

Tesis de Magíster

En Ingeniería de Software

Identificación de Riesgos de Proyectos de Software en Base a Taxonomías

Autor: Lic. Sebastián Darío Maniasi

Directores

M. Ing. Paola Britos
M. Ing. Eduardo Diez

Córdoba, 2005

Dedicatoria:

A mis padres María Rosa y Juan Carlos, por su cariño constante, por su esfuerzo permanente y por su apoyo incondicional.

A mis amigos, por estar desde siempre y para siempre y por darme magníficos consejos que a veces no se que necesito.

A mis compañeros y a mis profesores por alentarme a comenzar, por respaldarme para seguir y por ayudarme a terminar con este desafío.

Je Me Souviens!

Resumen

La Administración de Riesgos es la práctica compuesta de procesos, métodos y herramientas que posibilita la gestión de los riesgos en un proyecto y que provee de un entorno disciplinado para la toma de decisiones preactiva en base a determinar constantemente que puede ir mal, identificar cuales son los riesgos mas importantes en los cuales enfocarse e implementar estrategias para gestionarlos; esta actividad se inicia en la primera etapa de un proyecto de software y se desarrolla a lo largo de todo su ciclo de vida.

La Identificación de Riesgos en proyectos de software consiste en la determinación de elementos de riesgos potenciales mediante la utilización de algún método consistente y estructurado.

La Administración de Riesgos en base a Taxonomías implica el utilizar una estructura agrupadora de los mismos de acuerdo a sus diferentes clases como una lista de consulta durante la actividad de Identificación de los Riesgos.

El sistema desarrollado en este trabajo es una herramienta que brinda la posibilidad de efectuar tareas de Identificación de Riesgos en base a Taxonomías estándar en organizaciones de software de cualquier tipo.

Tabla de Contenidos

1. INTRODUCCIÓN 3

1.1. SOBRE LA TESIS DE MAGÍSTER 3
1.2. AUDIENCIA 3
1.3. ORGANIZACIÓN DE LA TESIS DE MAGÍSTER 4

2. DOMINIO DE LA APLICACIÓN 9

2.1. INTRODUCCIÓN 9
2.2. LOS RIESGOS 13
2.3. LA GESTIÓN DE RIESGOS 20
2.3.1. Modelo de Gestión de Riesgos 25
2.3.2. Estrategias para Tratar con Riesgos 27
2.4. IDENTIFICACIÓN DE RIESGOS 28
2.4.1. Técnicas de Identificación de Riesgos 30
2.5. TAXONOMÍAS 32
2.6. LA ADMINISTRACIÓN DE PROYECTOS Y CMMI 35
2.6.1. Gestión de Riesgos (CMMI) 39

3. ESTUDIO DE VIABILIDAD DEL SISTEMA (EVS) 47

3.1. INTRODUCCIÓN 47
3.2. EL ESTUDIO DE VIABILIDAD EN MÉTRICA VERSIÓN 3 47
3.3. ESTABLECIMIENTO DEL ALCANCE DEL SISTEMA 49
3.4. ESTUDIO DE LA SITUACIÓN ACTUAL 51
3.5. DEFINICIÓN DE REQUISITOS DEL SISTEMA 52
3.5.1. Descripción General 52
3.5.2. Requerimientos de Ingeniería 54
3.5.3. Preparación para la Distribución 56
3.6. ESTUDIO DE ALTERNATIVAS DE SOLUCIÓN 57
3.6.1. Active Risk Manager (ARM) 57
3.6.2. Technical Risk Identification and Mitigation System (TRIMS) 58
3.6.3. RiskTrak 60
3.6.4. WelcomRisk 61
3.6.5. Estudio de Viabilidad para el Proyecto de Desarrollo de Tesis de Magíster 63
3.7. VALORACIÓN DE LAS ALTERNATIVAS 67
3.8. SELECCIÓN DE LA SOLUCIÓN 68
3.8.1. El Sistema a Desarrollar 68

4. ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI) 73

4.1. INTRODUCCIÓN 73
4.2. EL ANÁLISIS DEL SISTEMA EN MÉTRICA VERSIÓN 3 73
4.3. UN NUEVO MODELO PARA LA IDENTIFICACIÓN DE RIESGOS EN BASE A TAXONOMÍAS 75
4.3.1. MIRT - Modelo para la Identificación de Riesgos en base a Taxonomías 76
4.4. TAXONOMÍA DE RIESGOS DEL MAGÍSTER 80
4.5. ENTORNO TECNOLÓGICO DEL SISTEMA 81
4.6. CATÁLOGO DE INVOLUCRADOS 82
4.7. DEFINICIÓN DEL SISTEMA Y ESTABLECIMIENTO DE REQUISITOS 82
4.7.1. Introducción 83
4.7.2. Descripción general 83
4.7.3. Requerimientos específicos 84
4.8. IDENTIFICACIÓN DE SUBSISTEMAS DE ANÁLISIS 90
4.9. ANÁLISIS DE CASOS DE USO Y CLASES 90
4.9.1. Diagrama de Casos de Uso 90
4.9.2. Definición de Actores 91
4.9.3. Definición de Casos de Uso 91
4.9.4. Diagrama de Clases 99

4.10. DEFINICIÓN DE INTERFACES DE USUARIO 110
4.10.1. Principios Generales de la Interfaz de Usuario 110
4.10.2. Menús del Sistema 110
4.10.3. Usando el Sistema 111
4.10.4. Formatos de Impresión 126
4.11. ANÁLISIS DE CONSISTENCIA 127
4.11.1. Verificación de los Modelos 127
4.11.2. Resultado del análisis de consistencia 127
4.12. PLAN DE PRUEBAS 127
4.12.1. Introducción 127
4.12.2. Alcance 127
4.12.3. Audiencia 128
4.12.4. Ítems y características a probar 128
4.12.5. Características que no van a ser probadas 128
4.12.6. Aproximación 128
4.12.7. Criterio de Paso/Falla 131
4.12.8. Criterio de suspensión y reiniciación de pruebas 131
4.12.9. Artefactos de las pruebas 132
4.12.10. Actividades de prueba 132
4.12.11. Necesidades de ambiente 132
4.12.12. Responsabilidades 133
4.12.13. Necesidades de personal, calendario y entrenamiento 134
4.12.14. Riesgos y contingencias de pruebas 134
4.13. APROBACIÓN DEL ANÁLISIS 134

5. DISEÑO DEL SISTEMA DE INFORMACIÓN (DSI) 137

5.1. INTRODUCCIÓN 137
5.2. EL DISEÑO DEL SISTEMA EN MÉTRICA VERSIÓN 3 137
5.3. DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA 139
5.3.1. Definición de Niveles de la Arquitectura 139
5.3.2. Identificación de Requisitos de Diseño y Construcción 140
5.3.3. Especificación de Excepciones 141
5.3.4. Especificación de Estándares y Normas de Diseño y Construcción 142
5.3.5. Identificación de Subsistemas de Diseño 143
5.3.6. Especificación del Entorno Tecnológico 145
5.3.7. Especificación de Requisitos de Operación y Seguridad 145
5.4. DISEÑO FÍSICO DE DATOS 146
5.5. DISEÑO DE LA ARQUITECTURA DE SOPORTE 151
5.5.1. mvc 151
5.5.2. JasperReports 155
5.5.3. persistence 157
5.5.4. i18n 160
5.5.5. JEOPS 169
5.6. DISEÑO DE CASOS DE USO REALES 173
5.6.1. Enfoque Seleccionado 173
5.6.2. Diagramas de Robustez 173
5.6.3. Revisión de la Interfaz de Usuario 180
5.7. REGLAS DE IDENTIFICACIÓN DE RIESGOS 181
5.7.1. Seudoreglas: 182
5.8. DISEÑO DETALLADO DE CLASES 201
5.9. VERIFICACIÓN Y ACEPTACIÓN DE LA ARQUITECTURA DEL SISTEMA 202
5.9.1. Verificación de las Especificaciones de Diseño 202
5.9.2. Análisis de Consistencia de las Especificaciones de Diseño 202
5.9.3. Aceptación de la Arquitectura del Sistema 202
5.10. GENERACIÓN DE ESPECIFICACIONES DE CONSTRUCCIÓN 202
5.10.1. Especificación del Entorno de Construcción 202
5.10.2. Definición de Componentes y Subsistemas de Construcción 203
5.10.3. Plan de Integración del Sistema 205
5.11. DISEÑO DE LA MIGRACIÓN Y CARGA INICIAL DE DATOS 205
5.12. ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS 206

5.13. ESTABLECIMIENTO DE REQUISITOS DE IMPLANTACIÓN 220
5.14. APROBACIÓN DEL DISEÑO DEL SISTEMA 220

6. CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI) 225

6.1. INTRODUCCIÓN 225
6.2. LA CONSTRUCCIÓN DEL SISTEMA EN MÉTRICA VERSIÓN 3 225
6.3. PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONSTRUCCIÓN 226
6.3.1. Implantación de Archivos 227
6.3.2. Preparación del Entorno de Construcción 227
6.4. GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES 227
6.5. EJECUCIÓN DE LAS PRUEBAS UNITARIAS 228
6.5.1. Preparación del Entorno de las Pruebas Unitarias 228
6.5.2. Realización y Evaluación de las Pruebas Unitarias 228
6.6. EJECUCIÓN DE LAS PRUEBAS DE INTEGRACIÓN 228
6.6.1. Preparación del Entorno de las Pruebas de Integración 228
6.6.2. Realización y Evaluación de las Pruebas de Integración 229
6.7. EJECUCIÓN DE LAS PRUEBAS DEL SISTEMA 229
6.7.1. Preparación del Entorno de las Pruebas del Sistema 229
6.7.2. Realización y Evaluación de las Pruebas del Sistema 229
6.8. ELABORACIÓN DE LOS MANUALES DE USUARIO 230
6.8.1. Requisitos de Hardware y Software 230
6.8.2. Instalación de la Aplicación Seldon 230
6.8.3. Configuración de la Aplicación Seldon 233
6.8.4. Ejecución de la Aplicación 235
6.9. DEFINICIÓN DE LA FORMACIÓN DE USUARIOS FINALES 244
6.10. CONSTRUCCIÓN DE LOS COMPONENTES Y PROCEDIMIENTOS DE MIGRACIÓN Y CARGA INICIAL DE DATOS 245
6.11. APROBACIÓN DEL SISTEMA 245

7. IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA DE INFORMACIÓN (IAS) 249

7.1. INTRODUCCIÓN 249
7.2. LA IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA EN MÉTRICA VERSIÓN 3 249
7.3. INSTALADOR DEL SISTEMA SELDON 250
7.3.1. Archivo de Configuración 251
7.3.2. Generación del Instalador 252
7.3.3. Ejecución del Instalador 252
7.4. TAREAS DE IMPLANTACIÓN Y ACEPTACIÓN 252

8. CONCLUSIONES FINALES Y FUTURAS LÍNEAS DE TRABAJO 257

8.1. CONCLUSIONES FINALES SOBRE LA METODOLOGÍA 257
8.2. CONCLUSIONES FINALES SOBRE LA TECNOLOGÍA Y HERRAMIENTAS 258
8.2.1. Rational Rose [Rational, 2002] 258
8.2.2. Eclipse [Eclipse, 2005] 259
8.2.3. Rational ClearCase [Rational ClearCase, 2004] 259
8.2.4. IzPack [IzPack, 2005] 259
8.3. CONCLUSIONES FINALES SOBRE EL TRABAJO DE TESIS DE MAGÍSTER 260
8.4. FUTURAS LÍNEAS DE TRABAJO 260

9. BIBLIOGRAFÍA 265

9.1. REFERENCIAS BIBLIOGRAFICAS 265
9.2. ABREVIATURAS 268

I. APÉNDICE A: PLAN DE GESTIÓN DEL PROYECTO 273

I.1. LA GESTIÓN DE PROYECTOS EN MÉTRICA VERSIÓN 3 273
I.2. INTRODUCCIÓN 274
I.3. DESCRIPCIÓN DEL PROYECTO 274
I.3.a. Entregables 274
I.3.b. Evolución del Plan 274
I.3.c. Documentos Referenciados 274
I.4. ORGANIZACIÓN DEL PROYECTO 275

