Neuronal de Errores de Usuario.

A continuación se revisa detalladamente el formulario cliente de manejo de la red neuronal de usuario.

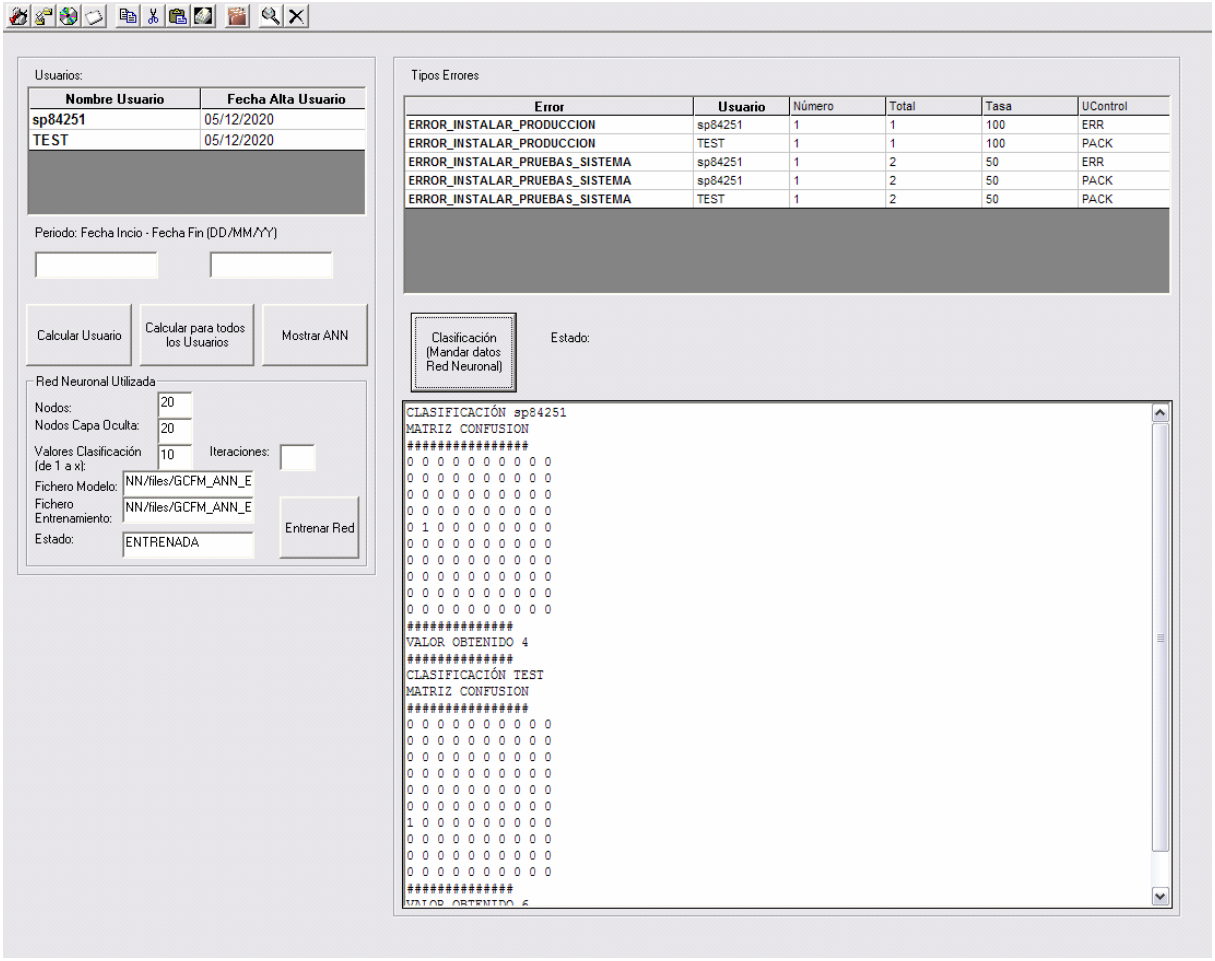


Ilustración 28 – Formulario de Manejo de Red Neuronal de Errores de Usuario

En el formulario se pueden diferenciar las siguientes funciones habilitadas para el manejo de la Red Neuronal de Errores de Usuario:

- *Selección de Usuario o Usuarios de Trabajo*: Se habilita un panel en el que se debe indicar qué usuario o usuarios y en qué periodo de tiempo se debe evaluar. Nótese que un mismo usuario en dos periodos de tiempo diferentes tendrá normalmente evaluaciones diferentes.

- *Entrenamiento de Red Neuronal de Errores*: Se dispone de un panel desde el que se definirán los parámetros configurables de la red neuronal de errores. Estos son: El número de nodos de la capa oculta, el número de valores posibles de clasificación, el número de iteraciones y los ficheros de modelo y de entrenamiento de la red. Se permite desde este panel entrenar la red sucesivas veces.

- *Tipificación y cómputo de errores*: Para los usuarios y el periodo de tiempo seleccionados, se realizará la cuenta de los errores cometidos, de los totales de iteraciones y de las tasas de cada tipo de error. Esta información se obtiene de realizar consultas directas a la base de datos.

- *Clasificación de datos*: Una vez preparados los datos de un determinado usuario para un periodo de tiempo, se pueden mandar a los servidores para realizar su entrenamiento. Se mandarán los datos de cada uno de los usuarios obteniendo una evaluación que será registrada en la base de datos a efectos de realizar informes futuros.

En el fichero de Configuración del cliente, se definirán los parámetros ANN_SERVER y PUERTO_ANN_SERVER para determinar la dirección y el puerto del servidor donde residan las redes neuronales.

5.6.2.2.1.2 Informes de la Red Neuronal de Errores de Usuario.

El informe que se habilitará para poder hacer accesible la información de la red neuronal es un *informe de evaluación de usuarios, donde se confrontan los periodos de evaluación de cada usuario y sus diferentes evaluaciones viendo su progreso por la organización*.

Para ello, es necesario incluir en la base de datos un nuevo informe que saque información de las tablas de clasificación de datos, y posteriormente definir una plantilla en la que se genere dicho informe y se presenten los resultados.

A continuación se muestra el diagrama de la pantalla de informes, donde se pueden observar los filtros que se pueden aplicar para sacar únicamente la información que se necesite.

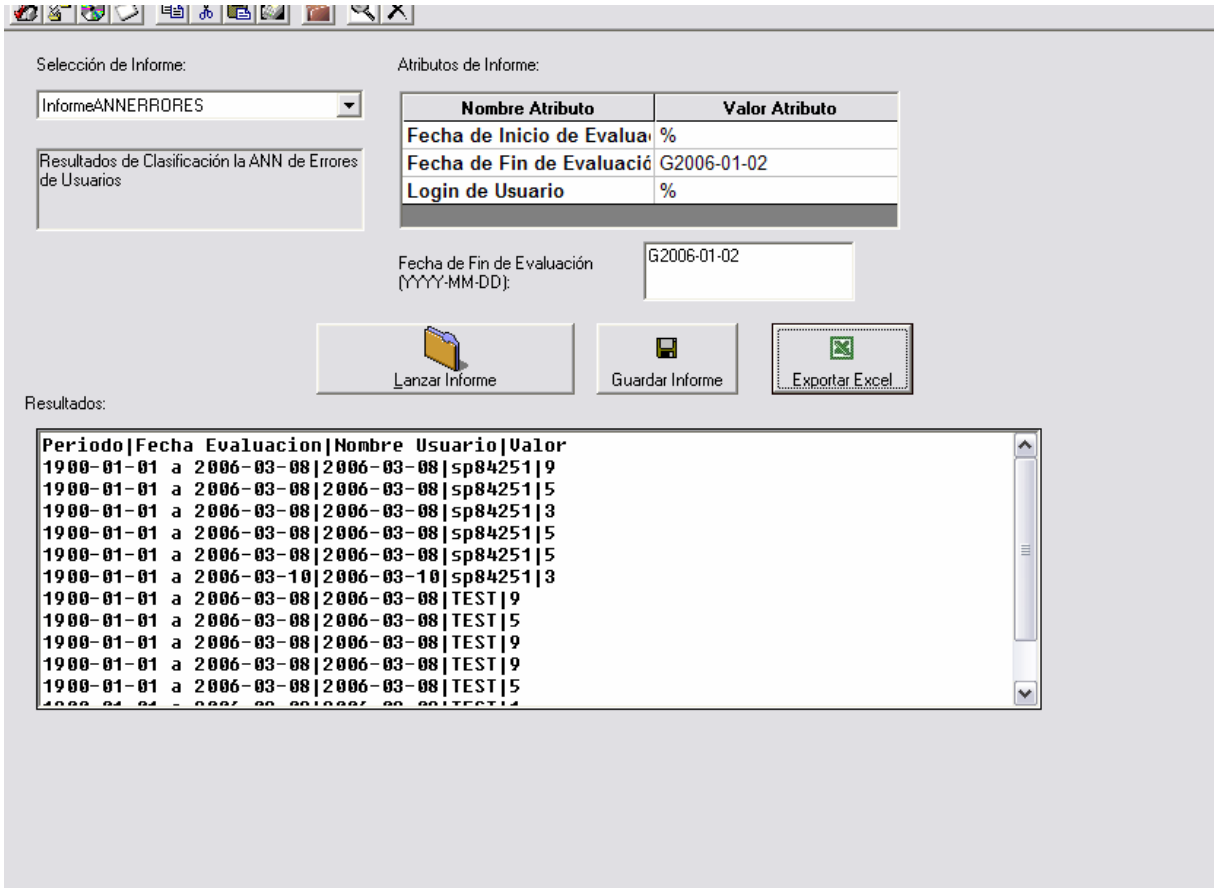


Ilustración 29 – Pantalla con el Informe de Red Neuronal de Errores de Usuario

A partir de dicho formulario se puede generar una forma de presentarlo de forma inteligible y visual. Para ello se ha aprovechado la integración del módulo de informes con plantillas y macros de Microsoft Excel. Así, se genera dinámicamente un *gráfico que confronta los usuarios frente al periodo en el que han sido evaluados*. Se muestra a continuación un ejemplo de dicho gráfico.

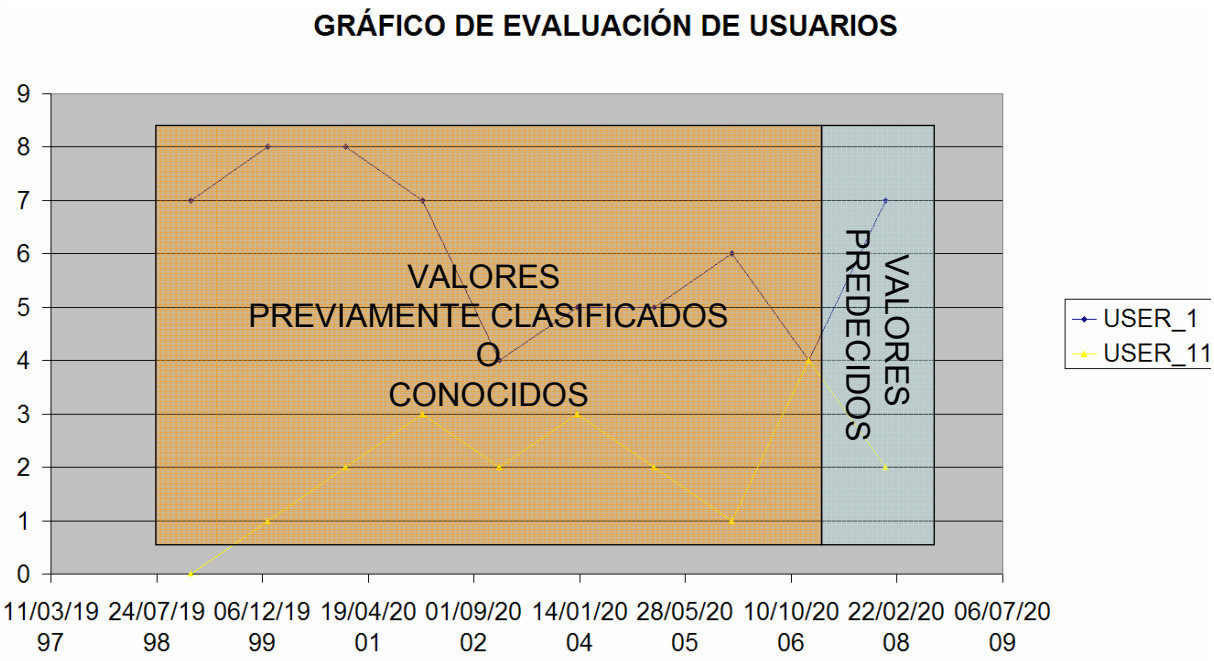


Ilustración 30 – Gráfico Generado para Informe de Red Neuronal de Errores de Usuario

5.6.2.2.2 Servidor

La parte Servidor de la aplicación, como se comentó anteriormente, tiene una rutina de atención de peticiones y preparación de datos, y otra parte de ejecución de cálculos utilizando las librerías de Redes Neuronales elegidas. Asimismo devolverá los resultados de los cálculos realizados una vez hayan sido procesados.

5.6.2.2.2.1 Preparación de Datos

Siguiendo el protocolo que se analizó en la sección “Comunicación entre Cliente y Servidor”, se envía la información del formulario de cliente a una rutina, que a su vez tiene una librería asociada.

La rutina recibe como parámetro la acción a realizar y las variables necesarias para esa acción. En este caso las acciones pueden ser ENTRENAR o CLASIFICAR.

A continuación se disponen ejemplos de llamada a la rutina para dichas acciones.

ENTRENAMIENTO:

```
ECFM_EjecutarNN.ksh ENTRENAR
'NOMBRE_ANN!ANN_ERRORES^NODOS_ANN!20^NODOS_CO!20^VALORES_CLASIFICACION!10^F
ICHERO_ENTRENAMIENTO!NN/files/ECFM_ANN_ERRORES.txt^FICHERO_MODELO!NN/files/
ECFM_ANN_ERRORES_model.txt^ITERACIONES!1000'
```

La llamada para entrenamiento consiste en indicar los parámetros topológicos básicos de la red y los ficheros de entrenamiento y modelo a utilizar. Nótese que debe existir el fichero de Entrenamiento que contendrá los ejemplos previamente clasificados.

CLASIFICACIÓN:

```
ECFM_EjecutarNN.ksh CLASIFICAR
'FICHERO_MODELO!NN/files/ECFM_ANN_ERRORES_model.txt^VALORES_CLASIFICACION!1
0' 0#0#0#0#0#0#1#100#1#50#0#0#0#0#0#0#0#0#2#40
```

La llamada para clasificación indica el fichero de modelo de la red, previamente entrenado, así como el ejemplo que se va a clasificar.

5.6.2.2.2.2 Utilización de la librería de Redes neuronales

Una vez preparados los datos se realizarán las llamadas a la API de manejo redes neuronales. Se trata de la librería "Torch" como se ha visto anteriormente. Consiste en un conjunto de funciones implementadas en C++ para manejo de técnicas de aprendizaje automático. Se ha preparado un módulo principal (llamado ML) para acceder a las funciones de la librería cómodamente desde las rutinas de preparación de datos.

Se pueden ver a continuación los ejemplos de llamadas a la librería para la clasificación y para el entrenamiento de redes neuronales.

ENTRENAMIENTO:

```
ML "$FICHERO_ENTRENAMIENTO" "$NODOS_ANN" "$VALORES_CLASIFICACION" -nhu
"$NODOS_CO" -save "$FICHERO_MODELO" -iter "$ITERACIONES"
```

Para entrenar la red se pasará el fichero de entrenamiento, el número de nodos, los valores posibles de clasificación, el número de unidades de la capa oculta, el fichero donde se guardará el modelo y el número de iteraciones.

El fichero de entrenamiento tendrá el formato:

```
NUMERO_EJEMPLOS NÚMERO_DE_CLASES+1

VALORES DEL EJEMPLO 1 SEGUIDOS DE VALOR DE CLASIFICACIÓN 1

VALORES DEL EJEMPLO 2 SEGUIDOS DE VALOR DE CLASIFICACIÓN 2

….
```

Los valores estarán separados por espacio.

CLASIFICACIÓN:

```
ML --test "$FICHERO_MODELO" tmp/fichClasifNN.tmp
```

Para realizar la clasificación de datos hay que indicar el fichero de modelo ya entrenado asociado a la red y un fichero con la información a clasificar, que estará en la forma siguiente:

```
1 NUMERO_DE_CLASES+1

VALORES SEPARADOS POR ESPACIO SEGUIDOS DE 0
```

5.6.2.2.2.3 Resultados de Entrenamiento y de Clasificación. Matrices de Confusión.

Cuando concluyen los cálculos se dan unos resultados. Las rutinas de servidor procesan esa información y la preparan para su envío al cliente. Se detalla a continuación cómo se presentan esos resultados y cómo interpretarlos.

ENTRENAMIENTO

```
FICHERO DE ENTRENAMIENTO

########################
```

```
NUMERO DE EJEMPLOS:  110

NUMERO DE VALORES:  20

NUMERO DE CLASES EN EL FICHERO:  10

PRIMERA CLASE:  0

ULTIMA CLASE:  9

MATRIZ CONFUSION

################

1 0 2 2 0 2 3 3 1 0

0 10 0 0 0 0 0 0 0 0

0 0 1 2 1 1 1 1 2 3

0 0 0 0 3 1 0 0 0 1

4 0 1 2 0 1 3 2 1 2

1 0 1 2 2 1 0 0 2 1

0 0 0 1 0 0 0 1 1 0

3 0 4 0 3 2 2 3 2 3

0 0 1 0 0 0 0 0 0 0

1 0 0 1 1 2 1 0 1 0

CLASIFICACIÓN

##############

VALOR OBTENIDO 5

##############
```

Para el entrenamiento se analiza que el fichero de entrada es correcto, y posteriormente se realiza dicho entrenamiento. El resultado se muestra con una matriz de confusión tal y como se explicó en la sección “Personalizaciones de TORCH”.

A la hora de realizar la clasificación, se obtiene el valor que devuelve la red para un determinado ejemplo. En este caso la matriz de confusión no será representativa ya que el ejemplo no está previamente clasificado con lo que no se tiene un valor previsto de la red. La matriz de confusión asociada al anterior ejemplo sería la siguiente:

```
MATRIZ CONFUSION

################

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
```

Como el valor de clasificación que se ha indicado en el fichero de ejemplo preparado ha sido 0, nos indica que se obtiene 5 (fila), con un valor previsto de 0 (columna). Este valor obtenido deberá ser registrado en la base de datos, ya que representa lo que la Red neuronal ha obtenido como valor de clasificación para un determinado periodo.

5.6.3 Minería de Datos sobre Análisis de Causas Origen de los Errores

Una de las aportaciones principales de este proyecto es la posibilidad de analizar, a partir de información relacionada con la gestión de la configuración del sistema, las causas que determinan los comportamientos erróneos del sistema. Para ello se va a hacer una minería de

datos sobre las causas y sobre los resultados de la clasificación realizada con las redes neuronales. Se utilizarán informes para poder analizar los resultados a través de la Visualización de los mismos.

5.6.3.1 Definición de Funciones de Minería de Datos para Análisis de Causas

Se definen a continuación las funciones que se han implementado en el sistema para poder realizar un análisis de causas de los errores adecuados. Nótese que las causas de los errores son dinámicas y dependen de múltiples factores tales como la aplicación sobre la que se trabaja o el tipo de error generado.

5.6.3.1.1 Entidades Implicadas

A la hora de determinar las causas de error se tienen las siguientes entidades implicadas:

- *Causas posibles*: Se tendrán que definir y tipificar todas las posibles causas de error.

- *Aplicación*: Dependiendo del sistema sobre el que se esté trabajando las causas pueden ser unas u otras. Por ejemplo, si se trabaja con un sistema de integración habrá errores relacionados con mensajes construidos o con conversiones de datos, errores que pueden no ser tan comunes por ejemplo en sistemas de atención de usuarios.

- *Unidad de Control*: Según la tipificación de unidades de control del sistema que se tenga, las causas pueden ser unas u otras. Por ejemplo, las causas de error de un requerimiento (REQ) podrían estar relacionadas con la toma del propio requerimiento o el análisis del mismo, si bien las causas de error producidas en un paquete de instalación (PACK) estarían más relacionadas con la codificación o la instalación.

- *Estado de Error*: El propio estado de error que se produzca puede tener diferentes causas. Por ejemplo, no serán las mismas las causas para un estado de error de instalación, que normalmente estarán relacionadas con la calidad del paquete de instalación o con el

procedimiento de dicha instalación, que las de un estado de error de pruebas funcionales, cuyas causas serán normalmente derivadas de problemas de implementación.

Como se puede ver el modelo de causas posibles tiene múltiples entidades dependientes y a priori no es sencillo. En la sección "Modelo de Datos" se puede ver en el diagrama de base de datos las implicaciones que tiene sobre las entidades afectadas.

5.6.3.1.2 Proceso de Registro de la información para detección de causas y posterior análisis

El proceso de registro de la información de causas que provocan errores, va ligado al proceso de cambio de estado de una unidad de control. Estos serían los pasos para registrar dichas causas:

- Ir al *formulario de cambio de estado de unidades de control*, para una determinada aplicación.
- *Detectar si se está cambiando de estado a un error*, comprobando el estado origen y el estado destino.
- En caso afirmativo, *extraer las posibles causas de error* para este estado de error, para esta unidad de control y para esta aplicación.
- *Registrar en la tabla de históricos* (ECFM_C_UCONTROL_LIST_HIST) *la causa* de error que ha provocado este cambio de estado.

Por tanto, la información final de las causas de error quedará registrada en la tabla de históricos de estados de las unidades de control que será por tanto una de las claves del análisis a realizar.

5.6.3.2 Componentes para la Minería de Datos sobre Análisis de Causas Origen de los Errores.

Se indican a continuación los componentes introducidos en el sistema para la realización de la minería de datos, que tendrá forma final de visualización de datos. El objetivo es que dicha visualización ayude en la detección real de las fuentes de los errores para ayudar en las correcciones futuras.

5.6.3.2.1 Cliente

En el cliente se han incluido los siguientes componentes:

- Formulario de registro de Causas de Errores.
- Formulario de Análisis de Causas y Relaciones con otras entidades.
- Informes.

A continuación se van a detallar cada uno de los componentes mencionados.

5.6.3.2.1.1 Formulario de Registro de Causas de Errores.

Como se ha indicado en la sección "Proceso de Registro de la información para detección de causas y posterior análisis", se trata de un proceso insertado dentro del proceso general de cambios de estado de unidades de control.

Por tanto dentro del formulario de cambio de estado, se debe detectar si el estado final es un estado de error (Comienza por "ERROR" y está marcado como "Estado Especial"), así como si el estado origen puede derivar en ese error. En la siguiente figura se puede ver un ejemplo de estado que puede provocar un error, ya que está en estado pendiente de instalar en un entorno.

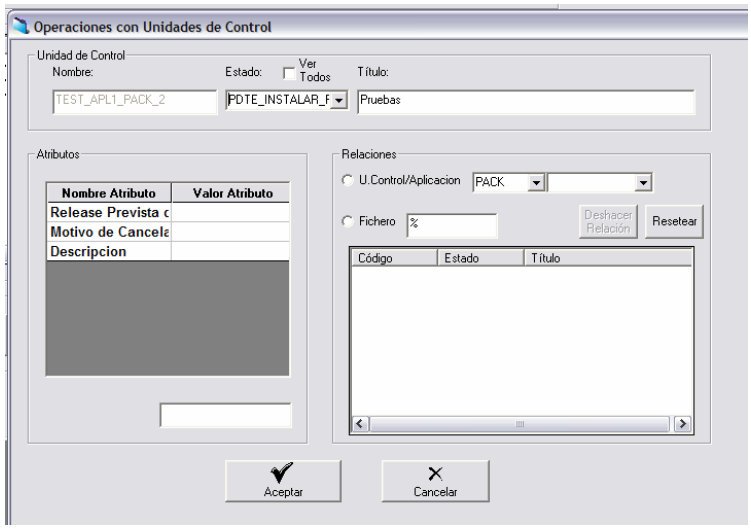


Ilustración 31 – Pantalla de Operaciones con Unidad de Control – Cambio de Estado

Si el estado final es un estado de error, además del proceso normal de cambio de estado, se incluirá un cuadro para seleccionar la causa que ha provocado el error, que dependerá de la unidad de control, de la aplicación y del estado de error. Se puede ver en la siguiente figura, donde el ejemplo anterior ha saltado a un estado de error.

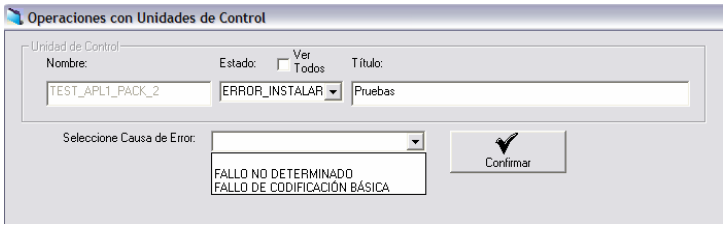


Ilustración 32 – Cambio de estado a Error. Detección de las causas que lo han provocado.

La información seleccionada será insertada en el histórico de causas para su posterior análisis.

5.6.3.2.1.2 Formulario de Análisis de Causas, y Relaciones con otras entidades del proceso de Gestión de Errores.

En la sección de "Entidades Implicadas" en las funciones de minería de datos, se comentaban las entidades implicadas en el proceso. El hecho de definir las causas con las dependencias de tantas entidades es un proceso muy dinámico. Se ha implementado un formulario en el que se pueden cruzar las diferentes entidades del sistema para la definición de causas de error relacionadas, y se pueden definir dinámicamente sus vínculos. Se muestra en el siguiente diagrama.