I.4.a. Modelo de Proceso ... *275*
I.4.b. Modelo de Estimación ... *275*
I.4.c. Estructura Organizacional ... *275*
I.4.d. Interfaces de Proyecto ... *275*
I.4.e. Responsabilidades de Proyecto ... *275*
I.5. PROCESO DE ADMINISTRACIÓN ... 276
I.5.a. Objetivos y Prioridades de la Administración ... *276*
I.5.b. Asunciones, Dependencias y Restricciones ... *276*
I.5.c. Manejo de Riesgos ... *276*
I.5.d. Plan de Prevención de Riesgos ... *276*
I.5.e. Plan de Personal ... *276*
I.5.f. Plan de Capacitación ... *277*
I.6. PROCESO TÉCNICOS ... 277
I.6.a. Métodos, Herramientas y Técnicas ... *277*
I.6.b. Funciones de Soporte del Proyecto ... *277*
I.7. ARTEFACTOS, CALENDARIO Y PRESUPUESTO ... 277
I.7.a. Artefactos ... *277*
I.7.b. Recursos Requeridos ... *277*
I.7.c. Calendario ... *278*
I.7.d. Presupuesto ... *278*

II. APÉNDICE B: ESTIMACIONES Y CALENDARIO ... 281

II.1. ESTIMACIONES ... 281
II.2. CALENDARIO ... 287

III. APÉNDICE C: PLAN DE GESTIÓN DE LA CONFIGURACIÓN ... 297

III.1. LA GESTIÓN DE LA CONFIGURACIÓN EN MÉTRICA VERSIÓN 3 ... 297
III.2. INTRODUCCIÓN ... 298
III.2.a. Propósito ... *298*
III.2.b. Alcance ... *299*
III.2.c. Definición de alto nivel del proceso de GCS ... *299*
III.3. ESPECIFICACIONES DE GESTIÓN ... 299
III.3.a. Organización ... *299*
III.3.b. Responsabilidades ... *300*
III.3.c. Implementación del plan de Gestión de Configuración ... *300*
III.3.d. Políticas, Directivas y Procedimientos Aplicables ... *303*
III.4. ACTIVIDADES DE GESTIÓN DE CONFIGURACIÓN ... 303
III.4.a. Identificación de la Configuración ... *303*
III.4.b. Control de la Configuración ... *307*
III.4.c. Contabilidad del estado de la configuración ... *308*
III.4.d. Auditoría de la Configuración ... *308*
III.5. CONTROL DE SUMINISTRADORES ... 308
III.6. RECOGIDA Y RETENCIÓN DE REGISTROS ... 308

IV. APÉNDICE D: PLAN DE ASEGURAMIENTO DE LA CALIDAD ... 311

IV.1. EL ASEGURAMIENTO DE LA CALIDAD EN MÉTRICA VERSIÓN 3 ... 311
IV.2. INTRODUCCIÓN ... 312
IV.2.a. Objetivos de Calidad ... *312*
IV.2.b. Métricas de Proyecto ... *313*
IV.2.c. Auditorías ... *315*

V. APÉNDICE E: PLAN DE RIESGOS ... 319

V.1. LA SEGURIDAD EN MÉTRICA VERSIÓN 3 ... 319
V.2. INTRODUCCIÓN ... 320
V.2.a. Propósito y Alcance ... *320*
V.3. ROLES Y RESPONSABILIDADES ... 320
V.4. DOCUMENTACIÓN DE RIESGOS ... 321
V.5. ACTIVIDADES DE GESTIÓN DE RIESGOS ... 321
V.6. HERRAMIENTAS DE GESTIÓN DE RIESGOS ... 322
V.7. CRITERIO DE CIERRE ... 322

V.8. Calendario para las Actividades de Gestión de Riesgos 322

VI. APÉNDICE F: TAXONOMÍA DE RIESGOS AJUSTADA 325

VI.1. Tipos de Proyectos 325
VI.2. Taxonomía de Riesgos Ajustada 326

VII. APÉNDICE G: MATRICES DE TRAZABILIDAD 359

VII.1. Trazabilidad Requerimientos-Casos de Uso 359
VII.2. Trazabilidad Casos de Uso-Requerimientos 360
VII.3. Trazabilidad Casos de Uso-Interfaces 360
VII.4. Trazabilidad Interfaces-Casos de Uso 361
VII.5. Trazabilidad Casos de Uso-Diagramas de Robustez 361
VII.6. Trazabilidad Diagramas de Robustez-Casos de Uso 362
VII.7. Trazabilidad Diagramas de Robustez-Clases 362
VII.8. Trazabilidad Clases-Diagramas de Robustez 363
VII.9. Trazabilidad Requerimientos-Casos de Prueba 364
VII.10. Trazabilidad Casos de Prueba-Requerimientos 365

VIII. APÉNDICE H: DOCUMENTACIÓN DE CLASES 369

VIII.1. Paquetes Principales 369
VIII.2. Paquete controller 370
VIII.3. Paquete model 384
VIII.4. Paquete mvc 407
VIII.5. Paquete persistence 425
VIII.6. Paquete view 428

IX. APÉNDICE I: MÉTRICAS FINALES DE DESARROLLO 457

IX.1. Resultados de la Ejecución de las Pruebas 457
IX.1.a. Primer Ciclo 458
IX.1.b. Segundo Ciclo 459
IX.2. Esfuerzo, Calendario, Líneas de Código y Documentación 460

Índice de Figuras

FIGURA 1 - MODELO DE GESTIÓN DE RIESGOS 10
FIGURA 2 - PA DE GESTIÓN DE RIESGOS EN CMMI 12
FIGURA 3 - PREOCUPACIONES, RIESGOS Y PROBLEMAS 13
FIGURA 4 - DESCRIPCIÓN DE RIESGOS 13
FIGURA 5 - LOS RIESGOS Y LAS FASES DE DESARROLLO 21
FIGURA 6 - PERFIL DE LOS RIESGOS EN PROYECTOS DE SOFTWARE 25
FIGURA 7 - LA INCERTIDUMBRE DESCRIPTIVA 28
FIGURA 8 - ORGANIZACIÓN DE LA TAXONOMÍA DE RIESGOS 33
FIGURA 9 - EL PROCESO DE TBQ 35
FIGURA 10 - LA GESTIÓN DE RIESGOS Y CMMI 38
FIGURA 11 - TAREAS DEL EVS 49
FIGURA 12 - PANTALLA PRINCIPAL DE GESTIÓN DE RIESGOS EN ARM 57
FIGURA 13 - PANTALLA PRINCIPAL DE TRIMS 58
FIGURA 14 - CARGA DE RIESGO EN TRIMS 59
FIGURA 15 - REPORTE EN TRIMS 59
FIGURA 16 - CATEGORÍAS DE RIESGOS EN RISKTRACK 60
FIGURA 17 - ANÁLISIS DE RIESGOS EN RISKTRACK 61
FIGURA 18 - PANTALLA PRINCIPAL DE WELCOMRISK 62
FIGURA 19 - RESUMEN DE RIESGOS 62
FIGURA 20 - REPORTE EN WELCOMRISK 63
FIGURA 21 - ACTIVIDADES DE LA TAREA DE IDENTIFICACIÓN DE RIESGOS 65
FIGURA 22 - ARQUITECTURA DE ALTO NIVEL 68
FIGURA 23 - TAREAS DEL ASI 75
FIGURA 24 - ESTRUCTURA DE MIRT 78
FIGURA 25 - PROCESO DE GENERACIÓN DE TAXONOMÍA DE RIESGOS DEL MAGÍSTER 81
FIGURA 26 - REQUERIMIENTOS FUNCIONALES 85
FIGURA 27 - REQUERIMIENTOS NO FUNCIONALES 89
FIGURA 28 - DIAGRAMA DE CASOS DE USO 90
FIGURA 29 - DIAGRAMA DE CLASES 99
FIGURA 30 - MENÚS DE OPCIONES 111
FIGURA 31 - INICIANDO NUEVA SESIÓN DE IDENTIFICACIÓN - INT-001 112
FIGURA 32 - IDENTIFICANDO EL PROYECTO - INT-002 113
FIGURA 33 - CARACTERIZANDO EL PROYECTO - INT-003 114
FIGURA 34 - CARACTERIZANDO POSIBLES RIESGOS - INT-004 115
FIGURA 35 - SOLICITANDO GENERACIÓN DE REPORTE DE RIESGOS - INT-005 116
FIGURA 36 - REVISANDO Y SOLICITANDO GUARDADO DEL REPORTE (1) - INT-006 117
FIGURA 37 - REVISANDO Y SOLICITANDO GUARDADO DEL REPORTE (2) - INT-007 117
FIGURA 38 - SOLICITANDO GUARDADO DEL REPORTE DE RIESGOS - INT-008 118
FIGURA 39 - GUARDANDO REPORTE DE RIESGOS - INT-009 119
FIGURA 40 - SOLICITANDO APERTURA DE REPORTE DE RIESGOS - INT-010 120
FIGURA 41 - ABRIENDO REPORTE DE RIESGOS - INT-011 121
FIGURA 42 - SOLICITANDO IMPRESIÓN DE REPORTE DE RIESGOS - INT-012 122
FIGURA 43 - IMPRIMIENDO REPORTE DE RIESGOS - INT-013 123
FIGURA 44 - SOLICITANDO GENERACIÓN DE REPORTE DE TAXONOMÍA BASE - INT-014 124
FIGURA 45 - CONSULTANDO REPORTE DE TAXONOMÍA BASE Y RIESGOS - INT-015 125
FIGURA 46 - FORMATO REPORTE DE RESULTADOS DE SESIÓN DE IDENTIFICACIÓN - INT-016 126
FIGURA 47 - PROCEDIMIENTO DE PRUEBAS 132
FIGURA 48 - TAREAS DEL DSI 138
FIGURA 49 - NIVELES DE LA ARQUITECTURA 139
FIGURA 50 - SUBSISTEMAS DE DISEÑO 144
FIGURA 51 - INSTANCIACIÓN DE LA ESTRUCTURA DE DATOS PARA LA TAXONOMÍA DE RIESGOS 150
FIGURA 52 - MODELO ESTÁTICO DE MVC 151
FIGURA 53 - MODELO DINÁMICO DE MVC – INICIALIZACIÓN 154
FIGURA 54 - MODELO DINÁMICO DE MVC – UTILIZACIÓN 155
FIGURA 55 - MODELO ESTÁTICO DE JASPERREPORTS 155

FIGURA 56 - MODELO DINÁMICO DE JASPERREPORTS – GENERACIÓN DE REPORTE 157
FIGURA 57 - MODELO ESTÁTICO DE PERSISTENCE 157
FIGURA 58 - MODELO DINÁMICO DE MVC – ESCRITURA 160
FIGURA 59 - MODELO DINÁMICO DE MVC – LECTURA 160
FIGURA 60 - MODELO ESTÁTICO DE I18N 161
FIGURA 61 - MODELO DINÁMICO DE I18N – UTILIZACIÓN 164
FIGURA 62 - MODELO ESTÁTICO DE JEOPS (PAQUETE JEOPS) 169
FIGURA 63 - MODELO ESTÁTICO DE JEOPS (PAQUETE COMPILER) 170
FIGURA 64 - MODELO ESTÁTICO DE JEOPS (PAQUETE CONFLICT) 170
FIGURA 65 - MODELO ESTÁTICO DE JEOPS (PAQUETE RETE) 171
FIGURA 66 - IDENTIFICANDO RIESGOS - DR-001 174
FIGURA 67 - GENERANDO REPORTE DE RESULTADOS - DR-002 175
FIGURA 68 - IMPRIMIENDO REPORTE DE RESULTADOS - DR-003 176
FIGURA 69 - ALMACENANDO RESULTADOS - DR-004 177
FIGURA 70 - RECUPERANDO RESULTADOS - DR-005 178
FIGURA 71 - GENERANDO REPORTE DE TAXONOMÍA - DR-006 179
FIGURA 72 - GENERANDO REPORTE DE RIESGOS - DR-007 180
FIGURA 73 - ACTIVIDADES DE IDENTIFICACIÓN DE RIESGOS EN SELDON 181
FIGURA 74 - DIAGRAMA DETALLADO DE CLASES 201
FIGURA 75 - MODELO DE COMPONENTES DE SELDON 204
FIGURA 76 - TAREAS DE LA CSI 226
FIGURA 77 - SELECCIÓN DEL LENGUAJE DEL INSTALADOR 230
FIGURA 78 - BIENVENIDA AL PROCESO DE INSTALACIÓN 231
FIGURA 79 - SELECCIÓN DEL DIRECTORIO DE INSTALACIÓN 231
FIGURA 80 - INFORMACIÓN DE CREACIÓN DEL DIRECTORIO DE INSTALACIÓN 232
FIGURA 81 - PROGRESO DE LA INSTALACIÓN 232
FIGURA 82 - FINALIZACIÓN DEL PROCESO DE INSTALACIÓN 233
FIGURA 83 - VENTANA PRINCIPAL DE SELDON 235
FIGURA 84 - INICIO DE SESIÓN DE IDENTIFICACIÓN DE RIESGOS 236
FIGURA 85 - ALTA DE DATOS DE LA SESIÓN DE IDENTIFICACIÓN 236
FIGURA 86 - ALTA DE CARACTERÍSTICAS DE LA SESIÓN DE IDENTIFICACIÓN 237
FIGURA 87 - PARÁMETROS DEL PROYECTO 237
FIGURA 88 - GENERACIÓN DEL REPORTE DE IDENTIFICACIÓN DE RIESGOS 238
FIGURA 89 - REPORTE DE IDENTIFICACIÓN DE RIESGOS 238
FIGURA 90 - ALMACENAMIENTO DE UNA SESIÓN - MENÚ PRINCIPAL 239
FIGURA 91 - ALMACENAMIENTO DE UNA SESIÓN - REPORTE DE IDENTIFICACIÓN 239
FIGURA 92 - ALMACENAMIENTO DE UNA SESIÓN - IDENTIFICACIÓN DE RIESGOS 240
FIGURA 93 - RECUPERACIÓN DE UNA SESIÓN DE IDENTIFICACIÓN DE RIESGOS 240
FIGURA 94 - IMPRESIÓN DE UNA SESIÓN DE IDENTIFICACIÓN DE RIESGOS 241
FIGURA 95 - CONSULTA DEL REPORTE DE TAXONOMÍA BASE DE RIESGOS 242
FIGURA 96 - REPORTE DE TAXONOMÍA BASE DE RIESGOS 242
FIGURA 97 - CONSULTA DE LA INFORMACIÓN DEL SISTEMA 243
FIGURA 98 - INFORMACIÓN DEL SISTEMA 243
FIGURA 99 - SALIDA DEL SISTEMA 244
FIGURA 100 - TAREAS DE LA IAS 250
FIGURA 101 - CALENDARIO DEL PROYECTO 287
FIGURA 102 - CALENDARIO EVS 288
FIGURA 103 - CALENDARIO ASI 289
FIGURA 104 - CALENDARIO DSI 290
FIGURA 105 - CALENDARIO CSI 291
FIGURA 106 - CALENDARIO IAS 292
FIGURA 107 - ESTRUCTURA DEL REPOSITORIO 304
FIGURA 108 - REPORTE DEL PRIMER CICLO DE PRUEBAS 458
FIGURA 109 - REPORTE DEL SEGUNDO CICLO DE PRUEBAS 459

Índice de Tablas

TABLA 1 - EJEMPLO DE ELEMENTOS DE UN RIESGO 20
TABLA 2 - HERRAMIENTAS PARA LA IDENTIFICACIÓN DE RIESGOS 52
TABLA 3 - COMPARACIÓN DE HERRAMIENTAS DE SOPORTE DE IDENTIFICACIÓN DE RIESGOS 67
TABLA 4 - CATÁLOGO DE INVOLUCRADOS EN EL DESARROLLO 82
TABLA 5 - OPERACIONES DE LA CLASE CATEGORIA 100
TABLA 6 - ATRIBUTOS DE LA CLASE CATEGORIA 100
TABLA 7 - ATRIBUTOS DE LA INTERFAZ DEFINICIONESCOMUNES 101
TABLA 8 - OPERACIONES DE LA CLASE IDENTIFICACION 101
TABLA 9 - ATRIBUTOS DE LA CLASE IDENTIFICACION 102
TABLA 10 - OPERACIONES DE LA CLASE INDICADOR 102
TABLA 11 - ATRIBUTOS DE LA CLASE INDICADOR 102
TABLA 12 - OPERACIONES DE LA CLASE PROYECTO 106
TABLA 13 - ATRIBUTOS DE LA CLASE PROYECTO 106
TABLA 14 - OPERACIONES DE LA CLASE RIESGO 107
TABLA 15 - ATRIBUTOS DE LA CLASE RIESGO 108
TABLA 16 - OPERACIONES DE LA CLASE SESION 108
TABLA 17 - ATRIBUTOS DE LA CLASE SESION 109
TABLA 18 - ASOCIACIÓN MENÚ-CASOS DE USO 111
TABLA 19 - SEVERIDAD DE DEFECTOS 131
TABLA 20 - REQUISITOS DE DISEÑO Y CONSTRUCCIÓN 141
TABLA 21 - EXC-001 141
TABLA 22 - EXC-002 141
TABLA 23 - EXC-003 142
TABLA 24 - EXC-004 142
TABLA 25 - EXC-005 142
TABLA 26 - CLASES E INTERFACES DE JEOPS 172
TABLA 27 - PLAN DE INTEGRACIÓN DE SELDON 205
TABLA 28 - ESTIMACIÓN DE ESFUERZO DE CONSTRUCCIÓN DEL SISTEMA 286
TABLA 29 - FORMACIÓN DEL CCC 301
TABLA 30 - INFORMACIÓN DE CC 302
TABLA 31 - PERMISOS EN EL VOB 303
TABLA 32 - ESTRUCTURA DE GRUPOS EN EL VOB 303
TABLA 33 - LÍNEAS BASE Y SUS ETIQUETAS 304
TABLA 34 - IC DEL PROYECTO 305
TABLA 35 - CONVENCIÓN PARA ETIQUETAS 306
TABLA 36 - TRIGGERS DEL VOB 307
TABLA 37 - ACTIVIDADES DE REPORTE 308
TABLA 38 - MÉTRICAS DE PROYECTO 314
TABLA 39 - ROLES Y RESPONSABILIDADES EN LA GESTIÓN DE RIESGOS 321
TABLA 40 - DOCUMENTACIÓN DE RIESGOS 321
TABLA 41 - ACTIVIDADES DE GESTIÓN DE RIESGOS 322
TABLA 42 - ACTIVIDADES DE GESTIÓN DE RIESGOS 322
TABLA 43 - TIPOS DE PROYECTO 326
TABLA 44 - TAXONOMÍA DE RIESGOS AJUSTADA 356
TABLA 45 - TRAZABILIDAD REQUERIMIENTOS-CASOS DE USO 359
TABLA 46 - TRAZABILIDAD CASOS DE USO-REQUERIMIENTOS 360
TABLA 47 - TRAZABILIDAD CASOS DE USO-INTERFACES 361
TABLA 48 - TRAZABILIDAD INTERFACES-CASOS DE USO 361
TABLA 49 - TRAZABILIDAD CASOS DE USO-DIAGRAMAS DE ROBUSTEZ 362
TABLA 50 - TRAZABILIDAD DIAGRAMAS DE ROBUSTEZ-CASOS DE USO 362
TABLA 51 - TRAZABILIDAD DIAGRAMAS DE ROBUSTEZ-CLASES 363
TABLA 52 - TRAZABILIDAD CLASES-DIAGRAMAS DE ROBUSTEZ 364
TABLA 53 - TRAZABILIDAD REQUERIMIENTOS-CASOS DE PRUEBA 365
TABLA 54 - TRAZABILIDAD CASOS DE PRUEBA-REQUERIMIENTOS 366
TABLA 55 - RESÚMEN DEL PRIMER CICLO DE PRUEBAS 458

TABLA 56 - RESÚMEN DEL SEGUNDO CICLO DE PRUEBAS 459
TABLA 57 - COMPARACIÓN DE ESTIMADOS Y RELAES 460

Capítulo

1

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Introducción

1. Introducción

En este capítulo se presenta brevemente el trabajo a desarrollar y la estructura de organización del mismo.

1.1. Sobre la Tesis de Magíster

Esta Tesis de Magíster trata sobre el desarrollo de una herramienta de software del tipo asistente, que facilita la identificación de riesgos en proyectos de desarrollo de software en organizaciones de cualquier tipo. El desarrollo de la misma se basa tecnologías Java y se apoya en la metodología Métrica V3 [Métrica V3, 2000].

El trabajo persigue como objetivo fundamental el de brindar una herramienta que permita facilitar las tareas de identificación de riesgos a las organizaciones de software, y como objetivo secundario el de utilizar la metodología Métrica V3 en un proyecto completo de desarrollo.

1.2. Audiencia

El presente trabajo de Tesis de Magíster se encuentra dirigido principalmente a ingenieros de software, profesionales de la industria del software y cátedras universitarias vinculadas al software y enfocados a la administración de proyectos de desarrollo.

La documentación contenida en este trabajo puede tomarse como referencia para la adopción de prácticas básicas y avanzadas de gestión de riesgos en un

entorno de procesos o de la metodología Métrica V3, y como punto de partida para futuras ampliaciones de la herramienta, o eventualmente para el desarrollo de herramientas similares.

1.3. Organización de la Tesis de Magíster

El documento se divide en nueve capítulos y en ocho apéndices que abarcan la totalidad del trabajo de Tesis de Magíster.

- El Capítulo 1.Introducción (el presente) sintetiza los objetivos del trabajo, la audiencia del mismo y su organización.
- El Capítulo 2.Dominio de la Aplicación presenta un panorama de la situación actual, mostrando el contexto que da origen al trabajo de Tesis de Magíster.
- El Capítulo 3.Estudio de Viabilidad del Sistema (EVS) contiene la documentación resultante del proceso "EVS: Estudio de la Viabilidad del Sistema" de la metodología Métrica V3. La documentación cubre la el análisis de la situación actual, la descripción general del sistema, el estudio de las diferentes alternativas de solución, y la selección de la alternativa más adecuada.
- El capítulo 4.Análisis del Sistema de Información (ASI) contiene la documentación resultante del proceso "ASI: Análisis del Sistema de Información" de la metodología Métrica V3. La documentación cubre el modelado en UML (en inglés, *Unified Modelling Language*) [Booch, et al., 2000] del los casos de uso, los prototipos de interfaces de usuario, y la estructura y comportamiento del sistema en términos de clases de análisis.
- El capítulo 5.Diseño del Sistema de Información (DSI) contiene la documentación resultante del proceso "DSI: Diseño del Sistema de Información" de la metodología Métrica V3. La documentación cubre el modelado en UML (en inglés, *Unified Modelling Language*) [Booch, et al., 2000] del diseño arquitectónico del sistema y los subsistemas que lo conforman, y la estructura y comportamiento del mismo en términos de clases de diseño.
- El capítulo 6.Construcción del Sistema de Información (CSI) contiene la documentación resultante del proceso "CSI: Construcción del Sistema de Información" de la metodología Métrica V3. La documentación cubre la descripción del entorno de construcción y los resultados de las pruebas unitarias, de integración y del sistema.
- El capítulo 7.Implantación y Aceptación del Sistema de Información (IAS) contiene la documentación resultante del proceso "IAS: Implantación y Aceptación del Sistema de Información" de la metodología Métrica V3. La documentación cubre la incorporación del sistema producido al entorno operativo y los resultados de las pruebas de implantación y aceptación.
- El capítulo 8.Conclusiones Finales y Futuras Líneas de Trabajo contiene un resumen de las conclusiones obtenidas luego de finalizado el trabajo final de Tesis de Magíster, y las propuestas futuras líneas de trabajo a

seguir por parte de aquellos interesados en el tema presentado en el presente documento.