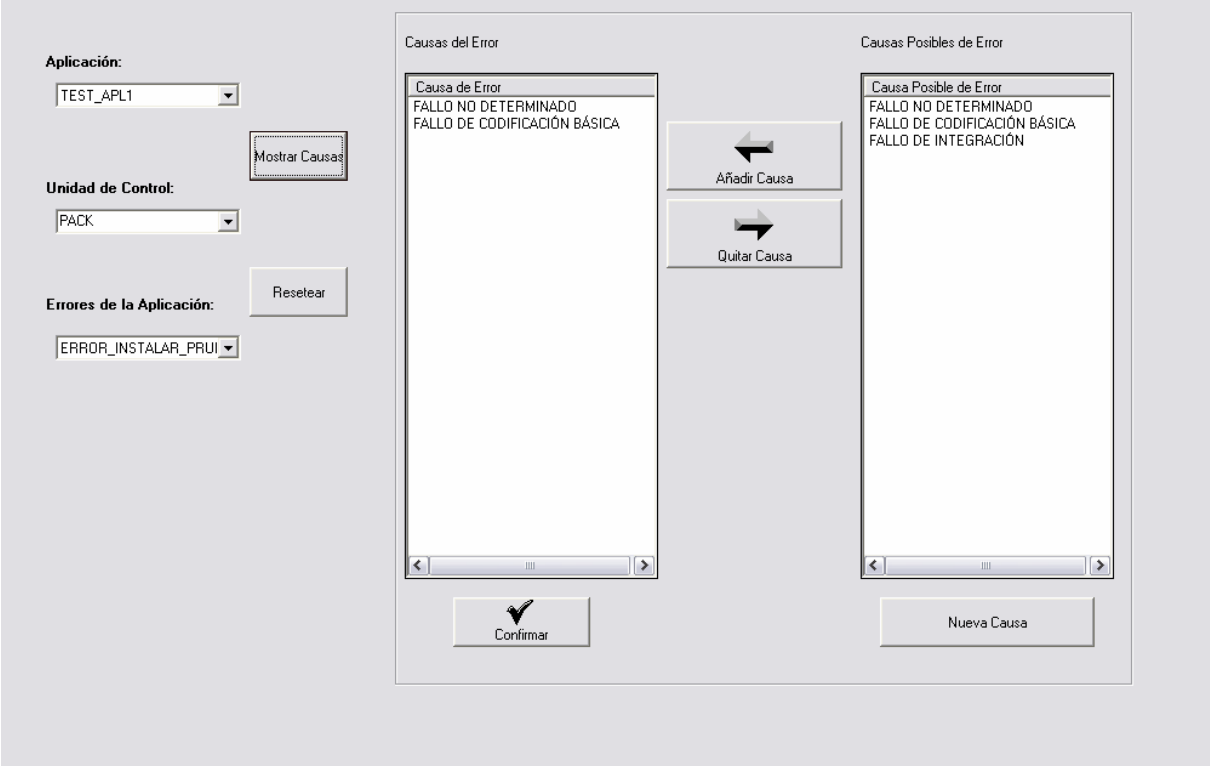


Ilustración 33 – Formulario de Análisis de Causas de Errores.

En el formulario de análisis de causas de errores, a partir de una aplicación y una unidad de control, se muestran los posibles estados de error. Si se pulsa el botón "Mostrar Causas", se mostrarán las causas posibles de error. Se pueden añadir nuevas causas del cuadro global de Causas Posibles. Asimismo se pueden definir nuevas Causas Posibles del Error seleccionado en concreto.

5.6.3.2.1.3 Visualización de Análisis de Causas. Informes.

Para la obtención de resultados, se obtendrán gráficos en tres dimensiones cruzando múltiples variables. La técnica de Minería de Datos será por tanto la Visualización, a través de informes dinámicos. Se cruza la información registrada de Causas de error, con la información generada por la red neuronal de análisis de error, relativa a las clasificaciones de datos.

Se trata de *encontrar las causas de error que provocan las evaluaciones más bajas o más altas*, comparando con diferentes variables para obtener diferentes visiones de los factores que hay que corregir o potenciar en cada caso en la compañía.

La solución generará un informe dinámicamente que muestra diferentes vistas de las causas de error:

- *Datos en modo Fila*, obtenidos directamente de la base de datos.
- *Causas que provocan las evaluaciones más altas.*
- *Causas que provocan las evaluaciones más bajas.*
- *Aplicaciones y causas que provocan las evaluaciones más bajas.*
- *Aplicaciones y causas que provocan las evaluaciones más altas.*
- *Usuarios y causas que provocan las evaluaciones más bajas.*
- *Usuarios y causas que provocan las evaluaciones más altas.*
- *Estados de Error y causas que provocan las evaluaciones más bajas.*
- *Estados de Error y causas que provocan las evaluaciones más altas.*

En cada uno de los gráficos a visualizar se pueden observar en los ejes:

- *Evaluaciones obtenidas*.
- *Variables medidas*, como por ejemplo Aplicaciones y Causas, o Errores y Causas.
- *Número de Ocurrencias*.

La variable “Periodo” en el que se mide servirá como filtro global al trabajo.

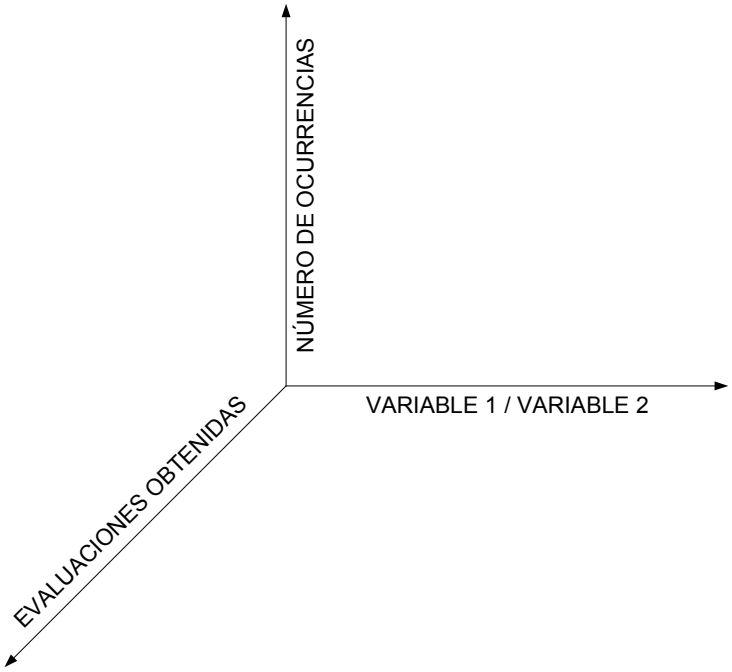


Ilustración 34 – Formato de los Gráficos de Análisis de Causas.

A continuación se muestra un gráfico en el que se miden los usuarios que han tenido las evaluaciones más altas y las causas generales de errores cometidos.

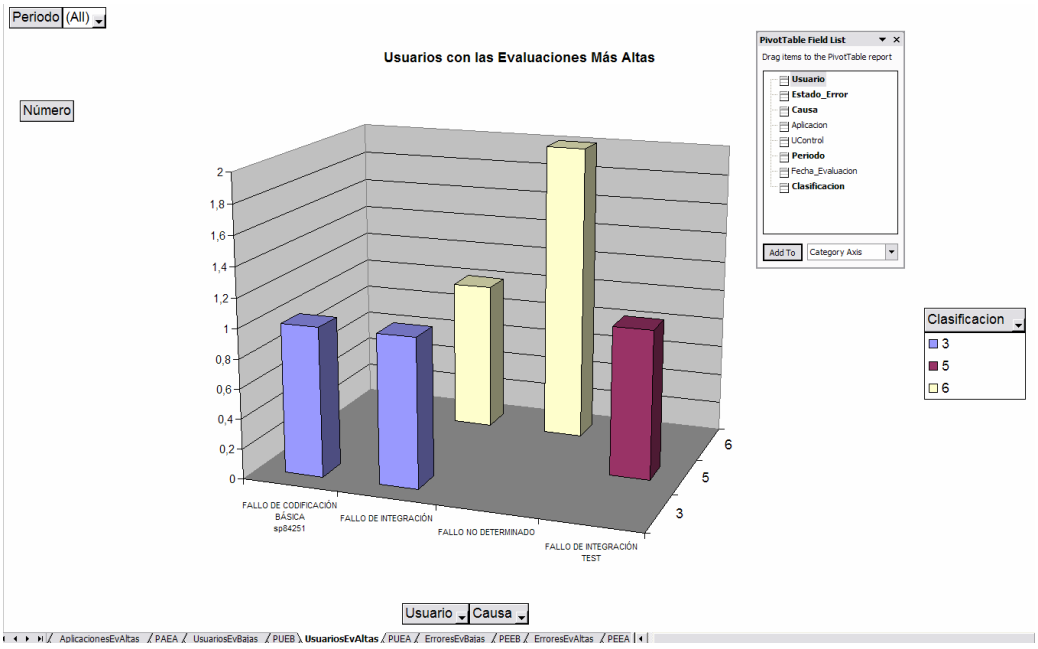


Ilustración 35 – Gráfico de Minería de Datos de Análisis de Causas.

Se puede ver en el ejemplo, que las tres evaluaciones más altas son 3, 5 y 6 respectivamente. En el eje de las abscisas se muestran los usuarios que tienen registradas estas y las causas que han tenido en los periodos evaluados también aparecen filtradas, de manera que por ejemplo, el usuario "TEST" ha tenido un error con causa "FALLO DE INTEGRACIÓN" en un periodo en el que ha tenido evaluación 5 (Se muestra el login del usuario bajo el nombre de la causa).

Se ha planteado como un informe dinámico, en el que añadir o quitar variables y datos para el análisis (se cogerían con "Drag & Drop" del cuadro "Field List") resulta muy sencillo.

5.6.3.2.2 Servidor. Librería de Consulta y Obtención de Informes de Causas.

En el servidor se han incluido las funciones necesarias para la consulta que permite obtener los datos para la realización de informes, cruzando las entidades de causas, con la información histórica y con la información de evaluaciones de los diferentes usuarios.

En la librería de funciones de informes, se ha incluido la función *InformeCausasErrores()*, con el informe que realiza el cruce de todas las entidades.

6 Análisis de Viabilidad

En esta sección se va a realizar un análisis de la viabilidad del sistema propuesto. Para ello se van a realizar dos tipos de pruebas: Las validaciones básicas y las validaciones sobre un modelo predictivo.

Con las validaciones básicas del sistema se va a mostrar la viabilidad de cada uno de los módulos y su funcionalidad.

El modelo predictivo consistirá en mostrar las conclusiones más relevantes de este trabajo de investigación: tratar de detectar los comportamientos de los usuarios basándose en la información de la gestión de la configuración y ser capaces de identificar las causas y los puntos de error que ocasionan las evaluaciones más bajas, por tanto los puntos a corregir para obtener mejores resultados en la compañía.

Corrección y Completitud de las Pruebas.

La *elección de estas pruebas se considera suficiente para la validación de la solución y con ella las hipótesis asociadas a este trabajo*.

Por un lado, un conjunto de pruebas de detalle de sistema asegura que las partes de la solución funcionan. Al ser un conjunto exhaustivo se va a asegurar la funcionalidad y utilidad de cada una de dichas partes. Así se asegura la usabilidad del sistema.

Las pruebas realizadas con el modelo van encaminadas a demostrar las hipótesis de este trabajo. Un modelo general que sea flexible y que responda a una situación que podría ser real en una determinada organización es necesario para validar la solución en su conjunto. Asimismo es necesaria la generación de un volumen grande de datos con aleatoriedad en los que se puedan encontrar ciertas tendencias, tal y como se va a hacer. Dichas tendencias deben ser conocidas a priori para comprobar la capacidad de clasificación y el comportamiento acorde a ellas de la red.

6.1 Validaciones Básicas del Sistema

Se han realizado una serie de validaciones básicas de los diferentes componentes del sistema, de las interfaces de comunicación, así como de los protocolos de funcionamiento establecidos con una batería de pruebas de sistema sobre los diferentes módulos del sistema.

Dichas pruebas están organizadas por módulos:

- Módulo de Instalación
- Módulo de Acciones sobre Repositorio
- Módulo de Control del Ciclo de Vida
- Módulo de Generación de Informes
- Módulo de Comunicación Off-line
- Módulo de Red Neuronal de Clasificación de Errores
- Módulo de Análisis de Causas de Error

El detalle de estas pruebas ha sido incluido en el anexo "Detalle de Pruebas de Sistema Realizadas" de este documento.

De forma resumida, las pruebas de detalle de sistema han sido superadas satisfactoriamente para cada uno de los módulos, garantizándose así que aportan cada uno de ellos la funcionalidad para la que han sido diseñados.

6.2 Validación de Modelo Predictivo del Sistema

Una vez se ha comprobado que la funcionalidad del sistema es adecuada, se va a construir un modelo con volumen de datos, inspirado en organizaciones reales para comprobar los resultados que el sistema genera y validar si son satisfactorios, así como obtener conclusiones sobre la validez y utilidad del modelo planteado.

En esta sección:

- Se definirá la *estructura y los datos* completos del sistema.
- Se hará un *análisis de los resultados* obtenidos.

En el anexo a este documento "Datos para realizar la validación del sistema" se encuentra la hoja con todos los datos del modelo de datos de validación a los que se hace referencia en esta sección.

6.2.1 Definición y Datos del Modelo de uso.

En esta sección se va a definir el modelo de trabajo completamente, así como los datos a utilizar en dicho modelo.

6.2.1.1 Estructura de la Compañía

La compañía sobre la que se va a realizar el estudio es una compañía ficticia. No obstante, se va a inspirar en otras organizaciones de diferentes sectores relacionadas con sistemas de información.

En la siguiente tabla se incluyen las características generales de la compañía.

CARACTERÍSTICA	DESCRIPCIÓN
Nombre de la Compañía	EVOLUTIVE SYSTEMS S.A.
Principal Dedicación	Desarrollo de Sistemas
Sector	Muchos (Comunicaciones, Farmacia, Seguros, Público…)
Número de Empleados	2000
Países con Presencia	España (1800 Empleados), Argentina (200)

La estructura de la compañía se incluye a continuación en el siguiente diagrama.

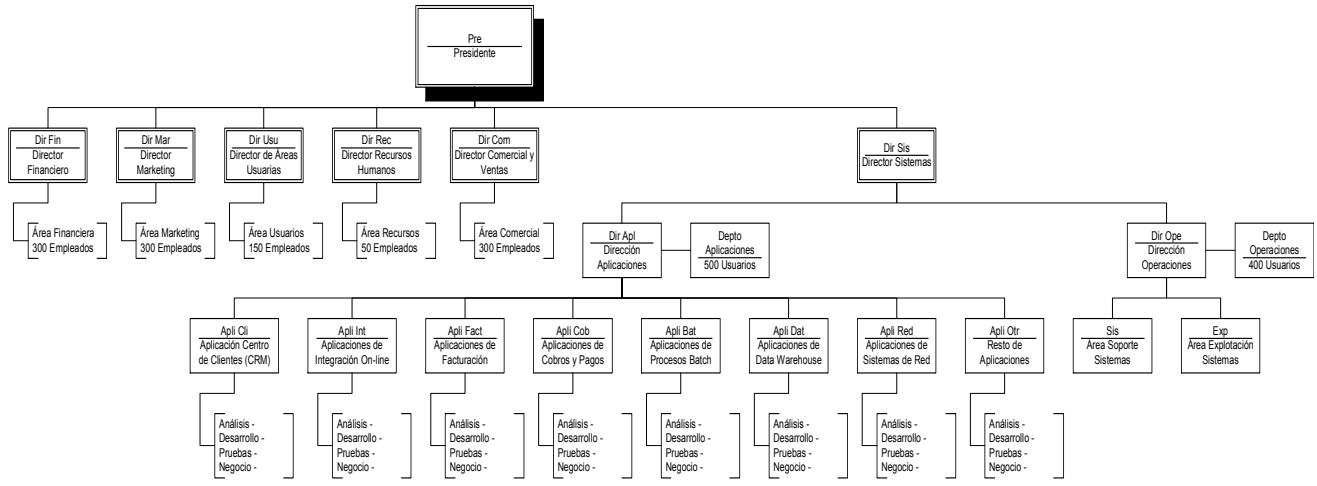


Ilustración 36 – Estructura de la compañía modelo.

En el diagrama se pueden ver las diferentes áreas de dirección, así como los empleados de cada una de las áreas. Se detalla sobre todo la estructura del área de sistemas, en la que se va a realizar el estudio. El área de sistemas se organiza en subdirecciones de aplicaciones y operaciones. Las subdirecciones de aplicaciones a su vez se distribuyen en equipos de trabajo especialistas en un tipo determinado de aplicación.

6.2.1.2 Estructura de Usuarios del Sistema

Una vez definida la estructura de la compañía globalmente, es necesario concretar los usuarios del sistema, así como los roles que tiene que ocupar cada uno de ellos. El estudio se va a centrar sobre el área de aplicaciones de la compañía. Los objetivos esenciales del estudio, tal y como se ha dicho en este documento son:

- Intentar detectar tendencias en las evoluciones de los usuarios del sistema y tratar de anticipar sus evaluaciones.
- Tratar de analizar las causas principales de los comportamientos erróneos, con el fin de establecer acciones correctivas.

6.2.1.2.1 Grupos y Permisos (Roles y Responsabilidades)

Para el sistema se van a definir los grupos que se detallan en la tabla que se incluye a continuación.

GRUPO	DESCRIPCIÓN
ADMIN	Administrador. Grupo de usuarios que podrán administrar completamente el sistema.
ANA	Análisis. Usuarios con rol de analistas.
DES	Desarrollo. Usuarios desarrolladores
PRU	Pruebas. Usuarios de pruebas
NEG	Negocio. Usuarios con roles de negocio, cercanos a los usuarios finales de las aplicaciones.

GRUPO	DESCRIPCIÓN
INST	Instalación. Usuarios con perfil de instalación de aplicaciones y soporte.

Los permisos que tendrán se asignarán según las siguientes premisas:

- El *usuario administrador* tendrá permisos para todas las pantallas de la aplicación. Será el encargado de lanzar las clasificaciones con redes neuronales, ya que es el único con privilegios para el uso de esta funcionalidad.

- Los *usuarios que no sean de instalación* podrán ver las pantallas de reporting y de trabajo con unidades de control.

- Los *usuarios de instalación*, además de las pantallas de reporting y de trabajo con unidades de control, podrán ver las pantallas de instalación.

6.2.1.2.2 Usuarios

El modelo se va a construir con un subconjunto de los usuarios de la compañía. Las premisas de definición de usuarios son las siguientes:

- La estructura de usuarios depende de las aplicaciones que se van a detallar más adelante.

- Se va a establecer un número de *40 usuarios activos*. Existen 8 aplicaciones como se puede ver en la sección anterior donde está el mapa estructural de la compañía. Las que más carga y actividad suponen tendrán 6 usuarios, y las que menos 4. De este modo, los usuarios de 1 al 6 desarrollarán su actividad principalmente en la aplicación 1, los del 6 al 12 en la 2, y así sucesivamente.

- Los usuarios 23 y 24 serán usuarios de dos aplicaciones, una de las críticas y otra de las no críticas, para tener muestras de usuarios de varios sistemas.

- Habrá asimismo 2 usuarios que harán actividad en diferentes aplicaciones, una de las más críticas y otra de las menos críticas, para representar posibles dedicaciones parciales a diferentes sistemas.

- Asimismo habrá *dos usuarios genéricos* de instalación.

En el apéndice "Datos para realizar la validación del sistema" se puede ver el listado detallado, así como su asignación de grupos según estas premisas.

6.2.1.3 Entornos de Trabajo

Se incluye en esta sección la información relativa a los entornos de trabajo. En la sección "Entornos de Trabajo" se incluye detalladamente la descripción de su uso en los sistemas, y su necesidad para la posterior definición de ciclos de vida y unidades de control en la aplicación.

6.2.1.3.1 Aplicaciones

Tal y como se ha comentado en la sección de descripción de la estructura de la compañía, estas son las aplicaciones disponibles.

APLICACIÓN	DESCRIPCIÓN
APLI_CLI	Aplicación de Centro de Clientes
APLI_INT	Aplicación de Integración
APLI_FACT	Aplicación de Facturación
APLI_COB	Aplicación de Cobros y Pagos

APLICACIÓN	DESCRIPCIÓN
APLI_BAT	Aplicación de Procesos Batch
APLI_DAT	Aplicación de Datawarehouse
APLI_RED	Aplicación de Red
APLI_OTR	Otras aplicaciones

En cuanto a criticidad, las más críticas son las de centro de clientes y de facturación, seguidas de las de integración y las del sistema de Cobros. Las aplicaciones más críticas tendrán más información histórica que el resto, así como más usuarios implicados.

6.2.1.3.2 Entornos Definidos

Para todas las aplicaciones se definirán tres entornos de trabajo diferentes, que son los siguientes.

ENTORNO	DESCRIPCIÓN
PRUEBAS_SISTEMA	Entorno donde se realizan las pruebas funcionales de los sistemas y sus integraciones.
PREPRODUCCIÓN	Entorno como producción donde se realizan pruebas funcionales, formaciones, validaciones con usuarios finales…
PRODUCCIÓN	Entorno de Explotación

Es obvio, que para todas ellas habrá un entorno de Desarrollo, si bien no se incluye al ser el estado inicial del modelo propuesto.

6.2.1.3.3 Máquinas

En cuanto a máquinas diferentes para el proyecto, no son una variable de decisión importante, pero para probar el modelo con sistemas diferentes se utilizarán hasta tres máquinas con sistemas distintos. A continuación se indican sus datos.

MÁQUINA	DESCRIPCIÓN
WINDOWS-SERVER	Desktop PC Windows XP
LINUX-SERVER	Servidor Línux Debian Woody 3.0
LINUX-SERVER_2	Servidor Línux Debian Woody 3.0

6.2.1.3.4 Información de cada entorno. Variables propias de entorno.

Para cada aplicación se definirán tres entornos: uno de desarrollo, uno de producción y uno de preproducción. Se pondrán en diferentes máquinas del sistema de forma aleatoria. Los directorios destino de instalación serán de la forma: DIR_BASE/NOMBRE_ENTORNO, donde DIR_BASE es un directorio de la máquina a partir del que quiero colgar información de entornos y NOMBRE_ENTORNO es el nombre que se le da a cada entorno. Para dar nombres a los entornos se utilizará el nombre de la aplicación seguido de las iniciales del entorno (por ejemplo APLI_FACT_SIS será el entorno de pruebas de sistema de la aplicación APLI_FACT).

Las Variables asociadas a cada entorno, únicamente son en este modelo, las necesarias para conectarse a través de Secure Shell (ssh) a la herramienta de gestión de versiones (CVS).

También se pueden ver en el apéndice de Datos detalladamente. En el apéndice "Datos para realizar la validación del sistema" se encuentran los datos correspondientes a cada entorno.

6.2.1.4 Definición de Procesos de Gestión de Cambios

En esta sección se describen los procesos de gestión que se van a cubrir con la herramienta. Se trata de definir cuáles son las unidades de control, qué es lo que representan, sus ciclos de vida y la información que deben llevar asociada.

6.2.1.4.1 Definición de Unidades de Control.

Se van a utilizar como unidades de control las que se han comentado anteriormente en la sección "Modelización del Proceso de Gestión de Cambios: Versiones, Releases, Errores y Requerimientos":

- *Errores (ERR)*: Control de flujo de información de los Errores que se producen en el sistema.
- *Requerimientos (REQ):* Unidad de control para el seguimiento de requerimientos de nueva funcionalidad o de alcance de proyectos.
- *Paquetes de instalación (PACK):* Esta unidad de control determina el flujo de los paquetes de código que agrupan una determinada funcionalidad.

6.2.1.4.2 Ciclos de Vida

El ciclo de vida de dichas unidades de control es similar al representado en la sección "Ciclos de Vida de Unidades de Control", si bien podría ser completamente diferente. Se introduce un nuevo entorno de preproducción con sus posibles errores en pruebas. Asimismo se incluye un

estado de pruebas en producción, que se corresponde a la aceptación o no de una funcionalidad una vez instalada en producción. Se determina el mismo ciclo de vida para las unidades de control ERR, REQ y PACK, aunque podrían ser distintos para cada una de ellas.