- El capítulo 9.Bibliografía contiene las referencias bibliográficas utilizadas para el trabajo final de Tesis de Magíster y, además, el listado de acrónimos empleados en el mismo.
- El resto del documento contiene documentación anexa al contenido de la Tesis de Magíster entre las que figuran los planes de gestión de proyecto, de gestión de riesgos, de aseguramiento de la calidad y de configuración del sistema, además de la taxonomía de riesgos empleada como base de la herramienta desarrollada, las matrices de trazabilidad del sistema y las métricas de desarrollo.

Capítulo

2

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Dominio de la Aplicación

2. Dominio de la Aplicación

En este capítulo se presenta brevemente el dominio de aplicación del sistema cuyo objetivo es brindar información sobre la cuál se sentarán las bases para llevar a cabo la identificación de riesgos de proyectos de software en base a taxonomías.

2.1. Introducción

El SEI (*Software Engineering Institute*) define al riesgo como "*la posibilidad de sufrir una pérdida*" [SEI, 2004] y a la **Gestión de Riesgos** como "*la práctica compuesta de procesos, métodos y herramientas que posibilita la gestión de los riesgos en un proyecto y que provee de un entorno disciplinado para la toma de decisiones preactiva en base a determinar constantemente que puede ir mal (riesgos), identificar cuales son los riesgos mas importantes en los cuales enfocarse e implementar estrategias para gestionarlos*" [SEI, 2004]; esta actividad se inicia en la primera etapa de un proyecto de software (durante la exploración de conceptos) y se desarrolla a lo largo de todo su ciclo de vida (hasta la aceptación del producto del proyecto).

Según [Rosenberg, *et al.*, 1999], *la **Gestión de Riesgos** es importante debido a que ayuda a evitar desastres, re-trabajo y sobre-trabajo, pero aún mas importante, porque estimula la generación de situaciones del tipo ganar-ganar*. Una correcta **Gestión de Riesgos** posibilita, por tanto, el aprovechamiento óptimo de recursos y provoca, como consecuencia, el aumento de ganancias y la disminución de pérdidas. La ausencia de una apropiada **Gestión de Riesgos** conlleva a la imposibilidad de lograr el control efectivo de un proyecto derivando esto en la imposibilidad de realizar una correcta administración del

mismo. En base a las consideraciones antes expuestas, la **Gestión de Riesgos** debe ser enfatizada y considerada como una actividad clave en todo tipo de proyectos y, particularmente, en proyectos de desarrollo de software.

Al realizar **Gestión de Riesgos**, es fundamental lograr una clara descripción del riesgo de forma tal de que el mismo pueda ser comprendido y manejado adecuadamente; cuando se lo enuncia, no solo debe considerarse el síntoma sino también sus consecuencias.

Existen varios modelos de **Gestión de Riesgos** pero el más aceptado consta de cinco pasos (Identificación, Análisis, Planificación, Seguimiento y Control) secuenciales e interactivos, en forma paralela a estos existen dos actividades comunes a ellos: las de documentación y comunicación (véase **Figura 1 - Modelo de Gestión de Riesgos**).

El siguiente gráfico representa el modelo antes descrito:

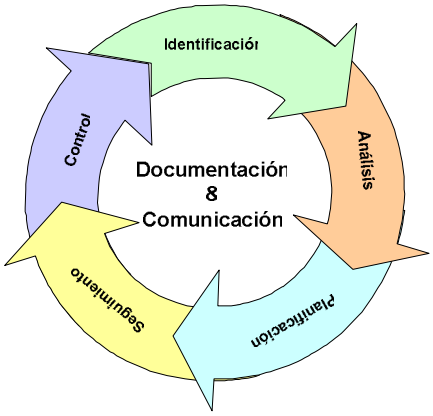


Figura 1 - Modelo de Gestión de Riesgos

La **Identificación de Riesgos** en proyectos de software consiste en la determinación de elementos de riesgos potenciales mediante la utilización de algún método consistente y estructurado; este es, probablemente, el paso más importante entre todos aquellos que componen las actividades de **Gestión de Riesgos**, ya que sin la correcta determinación de los mismos, no es posible desarrollar e implementar anticipadamente respuestas apropiadas a los problemas que puedan surgir en el proyecto [Futrell, *et al.*, 2002]. El resultado de la identificación de riesgos es una lista conteniendo los riesgos que se han identificados y su categoría correspondiente.

El diccionario Webster [Webster, 2004] define a las **Taxonomías** como *las clasificaciones ordenadas de plantas y animales de acuerdo a sus relaciones naturales presumidas*, esta definición suele ser ampliada especificando a las **Taxonomías** como *las clasificaciones ordenadas de elementos de acuerdo a sus relaciones presumidas*; así consideradas, puede emplearse a las taxonomías como herramientas de suma utilidad en diferentes ramas de la ciencia y la industria donde se pretende organizar y facilitar el acceso a un

número importante de elementos que se encuentran mutuamente relacionados de alguna manera relevante.

La **Gestión de Riesgos** en base a **Taxonomías** implica el utilizar una estructura agrupadora de los mismos de acuerdo a su diferentes clases como una lista de consulta durante la actividad de **Identificación de los Riesgos** (esta lista estructurada puede obtenerse tanto de la misma organización como de cualquier otra fuente disponible, tales como: [Marvin J. Carr *et al.*, 1993] o [Microsoft, 2002]).

Adicionalmente, CMMI (*Capability Maturity Model Integration*) [CMMI, 2002] que es un modelo para la determinación de la madurez de la capacidad de un proceso de software y se ha convertido en el nuevo estándar a nivel mundial para la medición de la calidad de los procesos de desarrollo de software, presenta como una de sus PA (*Process Area – Area de procesos*) fundamentales de Nivel 3 la **Gestión de Riesgos**.

Dentro del antes mencionado contexto las **Taxonomías** de fuentes de riesgos y la **Identificación de Riesgos** juegan un papel fundamental entre los objetivos planteados para el área de proceso asociada al manejo de riegos debido a que las tareas antes indicadas son consideradas como *Actividades* en el referido PA (véase **Figura 2 - PA de Gestión de Riesgos en CMMI**).

De lo antes expuesto, puede derivarse la enorme importancia que los aspectos mencionados tienen en el marco de las actividades de **Administración de Proyectos** (y por tanto en el área de **Ingeniería de Software**) considerando dentro de este contexto el continuo esfuerzo realizado y la permanente y creciente necesidad presentada por las compañías de software con relación a *herramientas* que permitan automatizar y estandarizar sus procesos de gestión en busca de una mayor madurez organizacional.

El siguiente gráfico [Williams Ray, 2003] resume el PA de **Gestión de Riesgos** y destaca la importancia de los componentes estudiados:

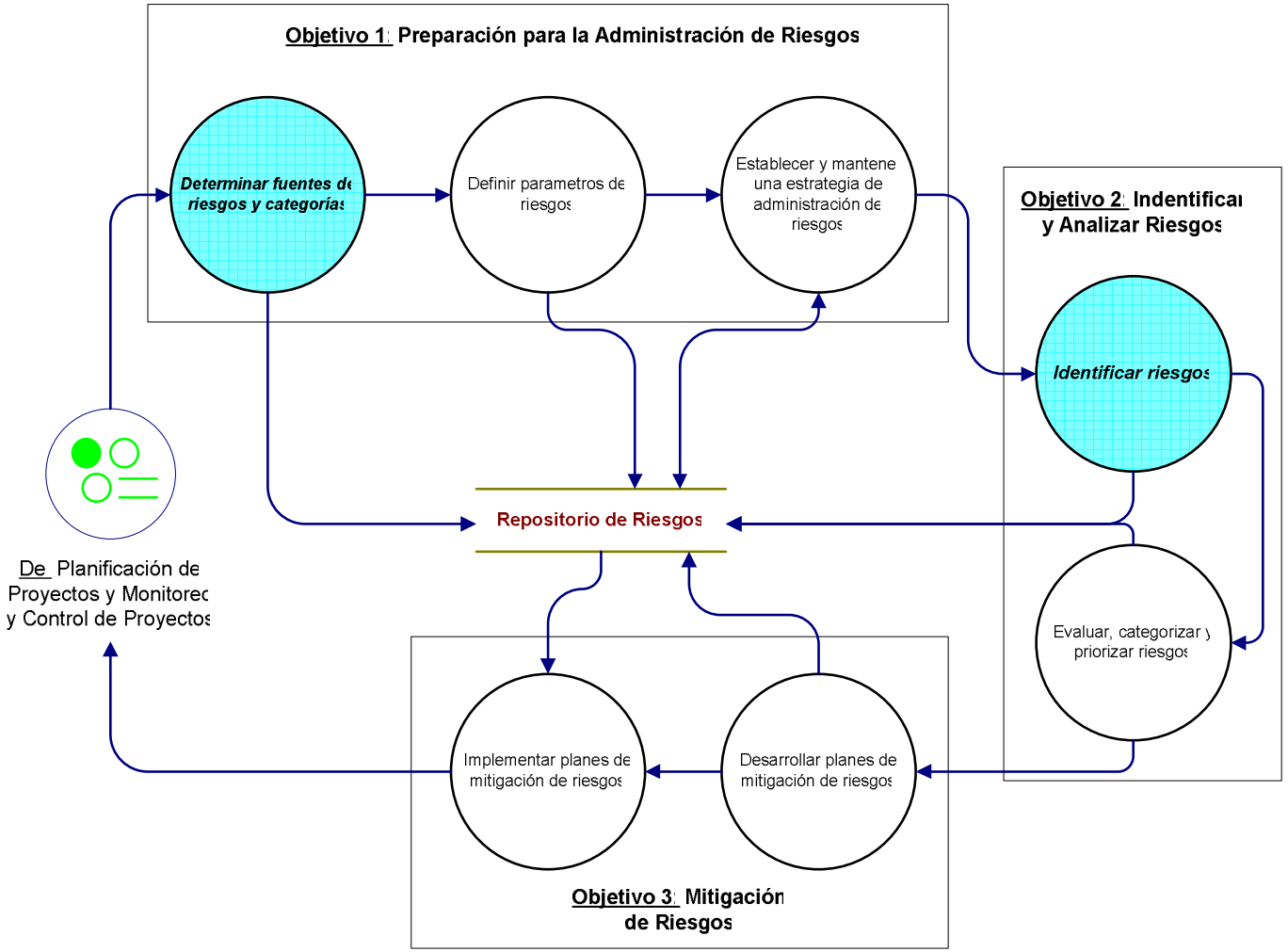


Figura 2 - PA de Gestión de Riesgos en CMMI

2.2. Los Riesgos

"En primer lugar, el riesgo concierne a lo que ocurra en el futuro. El hoy y el ayer no nos conciernen realmente, porque ahora ya estamos recogiendo los frutos de lo que sembramos en el pasado. La cuestión es si podemos, entonces, modificando nuestras acciones en este momento, crear una oportunidad para una situación diferente y más esperanzadora de nuestro mañana. Esto significa, en segundo lugar, que el riesgo implica un cambio, que puede venir dado por cambio de opiniones, acciones o lugares. En tercer lugar, el riesgo implica una elección, y la falta de certeza de que la elección sea correcta. Así, paradójicamente, el riesgo como la muerte o los impuestos, es una de las pocas cosas inevitables de la vida." [Charette, 1989]

Un riesgo es "*la posibilidad de sufrir una pérdida*" [SEI, 2004], sin embargo, en el entorno de la Ingeniería de Software, se prefiere hablar de la "*exposición a una pérdida*" debido a que se pretende considerar no solo la posibilidad de ocurrencia de un riesgo sino también su impacto asociado.

Suele ser común el confundir preocupaciones, riesgos y problemas: mientras que una **preocupación** es cualquier situación sobre la cual existen dudas al respecto en algún determinado contexto y que, por lo tanto, será evaluada como un posible riesgo, un **problema** es un riesgo que, efectivamente, se ha producido (véase **Figura 3 - Preocupaciones, Riesgos y Problemas**).

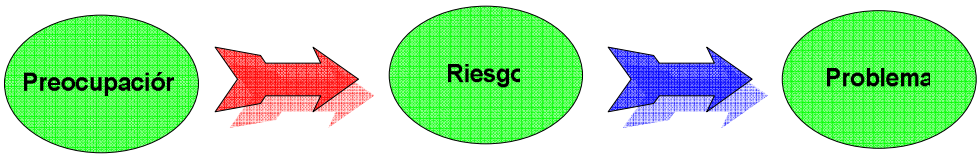


Figura 3 - Preocupaciones, Riesgos y Problemas

Un riesgo se describe mediante una expresión en lenguaje natural que expresa la relación entre una situación real de proyecto y otra situación no realizada. La primera parte de la definición se denomina *condición* y representa una situación existente en el proyecto que el equipo prevé que puede resultar en una pérdida o en una reducción de beneficios; mientras que la segunda parte de la declaración se denomina *consecuencia* y describe la situación no deseable para el proyecto que puede derivarse de la ocurrencia de la condición. Las dos oraciones deben unirse a través de algún indicador de relación que de idea de incertidumbre y que a su vez implique una relación de tipo causal, tales como "por lo tanto" o "como consecuencia" (véase **Figura 4 - Descripción de Riesgos**).

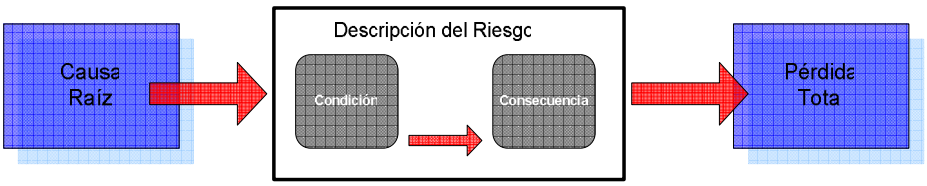


Figura 4 - Descripción de Riesgos

Es importante resaltar que la descripción de un riesgo no es una declaración del tipo “si-entonces” debido a que la primera intenta expresar sólo una posibilidad y no una consecuencia concreta y segura como lo hacen las antes mencionadas declaraciones.

Una frase para describir un riesgo debe ser lo suficientemente completa y directa como para posibilitar el análisis de su causa raíz, discutir su impacto y desarrollar un conjunto de respuestas o acciones para prevenirlo o reducir sus consecuencias. A su vez, en las frases empleadas se debe evitar tanto el uso de abreviaciones o acrónimos que resulten difíciles de comprender como de generalizaciones y detalles irrelevantes. Finalmente, en algunos casos suele ser de utilidad agregar información de contexto a un riesgo para facilitar su entendimiento a personas que puedan no conocer todos los detalles del proyecto (ya sea en la actualidad como en el futuro).

A pesar de poseer una gran importancia, la descripción de un riesgo no es la única información que debe generarse y verificarse de los riesgos, mas aún, es necesario considerar una gran cantidad de variables componentes con el propósito de llevar a cabo una administración eficiente. Comúnmente, y por cuestiones de facilidad operativa, el documento de descripción de riesgos o Base de Datos de Riesgos se elabora separadamente del Plan de Riesgos debido al dinamismo que debería caracterizar al primero de ellos. Adicionalmente, se han presentado múltiples propuestas respecto al formato que la Base de Datos de Riesgos debería respetar, una posibilidad se describe a modo de ejemplo a continuación (véase **Tabla 1 - Ejemplo de Elementos de un Riesgo**):

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES
Identificador del Riesgo	Identifica unívocamente a cada riesgo.	Formato libre.
Descripción Corta del Riesgo	Breve descripción del riesgo.	Texto en lenguaje natural de forma libre.
Descripción Larga del Riesgo	Completa descripción del riesgo.	Texto en lenguaje natural de forma libre.
Originador	Nombre de la persona que identificó el riesgo.	Nombre del recurso.
Dueño	Nombre de la persona a quién se asignó el riesgo.	Nombre del recurso.

<table>
<tr><th>NOMBRE DEL CAMPO</th><th>DEFINICIÓN</th><th colspan="3">POSIBLE LISTA DE VALORES</th></tr>
<tr><td>Ultima Actualización</td><td>Fecha y hora de última actualización de la información del riesgo.</td><td colspan="3">Fecha y hora.</td></tr>
<tr><td>Comentarios</td><td>Comentarios generales respecto al riesgo.</td><td colspan="3">Texto en lenguaje natural de forma libre.</td></tr>
<tr><td>Estrategia</td><td>Estrategia de respuesta elegida para el riesgo.</td><td colspan="3">Nombre de la estrategia.</td></tr>
<tr><td rowspan="6">Áreas de Riesgo</td><td rowspan="6">Áreas principales del proyecto a las cuales se asocia el riesgo.</td><td><u>Área</u></td><td colspan="2"><u>Definición</u></td></tr>
<tr><td>Negocios</td><td colspan="2">El riesgo representa un problema para los propósitos relacionados con los negocios.</td></tr>
<tr><td>Administración de Cambios</td><td colspan="2">El riesgo representa una medida de la resistencia a nivel de usuario o participante del proyecto.</td></tr>
<tr><td>Alcance</td><td colspan="2">El riesgo presenta un impacto en el alcance.</td></tr>
<tr><td>Presupuesto</td><td colspan="2">El riesgo presenta un impacto en el presupuesto.</td></tr>
<tr><td>Calendario</td><td colspan="2">El riesgo presenta un impacto en el calendario.</td></tr>
<tr><td rowspan="2">Estado</td><td rowspan="2">Especifica el estado actual del riesgo dentro del proceso</td><td><u>Nombre del estado</u></td><td><u>Cod.</u></td><td><u>Definición</u></td></tr>
<tr><td>Nuevo</td><td>10</td><td>Nuevo riesgo.</td></tr>
</table>

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES		
	seleccionado o definido.	Abierto	20	Riesgo aceptado y asignado.
		Cancelado	90	Un riesgo que ha dejado de ser verificado por el proyecto.
		Plan Creado	30	El plan para el riesgo ha sido creado y se encuentra pendiente de aprobación.
		Plan Aprobado	35	El plan para el riesgo ha sido aprobado y se encuentra en condiciones de ser ejecutado.
		Plan – Verde	40	El plan se esta ejecutando según lo planificado.
		Plan – Amarrillo	41	El plan se esta ejecutando con leves diferencias respecto a lo planificado.
		Plan – Rojo	42	El plan se esta ejecutando con severas diferencias respecto a lo planificado.
		Plan completo	50	El plan se ha ejecutado por completo y se encuentra pendiente la verificación de sus resultados.

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES		
		Completado	60	El plan ejecutado ha sido verificado y sus resultados se consideran apropiados.
		Re-Abierto	70	El plan ejecutado ha sido verificado y sus resultados no se consideran apropiados por lo cual se solicita una nueva ejecución del ciclo de vida o proceso del riesgo.
		Completo, luego de Re-Abierto	80	El plan ejecutado ha sido verificado y sus resultados se consideran apropiados.
Probabilidad de Ocurrencia	Indica la probabilidad de que un riesgo se produzca.	**Nombre del Estado**	**Cod.**	**Definición**
		Alta	3	Los datos y análisis indican una alta probabilidad de ocurrencia.
		Media	2	Los datos y análisis indican una moderada probabilidad de ocurrencia.
		Baja	1	Los datos y análisis indican una escasa probabilidad de ocurrencia.
Probabilidad del Impacto	Indica el impacto estimado para el riesgo sobre cada una de las áreas	**Alcance**		
		Nombre del Estado	**Cod.**	**Definición**

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES		
	anteriormente identificadas como afectadas por el mismo. Aquel que sea el mayor de los impactos individuales será considerado como el impacto final para el riesgo (*por ejemplo*: si el impacto sobre el alcance fuera alto y sobre todas las demás áreas fuera bajo, se consideraría el impacto total del riesgo como alto).	Alto	3	Por ejemplo, incremento en el alcance > 20%.
		Medio	2	Por ejemplo, incremento en el alcance < 15% y >10%
		Bajo	1	Por ejemplo, sin impacto o con impacto mínimo sobre el alcance.
		Presupuesto		
		Nombre del Estado	**Cod.**	**Definición**
		Alto	3	Por ejemplo, disminución del presupuesto > 25%.
		Medio	2	Por ejemplo, disminución del presupuesto < 25% y >15%
		Bajo	1	Por ejemplo, sin impacto o con impacto mínimo sobre el presupuesto.
		Calendario		
		Nombre del Estado	**Cod.**	**Definición**
		Alto	3	Por ejemplo, >25% de incremento en el calendario.
		Medio	2	Por ejemplo, <25% y >10% de incremento en el calendario.

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES		
		Bajo	1	Por ejemplo, sin impacto o con impacto mínimo sobre el calendario.
		Administración de Cambios		
		Nombre del Estado	**Cod.**	**Definición**
		Alto	3	Por ejemplo, resistencia significativa y a largo plazo para el uso de herramientas automáticas.
		Medio	2	Por ejemplo, resistencia moderada y a corto plazo para el uso de herramientas automáticas.
		Bajo	1	Por ejemplo, sin impacto o con impacto mínimo en usuarios y participantes del proyecto.
		Negocio		
		Nombre del Estado	**Cod.**	**Definición**
		Alto	3	Por ejemplo, degradación significativa del proceso de negocios por falta de ejecución o la imposibilidad de alcanzar objetivos.

NOMBRE DEL CAMPO	DEFINICIÓN	POSIBLE LISTA DE VALORES		
		Medio	2	Por ejemplo, reducción de los beneficios producidos por el proceso de negocios.
		Bajo	1	Por ejemplo, sin impacto o con impacto mínimo sobre los beneficios del negocio.
Exposición del Riesgo	Indicador de la importancia final del riesgo, en muchos casos se obtiene multiplicando los valores numéricos asociados a la probabilidad de ocurrencia y al impacto.	Valor numérico.		
Disparador del Riesgo	Un evento, medida, escenario u ocurrencia que dispara acciones tendientes a evitar que el riesgo se transforme en un problema.	Texto en lenguaje natural de forma libre.		
Plan del Riesgo	Plan completo para tratar con el riesgo según la estrategia seleccionada.	Documento separado.		

Tabla 1 - Ejemplo de Elementos de un Riesgo

2.3. La Gestión de Riesgos

Un efectivo proceso de **gestión de riesgos** es un importante componente en todo proyecto de desarrollo de software exitoso; el principal objetivo de dicho proceso lo constituye el posibilitar tanto al proyecto como a las organizaciones a las cuáles afecta el cumplimiento de su misión y de sus propósitos.

La gestión de riesgos permite definir en forma estructurada, operacional y organizacional, una serie de actividades para gestionar los riesgos de los proyectos a lo largo de todas las fases de su ciclo de vida de desarrollo de software. En la mayor parte de los casos, esto se traduce en la creación de planes tendientes a impedir que los riesgos se transformen en problemas o a minimizar su probabilidad de ocurrencia o impacto.