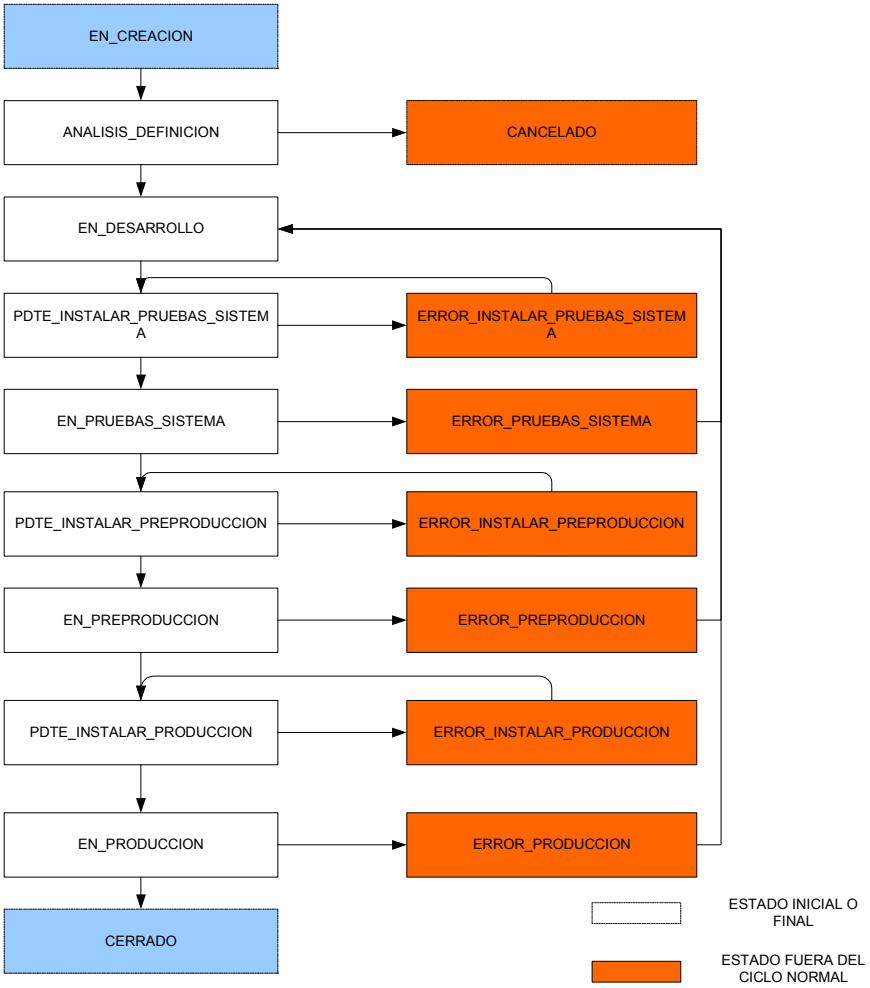


Ilustración 37 – Gráfico de Minería de Datos de Análisis de Causas.

Si bien sí que va a ser el mismo ciclo de vida para todas las unidades de control definidas, por simplificar el modelo.

6.2.1.4.3 Atributos propios de las Unidades de Control.

Se definen a continuación los atributos propios de cada unidad de control. Nótese que el código y el título son campos obligatorios y comunes a todas ellas.

6.2.1.4.3.1 Unidad de Control Errores

ATRIBUTO	TIPO	DESCRIPCIÓN
SEVERIDAD	TEXTO	Determina la Gravedad del Error.
RELEASE_ORIGEN	TEXTO	Release en la que se originó el Error.
RELEASE_PREVISTA	TEXTO	Release en la que se prevé que será solucionado.
ENTORNO_ORIGEN	TEXTO	Entorno en el que se originó la incidencia (¿Pruebas de Sistema? ¿Preproducción? ¿Producción?)
DESCRIPCION_DETALLADA	TEXTO	Descripción de la incidencia.
VALORACION_ESFUERZO	TEXTO	Estimación de esfuerzos en solucionarla.
MOTIVO_CANCELACION	TEXTO	Si se cancela, indicar aquí el motivo.

6.2.1.4.3.2 Unidad de Control Requerimientos

ATRIBUTO	TIPO	DESCRIPCIÓN
TIPO	TEXTO	Si es de Alcance Inicial, Evolutivo, Cambio de Alcance.

ATRIBUTO	TIPO	DESCRIPCIÓN
RELEASE_ORIGEN	TEXTO	Release en la que se originó el Requerimiento.
RELEASE_PREVISTA	TEXTO	Release en la que se prevé que será entregado.
DESCRIPCION_DETALLADA	TEXTO	Descripción de la solución propuesta.
VALORACION_ESFUERZO	TEXTO	Estimación de esfuerzos en solucionarla.
MOTIVO_CANCELACION	TEXTO	Si se cancela, indicar aquí el motivo.

6.2.1.4.3.3 Unidad de Control Paquetes de Instalación

ATRIBUTO	TIPO	DESCRIPCIÓN
RELEASE_PREVISTA	TEXTO	Release en la que se prevé que será entregado.
DESCRIPCION_DETALLADA	TEXTO	Descripción de las instrucciones de instalación.
VALORACION_ESFUERZO	TEXTO	Estimación de esfuerzos en solucionarla.
MOTIVO_CANCELACION	TEXTO	Si se cancela, indicar aquí el motivo.

6.2.1.5 Tipificación de las Causas de Errores

Para la *realización de la minería de datos*, es necesario tipificar las diferentes causas de error. Para ello se dará un listado de las causas de error principales y su relación con las diferentes aplicaciones y unidades de control.

Como se comentó en la sección "Minería de Datos sobre Análisis de Causas Origen de los Errores", es posible modificar dinámicamente la asignación de causas y sus relaciones con otras entidades, pero se va a partir de un modelo consolidado para esta validación.

6.2.1.5.1 Causas.

En la siguiente tabla se detallan las posibles causas de error.

CAUSA	DESCRIPCIÓN
FALLO NO DETERMINADO	Fallo por razón no determinada.
FALLO DE CODIFICACIÓN BÁSICA	Fallo de Codificación Básica de la Aplicación.
FALLO DE CODIFICACIÓN AVANZADA	Fallo por codificación avanzada. Compleja.
FALLO DE ANÁLISIS	Fallo en el análisis de la solución.
FALLO DE INTEGRACIÓN	Fallo de Componente de Integración con otros Sistemas.
FALLO DE INSTALACIÓN	Fallo en la instalación del Software.
FALLO DE PRUEBAS	Fallo en la realización de la Prueba. No ejecutó bien el caso de prueba.
FALLO DE USUARIO	Fallo del usuario final. No utilizó la aplicación según procedimiento.

CAUSA	DESCRIPCIÓN
FALLO DE REGRESIÓN	Fallo porque la nueva funcionalidad no incorpora la antigua.
FALLO DE ACEPTACIÓN	Fallo porque los usuarios finales consideran que no se cubre el objetivo.

6.2.1.5.2 Relación de las Causas con los estados de error, aplicaciones y unidades de control.

Para el modelo que se propone, se han relacionado todas las causas, con todos los estados de error de todas las unidades de control de todas las aplicaciones.

Estas relaciones suponen:

```
10 causas * 6 estados posibles de error * 3 unidades de control * 8 aplicaciones = 1440 combinaciones.
```

No se usarán todas, pero es la más general para poder sacar cualquier relación a priori.

6.2.1.6 Red Neuronal de Clasificación de Usuarios basándose en sus Errores

La definición de la *red neuronal de clasificación de usuarios basándose en sus errores* viene dada por la estructura de aplicaciones, unidades de control y ciclos de vida que se haya elegido, tal y como se detalla en la sección "Red Neuronal de Clasificación de Usuarios basándose en sus Errores".

Para el modelo a validar, se tratará de un perceptrón multicapa (MLP) con el siguiente diseño:

- *Nodos de la Capa de Entrada*:
    - Se tienen 8 aplicaciones.

  - Cada una con 3 unidades de control.
  - El ciclo de vida de las unidades de control tiene 6 estados posibles de error.
  - Habrá por cada una de los estados posibles, 2 unidades de entrada, una con el total de transiciones y otra con el total de errores.
  - Hay 2 entradas adicionales para la suma total de errores.
    - *El total son (8 * 3 * 6) * 2 + 2 = **290** Unidades*

- *Nodos de la Capa Oculta*: Inicialmente los mismos que los de la capa de Entrada, si bien se ajustará posteriormente por prueba y error en la fase de entrenamiento.
- *Valores de Clasificación*: Habrá una única unidad de salida con valores de 0 a 9 inclusive, es decir 10 clases.
- *Número de iteraciones de entrenamiento*: Inicialmente se pone a 1000, con posibilidad de variarlo según el problema.
- *Fichero de Entrenamiento*: Estará con el formato requerido por la Red Neuronal, y tendrá los datos de información histórica que se van a analizar en la siguiente sección.

6.2.1.7 Información Histórica

En esta sección se va a aportar la información histórica necesaria para las pruebas sobre el modelo. Esta información trata de reflejar el progreso de un grupo de usuarios de diferentes sistemas durante una serie de años.

6.2.1.7.1 Periodos

Los *periodos* en los que se van a hacer análisis son los correspondientes a los años naturales desde el año 1998, hasta el 2006. Se describen en la siguiente tabla.

FECHA INICIO DE PERIODO A EVALUAR	FECHA FIN DE PERIODO A EVALUAR
1998-01-01 00:00:00	1999-01-01 00:00:00
1999-01-01 00:00:00	2000-01-01 00:00:00
2000-01-01 00:00:00	2001-01-01 00:00:00
2001-01-01 00:00:00	2002-01-01 00:00:00
2002-01-01 00:00:00	2003-01-01 00:00:00
2003-01-01 00:00:00	2004-01-01 00:00:00
2004-01-01 00:00:00	2005-01-01 00:00:00
2005-01-01 00:00:00	2006-01-01 00:00:00
2006-01-01 00:00:00	2007-01-01 00:00:00
2007-01-01 00:00:00	2008-01-01 00:00:00

La idea es que se tiene información de las evaluaciones realizadas en todos los años menos en los dos últimos (según la tabla, del 2006 al 2008) que es en los que se pretende aplicar el modelo para adelantar resultados.

6.2.1.7.2 Evaluaciones de Usuarios Previas

Basándose en los periodos anteriormente mencionados se incluye información de las *evaluaciones* de los diferentes usuarios. Nótese que un usuario solamente podrá tener una evaluación para un determinado periodo.

Se propone una distribución aleatoria de clasificaciones por periodos de la siguiente manera:

- Se proponen *3 grupos de valores de clasificación*:
    - **0 a 3:** Evaluaciones Bajas
    - **4 a 6:** Evaluaciones Medias
    - **7 a 9:** Evaluaciones Altas
- Como se comentó anteriormente, *las aplicaciones más críticas* (de la 1 a la 4) tienen en promedio *6 usuarios y las menos críticas* (de la 5 a la 8) *tienen 4*. Todos los usuarios usuarios tienen que ser evaluados en los periodos 2006-2007 y 2007-2008 y tienen evaluaciones previas de 8 periodos anteriores.
- Se elegirán un porcentaje aproximado de *25%-30% de evaluaciones bajas, 25%-30% de evaluaciones altas y 40%-50% de evaluaciones medias*, repartidas de forma compensada entre los valores del grupo de evaluación.
- En general *los saltos de evaluación de un usuario son de un grupo al inmediatamente superior o inferior*. Es decir, no es común un usuario con evaluación alta que al periodo siguiente tenga una baja y viceversa.

Se muestran a continuación las distribuciones de los datos de evaluaciones con las que se va a trabajar.

PORCENTAJES	APLICACIÓN 1	APLICACIÓN 2	APLICACIÓN 3	APLICACIÓN 4	APLICACIÓN 5	APLICACIÓN 6	APLICACIÓN 7	APLICACIÓN 8	TOTALES
De 0	6,25	4,17	0,00	0,00	0,00	7,50	5,00	0,00	2,86
De 1	6,25	6,25	4,17	4,17	5,00	2,50	10,00	0,00	4,79
De 2	6,25	8,33	14,58	6,25	15,00	7,50	10,00	0,00	8,49
De 3	6,25	6,25	6,25	14,58	10,00	12,50	5,00	0,00	7,60
De 4	14,58	18,75	10,42	8,33	10,00	12,50	12,50	0,00	10,89
De 5	18,75	14,58	22,92	18,75	17,50	12,50	17,50	0,00	15,31
De 6	16,67	16,67	16,67	22,92	12,50	15,00	10,00	0,00	13,80
De 7	10,42	4,17	8,33	4,17	7,50	7,50	7,50	12,50	7,76
De 8	10,42	8,33	12,50	8,33	15,00	10,00	15,00	37,50	14,64
De 9	4,17	12,50	4,17	12,50	7,50	12,50	7,50	50,00	13,85
De evaluaciones bajas (0 a 3)	25,00	25,00	25,00	25,00	30,00	30,00	30,00	0,00	23,75
De evaluaciones medias (4 a 6)	50,00	50,00	50,00	50,00	40,00	40,00	40,00	0,00	40,00
De evaluaciones altas (7 a 9)	25,00	25,00	25,00	25,00	30,00	30,00	30,00	100,00	36,25
TOTAL	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**	**100,00**

6.2.1.7.3 Listado de Unidades de Control disponibles

Se propone un *listado de unidades de control*, construido según las siguientes premisas:

- De las *cuatro aplicaciones más críticas* habrá para los periodos finalizados:
    - 8 requerimientos (REQ) por periodo.
    - 20 errores (ERR) por periodo.
    - 20 paquetes de instalación (PACK) por periodo.
- De las *cuatro aplicaciones menos críticas* habrá los periodos finalizados:
    - 4 requerimientos (REQ) por periodo.
    - 10 errores (ERR) por periodo.
    - 10 paquetes de instalación (PACK) por periodo.

- *En el periodo* 2006-2007 que es uno de los que se pretenden sacar las conclusiones, *habrá menos datos disponibles* ya que haremos predicciones antes de la terminación del año. Para el 2007-2008 se utilizan datos de Unidades de Control similares a los del periodo 2002-2001

- *Los estados serán finales para todas las unidades de control generadas en periodos ya clasificados*, excepto para las de los periodos a clasificar que se repartirán por los diferentes estados.

De las anteriores conclusiones salen los siguientes números para confeccionar el listado de unidades de control del sistema:

```
4 Aplicaciones * { 8 periodos * (8 REQ + 20 PACK + 20 ERR) + 1 periodo * (4 REQ + 10 PACK + 10 ERR) } + 4 Aplicaciones * { 8 periodos * (4 REQ + 10 PACK + 10 ERR) + 1 periodo * (2 REQ + 5 PACK + 5 ERR) } = 2448 Unidades en el listado
```

6.2.1.7.4 Histórico de estados de Unidades de Control.

Para elaborar la información histórica *se ha diseñado un programa auxiliar de Generación de Datos Aleatorios.* Se ha implementado dentro de la misma interfaz cliente, si bien se trata de una pantalla que únicamente será visible con permisos de administración total del sistema.

6.2.1.7.4.1 Generación de Datos Históricos Aleatorios

El funcionamiento del programa generador aleatorio de datos se basa en la distribución probabilística de los mismos. Para ellos se han establecido los siguientes grupos:

- *Grupos de Evaluaciones*, los descritos en la sección “Evaluaciones de Usuarios Previas”:

    - Evaluaciones Bajas: 0 a 3

    - Evaluaciones Medias: 4 a 6

    - Evaluaciones Altas: 7 a 9

- *Grupos de Causas*: Se agrupan las causas descritas en "Tipificación de las Causas de Errores" por origen del error, de manera que hay tres grupos:
    - *Causas de Desarrollo*: Son causas que tienen su origen en el desarrollo. Se incluyen las siguientes causas.
        - FALLO NO DETERMINADO
        - FALLO DE CODIFICACIÓN BÁSICA
        - FALLO DE CODIFICACIÓN AVANZADA
        - FALLO DE ANÁLISIS
    - *Causas de Pruebas*: Estas causas tienen su origen en las pruebas de lo que genera desarrollo.
        - FALLO DE INTEGRACIÓN
        - FALLO DE INSTALACIÓN
        - FALLO DE PRUEBAS
    - *Causas de Usuario*: Los errores son motivados por el uso del sistema.
        - FALLO DE USUARIO
        - FALLO DE REGRESIÓN
        - FALLO DE ACEPTACIÓN
- *Grupos de Errores*: Se agrupan los errores según la etapa en la que se cometen en el ciclo de vida, de la siguiente manera:
    - *Errores en etapa de Desarrollo y Pruebas de Sistema*:
        - ERROR_INSTALAR_PRUEBAS_SISTEMA
        - ERROR_PRUEBAS_SISTEMA

- *Errores en etapa de Preproducción*:
  - ERROR_INSTALAR_PREPRODUCCION
  - ERROR_PREPRODUCCION
- *Error en etapa de producción*:
  - ERROR_INSTALAR_PRODUCCION
  - ERROR_PRODUCCION

El funcionamiento de la aplicación radica en la distribución probabilística de los casos de error. Se asignarán probabilidades a cada grupo de errores y a cada grupo de causas. Para los diferentes grupos de evaluaciones *se establece la siguiente distribución*:

- *Si la evaluación del usuario en el periodo es alta, la probabilidad de error es 0.*
- *Si la evaluación es baja, la probabilidad de caso con error es 1.*
- *Si la evaluación en media, la probabilidad de que el caso tenga error es 0,5.*

El programa sigue el siguiente pseudocódigo:

- `Recorrer la lista de Aplicaciones (x8).`
  - `Para cada Aplicación (x6 o x4) recorrer lo Usuarios que tiene asignados`
    - `Recorrer todos los Periodos disponibles (x8)`
      - `Mirar la evaluación del Usuario en el Periodo ya asignada`
        - `Recorrer la Unidades de Control del Periodo`
          - `Si la evaluación del usuario es baja en el Periodo entrar en el ciclo con errores (descrito posteriormente)`

- Si la evaluación del usuario es alta en el Periodo introducir un ciclo completo sin Errores
- Si la evaluación es media, con una probabilidad del 50% entrar en un ciclo con errores o sin errores.

Para las evaluaciones bajas se seguirá el siguiente proceso:

- Recorrer desde el principio el Ciclo de Vida de la Unidad de Control
  - Si el siguiente estado al que se salta puede ser de error:
    - Si es estado de Desarrollo, saltar a error con probabilidad de Grupo de Desarrollo.
    - Si es estado de Preproducción, saltar a error con probabilidad de Grupo de Preproducción.
    - Si es estado de Producción, saltar a error con probabilidad de Grupo de Producción.
    - Si he saltado a algún error la causa de error será:
      - Causa de desarrollo con probabilidad de Causas Origen de Desarrollo.
      - Causa de pruebas con probabilidad de Causas Origen de Pruebas.
      - Causa de usuario con probabilidad de Causas de Usuario.

Se muestra a continuación la pantalla de generación de datos desarrollada para cubrir esta funcionalidad.

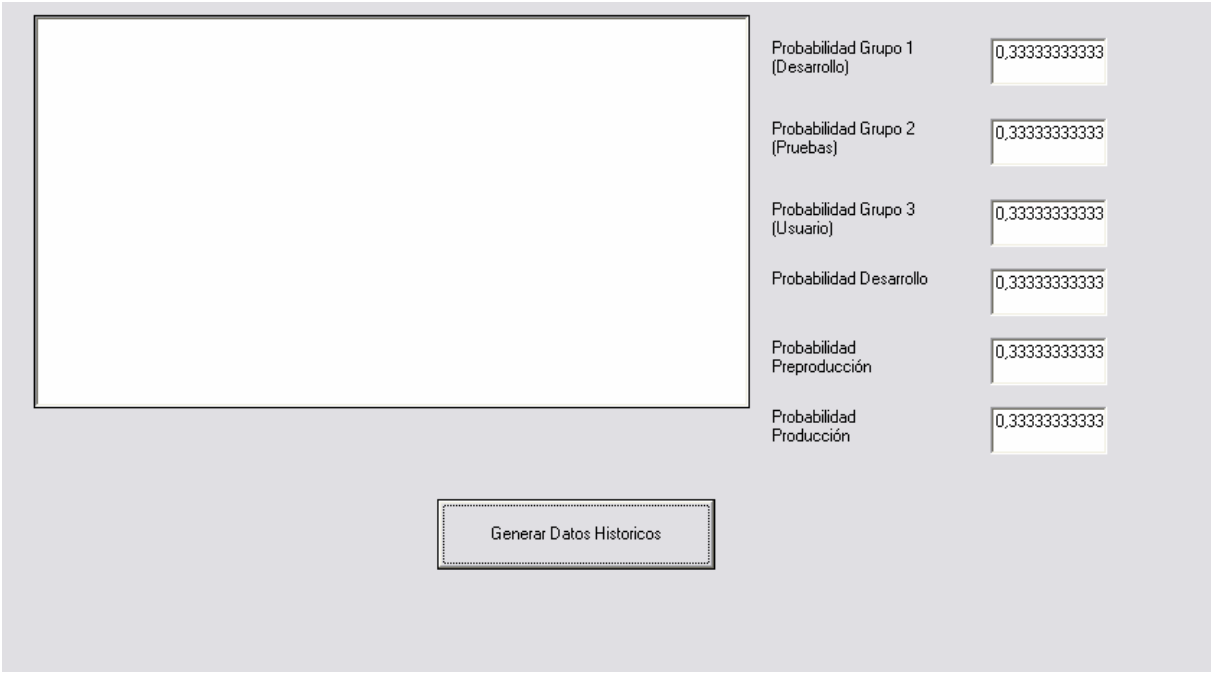


Ilustración 38 – Generador de Datos Históricos Aleatorios.

Nótese que las probabilidades finales elegidas para Grupo 1, 2 y 3 (0,7, 0,2 y 0,1 respectivamente) y para Desarrollo, Preproducción y Producción (0,7, 0,2 y 0,1 respectivamente) se van a describir a continuación. El funcionamiento es sencillo. Únicamente se deben introducir las probabilidades en las que se distribuye cada grupo de causas y de errores. Cuando se pulsa en el botón de Generar los Datos se borrará toda la información histórica de la base de datos y se generará la nueva.

Como se ha comentado, se han distribuido los errores y las causas según unas probabilidades. Para este modelo se escogerá una distribución probabilística que hace más probables los errores en las primeras etapas del ciclo de vida (desarrollo) y menos en las últimas (producción).

Las probabilidades asignadas a los errores son las siguientes

- *Probabilidad Grupo de Errores de Desarrollo: 0,7*
- *Probabilidad Grupo de Errores de Preproducción: 0,2*
- *Probabilidad Grupo de Errores de Producción: 0,1*

Si se produce error, para las causas, los valores de probabilidad de los diferentes grupos serán:

- *Probabilidad Grupo de Causas Origen en Desarrollo: 0,7*
- *Probabilidad Grupo de Causas Origen en Pruebas: 0,2*
- *Probabilidad Grupo de Causas Origen en Usuario: 0,1*

Una vez generados los datos, estos se han distribuido en la tabla de históricos de la siguiente manera:

- Un total de **48307 registros** en los periodos comprendidos entre 1998 y 2005.
- Los periodos 2006-2007 y 2007-2008 serán utilizados para la clasificación y tienen un total de **12555 registros**.
    - El periodo 2006-2007 tiene una distribución aleatoria de errores con el fin de que se pueda tener un periodo a evaluar con sucesos aleatorios. Es decir, no se tiene en cuenta evaluación alguna y en cualquier momento cualquier usuario puede cometer un error.
    - El periodo 2007-2008 se ha creado similar al periodo 2000-2001 con el fin de que se tenga un periodo a evaluar con un modelo de comparación. Se seleccionan para ello los registros del periodo 2000-2001 y se modifican levemente en el nuevo periodo de manera que sean similares en grandes números.

Se han generado un total de **60862 registros**.

6.2.1.7.4.2 Información para la Red Neuronal

Para la red neuronal se han generado ficheros de entrenamiento según el procedimiento indicado en la sección "Red Neuronal de Clasificación de Usuarios basándose en sus Errores", utilizando para ello la información histórica que se ha descrito.

Para cada uno de los 38 usuarios que tienen actividad en las diferentes aplicaciones (Nótese que hay 2 usuarios que actúan en 2 aplicaciones, ya que serían 40 en total si no existieran estos 2) se genera un registro por cada periodo, de manera que el *fichero de entrenamiento de la red es de 38*8=304 registros*. *Cada uno de los registros*, como se comentó anteriormente *tiene 291 campos*, 290 de las unidades de entrada y uno con el valor de clasificación del periodo.

6.2.2 Pruebas Realizadas sobre el Modelo Predictivo

A continuación se comentan las pruebas que se han realizado y los resultados obtenidos en este modelo. Nótese que los casos de prueba que se describen a continuación se han realizado para la parte predictiva de la aplicación. Las validaciones básicas de funcionalidad se muestran en la sección "Validaciones Básicas del Sistema".

6.2.2.1 Precondiciones y Bloques de Pruebas

Las precondiciones para la realización de pruebas son las siguientes:

- *El sistema funciona* desde el punto de vista de pruebas unitarias, especialmente en la parte predictiva de clasificación de redes neuronales, lo cual se ha mostrado anteriormente.

- *Se ha entrenado la red neuronal con datos históricos previos*, descritos en la sección "Información Histórica". En el conjunto de datos históricos con el que se trabaja se han realizado las extracciones debidas y la preparación de ficheros de datos para realizar el entrenamiento. *Los periodos que se han utilizado para el entrenamiento son desde 1998-1999 a 2005-2006, ambos inclusive, es decir, un total de 8*.

    - Se han obtenido *38 registros por periodo*, uno por cada usuario, tal y como se describe en la sección "Definición de Red Neuronal de Errores de Usuario"

- *El total de registros es por tanto 304 (38*8), de 290 campos* por registro más un último con el valor de la clasificación en ese periodo.
- Con este rango de datos *se han realizado varios entrenamientos de la red para tratar de ajustar su topología y conseguir mejores ratios de clasificación*. Dicha topología se ajusta como sigue:
  - El *número de nodos de la capa oculta se ha fijado en 146* (La mitad aproximada del total de nodos de la red).
  - La *tasa de aprendizaje se ha fijado a 0,0001*.
- *Se han generado los informes necesarios* para la realización de los análisis de evaluaciones de usuarios y de causas.

Se pueden ver los ficheros de prueba generados en el apéndice "Datos para realizar la validación del sistema". Asimismo se pueden ver en detalle los informes generados automáticamente por el sistema, de los que se va a realizar un análisis posteriormente.

A continuación se describen los bloques de pruebas a realizar para la validación del modelo.

NOTA: Para la obtención de estas extracciones, ficheros de datos y entrenamientos se han requerido aproximadamente 100 horas de proceso de 3 ordenadores monoprocesador de forma paralela.