[Wideman, 1998] destaca la importancia de la realización de una gestión de riesgos durante todo el ciclo de vida mediante el siguiente gráfico que ilustra la relación existente entre los riesgos y las diferentes fases del desarrollo (véase **Figura 5 - Los Riesgos y las Fases de Desarrollo**):

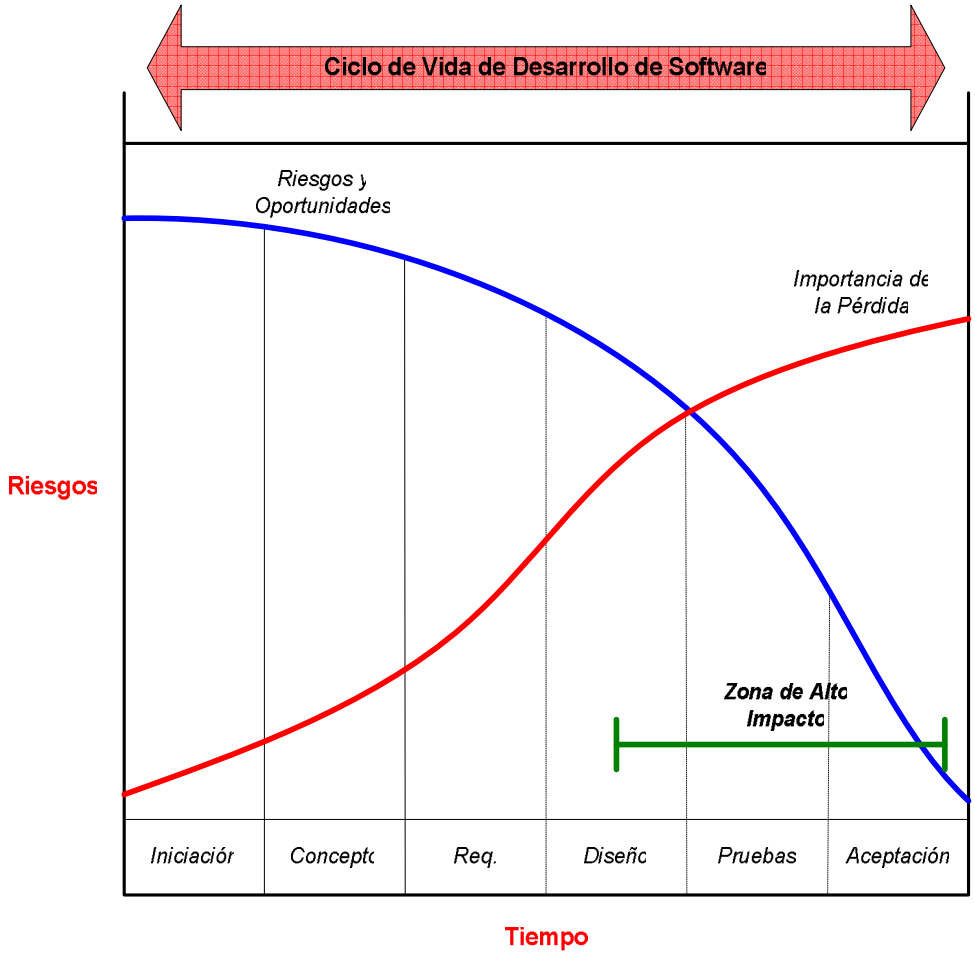


Figura 5 - Los Riesgos y las Fases de Desarrollo

Aún cuando la gestión de riesgos es fundamental para proyectos de cualquier tipo, para los proyectos de desarrollo de software, la gestión de riesgos constituye una actividad fundamental debido a que dichos emprendimientos se ven caracterizados por una serie de aspectos particulares que incrementan la presión sobre el proyecto a medida que dificultan la toma de decisiones de los responsables por la administración, entre ellos:

- Las constantes presiones internas y externas.

- Los cambios normativos y la evolución de las técnicas que pueden derivar en una alteración de los planes y de las estrategias.
- Los continuos y comunes cambios en los requerimientos de los usuarios.
- Las nuevas herramientas y tecnologías existentes en el mercado.
- Las periódicas amenazas de seguridad.
- La variabilidad de los equipos de proyecto.

A nivel organizacional, una política de gestión de riesgos debería considerar, al menos, los siguientes aspectos [Motorola LMPS, 1999]:

- Todos los riesgos relacionados con los proyectos de desarrollo de software deben ser identificados, analizados, priorizados y revisados siguiendo un Plan de Gestión de Riesgos.
- El Plan de Gestión de Riesgos debe ser documentado apropiadamente, ya sea como un documento independiente o formando parte del Plan de Proyecto.
- Como consecuencia de los constantes cambios, la lista de los riesgos y la información relacionada con su estado actual e historia reciente, deben ser mantenidos en una Base de Datos de Riesgos de Proyecto separada.
- La información contenida en la Base de Datos de Riesgos debe ser utilizada para acrecentar la información contenida en una Base de Datos de Riesgos Organizacional.

Según American Systems Corporation [ASC, 2003], existen un conjunto de reglas básicas relacionadas con la Gestión de Riesgos, las mismas se enumeran a continuación:

- **Regla #1:**
  "*Los proyectos que no hacen gestión de riesgos se encuentran progresando a riesgo*".
- **Regla #2:**
  "*La gestión de riesgos no es gratuita, es necesario comprometer recursos, establecer planes y procesos y disponer de reservas*".
- **Regla #3:**
  "*Centralice las responsabilidades de gestión de riesgos porque la responsabilidad de administración distribuida debe ser coordinada*".
- **Regla #4:**
  "*Priorice los riesgos y concéntrese en aquellos que sean mas críticos; a pesar de esto, todos los demás riesgos deberían contar con una estrategia de mitigación*".
- **Regla #5:**
  "*Los administradores de un proyecto son responsables por las acciones mientras que los administradores de los riesgos del proyecto son responsables por la identificación y seguimiento de los riesgos*".
- **Regla #6:**
  "*El proceso de gestión de riesgos debe definirse y seguirse de forma consistente en toda la organización. Todas las actividades deben ser tendientes a cumplimentar los requisitos de una política organizacional de gestión de riesgos*".

Finalmente, [Microsoft, 2002] ha definido una serie de principios básicos sobre los cuales se debería basar un proceso de gestión de riesgos:

- **Agilidad:**
  La agilidad exige que los equipos de proyectos valoren ininterrumpidamente y administren de manera proactiva los riesgos durante *todas* las fases del ciclo de vida porque los continuos cambios en todas las facetas del proyecto implican, además, cambios a nivel de los riesgos. Un enfoque proactivo permite que un equipo acepte los cambios y los transforme en una oportunidad.

- **Potenciar la comunicación:**
  Los riesgos deberían discutirse de forma abierta, tanto dentro como fuera del equipo de proyecto. Todos los integrantes del equipo deben participar, en mayor o menor medida, de las diferentes tareas relacionadas con la gestión de riesgos. Los miembros de un equipo no deberían tener reservas de ningún tipo en comunicar sus opiniones con libertad para, de esta forma, evaluar con mayor precisión el estado del proyecto y tomar decisiones consensuadas.

- **Aprender de todas las experiencias:**
  Ya que el aprendizaje sólo puede ayudar a mejorar los resultados, el conocimiento obtenido en un proyecto puede reducir la incertidumbre de la toma de decisiones en otros proyectos cuando la información es poco fiable. El análisis directo de los resultados de proyectos anteriores fomenta el aprendizaje dentro del equipo mediante el intercambio de opiniones entre sus miembros.

- **Participación necesaria:**
  La participación activa en el proceso de gestión de riesgos es responsabilidad de todos los integrantes del equipo, aún cuando la responsabilidad por la administración de los riesgos deba ser centralizada. Asimismo, los diferentes miembros del equipo de proyecto reciben asignaciones de tareas específicas de las cuales son responsables.

- **El riesgo es inherente en cualquier proyecto o proceso:**
  Aunque un proyecto puede tener más o menos riesgos que otro, no existe ningún proyecto que no se vea amenazado por algún riesgo.

- **Proactividad:**
  La gestión de riesgos proactiva es la forma más eficaz de hacer gestión de riesgos ya que posibilita:
  - Anticiparse a los problemas en lugar de reaccionar ante ellos.
  - Tratar la raíz de un problema y no sus síntomas.
  - Reducir los tiempos de respuestas y minimizar el daño ocasionado por un problema en momentos de crisis.

- **La identificación de un riesgo es algo positivo:**
  Para llevar a cabo una gestión de riesgos de forma efectiva es preciso comprender los riesgos a los que se enfrenta el equipo del proyecto. Cuando la diversidad de los retos y la magnitud de las pérdidas potenciales son muy evidentes, las actividades relacionadas con la identificación de riesgos pueden influir negativamente en la moral de los miembros del equipo, algunos de los cuales pueden, incluso, tener la impresión de que dicha tarea agrega mayor complejidad a la consecución final del proyecto. En realidad, el proceso de identificación de riesgos permite al equipo gestionar los riesgos de forma más eficaz ya que posibilita el ponerlos de manifiesto y de este modo poder tratarlos. Esto implica considerar los riesgos como la única forma de crear la oportunidad adecuada para conseguir los objetivos del proyecto.

- **Valoración continua:**
  Los cambios continuos en los proyectos y en los entornos operativos obligan a los equipos a realizar valoraciones frecuentes del estado de los riesgos existentes y a valorar o a actualizar los planes para prevenir o actuar ante ellos. Se debe buscar, además, y de forma constante, la posible aparición de nuevos riesgos. Las actividades de gestión de riesgos deben integrarse en el ciclo de vida general del proyecto.

- **Primero especificar y luego administrar:**
  El punto de conjunción entre la gestión de riesgos y la toma de decisiones lo constituye la incertidumbre. Las definiciones genéricas de un riesgo no hacen desaparecer la incertidumbre y dan lugar a distintas interpretaciones del mismo. Las definiciones que no dejan lugar a dudas permiten:
    - Asegurar que todos los participantes comprenden el riesgo de la misma forma.
    - Comprender la causa o causas de los riesgos y la relación con los problemas que puedan derivarse.
    - Disponer de una base para realizar un análisis formal y cuantitativo y planear los esfuerzos asociados a las acciones relacionadas con el proceso de administración del riesgo.
    - Aumentar la confianza que los patrocinadores tienen depositada en la capacidad del equipo.

- **Las situaciones no deben juzgarse sólo por el número de riesgos:**
  A pesar de que los miembros del equipo y los patrocinadores suelen percibir los elementos de riesgo como algo negativo, los proyectos nunca deben juzgarse por el número de riesgos detectados. Es importante recordar que, un riesgo es sólo una posibilidad, no la certeza de una pérdida ni un resultado por debajo de lo esperado.

La siguiente figura [Walker, 1998] ilustra la relación existente entre un enfoque proactivo y un enfoque tradicional para el desarrollo de actividades de gestión

de riesgos a lo largo del ciclo de vida de un proyecto de desarrollo de software (véase **Figura 6 - Perfil de los Riesgos en Proyectos de Software**):

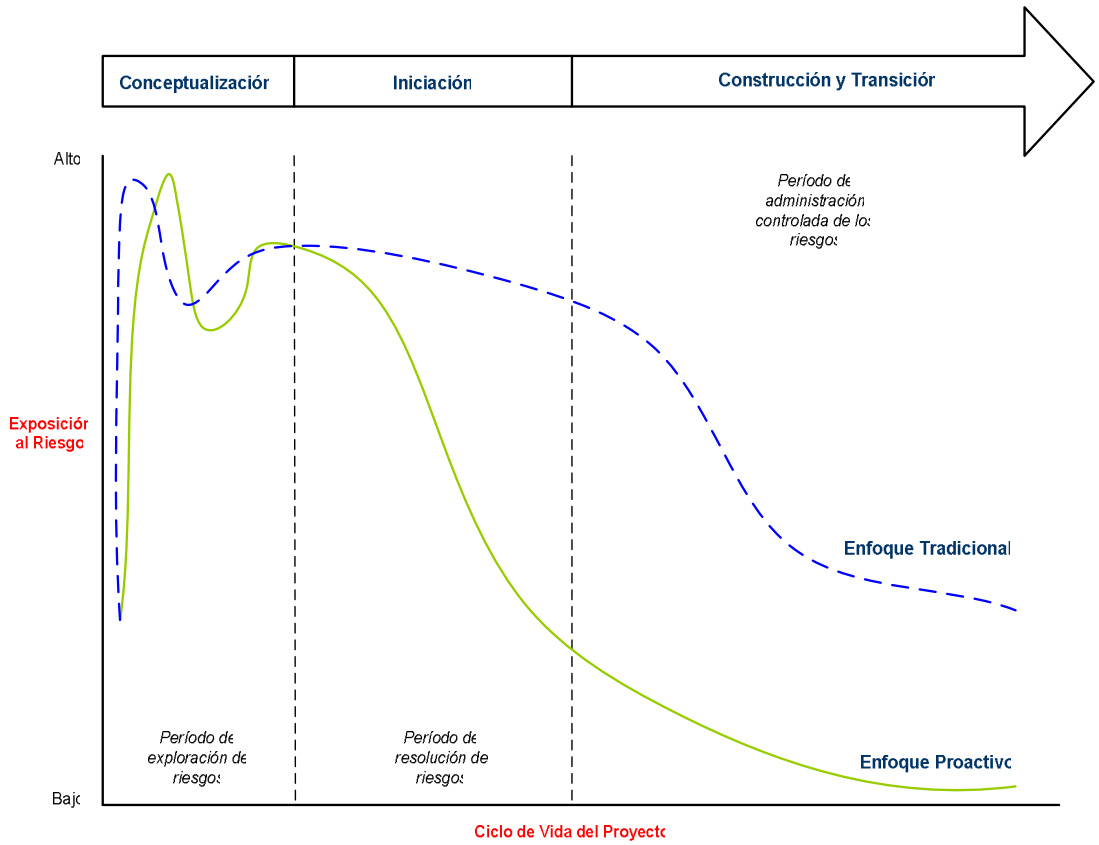


Figura 6 - Perfil de los Riesgos en Proyectos de Software

En resumen, la gestión de riesgos se basa en la noción de que los mismos deben tratarse de forma proactiva y de que dicha administración forma parte de un proceso formal y sistemático que debe considerarse como una iniciativa positiva.