6.2.2.1.1 Predicción de un ciclo aleatorio

Un primer bloque de pruebas consiste en crear para un periodo determinado de tiempo un conjunto de datos históricos aleatorios y un *comportamiento por tanto diferente al de otros periodos*. Este conjunto de datos es generado de manera que cualquier usuario en cualquier momento puede cometer un error. No se atiende a evaluaciones previas.

Sí que seguirá una determinada distribución probabilística en cuanto al porcentaje de errores general que se producirán, sin embargo no estará condicionado por las evaluaciones de los usuarios en ese periodo.

Después se debe hacer una analítica tanto de los datos de entrada como de los datos de salida para determinar si los resultados son de cierto sentido.

La *preparación de los datos* ha consistido en generar un periodo completo con el generador de datos automáticos, el 2006-2007 y posteriormente clasificar los datos obtenidos en este periodo.

6.2.2.1.2 Predicción de un ciclo similar a otro anterior

La prueba consiste en, una vez tenemos los datos debidamente clasificados en un modelo de red neuronal, *seleccionar un periodo que se tenga previamente clasificado e introducir otro similar con pequeñas variaciones*.

Los datos esperados de salida deben ser parecidos a los del periodo original. El objetivo de esta prueba es demostrar que si se dan condiciones parecidas entre diferentes periodos, la red va a ser capaz de detectar dichas similitudes y dar resultados similares que muestren *tendencias parecidas*.

La *preparación de datos* ha consistido en escoger un periodo al azar, en este caso 2000-2001, e introducir los datos con mínimas variaciones en algunas iteraciones de error y algunas evaluaciones de usuarios, sin cambiar de grupo de evaluación, en el periodo 2007-2008, para posteriormente lanzar la clasificación de dicho periodo y comparar los resultados.

6.2.2.2 Análisis de Resultados

Se analizan a continuación los resultados obtenidos en las diferentes pruebas realizadas. Las *fuentes de análisis* son:

- Los propios *datos generados* en la base de datos.

- Los *datos resultantes* que se han mandado para realizar la clasificación y entrenamiento de la ANN.

- Los *datos de los informes* de Errores y de detección de causas.

6.2.2.2.1 Distribución de Resultados

A continuación se muestra una tabla resumen con la *distribución de evaluaciones obtenidas en cada uno de los periodos*. Los periodos de análisis son el periodo 2006 a 2007 y el periodo 2007 a 2008.

Periodo	Valor Evaluación										
	0	1	2	3	4	5	6	7	8	9	Total
1998 01-01 a 1999-01-01	5,26%	2,63%	2,63%	7,89%	13,16%	23,68%	15,79%	13,16%	5,26%	10,53%	100,00%
1999 01-01 a 2000-01-01	2,63%	5,26%	7,89%	2,63%	7,89%	31,58%	13,16%	2,63%	23,68%	2,63%	100,00%
2000 01-01 a 2001-01-01	0,00%	2,63%	13,16%	2,63%	13,16%	2,63%	36,84%	5,26%	10,53%	13,16%	100,00%
2001 01-01 a 2002-01-01	0,00%	7,89%	2,63%	7,89%	18,42%	15,79%	18,42%	7,89%	5,26%	15,79%	100,00%
2002 01-01 a 2003-01-01	2,63%	2,63%	7,89%	23,68%	15,79%	2,63%	18,42%	13,16%	7,89%	5,26%	100,00%
2003 01-01 a 2004-01-01	2,63%	0,00%	21,05%	13,16%	5,26%	28,95%	2,63%	7,89%	18,42%	0,00%	100,00%
2004 01-01 a 2005-01-01	2,63%	7,89%	13,16%	13,16%	7,89%	18,42%	10,53%	5,26%	13,16%	7,89%	100,00%
2005 01-01 a 2006-01-01	5,26%	21,05%	10,53%	0,00%	13,16%	10,53%	13,16%	2,63%	7,89%	15,79%	100,00%
2006 01-01 a 2007-01-01	0,00%	10,53%	18,42%	10,53%	15,79%	21,05%	23,68%	0,00%	0,00%	0,00%	100,00%
2007 01-01 a 2008-01-01	0,00%	0,00%	13,16%	7,89%	10,53%	2,63%	36,84%	5,26%	13,16%	10,53%	100,00%
Total	**2,11%**	**6,05%**	**11,05%**	**8,95%**	**12,11%**	**15,79%**	**18,95%**	**6,32%**	**10,53%**	**8,16%**	**100,00%**

Tal y como se ha comentado anteriormente, se han agrupado los datos en grupos de *evaluaciones Altas (7, 8 y 9), Medias (4, 5 y 6) y Bajas (0, 1, 2 y 3)*, con las distribuciones indicadas en la sección "Evaluaciones de Usuarios Previas". Si atendemos a estos grupos de evaluaciones, la distribución se muestra en la siguiente tabla.

Porcentaje	TIPO			
Periodo	ALTA	BAJA	MEDIA	Total
1998-01-01 a 1999-01-01	28,95%	18,42%	52,63%	100,00%
1999-01-01 a 2000-01-01	28,95%	18,42%	52,63%	100,00%
2000-01-01 a 2001-01-01	28,95%	18,42%	52,63%	100,00%
2001-01-01 a 2002-01-01	28,95%	18,42%	52,63%	100,00%
2002-01-01 a 2003-01-01	26,32%	36,84%	36,84%	100,00%
2003-01-01 a 2004-01-01	26,32%	36,84%	36,84%	100,00%
2004-01-01 a 2005-01-01	26,32%	36,84%	36,84%	100,00%
2005-01-01 a 2006-01-01	26,32%	36,84%	36,84%	100,00%
2006-01-01 a 2007-01-01	0,00%	39,47%	60,53%	100,00%
2007-01-01 a 2008-01-01	28,95%	21,05%	50,00%	100,00%
Total	**25,00%**	**28,16%**	**46,84%**	**100,00%**

En la gráfica generada en el informe de análisis de evaluaciones se puede observar como se han buscado los *cambios suaves en los usuarios*. Es decir, no pasa de evaluación baja a evaluación alta sin pasar un periodo por evaluación media. Se observa también que en el periodo 2006-2007, uno de los periodos que se analizarán posteriormente, sí que hay un cambio de dicha tendencia.

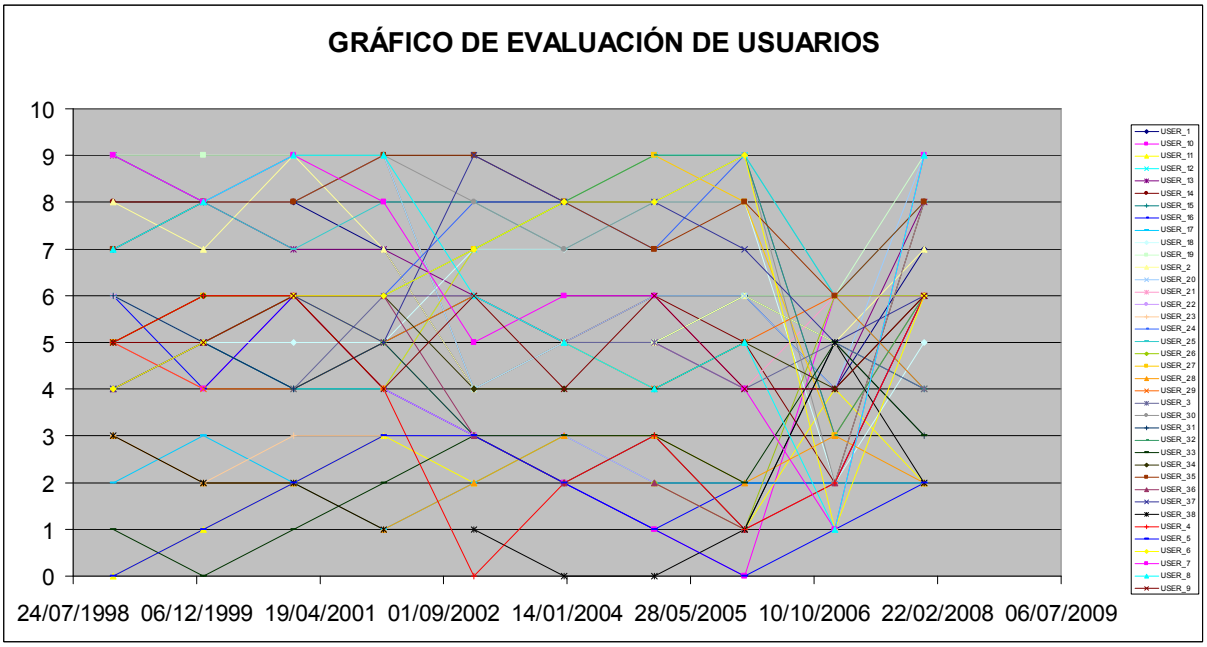


Ilustración 39 – Grafico general de Errores Evaluaciones de todos los usuarios

Se analizan a continuación los periodos 2006-2007 y 2007-2008 que son los que han sido evaluados por el sistema.

6.2.2.2.2 Predicción de un ciclo aleatorio

La generación de datos del ciclo 2006-2007, se ha realizado de forma aleatoria, de manera que *la probabilidad de un determinado usuario de cometer un determinado error viene determinada por la probabilidad general de ese error*, según las distribuciones indicadas en la sección "Datos para la Clasificación".

Como se ha comentado anteriormente, los usuarios con evaluaciones altas, en sus ciclos, no cometen errores. Esto explica la circunstancia de que la red neuronal haya predicho que el número de evaluaciones altas del periodo sea del 0%. Esto motiva en el gráfico de progreso de evaluaciones de los usuarios se vea una *caída general de las evaluaciones*. Se repartirán por tanto entre evaluaciones bajas y medias. Los usuarios que más hayan caído en errores tendrán evaluaciones bajas y los que menos tendrán medias.

Las evaluaciones quedan distribuidas en un 60% de medias y un 40% de bajas. Si se observan en la tendencia general las evaluaciones medias (46,84%), representan 1,5 veces aproximadamente las evaluaciones bajas (28,16%). Por lo tanto *lo que predice la red neuronal para el periodo aleatorio sigue una distribución similar a la distribución media general*.

También se puede ver en la Gráfica de Distribución de Resultados de la sección "Distribución de Resultados" *que se concentran el mayor número de evaluaciones en resultados de 5 y 6*. Se puede observar que en los totales generales es exactamente esta la distribución general. Los datos que más se repiten respectivamente en la distribución general son el 6 (18,95%) y el 5 (15,79%).

Igualmente, si se revisa el detalle de evaluaciones bajas, en el periodo aleatorio los valores que más se repiten son 2, 3, 1 y 0, en orden de más frecuente a menos frecuente. Si se observa en la tendencia general, *sigue este orden*.

También se puede ver que en particular el dato 0 tiene un número muy bajo de ocurrencias (2,11%) en el caso general, lo que coincide con el periodo evaluado, en el que no tiene ni una sola ocurrencia. Esto viene derivado de que la red tiende a podar los casos extremos en evaluaciones de usuarios lo cual es esperable.

6.2.2.2.2.1 Análisis de Causas y Errores

Se analizan los resultados de la minería de datos de Visualización sobre este periodo, revisando el informe general de causas de errores. Se visualiza a continuación el gráfico general de *errores que provocan las evaluaciones más bajas*.

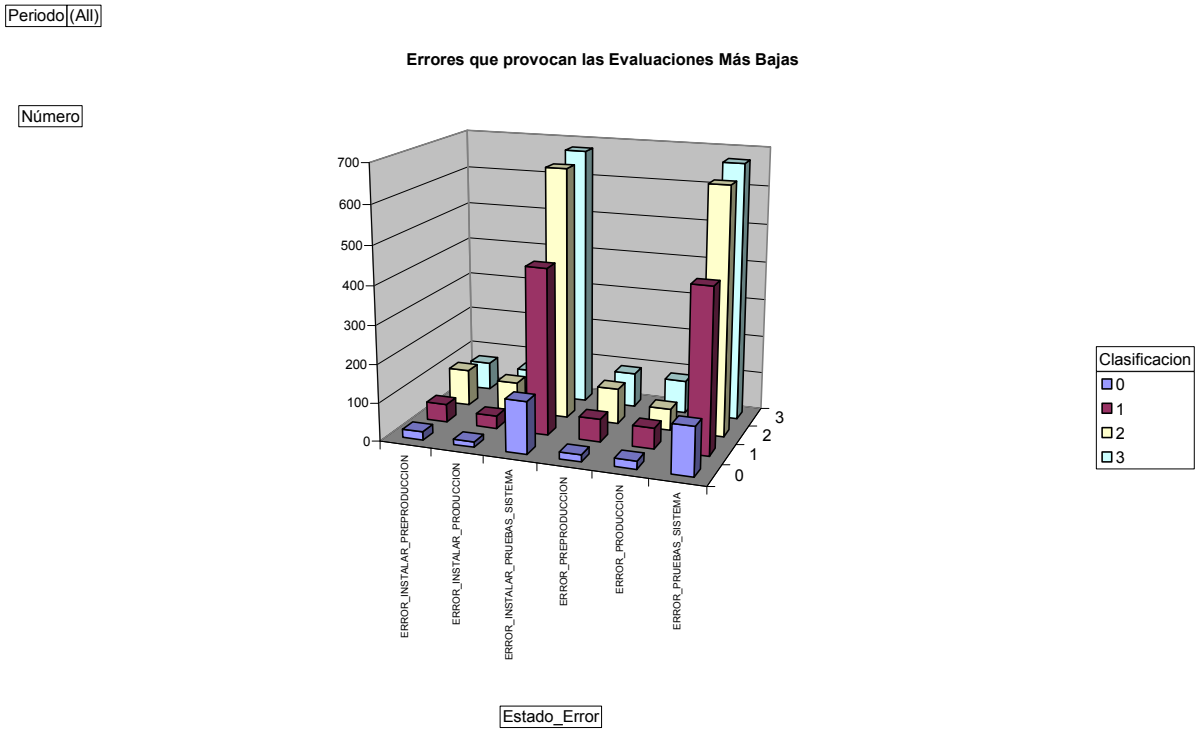


Ilustración 40 – Grafico general de Errores que provocan las evaluaciones más bajas

Se puede observar que para todos los periodos, los errores siguen las distribuciones previstas en la sección “Datos para la Clasificación”. Los errores más cometidos son los del grupo que se han llamado Errores de Desarrollo (ERROR_PRUEBAS_SISTEMA y ERROR_INSTALAR_PRUEBAS_SISTEMA en este caso), siendo los errores del grupo Preproducción y Producción, prácticamente equiprobables.

Nótese que esto lo que está indicando es los puntos del ciclo de vida donde más errores se cometen, lo que ya da un importante dato para la toma de decisiones de la compañía. En este caso se puede observar que *la fase de desarrollo y pruebas de sistema es el mayor foco de error de la compañía*.

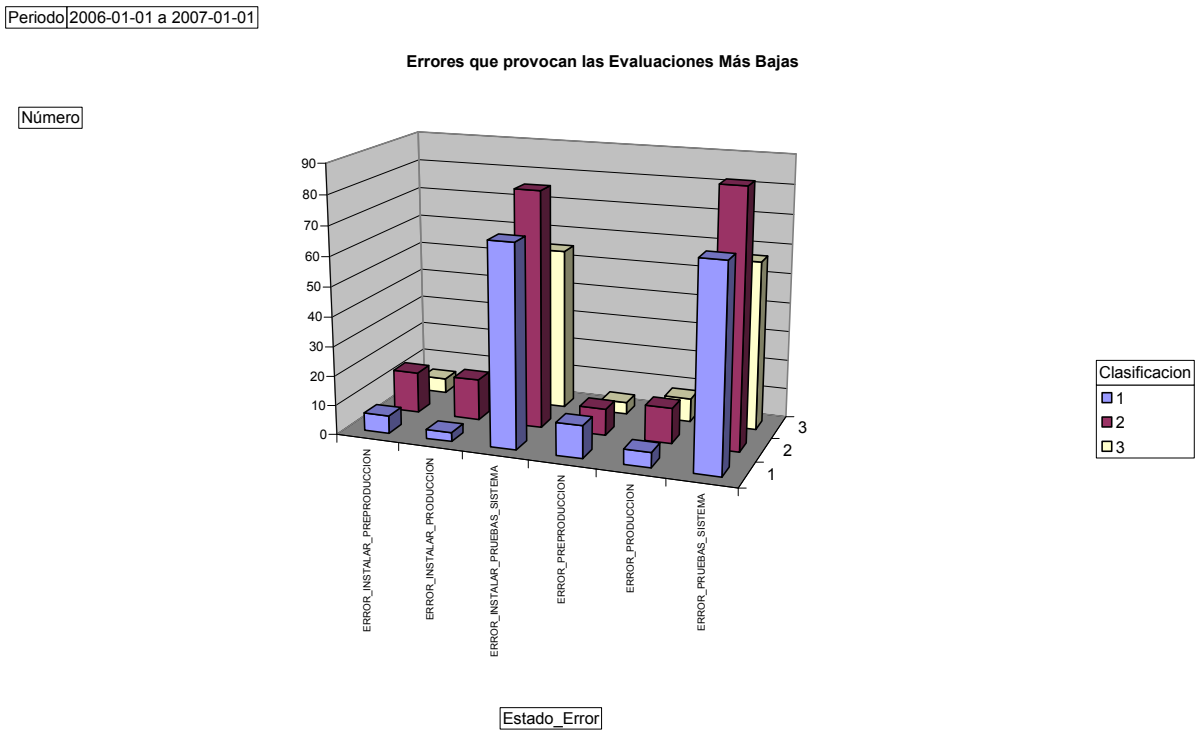


Ilustración 41 – Grafico del periodo 2006-2007 de Errores que provocan las evaluaciones más bajas

En el periodo 2006-2007 se tiene una *distribución de errores similar a la del caso general*. La mayoría de los errores cometidos quedan en el grupo de Desarrollo quedando la distribución de Preproducción y Producción con probabilidad parecida, mucho menor. Al haber sido generado el periodo siguiendo las distribuciones generales de probabilidad de errores en cada fase. este resultado era esperable.

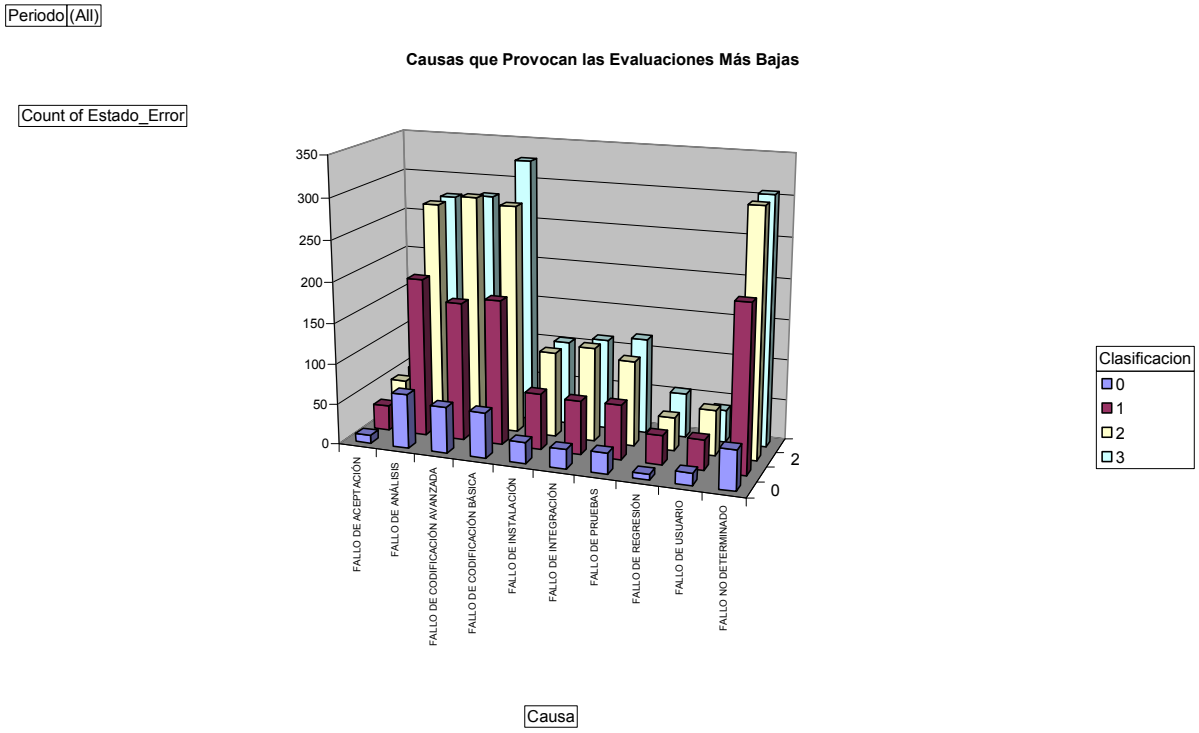


Ilustración 42 – Grafico general de Causas que provocan las evaluaciones más bajas

Si se observa el dato general de las *causas que provocan las evaluaciones más bajas*, se ve que se tienen tres grupos claramente identificados, tal y como se indicó en la sección "Datos para la Clasificación", que son las *causas de origen en Desarrollo* (FALLO NO DETERMINADO, FALLO DE CODIFICACIÓN BÁSICA, FALLO DE CODIFICACIÓN AVANZADA, FALLO DE ANÁLISIS), que representan la mayor parte, el grupo de causas Origen en Pruebas (FALLO DE INSTALACIÓN, FALLO DE INTEGRACIÓN, FALLO DE PRUEBAS) y el grupo de causas de Usuario (FALLO DE USUARIO, FALLO DE REGRESIÓN, FALLO DE ACEPTACIÓN).

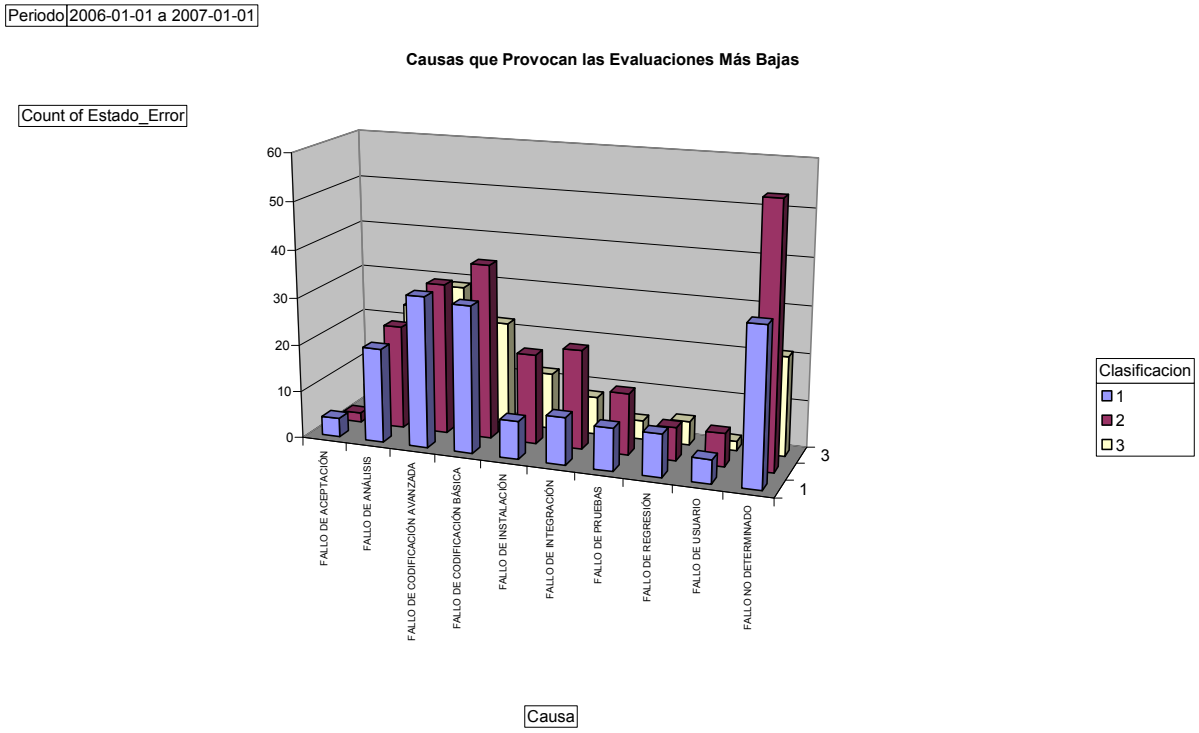


Ilustración 43 – Grafico del periodo 2006-2007 de Causas que provocan las evaluaciones más bajas

En el periodo evaluado la *distribución de causas obtenidas mantiene un esquema que sigue la tendencia general*, si bien la causa que más se repite dentro del grupo de Desarrollo es "FALLO NO DETERMINADO". Se debe simplemente a razones de aleatoriedad de datos. Si bien el grupo de causas con Origen Desarrollo es el más numeroso según se han generado los datos, esta causa en concreto ocurre más veces en este periodo. Como se puede ver en el esquema general, se compensa.

La información que reporta esta minería de datos puede ser clave para la compañía. Lo que arroja este esquema es que las causas que están provocando las evaluaciones más bajas de la compañía vienen dadas por fallos de análisis, o de codificación de las diferentes aplicaciones. También se puede obtener este informe por sistema. Esta información orientaría esfuerzos de la compañía de formación, de personal o acciones correctivas de dirección.

Por tanto, se demuestra que *la distribución de resultados dentro de un periodo aleatorio está de acuerdo con lo esperable*, con lo que se puede concluir que *el modelo generado para un periodo de distribución aleatoria es capaz de predecir los resultados adecuadamente*.