2.3.1. Modelo de Gestión de Riesgos

El modelo de gestión de riesgos seleccionado consta de cinco pasos, etapas o fases: *Identificación*, *Análisis*, *Planificación*, *Seguimiento* y *Control*, compartiendo como actividades comunes las de *Documentación* y *Comunicación* (véase **Figura 1 - Modelo de Gestión de Riesgos**).

El modelo se grafica de manera circular de forma tal de demostrar que la gestión de riesgos constituye un proceso continuo mientras que la dirección de las flechas intenta indicar el sentido lógico de ejecución e intercambio de información entre los pasos que componen el modelo. Finalmente, la comunicación y la documentación se sitúan en el centro con un doble propósito: primero, el de resaltar el hecho de que conforman los elementos conectores del modelo y, segundo, el de remarcar el hecho de que en muchos

casos constituyen el mayor de los obstáculos del proceso de gestión de riesgos. [Marvin J. Carr *et al.*, 1993]

La primera de las actividades a llevar a cabo en el modelo de gestión de riesgos es su **identificación** que permite anticipar los riesgos antes de que los mismos se transformen en problemas reales que puedan afectar adversamente el desarrollo del proyecto. Es importante destacar que la identificación de riesgos (al igual que cada una de las etapas del modelo) debería ser llevada a cabo de forma disciplinada y consistente, alentando a cada uno de los miembros del equipo de proyecto a expresar sus preocupaciones y posibilitando de este modo su análisis posterior.

Durante la etapa de **análisis** se pretende transformar una serie de datos que han sido obtenidos en la etapa de identificación, en información que permita llevar adelante una toma de decisiones enfocada en los riesgos más importantes para el proyecto. El análisis de riesgos es un proceso sistemático de estimación de la probabilidad de ocurrencia y la magnitud de la pérdida o impacto de cada uno de los riesgos identificados mediante el cual se logra reducir la incertidumbre de la medida y del resultado del acontecimiento asociado a un riesgo.

En la fase de **planificación**, se transforma a la información relacionada con los riesgos en decisiones y acciones presentes y futuras. La planificación involucra tanto el desarrollo de acciones individuales para cada uno de los riesgos como la priorización de dichas acciones y la creación de un plan integral de administración.

El **seguimiento** consiste en monitorizar constantemente tanto del estado de los riesgos como de las acciones tomadas con el propósito de evitarlos o mitigar su impacto negativo. El seguimiento de los riesgos trae inevitablemente asociada la necesidad de la toma de una serie de métricas relacionadas con la gestión de riesgos que permitan a los responsables de la administración del proyecto el llevar a cabo una estricta y permanente verificación de los planes.

El **control** de riesgos posibilita el corregir las desviaciones que puedan producirse a los planes de acción implementados.

La **comunicación** y **documentación** del proceso son claves en el modelo debido a que una falta de efectividad en las mismas impide la viabilidad de cualquier estrategia de administración aplicable. La comunicación esta presente y se manifiesta a diferentes niveles en el modelo: un primer nivel lo constituye la comunicación que debe realizarse entre los diferentes pasos del proceso, un segundo nivel esta caracterizado por la comunicación interna del equipo de proyecto y un tercer nivel por la que surge entre el proyecto y los diferentes participantes del mismo.

2.3.2. Estrategias para Tratar con Riesgos

Una de las más importantes decisiones que deben tomarse al hacer gestión de riesgos tiene que ver con la estrategia de respuesta a adoptar para cada uno de los riesgos sobre los cuales se focaliza la atención del proyecto.

Es importante considerar el hecho de que para cada uno de los riesgos identificados puede seleccionarse una estrategia diferente y que la estrategia elegida puede ser reemplazada por una diferente a medida que se producen revisiones sucesivas de los riesgos.

Formalmente, se define a una **estrategia de respuesta** como una actividad que puede ser realizada con el propósito de manejar un riesgo antes de que el mismo se dispare.

Existen diferentes estrategias de respuesta, entre las más reconocidas y utilizadas pueden destacarse:

- **Aceptar:**
  El uso de esta técnica indica un reconocimiento explícito hacia el riesgo y su incontrolabilidad. La aceptación es una técnica *pasiva* que se enfoca en la admisión de cualquier consecuencia derivada de un riesgo sin tratar de prevenirla. Esta estrategia se emplea generalmente para riesgos considerados de baja o muy baja importancia donde no es posible detectar una ganancia significativa al intentar lograr una reducción del riesgo.

- **Evitar:**
  Esta técnica emplea un enfoque centrado en el evitar la posibilidad de ocurrencia de un riesgo. En muchos casos, el empleo de esta estrategia, implica la realización de cambios sobre las condiciones inicialmente pautadas para el proyecto, por ejemplo:
  - Reducción del alcance del trabajo.
  - Cambios de requerimientos y/o especificaciones.
  - Cambios en el SOW (*Statement of Work*).
  - Cambios en las líneas base del proyecto.

- **Controlar:**
  Esta técnica se caracteriza por la toma de *acciones concretas y puntuales* tendientes a disminuir la probabilidad de ocurrencia o el impacto para un riesgo. Dichas acciones se llevan a cabo generalmente a lo largo de todo el ciclo de vida del proyecto y se constituyen en el tipo de respuesta más común para los riesgos. En muchos casos, las acciones asociadas a este tipo de estrategia se consideran como un *producto* del proyecto y se controlan y monitorean como parte de las acciones normales y habituales de seguimiento.

- **Investigar:**
  En esta técnica se difieren todas las acciones hasta que se haya realizado una mayor cantidad de trabajo y/o hasta que se conozcan

una mayor cantidad de hechos. Es importante destacar que no se define ningún tipo de respuesta para la reducción de un riesgo, simplemente porque la estrategia se emplea cuando no es posible identificar una solución clara y se requiere llevar a cabo una investigación. Puede incluirse como parte del desarrollo de la técnica el realizar análisis de causa raíz de problemas.

- **Reducir:**
  En esta técnica se trabaja *activamente* para lograr una disminución de los riesgos en base a la realización de una serie de *acciones planificadas y constantes*, tales como: uso de prototipado, revisiones con especialistas, formación temprana de un equipo multidisciplinario, simulaciones, modelados, reuniones del equipo de trabajo, reducción de dependencias, involucramiento del cliente, etc.

- **Transferir:**
  Transferir es el proceso de mover algo de un lugar a otro o de una persona a otra. Esta técnica hace referencia, justamente, a esta idea o concepto, por ejemplo: transferir un riesgo del equipo de proyecto a un cliente o un proveedor. Típicamente la transferencia implica el hacer uso de proveedores especializados en una determinada área capaces de reducir la exposición final del riesgo.

2.4. Identificación de Riesgos

La **identificación de riesgos** es la primera de las fases en el proceso de administración y consiste en la determinación integral de elementos potenciales de riesgo empleando algún método estructurado y consistente con el fin de lograr la reducción de la "incertidumbre descriptiva", es decir, con el propósito de lograr una apropiada caracterización de los posibles inconvenientes que puedan presentarse en el proyecto posibilitando de esta forma la administración de los mismo (véase **Figura 7 - La Incertidumbre Descriptiva**).



Figura 7 - La Incertidumbre Descriptiva

Este es el más importante de todos los pasos del modelo debido a que si los riesgos no son identificados de forma apropiada, es imposible el tomar acciones posteriores para mitigarlos y, además, debido a que da comienzo efectivo al proceso de gestión de riesgos.

La identificación de riesgos depende fuertemente de dos aspectos fundamentales: el primero, es la comunicación abierta y, el segundo, la capacidad de prever inconvenientes a futuro. Aún cuando es importante la participación individual de los miembros del equipo de trabajo, el éxito en esta actividad se basa en la participación conjunta para la generación de ideas en base a la combinación de experiencias y opiniones.

Quizás el aspecto mas importante en esta etapa sea el hecho de capturar tantos riesgos como sea posible; seguramente no se trabajará sobre todos esos riesgos porque un análisis posterior permitirá seleccionar aquellos sobre los cuales el proyecto deberá centrarse, pero de esta forma se garantiza el abarcar la mayor cantidad de potenciales inconvenientes futuros.

Debido a que puede no ser posible reconocer todos los riesgos asociados a un proyecto en un determinado momento, es importante reiniciar este paso del proceso de forma periódica a lo largo del ciclo de vida del proyecto de manera tal de lograr la identificación de algunos riesgos que podrían haber permanecido ocultos o ser inexistentes en momentos previos del desarrollo.

Las *entradas* en la identificación de riesgos son toda la información disponible acerca de riesgos generales y específicos del proyecto en las áreas de negocios, técnicas, organizacionales y de entorno que resulten importantes y destacables. Otros aspectos adicionales a considerar son la experiencia del equipo y el enfoque organizativo actual ante los riesgos en forma de políticas, procesos, plantillas e información acerca del proyecto, incluido su estado actual e histórico. Debe tenerse en cuenta todo aquello que el equipo de proyecto considere de ayuda para identificar los riesgos. Al iniciar las actividades durante las primeras fases del ciclo de vida, es muy útil poner en común una serie de conocimientos, realizando sesiones dirigidas o incluso talleres para obtener información sobre las percepciones que cada uno de los participantes del proyecto tienen acerca de los riesgos y las oportunidades. Para identificar los riesgos relevantes de los proyectos, se puede recurrir a esquemas de clasificación de la industria, listas de control de proyectos, informes de resumen de proyectos anteriores o cualquier fuente de información publicada.

Durante esta fase es necesario usar la estructura del proyecto como base para la tarea a realizar, deben asignarse las actividades de identificación a los miembros mas apropiados del equipo, es preciso ser minucioso pero no absurdo en el proceso, debe intentarse el desarrollo de una lista de riesgos tan completa como sea posible y no deben analizarse los riesgos identificados.

El *resultado* esperado de las actividades antes mencionadas es una declaración clara, sin ambigüedades y consensuada registrada en forma de lista de riesgos con los que debe enfrentarse el equipo de proyecto. La lista de riesgos en forma de tabla constituye la entrada principal para la siguiente etapa de análisis del proceso de gestión de riesgos. La identificación de riesgos suele generar una gran cantidad de información útil, incluida la identificación de las causas raíz y los efectos que provocan, las partes afectadas, el propietario, etc. (véase **Tabla 1 - Ejemplo de Elementos de un Riesgo**).