6.2.2.2.3 Predicción de un ciclo similar a otro anterior

Los datos del periodo 2007-2008 han sido generados a partir de los datos del periodo 2000-2001. Lo que es esperable del sistema es que se obtengan datos de clasificación *similares en tendencias a los obtenidos en ese último periodo*. También es esperable que *no sean exactamente iguales* porque se podría estar ante un problema de sobreajuste de la red de entrenamiento a los ejemplos.

Debieran ser parecidos en cuanto a las tendencias de los datos, es decir, que las evaluaciones de los distintos empleados que fueron altas (7, 8, 9) en el periodo 2000-2001 tiendan a ser altas en el periodo 2007-2008, las medias tiendan a ser medias (6, 5, 4), y lo mismo para las bajas (3, 2, 1, 0).

Si se observa en la sección "Distribución de Resultados", la distribución por evaluaciones, se puede ver que la cantidad de evaluaciones por porcentaje no coincide en la mitad de los posibles valores. Coincide en la cantidad de evaluaciones 7, 6, 5, 2 y 0. Por lo tanto esto nos indica que la red *está generalizando debidamente*. No está completamente ajustada a los ejemplos sino que detecta tendencias generales.

Si se analiza la distribución por grupos de evaluaciones, se puede ver una coincidencia casi total en todos los grupos de evaluaciones. En concreto, los periodos 2000-2001 y 2007-2008 *coinciden en las distribuciones generales de evaluaciones altas y en las medias y bajas hay una muy pequeña diferencia*. Esta diferencia, analizada en detalle es porque la red ha detectado que el patrón seguido por un usuario (USER_31) se parece a uno de evaluación baja y lo ha clasificado como tal, dándole un valor 3. Nótese que la red le da, de las evaluaciones bajas, la más alta (un 3). Asimismo, nótese que el usuario en el periodo 2000-2001 tenía la más baja de las evaluaciones medias posibles (un 4). Desde el punto de vista probabilística, al ser las evaluaciones medias equiprobables entre casos con error y casos sin error, se pueden dar casos de usuarios con más errores de la media.

Por todo lo anterior se puede decir que se está ante un modelo que realiza las predicciones dentro de lo esperable y que no cae en el sobreajuste de los ejemplos. *Si se repiten situaciones de uno a otro periodo, el modelo se comportará de una manera similar*.

6.2.2.2.3.1 Análisis de Causas y Errores

Se comparan a continuación algunos aspectos entre los periodos 2000-2001 y 2007-2008 de forma gráfica.

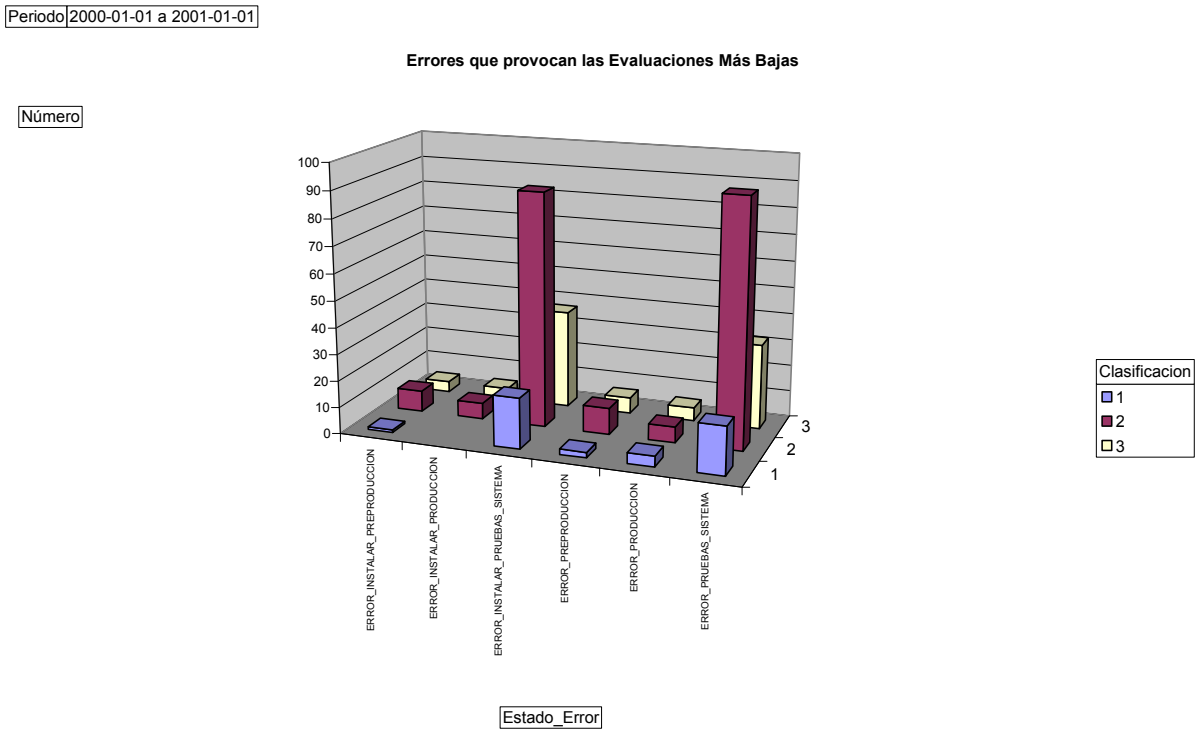


Ilustración 44 – Grafico del periodo 2000-2001 de Errores que provocan las evaluaciones más bajas

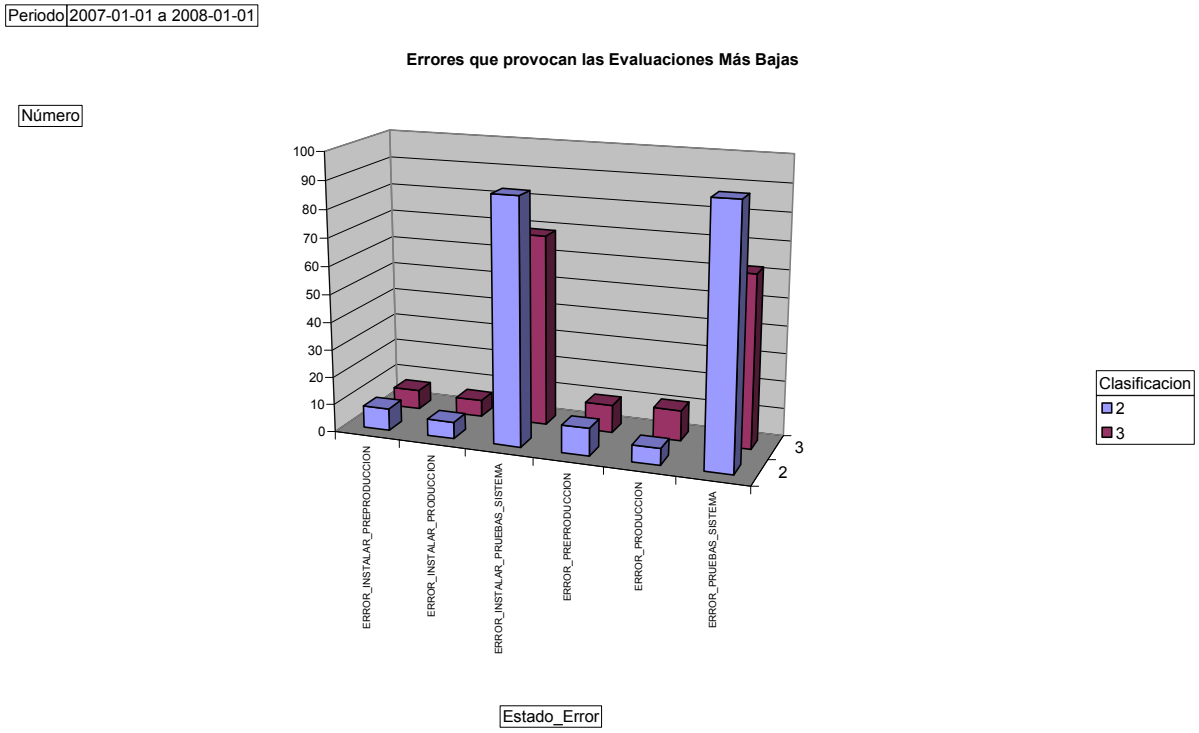


Ilustración 45 – Grafico del periodo 2007-2008 de Errores que provocan las evaluaciones más bajas

Tanto en causas más comunes que provocan error como en errores que provocan las evaluaciones más bajas, ambos periodos siguen las mismas distribuciones. Se observa, como se comentó anteriormente cómo ambos periodos tienen distintas evaluaciones en número debido a que el modelo predice *resultados parecidos*, no iguales en las evaluaciones. Si nos fijamos, la red está podando los valores extremos. En este caso no aparecen valores "1" en el nuevo periodo, ya que la red está otorgando valores 2 y 3.

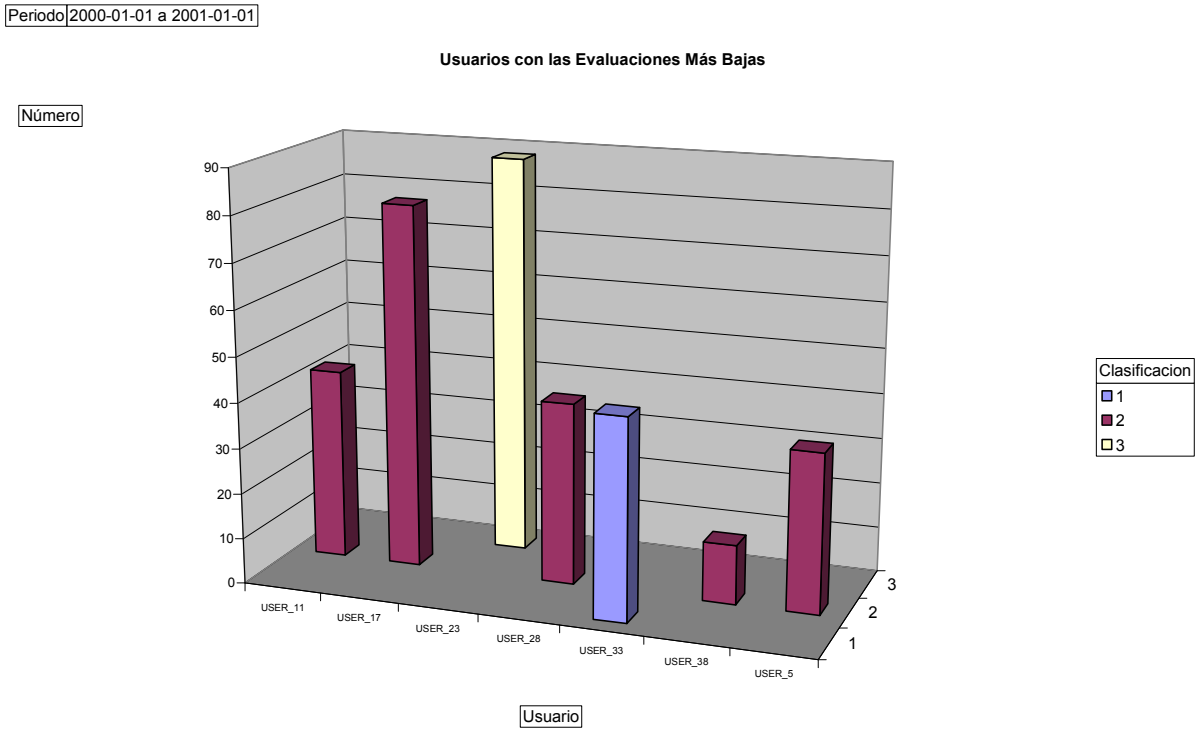


Ilustración 46 – Grafico del periodo 2000-2001 de Usuarios con las evaluaciones más bajas

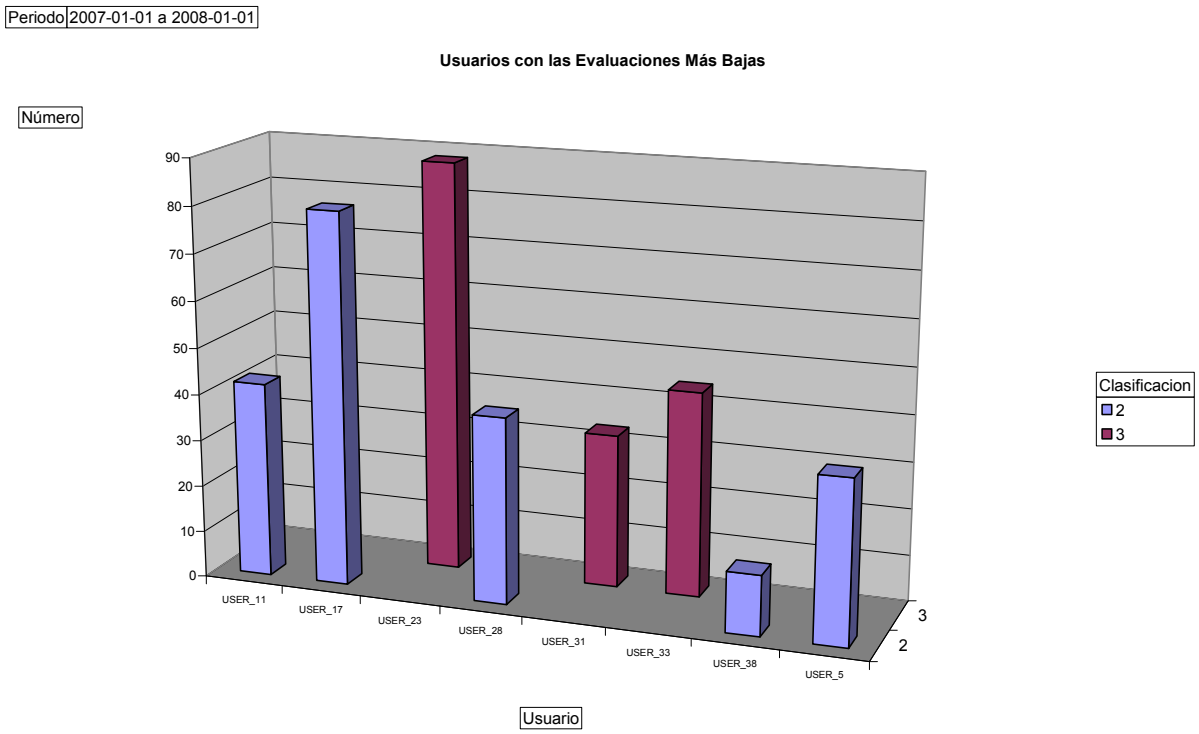


Ilustración 47 – Grafico del periodo 2007-2008 de Usuarios con las evaluaciones más bajas

Si se analizan los usuarios con las evaluaciones más bajas, únicamente se observa una diferencia. El USER_31, aparece en el periodo 2007-2008 y no en el 2000-2001. El resto aparecen en ambos. Esta diferencia se comentó anteriormente y se debe a que con los datos de dicho usuario el sistema interpreta que tiene una evaluación baja (3) ya que sigue un patrón similar al de las evaluaciones bajas a pesar de tener una evaluación media.

6.3 Conclusión a las pruebas realizadas

Una vez realizado un conjunto de pruebas de sistema y sobre un modelo predictivo, se puede afirmar que ***la solución cubre la Hipótesis de este trabajo, así como las subhipótesis planteadas***. Se detallan estas y otras conclusiones generales en la siguiente sección.

7 Conclusiones. Líneas Futuras de Trabajo.

En esta sección se comentan las conclusiones derivadas de este trabajo así como las líneas futuras de investigación que se derivan de él.

7.1 Conclusiones

Se ha implementado una solución consistente en una Herramienta para establecer un modelo predictivo en los procesos de GCS. Una vez realizadas las pruebas tanto de funcionalidad básica como de un completo modelo de validación con gran cantidad de datos históricos, se pueden derivar las siguientes conclusiones de la solución que se ha generado:

- Se ha implementado una solución de gran *utilidad para la toma de decisiones y detección de tendencias en organizaciones* utilizando información insertada en los procesos de GCS. Este aspecto es importante resaltarlo ya que va asociado a la misma motivación de la investigación realizada.

- La solución es *generalizable y adaptable* a modelos diferentes en distintas organizaciones. Cada uno de los componentes se ha pensado para su fácil adaptación a diferentes arquitecturas tanto funcionales y de proceso, como tecnológicas.

- El sistema *integra* desde el versionado de código en un repositorio, hasta la predicción de evaluaciones y detección de causas de error, pasando por la arquitectura e infraestructura de entornos de la compañía.

- El sistema permite establecer permisos sobre los diferentes módulos, según los perfiles que tenga cada uno de los usuarios, dando así la *funcionalidad de seguridad* sobre los diferentes módulos.

- Se manejan los *entornos de trabajo remotamente* desde una interfaz gráfica, dando una visión completa y general de los sistemas y entornos de trabajo de la compañía.

- Se permite realizar la *distribución del código* del repositorio en las diferentes máquinas y entornos del sistema desde dicha interfaz gráfica. Esto facilita las tareas de

despliegue de software que ocurren sobre los diferentes entornos de las diferentes aplicaciones.

- El sistema implementa una estructura de unidades de control y sus relaciones para los *procesos de Gestión de Errores, Requerimientos y Releases*. Asimismo permite relaciones entre las unidades de control y los objetos de software. Mantiene información sobre los atributos y los cambios de estado.

- El sistema permite generar *informes* de forma dinámica a través de un módulo de reportes específico. Dichos informes representan información clave para el análisis. Se integra con una herramienta de uso común.

- El sistema permite *intercambiar mensajes* entre los diferentes usuarios de la aplicación a través de un módulo de comunicación off-line.

- El sistema permite *generar datos históricos aleatorios* basándose en el número de periodos, número de usuarios, número de unidades de control y una lista de probabilidades que el usuario puede personalizar.

- El sistema permite realizar *entrenamientos y clasificaciones con una red neuronal de predicción de clasificación basándose en los patrones definidos en los ciclos de vida y en los estados de error de los mismos. Los datos de entrenamiento y clasificación son los datos de la gestión de la configuración* para las diferentes unidades de control y usuarios de la aplicación.

- *El modelo predictivo generado para un caso aleatorio es capaz de predecir los resultados en lo que se refiere a las subhipótesis de este trabajo*. Se ha mostrado que un modelo aleatorio de datos históricos sigue la distribución general de datos con los que se ha entrenado al sistema, lo que es lo previsible según una distribución aleatoria normal.

- *El modelo predictivo generado también cumple las hipótesis de trabajo en el caso de un conjunto de datos similares a los previamente clasificados*. Se ha visto que si en dos periodos diferentes se dieran unas condiciones similares, se obtendrían unos datos similares de clasificación, lo que es lo esperable para un modelo como el propuesto.

Se puede decir que ***se cumplen y validan los aspectos que se plantean como "Hipótesis" y subhipótesis en este trabajo para los modelos propuestos***. Es capaz de ayudar en la predicción de evaluaciones de los diferentes usuarios en función de los datos registrados en los procesos de GCS. Asimismo puede inferir cuáles son las causas que provocan más comúnmente los errores en los diferentes sistemas con el fin de establecer acciones correctoras para tratar de paliar la raíz de dichos errores.

Por consiguiente, *el sistema, en sus diferentes módulos y componentes, es operativo y funcional*.

7.2 Líneas de Investigación

Se indican a continuación algunas líneas por las que podrían progresar las investigaciones realizadas en este trabajo. Se van a comentar asimismo mejoras que se han detectado para el sistema de Gestión de la Configuración Evolutivo que se ha implementado.

7.2.1 Líneas futuras de Trabajo

Implementación en una Organización del Modelo.

Las pruebas realizadas en el Análisis de Viabilidad se han hecho diseñando un modelo completo que podría emular a una organización determinada. Esto ha sido necesario para poder prever los comportamientos de una organización basándose en unos datos históricos que seguían unas tendencias conocidas.

Un proyecto en sí mismo sería la implementación en una o varias organizaciones de este modelo, ya que implicaría el análisis completo de la organización, de sus procesos, de los roles y responsabilidades y el cambio cultural de los usuarios del área de sistemas para adaptarse al modelo propuesto con el fin de poder extraer conocimiento de los procesos de GCS para la ayuda en la toma de decisiones.

Descubrimiento de Conocimiento del módulo de preguntas y respuestas implementado.

Se ha implementado un "Formulario de Acciones sobre Repositorio" en el que los usuarios de las diferentes aplicaciones registran para cada cambio de versión sobre el repositorio, la respuesta a una serie de preguntas que se realizan y que son personalizables por aplicación.

Se puede continuar la investigación tratando de *inferir conocimiento con algún mecanismo de Descubrimiento de Conocimiento*, tal como las Redes Neuronales, en estas respuestas a estas preguntas. Se puede obtener *información valiosa sobre las razones* que están motivando los cambios en los diferentes objetos del repositorio de software.

En esta investigación se ha buscado el nuevo conocimiento en los elementos de control de procesos, tales como los errores, los requerimientos o los paquetes de software. Sin embargo esto aportaría nuevo conocimiento derivado del manejo de versiones, lo que representa el nivel de código fuente.

Generalización del modelo de Redes Neuronales.

El mecanismo utilizado para el Descubrimiento de Conocimiento es una Red Neuronal cuyo tipo es un perceptrón multicapa. En la implementación realizada, esta estructura permite una única salida para un conjunto de entradas, dado un número de unidades de una capa oculta. Esta elección limita el conocimiento que se puede inferir a una única variable que puede tomar un rango de valores.

Se puede estudiar *utilizar Perceptrones Multicapa con varias unidades de salida o clases, diferentes redes simultáneamente o bien otros tipos de redes neuronales con otras funciones y estructuras*, de manera que a partir de una serie de entradas se puedan inferir diferentes variables de conocimiento. Por ejemplo, a partir de las entradas propuestas para la inferencia de las Evaluaciones de los usuarios en un periodo, se podrían tratar de descubrir más datos del usuario como la capacidad de ejecución que tiene, la precisión en los trabajos o la posición del usuario en la realización de pruebas.

Evolución de topologías de Redes Neuronales.

En el esquema propuesto, se han utilizado redes neuronales para clasificar determinadas informaciones contenidas en el sistema. Las estructuras que se manejan son estructuras por defecto en función de ciertos parámetros. Una vez definidas y entrenadas, las redes quedan fijas hasta que se volviera a hacer un nuevo entrenamiento o una redefinición de estructura.

Sería una línea de investigación interesante la *posibilidad de que las arquitecturas de las redes neuronales sean evolucionadas automáticamente con la propia marcha de la empresa y acoplasen su estructura al momento de la empresa*. Para ello se podría trabajar con Algoritmos Genéticos, que pueden encontrar estructuras adecuadas a cada momento de la evolución.

Es decir, ese conjunto de parámetros fijo, tales como el número de unidades de la capa oculta, el número de iteraciones a entrenar un número determinado de ejemplos o la tasa de aprendizaje de la red, en primer lugar se tienen que definir manualmente por prueba y error hasta dar con un conjunto preciso. Igualmente se quedan fijos con el tiempo cuando la empresa está cambiando.

Se propone utilizar algún tipo de estructura como los *Algoritmos Genéticos*, para adaptar al tiempo la estructura de la red y realizar el ajuste de los parámetros según corresponda de forma dinámica.

Publicaciones.

A partir de este trabajo de investigación, se está trabajando en la generación de publicaciones para congresos, en concreto para "Internacional Conference in Software Engineering" (http://www.icse-conferences.org/) y revistas de investigación, en concreto para la revista REICIS (http://www.ati.es/reicis**)**. Asimismo se puede evaluar su presentación o la de trabajos anexos futuros en otros seminarios, congresos y revistas científicas.

7.2.2 Mejoras del Sistema

A continuación se mencionan una serie de mejoras que se han detectado para el sistema de Gestión de la Configuración Evolutivo.

Generalización del Cliente.

El cliente gráfico que se utiliza está realizado en una herramienta de uso general para interfaces gráficas para usuario: Microsoft Visual Basic. Esto implica que los clientes deben estar alojados en sistemas operativos Microsoft Windows. Pues bien, sería una mejora del sistema para posibilitar la distribución del mismo la *construcción de un cliente gráfico equivalente, o bien con las funciones principales del sistema, en otras plataformas gráficas*.

En un primer momento debiera realizarse dicha construcción para entornos X Windows, con lo que se cubrirían muchas arquitecturas UNIX/Linux. También existe la posibilidad de implementar una aplicación Java o bien una aplicación Web con un servidor http.

Acceso a información de Base de Datos desde la GUI.

Se ha observado que el Modelo de Datos de la Aplicación es complejo al tener múltiples entidades con relaciones entre ellas. La administración de los datos de este modelo en el sistema implementado se hace para la totalidad de las entidades a través de carga y modificación directa en la base de datos.

Sería una mejora en la mantenibilidad del sistema la posibilidad de *administrar estos datos desde pantallas de la aplicación*, ya que los modelos de sistemas de una compañía son muy cambiantes en el tiempo y resulta difícil manejar directamente la base de datos sin un conocimiento profundo de la aplicación.

Combinación con otras herramientas de GCS.

El sistema se ha diseñado de forma flexible para integrarse con diferentes herramientas de Gestión de la Configuración del Software, pero se ha utilizado CVS como repositorio único de versiones.

Si se pretendiera encajar la herramienta sobre una compañía que tuviera otra herramienta de GCS con otro repositorio, habría que hacer la adaptación de las interfaces correspondientes de la herramienta. Sería una línea de mejora a explorar, *identificar nuevas herramientas de uso común e integrarlas en el sistema* con el fin de flexibilizarlo aun más y poder realizar su encaje más deprisa en diferentes compañías.

Gestión desatendida de tareas a través de mensajes.

Está disponible un módulo de envío y recepción mensajes para habilitar la comunicación offline. Esto permitiría dejar tareas en ejecución o pendientes tanto para servidores como para otros usuarios del sistema.

En esta primera versión la gestión es atendida. Por ejemplo, los cálculos de redes neuronales y de gestión de informes que son ciertamente pesados computacionalmente hablando, se hacen en una sesión en línea. La mejora consistiría en *dejar lanzadas las tareas de cálculo y proceso en el servidor de forma desatendida* y seguir con el trabajo, para que cuando termine en el servidor que corresponda, deje un mensaje en el buzón del usuario para indicarle que está listo su trabajo.

Estas son las líneas de Investigación se han identificado, si bien las posibilidades abiertas son múltiples en todos los campos de aplicación de este trabajo. Tanto en el campo de la GCS, como en el del Descubrimiento de Conocimiento, la evolución e inferencia de conocimiento en procesos de GCS puede derivar conclusiones científicas de interés.

8 Apéndices

8.1 Manual de Usuario de la Aplicación ECFM

Se describe a continuación el uso básico de la aplicación ECFM. Esta sección está destinada a los usuarios de la aplicación que no sean administradores de la misma, sino que utilicen algunas de sus funcionalidades.

8.1.1 Cliente ECFM

Se describe a continuación la Interfaz Gráfica de ECFM a través de la cual el usuario podrá realizar la totalidad de sus acciones sobre el sistema.

8.1.1.1 Login en el Sistema

Al entrar en el ejecutable de la aplicación, una vez instalada en un PC de escritorio por un administrador[1], aparecerá la pantalla de login en el sistema.

Como login por defecto aparecerá el de sistema operativo del usuario que abra la sesión. Es necesario introducir el login y la password de la aplicación para acceder a la misma y

[1] Nótese que un administrador de la aplicación debe instalar la aplicación y configurar los ficheros de la misma antes de comenzar su uso.

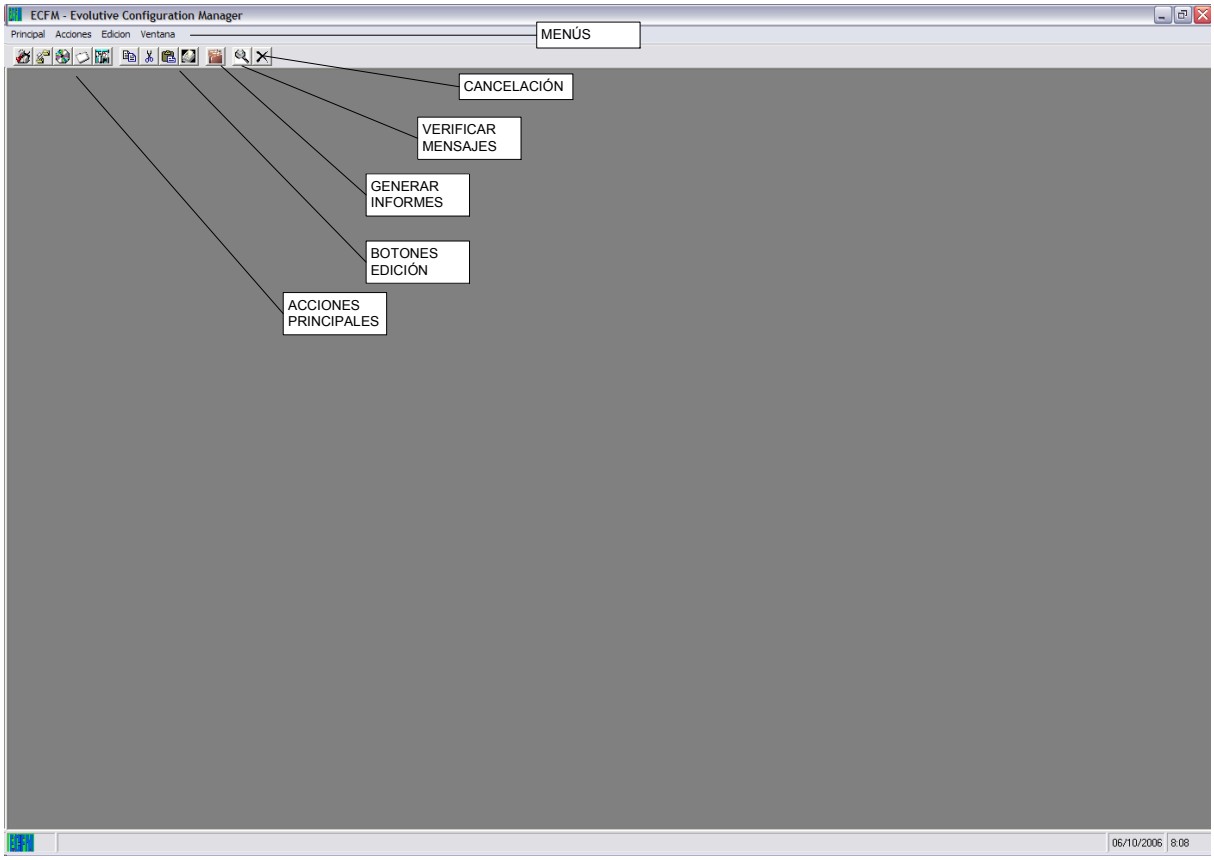
posteriormente pulsar en “Aceptar”. Aparecerá un mensaje de Error en el caso de que la contraseña introducida no sea correcta.

Este login y esta password dan acceso a un conjunto de funcionalidades del sistema, dependiendo de cómo el administrador haya asignado los permisos en el sistema. Si un usuario intenta acceder a una funcionalidad sobre la que no tiene permisos aparecerá una pantalla como la siguiente:

El usuario debe pedir permisos de acceso al administrador en caso de necesitarlos sobre las diferentes funcionalidades.

8.1.1.2 Pantalla Principal de la Aplicación

Una vez se ha hecho login en la aplicación, aparecerá la pantalla principal de la misma desde donde se controlará el resto.



En la pantalla principal de la aplicación se pueden diferenciar los diferentes Menús de Acceso, así como los botones que permiten acceder directamente a las pantallas de acciones.

La barra de menús tiene las siguientes opciones:

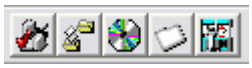
- *Principal*: Se pueden lanzar las principales acciones funcionales de la aplicación. Las pantallas que se pueden lanzar son las de Instalación, Control del Ciclo de Vida, Red Neuronal de Clasificación de Errores (sólo para Administradores), Análisis de Causas de Error e Informes y Consulta de Datos.

- *Acciones*: Se pueden lanzar desde este menú las acciones de Subir Fichero Repositorio, Recepción de Mensajes y Generar Datos (sólo para Administradores de la Aplicación)

- *Edición*: Las acciones de Copiar, Cortar, Pegar e Imprimir son lanzadas desde este menú.

- *Ventana*: Desde el menú de ventana se pueden mostrar en diferentes modos las ventanas abiertas en la aplicación.

La barra de botones de la aplicación permite acceso directo a las diferentes funcionalidades de la misma. Se dispone en diferentes secciones:

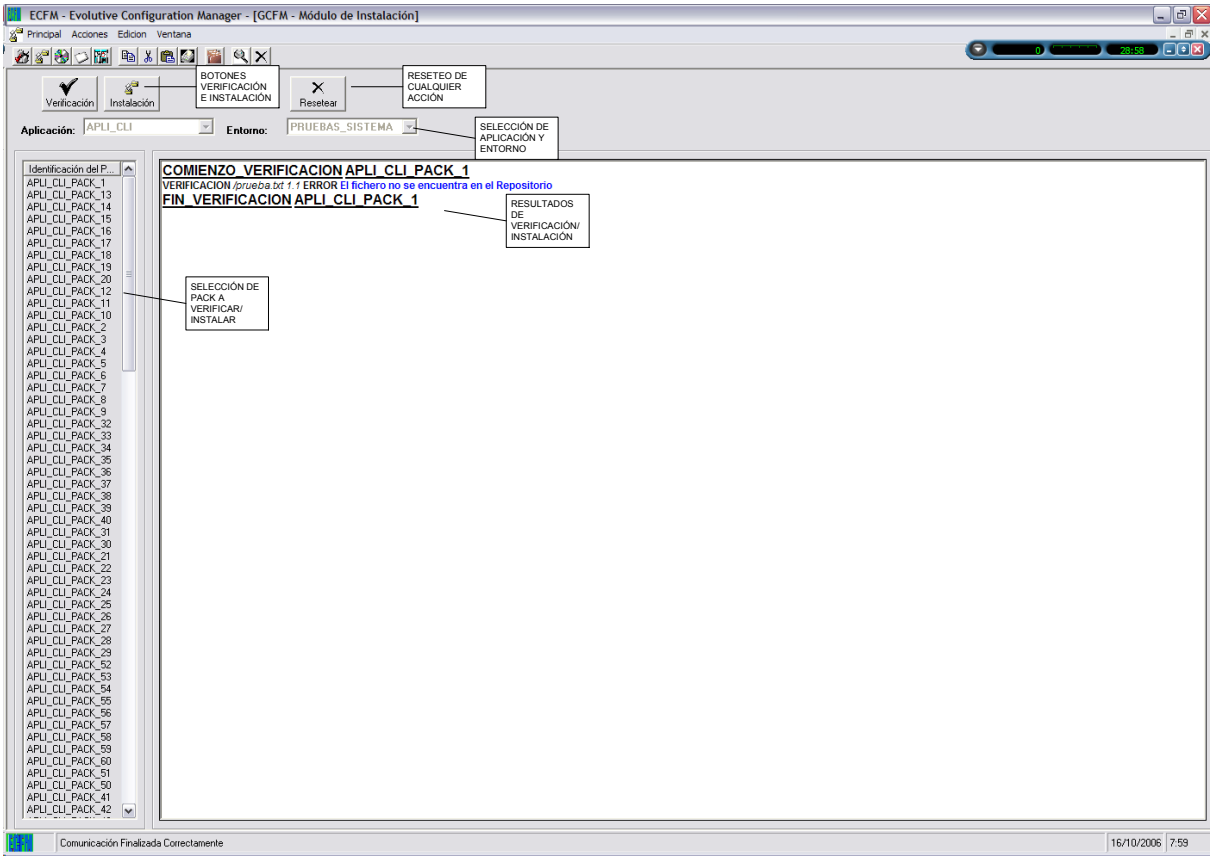
- *Acciones Principales*:  Se pueden lanzar desde este menú de botones las acciones de instalación, Actualización de Repositorio, Control del Ciclo de Vida, Red Neuronal de Gestión de Errores (solo Administradores) y Análisis de Causas de Error.

- *Edición*:  Botones para copiar, cortar, pegar e imprimir el texto seleccionado en cada momento.

- *Otras Acciones*:  Para lanzar la generación de Informes, la recuperación de mensajes y de cancelación de operaciones.

8.1.1.3 Pantalla de Instalación

Desde la pantalla de instalación se permite desplegar el software directamente contra los diferentes directorios de trabajo de cada uno de los entornos de las diferentes aplicaciones.



Como se puede observar, se tienen diferentes botones y listas de chequeo dentro de la pantalla. A continuación se describe la operativa a seguir y lo que se debe tener en cuenta en cada uno de los pasos.

- *Selección de Aplicación y Entorno*: Lo primero que se debe seleccionar es la aplicación y el entorno de trabajo de la misma sobre el que se quiere realizar la instalación o despliegue de software. Una vez seleccionados, se mostrará un listado de los Paquetes de Instalación (Unidad de Control "PACK") que están disponibles para dicha aplicación y entorno.

- *Selección de PACK a Desplegar*: Una vez mostrado el listado de PACK disponibles para la aplicación y el entorno a instalar, se deberá seleccionar aquel que se desee desplegar sobre la aplicación y el entorno seleccionado.

- *Verificación de Despliegue*: Cuando se tiene correctamente seleccionado el PACK que se desea instalar en el entorno correspondiente de trabajo, únicamente estará activo el botón de Verificación. Esto se debe a que obligatoriamente, antes de realizar una

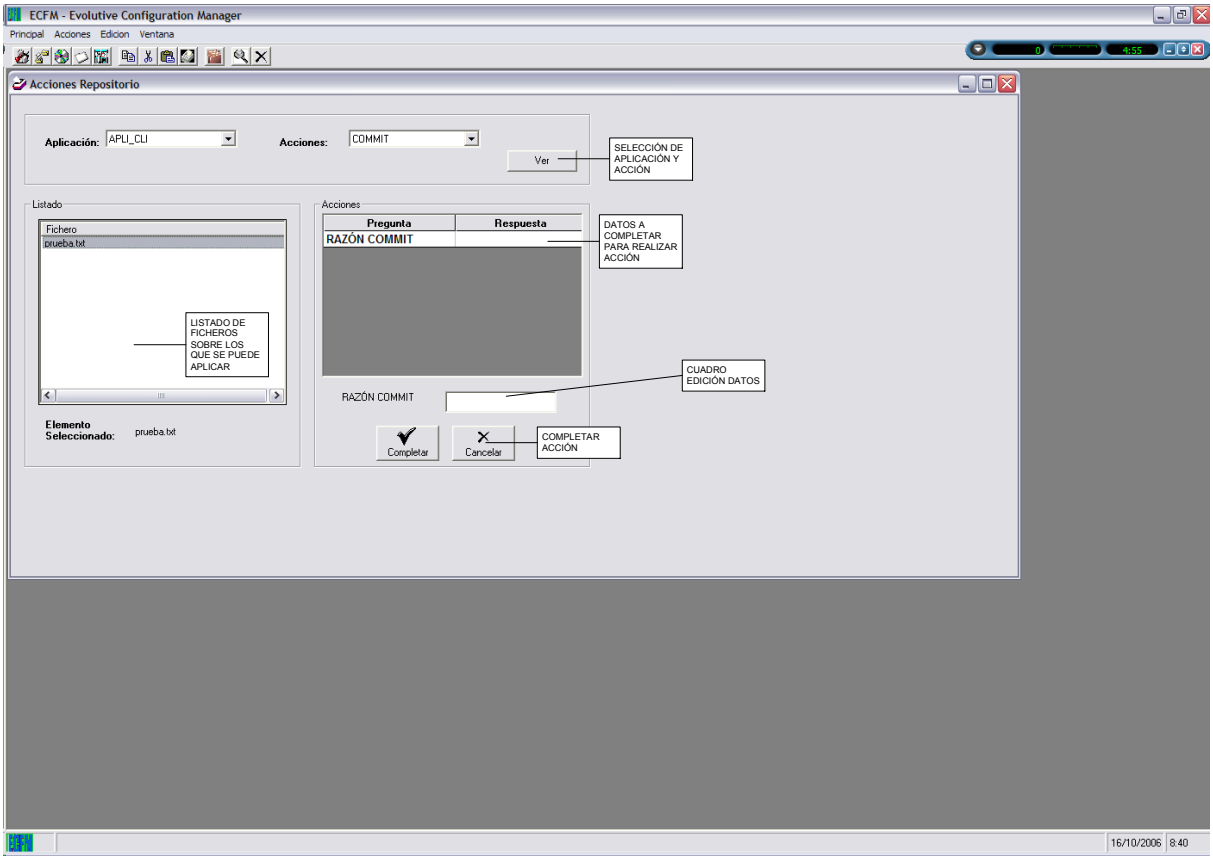
instalación, se debe hacer una verificación de coherencia del software. Este proceso chequeará cosas como que el software relacionado está en el repositorio, que no se han instalado versiones superiores o que no existen versiones superiores en el repositorio. Se mostrarán los mensajes en el cuadro de resultados de la verificación.

- *Instalación*: Al finalizar la verificación se activará el botón de instalación, pudiendo ya realizarse el despliegue de software sobre el entorno. Esta acción se realiza pulsando ese botón. Se copiará cada uno de los ficheros relacionados al PACK en el directorio correspondiente de trabajo relativo al entorno en el que se instale. Se chequeará asimismo la corrección de la copia, dejando un log o registro de la instalación.

Nótese que en medio de cualquier operación, para volver al estado inicial, se puede pulsar el botón Resetear, que interrumpirá la acción en curso en cualesquiera sea su estado.

8.1.1.4 Pantalla de Acciones de Actualización de Repositorio

Desde esta pantalla se permite realizar actualizaciones sobre ficheros de repositorio, registrando información adicional a las mismas. Se muestra a continuación la pantalla de trabajo para esta funcionalidad, con sus diferentes partes.



Se describen los *pasos y operativas* asociados a esta funcionalidad.

- *Selección de Aplicación y Acción*: En primer lugar se debe seleccionar la Aplicación y la Acción a realizar sobre la misma de las listas de "Aplicación" y "Acciones" respectivamente. Nótese que en esta versión de la aplicación, únicamente está habilitada la acción "COMMIT" que es para realizar el COMMIT con informaciones adicionales sobre los ficheros en edición de cada una de las aplicaciones. Una vez seleccionados, pulsar el botón "Ver" para mostrar el listado de elementos sobre los que aplicar la acción.

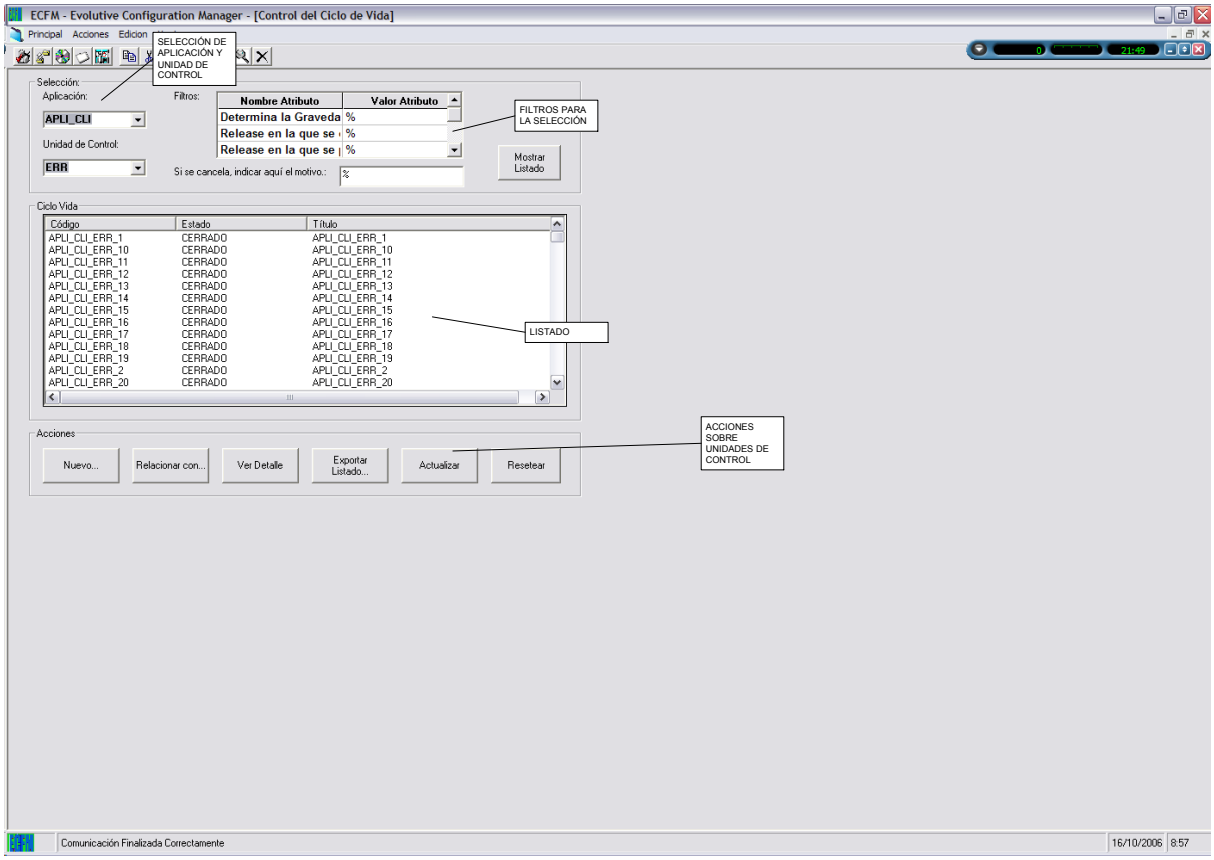
- *Selección de Fichero*: Se mostrarán los elementos que estén pendientes de realizar la acción, en caso de haberlos. En el caso de la acción COMMIT se mostrarán los ficheros que estén en edición en la copia local de trabajo, pendientes de realizar el Check In definitivo. Al seleccionar el fichero en cuestión, se mostrará en el cuadro "Acciones", la información que se debe de completar.

- *Completar Información Adicional*: En el recuadro denominado "Acciones" se mostrarán los requerimientos para completar la Acción seleccionada previamente. En el caso de COMMIT, se mostrarán una serie de preguntas, cuyas respuestas darán información adicional a la acción de Check In y que podrán ser utilizadas posteriormente. Las respuestas a dichas preguntas se completarán desde el cuadro de edición de texto.

Para finalizar la acción en curso y guardar toda la información adicional, se pulsará el Botón "Completar". Si se desea interrumpir la acción, se pulsará "Cancelar".

8.1.1.5 Pantalla de Control de Ciclo de Vida

Desde esta pantalla se realiza la gestión de las Unidades de Control de cada aplicación. Es desde esta interfaz desde donde se permitirán la creación de nuevas unidades, la relación de las mismas, los diferentes listados, los cambios de estado o la edición de parámetros de las Unidades de Control. Se muestra a continuación la pantalla de trabajo.



El modo de funcionamiento de esta pantalla implica las siguientes *operaciones*:

- *Selección de Aplicación y Tipo de Unidad de Control*: En primer lugar se seleccionará la aplicación con la que se quiere trabajar en la lista "Aplicación" y el Tipo de Unidad de Control en la lista "Unidad de Control" dentro del marco "Selección". Una vez que se seleccione aparecerán en el cuadro "Filtros", los diferentes atributos correspondientes a esa selección.

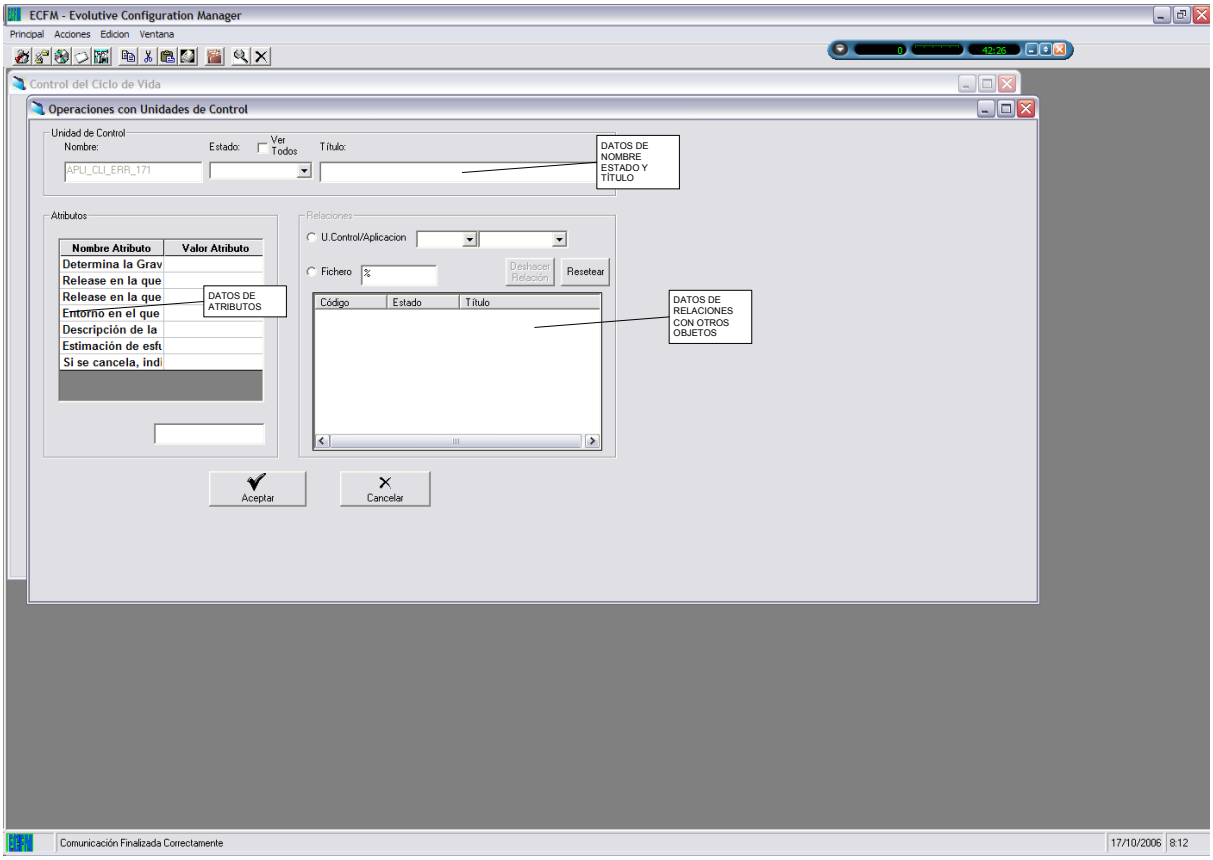
- *Aplicar filtros*: Estos filtros pueden servir para restringir el listado obtenido basándose en dichos atributos. El carácter "%" es el que indica que se busque cualquier cadena. Por ejemplo, la cadena AP%1, indica todos los literales que empiecen por AP y terminen en 1. Por defecto el valor será "%" para todos los filtros. Una vez que se han preparado los filtros se deberá pulsar "Mostrar Listado" para sacar el listado de unidades de control.

- *Selección de Unidad de Control*: En el marco "Ciclo Vida" se mostrarán diferentes datos de las unidades de control. Concretamente el Código o Identificador, el Estado y el Título de la misma. Para hacer operaciones sobre las unidades de control se seleccionará aquella con la que se desee trabajar.

Una vez seleccionada la unidad de control de trabajo, se pueden realizar las siguientes *operaciones*, desde el Marco "Acciones":

- *Crear una nueva Unidad de Control*: Con el botón "Nuevo…"
- *Relacionar con otras unidades de Control* o con otros Ficheros: Con el botón "Relacionar con…"
- *Modificar o analizar unidades de Control*: Con el botón "Ver Detalle…"
- *Sacar un listado de unidades de control a un fichero*: Con el botón "Exportar Listado" saldrá un cuadro de diálogo en el que se le indicará una ruta en la que se exportará un fichero con los datos del listado.
- *Actualizar el listado*: Con el botón "Actualizar" se actualizará el listado generado con la última información de la base de datos.
- *Resetear*: Con el botón "Resetear" se borrará el listado y las selecciones, volviendo al inicio de la pantalla.

Para las Acciones *de Creación, de Relacionar Unidades de Control y de Modificación*, se trabajará desde una pantalla como la que se muestra en el siguiente diagrama



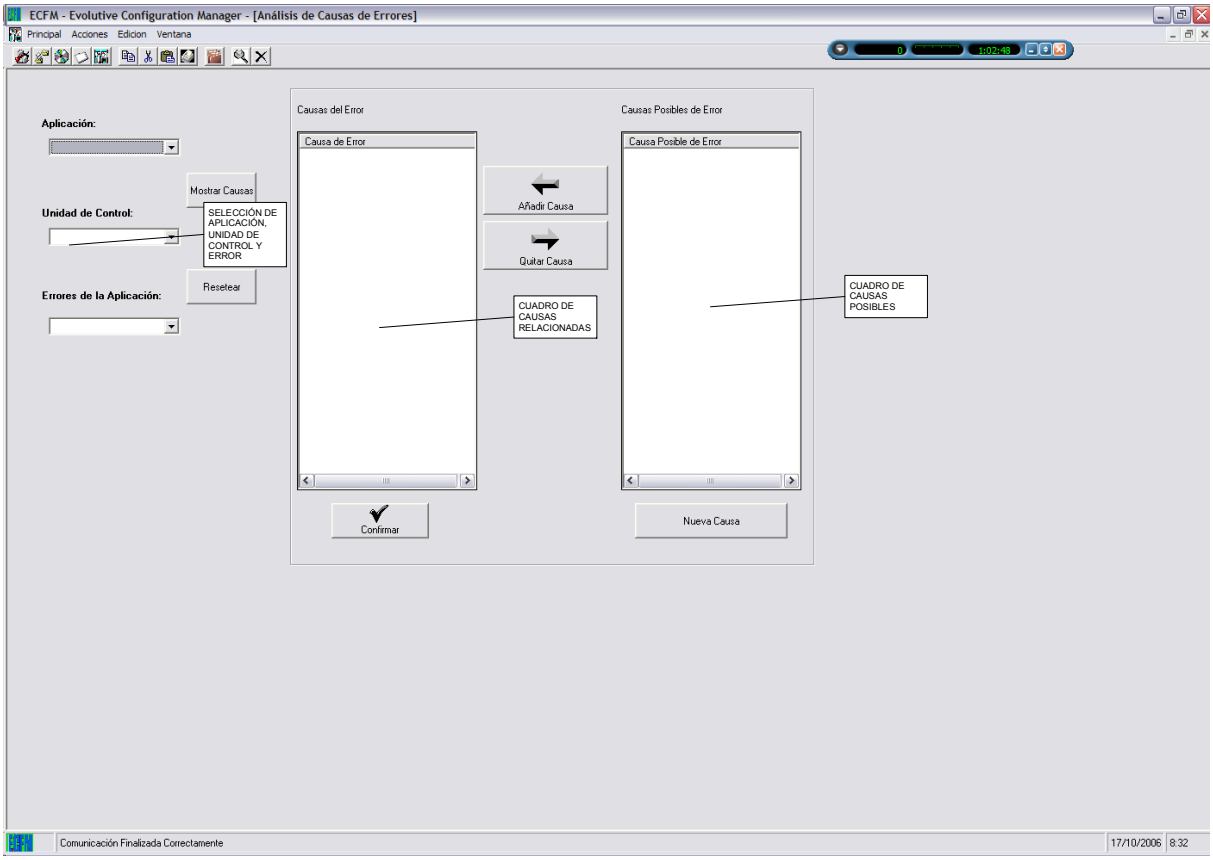
El nombre de la Unidad de Control no será modificable en ningún momento pues es un código generado automáticamente por la base de datos. En tiempo de creación y modificación se puede modificar el *Estado* en el que está del ciclo de vida, el *Título* de la Unidad de Control desde el marco "Unidad de Control", así como los datos de los diferentes *atributos* desde el marco "Atributos".

En el marco "Relaciones" se podrá *relacionar* una Unidad de Control bien con Ficheros del Repositorio, bien con otra Unidad de Control de Cualquier aplicación y tipo.

Al pulsar el Botón "Aceptar" se validarán los cambios realizados y si se pulsa "Cancelar" no se aplicarán las modificaciones realizadas.

8.1.1.6 Pantalla de Análisis de Causas de Error

Desde la Pantalla de Análisis de Causas de Error, se pueden gestionar las diferentes causas que pueden provocar los errores de las diferentes aplicaciones, así como las relaciones entre ellas. La pantalla de manejo es la que se indica a continuación.



Se explica a continuación el proceso de *manejo* de esta funcionalidad.

- *Selección de Aplicación, Unidad de Control y estado de Error*: En primer lugar, se deberá seleccionar la Aplicación con la que se trabajará desde la lista "Aplicación" y la Unidad de Control de dicha aplicación en la lista "Unidad de Control". Una vez seleccionada esta dupla la lista "Errores de la Aplicación" contendrá los estados del ciclo de vida de la aplicación que sean estados de error, y se deberá seleccionar uno de ellos.

- *Listado de Causas de Error*: Una vez seleccionadas la aplicación, la unidad de control y el estado de error a analizar, se pulsará "Mostrar Listado" para sacar el la lista "Causas del Error", las causas que pueden estar asociadas a este error.

- *Añadir o Quitar Causas*: En el recuadro "Causas Posibles de Error" se mostrarán las causas posibles para un error. Si se selecciona una causa de las posibles y se pulsa el botón "Añadir Causa", se añadirá a las causas asociadas al error seleccionado. Si se selecciona una de las causas del error y se pulsa "Quitar Causa", desaparecerá de las causas que pueden provocar ese error. Nótese que quitar una causa implicará que todas aquellos errores de ese tipo en unidades de control de esa aplicación que hubieran estado asociados a esa causa, perderán su relación.

- *Creación de Nuevas Causas Posibles de Error*: Parar añadir nuevas posibles causas de error en la base de datos, se pulsará "Nueva Causa" donde se mostrará un diálogo para introducir la nueva tipificación de causa.

A esta vista únicamente tendrán acceso los administradores de la aplicación porque conlleva cambios de índole estructural en la misma.

8.1.1.7 Pantalla de Generación de Informes

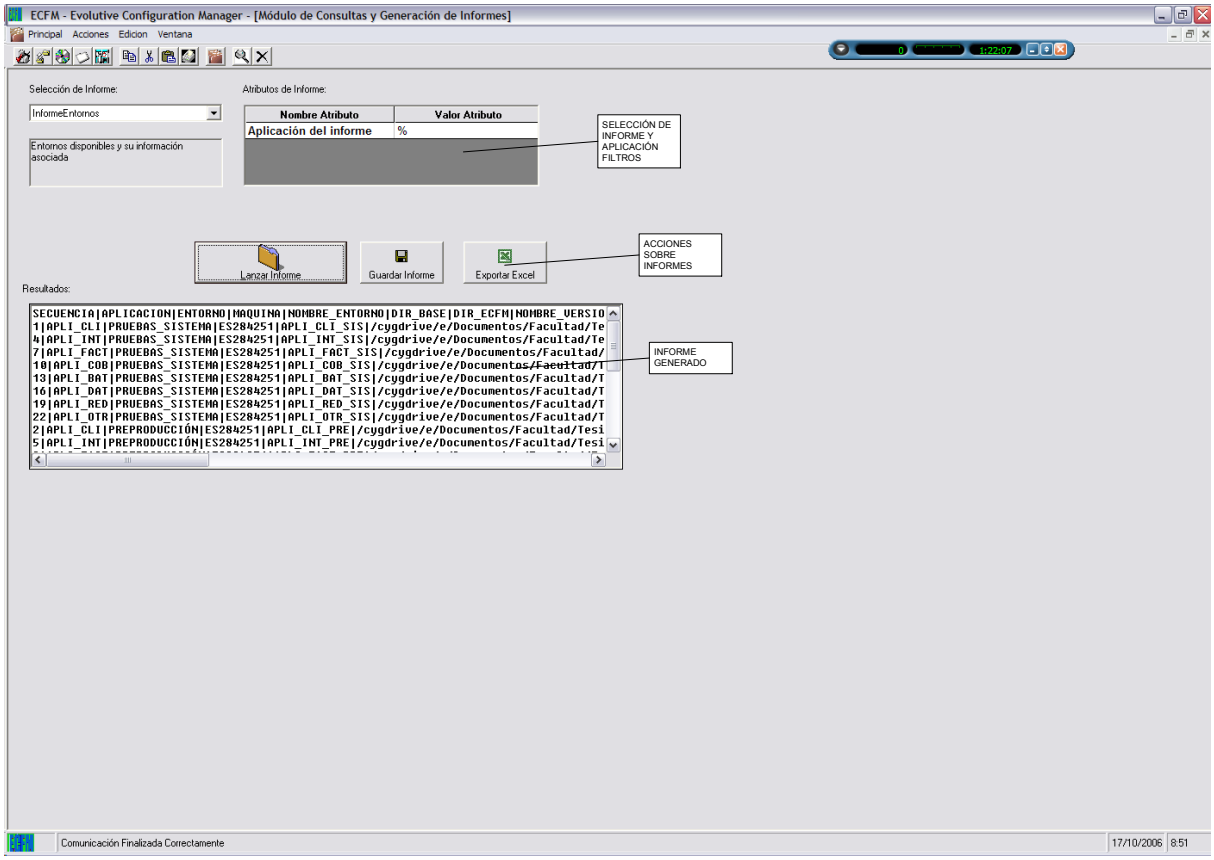
Desde esta pantalla se permite generar diferentes informes basándose en información guardada en la base de datos. Los *informes disponibles* en esta versión de la aplicación son los siguientes:

- *InformeEntornos*: Informe con la información asociada a entornos de trabajo en el que se muestra para cada uno de los entornos de trabajo las variables y directorios principales.

- *InformeANNERRORES*: Informe que extrae la clasificación de los diferentes usuarios obtenida por la red neuronal de clasificación de errores.

- *InformeCausasErrores*: Informe de análisis de las causas que han provocado los diferentes errores en los diferentes periodos.

La pantalla de informes se muestra a continuación.



La *funcionalidad* de informes se maneja como se indica a continuación:

- *Selección de Informe y Aplicación de Filtros*: Se selecciona el informe a generar y en el cuadro "Atributos de Informe" se aplican los filtros que se deseen al informe. Ver la sección "Pantalla de Control de Ciclo de Vida" para ver cómo aplicar los informes.

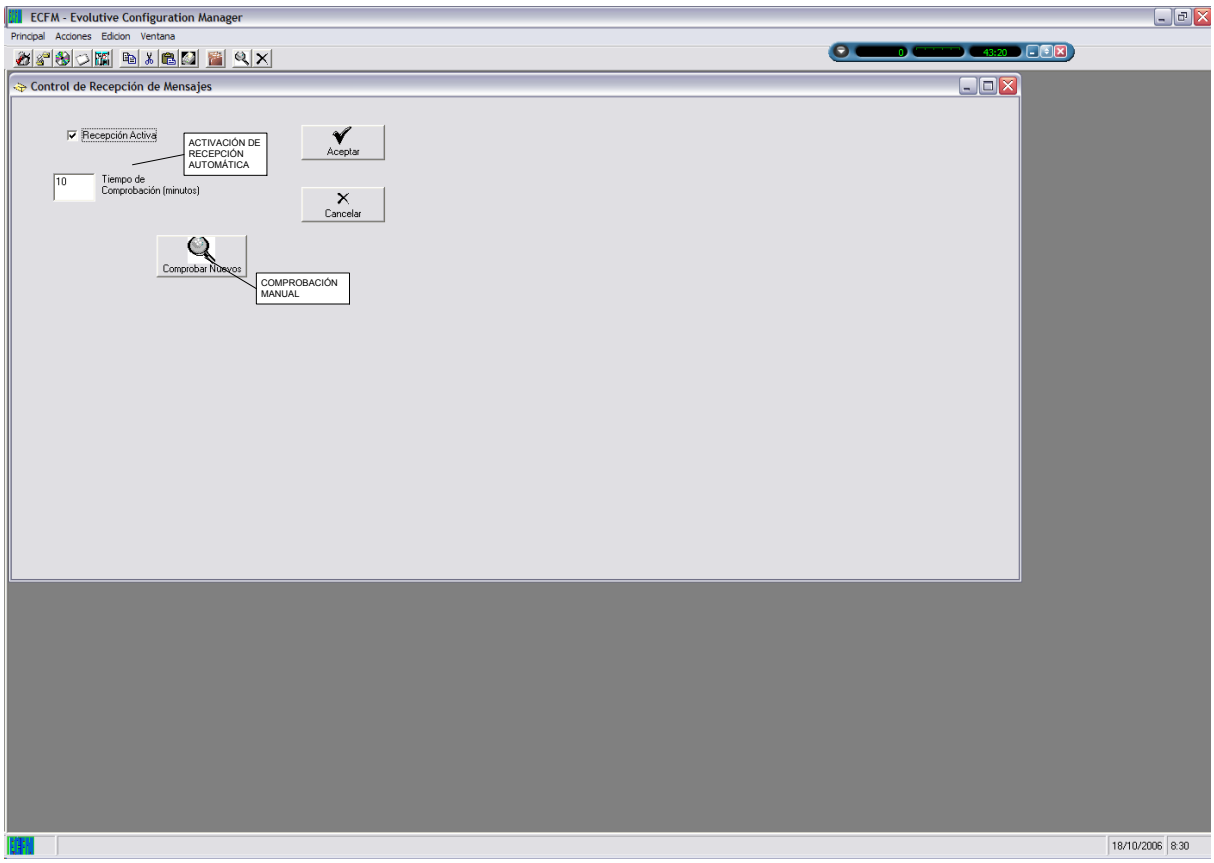
- *Lanzar Informe*: Una vez seleccionado el informe se lanzará la generación del informe. Es necesario esperar a que termine para poder exportar dicho informe.

- *Guardar Informe*: Se podrá guardar el informe en archivo separado por tabuladores con el botón "Guardar Informe".

- *Exportar Excel*: Para exportar el informe en un archivo de Excel, pulsar el botón "Exportar Excel". Esta funcionalidad permite utilizar unas plantillas prediseñadas de informes en las que se podrán ver gráficos y visualizar diferentes vistas a partir de un mismo informe.

8.1.1.8 Pantalla de Recuperación de Mensajes

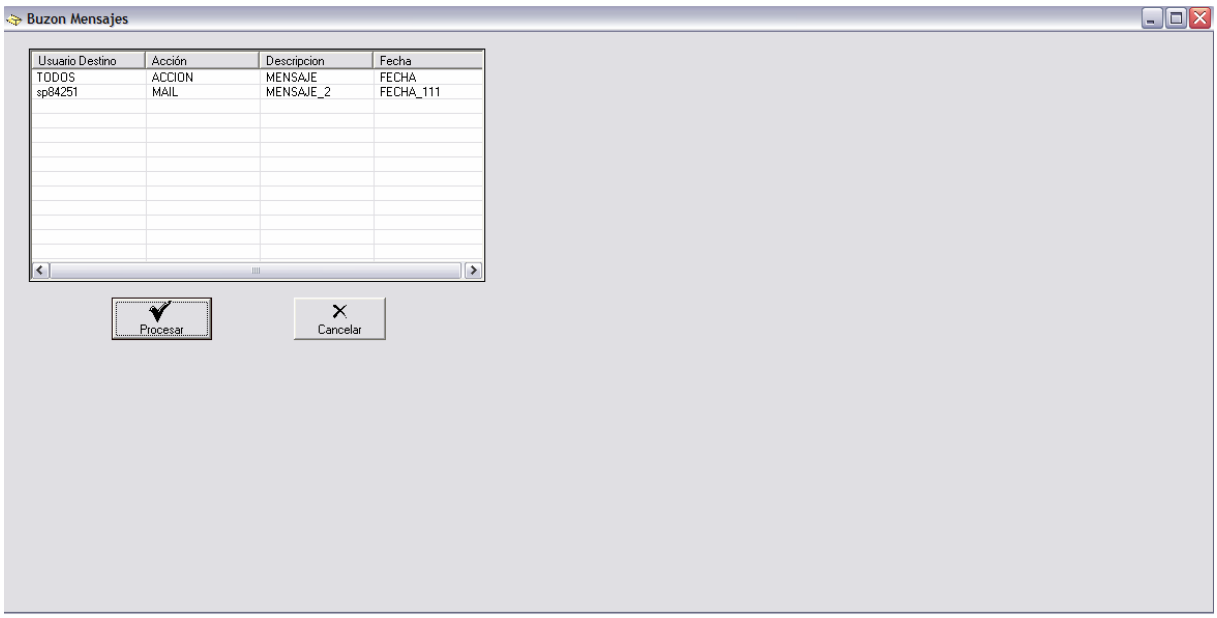
Desde esta pantalla se gestiona la *recepción de nuevos mensajes* y la comprobación automática de los buzones.



Para activar la *comprobación automática* de mensajes en el buzón de entrada, se indicará el tiempo en minutos para realizar comprobaciones de nuevos mensajes y se indicará la activación de la recepción marcando la casilla de Recepción automática y pulsando aceptar.

Para realizar una *comprobación manual*, simplemente se pulsará el botón de Comprobar Nuevos.

En el caso de existir mensajes en el buzón de entrada se mostrarán en una pantalla como la siguiente.



Para *recoger un mensaje* se debe pulsar encima y posteriormente apretar el botón "*Procesar*". Se mostrará el contenido del mensaje en un cuadro de texto en caso de que la acción sea de tipo "MAIL". Si la acción es de cualquier otro tipo, en esta versión de la aplicación no se desencadenará ningún proceso.

8.1.2 Repositorio ECFM

En la versión actual del sistema ECFM, se debe utilizar *CVS*. Es necesario que el administrador del sistema prepare el fichero de configuración "ECFM.cfg" situado en el

directorio de instalación del sistema ECFM de manera que esté relacionado con la configuración de CVS. Ver el Manual de Administración de la Aplicación para analizar la configuración en detalle de esta parte.

8.2 Manual de Administración de la Aplicación ECFM

En esta sección se van a mostrar los aspectos relacionados con la administración de la aplicación ECFM. La sección va dirigida a los usuarios administradores del sistema ECFM.

8.2.1 Software Base Relacionado

En la siguiente tabla se muestra una lista del Software necesario para cada una de las partes del sistema ECFM. No se incluye en esta relación el software propio desarrollado para el sistema, es decir el código generado que se verá posteriormente. Se analizan con detalle en la sección "Arquitectura Tecnológica".

PARTE	SOFTWARE	VERSIÓN	DESCRIPCIÓN
CLIENTE	Microsoft Windows	95 o Superior	El sistema operativo del cliente debe ser Microsoft Windows 95 o superior
CLIENTE	Visual Basic Run Time Libraries	5.0 o superior	Librerías que deben ser incluidas en los puestos cliente para que funcione el ejecutable. Vienen en el paquete de instalación.
CLIENTE	MySQL ODBC Driver	3.51	Para conectar desde aplicación con Base de Datos MySQL
CLIENTE	Microsoft Winsock Control y Otros Controles	6.0	Para realizar la comunicación con servidores se utiliza el objeto Winsock

PARTE	SOFTWARE	VERSIÓN	DESCRIPCIÓN
CLIENTE	CVS Client	1.11.17	Cliente CVS para poder acceder a los ficheros de repositorio en servidor
CLIENTE	SSH Client	2	Es necesario tener un cliente ssh2 para las conexiones al repositorio.
SERVIDOR	UNIX Windows + Cygwin	Linux, HP-UX, Sun Os, … Windows 95 o Superior Cygwin 1.5.21	El servidor puede estar alojado en un servidor cualquiera UNIX (Se ha probado con Linux Debian, HP-UX 11.1 y Sun OS) También puede estar alojado en un servidor Windo
SERVIDOR	CVS Client	1.11.17	Cliente CVS para poder acceder a los ficheros de repositorio en servidor
SERVIDOR	Librerías TORCH	3	Librerías y API de gestión de estructuras de aprendizaje automático. Se debe disponer de compilador en el sistema operativo gcc, g++ o similar según requerimientos de la librería para cada sistema.
BASE DE DATOS	MySQL Server	4.1.2	Servidor de Base de datos en el que se alojará la base de datos de la aplicación
REPOSITORIO	CVS Server	1.11.6	Servidor de Repositorio CVS donde estará creado el repositorio de trabajo con versiones
REPOSITORIO	SSH Server	2	Servidor SSH para las

PARTE	SOFTWARE	VERSIÓN	DESCRIPCIÓN
			conexiones desde cliente

8.2.2 Cliente ECFM

Se describen a continuación los aspectos de administración del cliente ECFM.

8.2.2.1 Estructura de Directorios y Archivos

Se dispone a continuación el *esquema de archivos y de directorios de la parte cliente* de la aplicación. Nótese que [PATH] es la ruta donde estará instalada la aplicación

DIRECTORIO	ARCHIVO	DESCRIPCIÓN
[PATH]	ECFM.exe	Ejecutable de la aplicación
[PATH]	ECFM_DB.dsn	Fichero con parámetros de conexión a la base de datos.
[PATH]	ECFM.cfg	Fichero de configuración de la aplicación
[PATH]/images	ECFM_logo.gif Comunicación.avi	Ficheros de imagen y vídeo que se cargan desde la aplicación.
[PATH]/informes/Plantillas/	Para cada uno de los informes habrá una subcarpeta con un fichero de plantilla de informe. InformeANNERRORES/PlantillaInformeANNERRORES.xls	Ficheros de plantilla de carga y generación de informes. Contendrán los

DIRECTORIO	ARCHIVO	DESCRIPCIÓN
	InformeCausasErrores/PlantillaInformeCausasErrores.xls InformeEntornos/PlantillaInformeEntornos.xls	macros y gráficos necesarios.
[PATH]/informes/Generados	Para cada uno de los informes habrá una subcarpeta en la que se guardarán los informes generados de cada tipo. InformeANNERRORES/ InformeCausasErrores/ InformeEntornos/	Informes generados de cada uno de los tipos disponibles en la aplicación
[PATH]/informes/Texto	Para cada uno de los informes habrá una subcarpeta en la que se guardarán los informes generados de cada tipo en modo sólo texto. InformeANNERRORES/ InformeCausasErrores/ InformeEntornos/	Informes generados de cada uno de los tipos disponibles en la aplicación en modo solo texto
[PATH]/tmp	Directorio	Debe existir el directorio para realizar operaciones temporales.

8.2.2.1.1 Archivos de Configuración

En esta sección se detallan los *archivos de configuración* necesarios, así como los valores que se deben configurar en los mismos.

FICHERO	PARÁMETROS	DESCRIPCIÓN
ECFM_DB.dsn	[ODBC] DRIVER=MySQL ODBC 3.51 Driver UID=ECFM STMT= OPTION= PORT= SERVER=LINUX-SERVER DATABASE=ECFM DESC=	Driver ODBC a utilizar Usuario de conexión Puerto (en blanco por defecto) Servidor de Base de Datos Base de datos a conectar
ECFM.cfg	[CVSWORKFOLDER] E:\Documentos\Facultad\Tesis\Work_Folder_TEST [/CVSWORKFOLDER] [CVSEXE] C\ Program Files\GNU\WinCvs 1.3\CVSNT\cvs.exe [/CVSEXE] [CVS_RSH] C\ CYGWIN\BIN\ssh.exe [/CVS_RSH] [CVS_HOME] E:\Documentos\Facultad\Tesis\Work_Folder_Tesis\TESIS\Tesis\Source\Repository\KEYS [/CVS_HOME] [MENSAJES_SERVER] NT-SERVER [/MENSAJES_SERVER] [PUERTO_MENSAJES_SERVER] 14651 [/PUERTO_MENSAJES_SERVER] [ANN_SERVER] LINUX-SERVER [/ANN_SERVER] [PUERTO_ANN_SERVER] 14651 [/PUERTO_ANN_SERVER]	Carpeta de trabajo local de CVS Ejecutable de CVS Ejecutable del cliente SSH Directorio de CVS donde estarán las claves privadas de los servidores de repositorio para conectar por SSH Servidor donde estarán los mensajes off line alojados Puerto del servidor de mensajes Servidor de Redes Neuronales Puerto del servidor de Redes Neuronales

El uso del cliente se detalla en el manual de usuario de la aplicación.

8.2.3 Servidor ECFM

Se detallan en esta sección los aspectos relativos a la Administración del Servidor ECFM.

8.2.3.1 Estructura de Directorios y Archivos

Se dispone a continuación el *esquema de archivos y de directorios de la parte servidor* de la aplicación. Nótese que [PATH] es la ruta donde estará compilado el servidor de la aplicación.

DIRECTORIO	ARCHIVO	DESCRIPCIÓN
[PATH]	ECFM_EjecutarComando.ksh ECFM_EjecutarNN.ksh ECFM_EjecutarInforme.ksh ECFM_FuncionesGenerales.ksh ECFM_ProcesarMensaje.ksh	Scripts y librerías del servidor
[PATH]	ECFM_Server	Ejecutable del Servidor
[PATH]	ECFM_Server.cfg	Fichero de configuración de la aplicación
[PATH]/CVS_	.ECFM_parametersBD.cfg	Fichero con los parámetros de conexión a la base de datos
[PATH]/CVS_	ECFM_Instala_Pack.ksh ECFM_Listado_UControl.ksh ECFM_Listado_Editables.ksh ECFM_Verifica_Pack.ksh ECFM_Listado_Ficheros.ksh ECFM_libConsultaBD.ksh ECFM_Listado_Ficheros_debug.ksh ECFM_libConsultaCVS.ksh ECFM_Listado_Packs.ksh	Scripts para la gestión de información de CVS
[PATH]/inbox/in [PATH]/inbox/processed [PATH]/inbox/in_progress [PATH]/inbox/error	Directorio de mensajes entrantes Directorio de mensajes Procesados Directorio de mensajes en proceso Directorio de mensajes erróneos	Directorios para la gestión de mensajes que utilizará el servidor de informes

DIRECTORIO	ARCHIVO	DESCRIPCIÓN
[PATH]/src_C	ECFM_Server.c LibECFM.h LibSock.h LibSock.c LibECFM.c Makefile	Directorio donde se alojan los fuentes del servidor. Se compilan con el Makefile de ese mismo directorio
[PATH]/tmp [PATH]/log	Directorio	Debe existir el directorio para realizar operaciones temporales y generar logs.

8.2.3.1.1 Archivos de Configuración

En esta sección se detallan los archivos de configuración necesarios, así como los valores que se deben configurar en los mismos.

FICHERO	**PARÁMETROS**	**DESCRIPCIÓN**
.ECFM_parametersBD.cfg	export USER_BD=ECFM export PASSWORD_BD=ppppp export BD=ECFM export HOST_BD=LINUX-SERVER	Usuario Conexión Password Conexión Base de Datos de Conexión Host del servidor de Base de Datos
ECFM_Server.cfg	PUERTO_SERVIDOR_ECFM 14651 LOG_FILE ./log/logfile.log ECFM_SERVER_DIR /cygdrive/c/Documentos/Facultad/Tesis/Work_Folder_Tesis/TESIS/Tesis/Source/Server ECFM_SERVER_MAIL ECFM_User@localhost	Puerto del servidor Fichero de log Directorio Base ECFM Dirección de envío de correos

8.2.3.2 Arranque y Parada

El Arranque se hará ejecutando, desde el PATH donde esté el ejecutable:

```
ECFM_Server
```

Se puede dejar en Background:

```
ECFM_Server &
```

Incluso, se puede ejecutar independientemente del terminal:

```
nohup ECFM_Server &
```

Para comprobar que el servidor está arriba se analizarán los procesos ECFM de la máquina:

```
ps -ef|grep ECFM_Server
```

La información que resultará será parecida a la siguiente:

```
ECFM_User          5484                    1    con       08:09:26
/cygdrive/e/Documentos/Facultad/Tesis/Work_Folder_Tesis/TESIS/Tesis/Source/
Server/ECFM_Server/ECFM_Server
```

El dato más importante es el número de proceso, resaltado en negrita en la anterior salida.

La parada del servidor se realizará mandando la señal KILL al proceso.

```
kill NUMERO_PROCESO
```

8.2.4 Repositorio ECFM

El sistema puede tener tantos servidores de repositorios como se desee, ya que cada aplicativo puede tener su código fuente en un lugar diferente.

El servidor de repositorios debe tener configurado un *CVSROOT con un repositorio inicializado debidamente,* según la configuración estándar de CVS. Este CVSROOT será referenciado desde la aplicación para obtener ficheros del repositorio.

8.2.4.1 Configuración

Es necesario para el funcionamiento del servidor, *habilitar un servidor de SSH* en el mismo. Para que todos los clientes puedan conectar por esta vía se debe configurar el acceso a través de claves públicas y privadas.

Para poder establecer y mantener una comunicación entre dos máquinas a través del protocolo SSH es necesario que esté *generada la clave pública y privada de dos usuarios*, uno en cada máquina, y se han de intercambiar las claves. Para ello:

Se añade la clave pública del servidor A al fichero known_hosts del usuario B y viceversa:

```
$ cat /opt/openssh2/etc/ssh_host_key.pub >> [HOME_DEL_USUARIO]/.ssh/known_hosts
```

Después se edita el [HOME_DEL_USUARIO]/.ssh/known_hosts, incluyendo antes de la clave, y en la misma línea, la dirección IP; y al final el nombre del usuario. Por ejemplo:

```
192.168.0.41 ssh-rsa […CÓDIGOS de CLAVE…]
```

Se añade al fichero authorized_keys de cada usuario la clave pública del otro usuario:

```
$ cat $HOME/.ssh/identity.pub >> $HOME/authorized_keys
```

Para conectarse, una vez configurado correctamente desde un cliente, la cadena CVSROOT será como sigue:

```
:ext:USER@SERVER:PATH_SERVIDOR_CVSROOT
```

Donde:

- *USER* es el usuario configurado en el servidor.
- *SERVER* es el host de repositorio cuya clave privada ha sido instalada en el cliente en el directorio HOME.
- *PATH_SERVIDOR_CVSROOT* es la ruta completa al directorio CVSROOT del servidor.

Se exportará la variable HOME en el cliente al directorio donde se tenga un subdirectorio .ssh en el que estará el fichero known_hosts en el que se incluirá la clave privada del servidor al que se va a conectar, tal y como se ha indicado anteriormente.

Deberá asimismo tener exportada la variable "CVS_RSH" al valor de la ruta donde esté el ejecutable "ssh.exe" (Por ejemplo: C:\CYGWIN\BIN\ssh.exe)

8.2.5 Base de Datos ECFM

La base de datos se creará en un servidor con *una instalación estándar de MySQL*. No se requieren características especiales. Deben permitirse las conexiones realizadas desde los posibles clientes de la aplicación. Dichas conexiones serán de tipo ODBC y también directas de SQL.

El modelo de datos de la aplicación se genera sobre una base de datos de tipo "InnoDB", propio de MySQL. Esto permite mantener las restricciones de integridad referencial entre las tablas.

Desde un cliente se podrá comprobar la conexión al servidor, tanto desde un cliente MySQL como desde las conexiones ODBC del sistema, configurando nombre de la máquina servidor, puerto (será el estándar) y usuario y contraseña de la base de datos.

8.3 Datos para realizar la validación del sistema

En este apéndice se anexan los datos para realizar la validación completa del modelo del sistema.


ModeloValidacion.zip

8.4 Detalle de Pruebas de Sistema Realizadas

Se muestran a continuación el detalle de las pruebas de sistema realizadas.

8.4.1 Módulo de Instalación

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
MI-1	Listado de PACKs de diferentes aplicaciones	Cargar datos en Base de Datos	Seleccionar aplicaciones y entornos en el módulo de instalación	Listado de PACK disponibles para la aplicación	OK	
MI-2	Verificación de PACK contra entorno	Tener Pack con ficheros relacionados	Seleccionar Pack con ficheros relacionados y después de listar verificar	Resultados de la verificación sin detectar problemas	OK	APLI_CLI_PACK_1
MI-3	Verificación de PACK sin ficheros relacionados	Tener Pack sin ficheros relacionados	Seleccionar Pack sin ficheros relacionados y después de listar verificar	Indicará que el Pack no tiene ficheros relacionados	OK	APLI_CLI_PACK_3
MI-4	Verificación de PACK contra entorno con versiones superiores instaladas	Tener Pack con ficheros relacionados. En el "log" del entorno debe haber registrada una instalación de una versión superior	Seleccionar Pack con ficheros relacionados y después de listar verificar	Indicará la versión que ha detectado como superior, dando Warning	OK	APLI_CLI_PACK_2 / APLI_CLI_PACK_4
MI-5	Verificación de PACK con versiones superiores en repositorio	Tener Pack con ficheros relacionados. En el repositorio habrá una versión superior de alguno de los ficheros relacionados	Seleccionar Pack con ficheros relacionados y después de listar verificar	Indicará el fichero que ha detectado que tiene versiones superiores en repositorio, dando Warning	OK	APLI_CLI_PACK_2
MI-6	Instalación de PACK contra entorno	Tener Pack con ficheros relacionados	Seleccionar Pack con ficheros relacionados y después de listar y verificar, instalar	Ficheros instalados en el entorno	OK	APLI_CLI_PACK_1
MI-7	Instalación de PACK sin ficheros relacionados	Tener Pack sin ficheros relacionados	Seleccionar Pack con ficheros relacionados y después de listar y verificar, instalar	Error en la instalación	OK	APLI_CLI_PACK_3
MI-8	Instalación de PACK contra entorno sin permisos de escritura	Tener Pack con ficheros relacionados y quitar los permisos de escritura de un entorno	Seleccionar Pack con ficheros relacionados y después de listar y verificar, instalar	Error en la instalación	OK	APL_CLI_PACK_4
MI-9	Resetear instalaciones y verificaciones en progreso	Tener Pack con ficheros relacionados	Lanzar instalaciones y verificaciones y pulsar "Resetear"	Se debe detener la operación en curso	OK	

8.4.2 Módulo de Acciones sobre Repositorio

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
RE-1	Listado de Ficheros pendientes en aplicación sin ficheros disponibles	Tener una aplicación en la que no haya ficheros pendientes de COMMIT	Seleccionar una aplicación y la acción COMMIT y posteriormente listar los ficheros	No saldrá ningún fichero	OK	APLI_BAT sin ficheros pendientes
RE-2	Listado de Ficheros pendientes en aplicación con ficheros disponibles	Tener una aplicación en la que haya ficheros pendientes de COMMIT	Seleccionar una aplicación y la acción COMMIT y posteriormente listar los ficheros	Saldrán los ficheros pendientes de COMMIT en el repositorio	OK	APLI_CLI
RE-3	Hacer COMMIT de un fichero disponible	Tener una aplicación en la que haya ficheros pendientes de COMMIT	Seleccionar una aplicación y la acción COMMIT y posteriormente listar los ficheros. Hacer COMMIT de un fichero	El fichero quedará subido de versión en el repositorio	OK	
RE-4	Introducir razones COMMIT	Tener una aplicación en la que haya ficheros pendientes de COMMIT	Seleccionar una aplicación y la acción COMMIT y posteriormente listar los ficheros. Hacer COMMIT de un fichero. Introducir las contestaciones a las preguntas que se hacen sobre razones de COMMIT	Comprobar razones introducidas en la base de datos	OK	

8.4.3 Módulo de Control del Ciclo de Vida

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
CC-1	Listado de Unidades de control de diferentes aplicaciones y tipos	Tener aplicaciones con diferentes unidades de control	Seleccionar aplicaciones y tipos y hacer listados	Listado de unidades de control de esa aplicación y tipo	OK	
CC-2	Listado de Unidades de control de diferentes aplicaciones y tipos aplicando filtros por atributo	Tener aplicaciones con diferentes unidades de control y algunos filtros	Seleccionar aplicaciones y tipos y hacer listados aplicando filtros por diferentes atributos	Listado de unidades de control que cumplen los filtros	OK	

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
CC-3	Creación de Unidad de Control	Seleccionar aplicación y tipo donde crear la unidad de control	Seleccionar la aplicación y tipo y pulsar "Nuevo". En la pantalla de "Operaciones con Unidades de Control" introducir el título y el valor de los atributos. Comprobar que el nombre que le da la aplicación es correcto. Pulsar Aceptar. Posteriormente en "Ver Detalle" comprobar que se guardaron correctamente los datos registrados	Unidad de Control creada correctamente	OK	
CC-4	Relacionar unidad de control con otras unidades de control	Tener aplicación con unidades de control	Listar unidades de Control y posteriormente pulsar "Relacionar con". En la pantalla de "Operaciones con Unidades de Control" pinchar en U.Control/Aplicación y seleccionar el tipo y la aplicación destino. Posteriormente seleccionar con la que se relacionará y aceptar. Posteriormente en "Ver Detalle" comprobar con las que está relacionado	Unidad de Control relacionada correctamente	OK	
CC-5	Relacionar unidad de control con otros ficheros	Tener aplicación con unidades de control	Listar unidades de Control y posteriormente pulsar "Relacionar con". En la pantalla de "Operaciones con Unidades de Control" pinchar en Fichero y seleccionar el fichero con el que hay que relacionarlo. Posteriormente seleccionar con la que se relacionará y aceptar. Posteriormente en "Ver Detalle" comprobar con las que está relacionado	Unidad de Control relacionada correctamente	OK	
CC-6	Relacionar unidad de Control con objetos ya relacionados	Tener aplicación con unidades de control previamente relacionadas con otros objetos	Tratar de relacionar la aplicación con otros objetos	Debe mostrar mensaje de error al ya estar relacionada	OK	
CC-7	Consultar Detalles de unidad de Control	Tener aplicación con unidades de control	Pulsar en "Ver Detalle" y saldrán las diferentes características de la Ucontrol, tales como Atributos, Ficheros relacionados, relaciones con otras Ucontrol	Saldrá el detalle de la Unidad de Control	OK	

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
CC-8	Exportar listado generado de unidades de control	Tener aplicación con unidades de control	Pulsar en "Exportar Listado" una vez seleccionada aplicación y tipo de Ucontrol	Se generará un listado de unidades de control	OK	
CC-9	Refrescar y resetear listados generados	Pulsar los botones "Actualizar" y "Resetear" en la pantalla de control del ciclo de vida	Pulsar los botones "Actualizar" y "Resetear" en la pantalla de control del ciclo de vida	Se limpiará el listado en caso de "Resetear" y en caso de "Actualizar" se actualizará	OK	

8.4.4 Módulo de Generación de Informes

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
IN1	Generar Informe de Entornos	Tener entornos configurados en la base de datos	Ejecutar el Informe de Entornos	Informe generado correctamente	OK	
IN2	Generar Informe de Entornos aplicando filtros	Tener entornos configurados en la base de datos	Ejecutar el Informe de Entornos aplicando filtros	Informe generado correctamente	OK	
IN3	Generar Informe de ANN de Errores	Tener datos en la base de datos para generar los diferentes errores de un periodo de ANN	Ejecutar el informe de la ANN de Errores	Informe generado correctamente	OK	
IN4	Generar Informe de Causas de Errores	Tener datos en la base de datos para generar la minería de datos sobre errores en la base de datos	Ejecutar el informe de Causas de Errores	Informe generado correctamente	OK	

8.4.5 Módulo de Comunicación Off-line

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
CO-1	Comprobar nuevos mensajes en buzón vacío	Tener correctamente configurado el servidor de mensajes. Buzón vacío del usuario	Comprobar recepción de mensajes pulsando el botón de análisis de mensajes	No aparecerá ningún mensaje	OK	
CO-2	Comprobar nuevos mensajes en buzón lleno con mensaje personal	Tener correctamente configurado el servidor de mensajes. Buzón lleno	Comprobar recepción de mensajes pulsando el botón de análisis de mensajes	Aparecerá el mensaje que hay en el buzón	OK	

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
CO-3	Comprobar nuevos mensajes en buzón lleno con mensaje multicast	Tener correctamente configurado el servidor de mensajes. Buzón lleno	Comprobar recepción de mensajes pulsando el botón de análisis de mensajes	Aparecerá el mensaje que hay en el buzón	OK	
CO-3	Activar y comprobar recepción automática	Tener correctamente configurado el servidor de mensajes. Buzón lleno	Comprobar que se reciben mensajes activando previamente la recepción automática	Saltarán las alarmas de mensajes	OK	
CO-4	Desactivar recepción automática	Tener correctamente configurado el servidor de mensajes. Buzón lleno	Comprobar que no se reciben mensajes desactivando previamente la recepción automática	No saltarán las alarmas de mensajes	OK	

8.4.6 Módulo de Red Neuronal de Clasificación de Errores

Las pruebas de este módulo se muestran en la sección "Validación de Modelo Predictivo del Sistema" ya que su funcionamiento debe ser probado con volumen de datos y comprobando que realiza los análisis y predicciones adecuados.

8.4.7 Módulo de Análisis de Causas de Error

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
AE-1	Mostrar las causas asociadas a diferentes aplicaciones, unidades de control y posibles Errores de la aplicación	Tener en la base de datos causas asociadas a tipos de error, aplicaciones, y unidades de control	Seleccionar aplicación, unidad de control y tipo de error. Pulsar Mostrar Causas	Se mostrarán las causas asociadas a la selección	OK	
AE-2	Añadir causa a una aplicación, unidad de control y error de la aplicación	Tener en la base de datos causas asociadas a tipos de error, aplicaciones, y unidades de control	Seleccionar aplicación, unidad de control y tipo de error. Pulsar Mostrar Causas. Posteriormente añadir una causa de la columna total de causas	Causa añadida a la aplicación, al tipo y a la causa de error	OK	

CASO	DESCRIPCIÓN	PRECONDICIONES	ENTRADAS Y PASOS A SEGUIR	RESULTADOS ESPERADOS	RESULTADOS OBTENIDOS	COMENTARIOS
AE-3	Borrar causa a una aplicación, unidad de control y error de la aplicación	Tener en la base de datos causas asociadas a tipos de error, aplicaciones, y unidades de control	Seleccionar aplicación, unidad de control y tipo de error. Pulsar Mostrar Causas. Posteriormente eliminar una causa de la columna de causas de la aplicación	Causa eliminada de la aplicación, al tipo y a la causa de error	OK	
AE-4	Introducir una nueva causa posible	Tener en la base de datos causas asociadas a tipos de error, aplicaciones, y unidades de control	Seleccionar aplicación, unidad de control y tipo de error. Pulsar Mostrar Causas. Posteriormente pulsar "Nueva Causa" que introducirá un campo en la base de datos	Causa añadida al total de tipificaciones de causas	OK	

9 Bibliografía

En esta sección se incluyen las referencias bibliográficas de soporte a este trabajo de Investigación.

CÓDIGO	REFERENCIA
[Aha-91]	David W. Aha, D. W., Kibler, D. Albert, M. K. "Instance-based learning algorithms.", Machine Learning, 6(1):37-66 (1991)
[Ber-90]	Berliner, B. "CVS II: Parallelizing Software Development" (1990)
[Bru-03]	Brucker, A. "The CVS-Server Case Study: A Formalized Security Architecture", Electronic Notes in Theoretical Computer Science 80 (2003)
[Ced-05]	Cederqvist et al, "Version Management with CVS" (2005)
[Che-01]	Chen, A. et al, "CVSSearch: Searching through Source Code using CVS Comments" (2001)
[Cla-89]	Clark, P. and Boswell, R. "Rule induction with CN2: Some recent improvements.", In Y. Kodratoff, editor, Proceedings of the 5th European Working Session on Learning, Springer-Verlag, 151–163 (1989)
[Col-02]	Collobert, R., Bengio, S., Marièthoz, J., "TORCH: A Modular Machine Learning Software Library", IDIAP Research Report (2002)
[Dar-92]	Dart, S., "Configuration Management Tutorial", Carnegie-Mellon University Technical Report (1992)
[Dar-93-1]	Dart, S., "Concepts in Software Configuration Management", Carnegie-Mellon University Technical Report (1993)

CÓDIGO	REFERENCIA
[Dar-93-2]	Dart, S., "Configuration Management Plans: The beginning of your CM solution", Carnegie-Mellon University Technical Report (1993)
[Dei 90]	Deitz. D. "Pulling the Data Together". Mechanical Engineering , (February 1990).
[del-01]	de las Peñas, J. “Sistemas de Gestión de la Configuración en Telecomunicaciones”, Trabajo Independiente Doctorado, UAM (2001)
[del-02]	de las Peñas, J., "Evolución de Redes Neuronales Mediante Algoritmos Genéticos", Trabajo de Iniciación a la Investigación, UAM (2002)
[Esp-97]	Esposito, F., Malerba, D., Semeraro, K., Kay, Y., “A comparative analysis of methods for pruning decision trees.”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5):476-491 (1997).
[Est-00]	Estublier, J. "Software Configuration Management: A RoadMap", Technical Report, Grenoble University(2000)
[Est-94]	Estublier, J., Casallas, R., "Configuration Management", Chapter 4, Ed. Tichy (1994)
[Fah-90]	Fahlman, S., Lebiere, C., “The Cascade-Correlation Learning Architecture.”, Advances in Neural Information Processing Architecture 2, D.S. Touretzky, Ed. pp. 524-532. Morgan Kauffman, Los Altos CA (1990)
[Fei-90]	Feiler, P. H., "Software Configuration Management, Advances in software development environments", Carnegie-Mellon University Technical Report (1990)

CÓDIGO	REFERENCIA
[Fru-99]	Frühauf, K., Zeller, A., "Software Configuration Management: State of the Art, State of the Practice", Proceedings of the Dagstull Seminar (1999)
[Gol-89]	Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning. Adisson-Wesley Publishing Company (1989)
[Gra-99]	Graves, T., Karr, A., Marron, J., Siy, H. "Predicting fault incidence using software change history", IEEE Transactions on Software Engineering, VOL. XX (1999)
[Gro-00]	Groth, R., Data Mining: Building Competitive Advantage. (2000)
[Has-05]	Hassan,A. "Mining Software Repositories to Guide Software Development" In Submitted to the 26th International Conference on Software Engineering (2005)
[Heb-49]	Hebb, D. O., "The Organization of Behavior". New York: John Wiley (1949)
[Hol-75]	Holland, J.H., Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
[IEE-87]	IEEE Guide to Software 1987. IEEE/ANSI Standard 1042-1987.
[Jen-96]	Jensen, F. V. "An introduction to Bayesian networks". Taylor and Francis, London (1996)
[Kit-94]	Kitano, H., "Neurogenetic Learning: An Integrated Method of Designing and Training Neural Networks Using Genetic Algorithms", Physica D, vol. 75, pp. 225 228 (1994)

CÓDIGO	REFERENCIA
[Koz-92]	Kozza, J., Genetic Programming: On the Programming of Computers Means Natural Selection, MIT Press (1992)
[MAP-01]	Ministerio de Aministraciones Públicas, "Metodología Métrica - Versión 3. Gestión de la Configuración" (2001)
[Mar-86]	Marzullo, K. and Wiebe, D., "Jasmine: A Software System Modelling Facility". Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, pp. 121 130 (1986)
[Mat-97]	Mattison, R., Data Mining and Data Warehousing in Telecommunications. John Wiley & Sons, New York (1997)
[McC-43]	McCulloch, W.S., Pitts, W., "A logical calculus in the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, 5. (1943)
[McL-86]	McLelland, J., Rumelhart, D., Parallel Distributed Processing: Explorations in the Microestructure of Cognition, MIT Press, pp. 318-362 (1986)
[Min-69]	Minsky, M. L. and Papert, S. A. Perceptrons: Introduction to Computational Geometry. MIT Press (1969)
[Mit-97]	Mitchell, T. Machine Learning. McGraw-Hill (1997)
[Mon-89]	Montana, D., "Training Feedforward Networks Using Genetic Algorithms.", Proceedings of the 11th International Conference in Artificial Intelligence, pp. 762-767 (1989)
[Pea 98]	Pearl, J., “Probabilistic reasoning in intelligent systems networks of plausible inference.”, Morgan Kaufmann (1988)

CÓDIGO	REFERENCIA
[Per-99]	Pérez, E., "Minería de Datos: Fundamentos, Métodos y Aplicaciones.", Revista de la Asociación de Técnicos de Informática, 138 (1999)
[Puj-99]	Pujol, J. C., Poli, R., "Evolving the Architecture end Weights of Neural Networks Using a Weight Mapping Approach.", Technical Report CSRP-99-05 (1999)
[Qui-86]	Quinlan, J. R., "Induction of decision trees", Machine Learning, 1(1):81-106, (1986)
[Qui-93]	Quinlan, J. R. "C4.5 programs for machine learning.", Morgan Kaufmann (1993)
[Qui-96-1]	Quinlan, J. R. "Bagging, boosting, and C4.5.", In Proc. 13th National Conference on Artificial Intelligence, pag. 725-730, Cambridge, MA (1996).
[Qui-96-2]	Quinlan, J. R. "Improved use of continuous attributes in C4.5". Journal of Artificial Intelligence Research, 4:77-90 (1996)
[Rad-98]	Radi, A., Poli, R. "Genetic Programming Can Discover Fast and General Learning Rules.", Proceedings of the Third Annual on Genetic Programming Conference, pp. 314-322 (1998)
[Roh-95]	R. Rohwer , M. Wynne-Jones , F. Wysotzki, "Neural networks, Machine learning, neural and statistical classification" - Chapter 6, Ellis Horwood, Upper Saddle River, NJ (1995)

CÓDIGO	REFERENCIA
[Ros-58]	Rosenblatt F., “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, Psychological Review, vol.65, 386-408. (1958)
[Say-01]	Sayjad, J., Lethbridge, T., Matwin, S., "Supporting Software Maintenance by Mining Software Update Records" (2001)
[Sig-93]	Sigüenza, J.A., Neurocomputación: Cómo Funciona el Cerebro., Ed. Eudema(1993)
[Whi-01]	Whitgift, D, "Software Configuration Management:Methods and Tools." (2001)
[Wid-60]	Widrow, B., Hoff, M.E., "Adaptative switching circuits.", IRE Wescon Convention Record, 4 (1960)
[Yao-99]	Yao, X., "Evolving Artificial Neural Networks.", Proceedings of the IEEE, 87, pp. 1423-1447 (1999)