2.4.1. Técnicas de Identificación de Riesgos

Existen varios métodos y herramientas que suelen ser empleados durante la etapa de identificación de riesgos; en muchas oportunidades, la decisión respecto a qué método utilizar se basa en las características particulares del equipo de proyecto con el que se trabaja. Es importante destacar además que, comúnmente, estas técnicas suelen emplearse de manera combinada. Algunas de las más popularmente aceptadas son:

- **Tormenta de ideas:**
  En este método, los miembros identifican riesgos sin emplear ninguna estructura como base y de forma oral para permitir a los demás participantes el construir nuevas ideas en base a los pensamientos de otros. Para lograr los resultados esperados, es importante elegir como miembros del grupo a personas que estén familiarizadas con los temas a discutir y, en mucho casos, suele ser común el entregar documentación pertinente antes de las sesiones de identificación de riesgos con el fin de lograr una mayor efectividad. El uso de esta técnica suele trasladarse además a otras etapas (por ejemplo, al momento de generar estrategias de mitigación).

- **Encuestas:**
  Las encuestas contienen una serie de preguntas tendientes a realizar la identificación de riesgos en un área particular de interés. Un inconveniente intrínseco de este método es que la gente suele no sentirse atraída por las mismas, motivo por el cual pueden brindar información inapropiada; además, puede ser dificultoso evaluar las respuestas por la evidente subjetividad asociada a las mismas. Su ventaja principal lo constituye la posibilidad de obtener una amplia variedad de opiniones sin necesidad de organizar una reunión para contar con todos los participantes (algunos de los cuales pueden ser de difícil acceso).

- **Entrevistas:**
  Es una técnica muy poderosa y se basa en la idea del uso de una serie de consultas para determinar la línea base de riesgos. Asimismo, debe considerarse que el proceso de entrevistas no es más que un mecanismo de preguntas y respuestas cuyo resultado final será fuertemente dependiente de la capacidad del entrevistador y de la corrección de las preguntas. Suele combinarse el uso de entrevistas con el de la tormenta de ideas, ya sea antes o después de la sesión, como entrada para la generación de ideas o mecanismo de validación respectivamente.

- **Grupos de trabajo:**
  Son un mecanismo muy importante para el análisis de un área o tema particular al realizar un proceso de discusión que posibilita el descubrimiento de riesgos que podrían no ser obvios para el grupo

encargado específicamente de la identificación. Generalmente, estos grupos de trabajo están compuestos por personas especializadas en alguna determinada temática relacionada con el proyecto.

- **Conocimiento por experiencia o documentado:**
  El *conocimiento por experiencia* es una colección de información que una persona ha obtenido mediante su experiencia. El *conocimiento documentado* es una colección de información que ha sido documentada respecto a un determinado tema (en este caso particular, riesgos en una determinada área de interés). Es importante considerar que debe validarse la relevancia y aplicabilidad de los conocimientos del experto y de la documentación pertinente al hacer uso de estas técnicas.

- **Taxonomías de riesgos – Lecciones aprendidas:**
  Las *taxonomías* son listados de riesgos que han sido encontrados en programas, proyectos o situaciones similares a las que se intenta analizar. Es importante considerar que debe validarse la relevancia y aplicabilidad de la información al hacer uso de esta técnica. Las taxonomías se basan en el uso de preguntas que hacen referencia a situaciones o eventos sobre un área particular de un proyecto o programa y que pueden derivar en una serie de riesgos para el mismo; generalmente estas preguntas se encuentran agrupadas por áreas temáticas (rendimientos, costos, calendario, por ejemplo). Las *lecciones aprendidas* son conocimiento basado en la experiencia y que han sido compiladas a lo largo del tiempo como información que puede ser relevante durante la identificación de riesgos para los diferentes proyectos que se están ejecutando en una organización. Es importante considerar que debe validarse la relevancia y aplicabilidad de la información consultada al hacer uso de esta técnica.

- **Salidas del Análisis Orientado a Riesgos:**
  Existen varios tipos de análisis orientado a riesgos, uno de ellos se basa en el uso de *árboles de análisis de fallas* y otro en el uso de árboles de *análisis de eventos*; ambos constituyen un razonamiento de arriba hacia abajo que intenta determinar que eventos, condiciones o fallas pueden llevar a una situación indeseable. Esta situación con su consecuencia asociada puede constituir un riesgo para el proyecto o programa.

- **Información histórica:**
  Es básicamente lo mismo que la información documentada y suele usarse del mismo modo, la diferencia entre ambas surge porque la información histórica suele ser ampliamente considerada como un hecho válido.

- **Plantillas de ingeniería:**
  Constituyen un conjunto de diagramas de flujo respecto a varios aspectos del proceso de desarrollo y se usan como una guía contra la cual validar, empleando una visión de arriba hacia abajo, las actividades a realizar durante un proyecto. Debido a que poseen información de

entradas, salidas y acciones esperadas para cada parte del proceso como así también de los factores involucrados en las mismas, suelen ofrecer un importante marco de referencia. Su principal inconveniente lo constituye el hecho de que estas plantillas pueden estar incompletas por lo cual se recomienda hacer una exhaustiva revisión de las mismas en caso de optarse por su utilización.

- **Plantillas de camino crítico:**
  Estas plantillas reciben también el nombre de *plantillas Willoughby* y se caracterizan por mostrar áreas de riesgos y proveen un mecanismo de reducción de los mismos para cada una de las etapas de un ciclo de vida de desarrollo de software por el hecho de posibilitar fácilmente su identificación. Debido a que se realizan constantemente nuevas publicaciones actualizadas de estas plantillas, es importante considerar el contar con información reciente al emplear esta técnica.

2.5. Taxonomías

El método de **identificación de riesgos basado en taxonomías** fue desarrollado originariamente por el SEI [Marvin J. Carr *et al.*, 1993], y, en su versión original, trabaja agrupando las distintas fuentes de riesgos en varias categorías y proveyendo de un cuestionario llamado TBQ (*Taxonomy-Based Questionnaire*) para realizar un proceso sistemático de identificación.

La taxonomía de riesgos presentada sigue el tradicional ciclo de vida de desarrollo de software en cascada y provee un marco para organizar los datos y la información. Adicionalmente, la estructura fundamental del método consta de un *Cuestionario Basado en Taxonomías* y un proceso para su aplicación.

La taxonomía organiza los riesgos de desarrollo de software en tres niveles (véase **Figura 8 - Organización de la Taxonomía de Riesgos**):

- Clases.
- Elementos.
- Atributos.

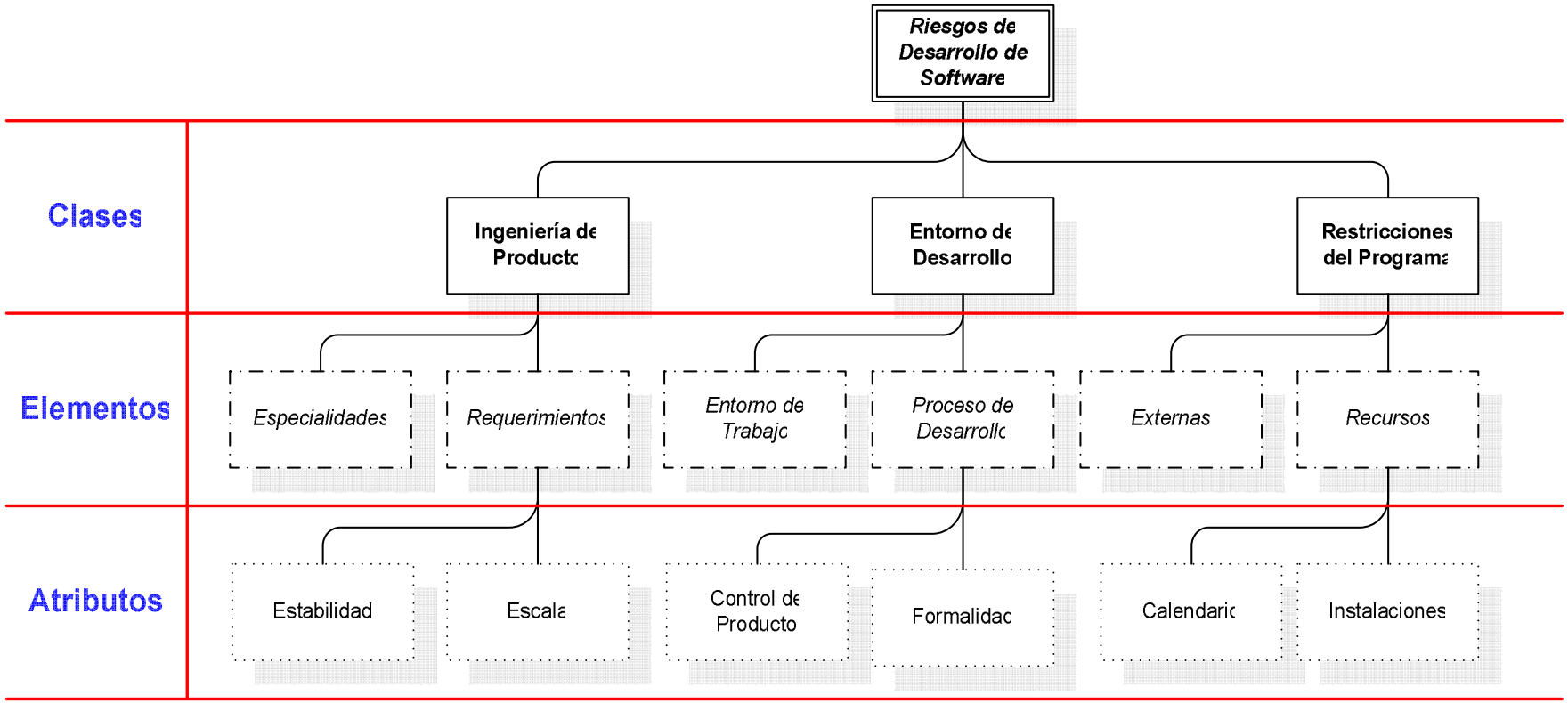


Figura 8 - Organización de la Taxonomía de Riesgos

Tal como puede observarse, la taxonomía esta organizada en tres grandes clases que a su vez se dividen en los elementos que las componen, cada uno de los cuales se caracteriza por una serie de atributos. Las tres clases antes mencionadas son:

- **Ingeniería de Producto:**
  Incluye los aspectos técnicos del trabajo que debe ser realizado.

- **Entorno de Desarrollo:**
  Involucra los métodos, procedimientos y herramientas empleadas para producir el producto.

- **Restricciones del Programa:**
  Abarca los factores contractuales, organizacionales y operacionales dentro de cuyo marco se desarrolla el software pero que, generalmente, se encuentran fuera del marco de la operatoria de la administración del proyecto.

El cuestionario, a su vez, consta de una serie de preguntas objetivas y relacionadas a cada uno de los atributos de la taxonomía y diseñadas para educir un conjunto de riesgos y preocupaciones que potencialmente puedan afectar al producto o al proyecto en el cual se trabaja. El proceso ha sido desarrollado considerando, además, la posibilidad de realizar una aplicación práctica, eficiente y consistente del cuestionario en los diferentes proyectos de una organización. La aplicación del cuestionario es semiestructurada y tanto las preguntas como sus consecuencias se emplean como un instrumento de definición de riesgos que no limita el empleo de otras metodologías; mas aún, este cuestionario suele ser definido como una tormenta de ideas estructurada. El TBQ produce mejores resultados cuando se lo aplica a través de un equipo independiente.

El proceso para la aplicación de TBQ consta de cuatro etapas secuenciales (véase **Figura 9 - El proceso de TBQ**):

1. **Compromiso de la Gerencia:**
   Antes de comenzar con la realización del cuestionario es necesario contar con la aceptación y el compromiso de los ejecutivos de la organización como así también el llevar a cabo la selección del proyecto sobre el cual se ejecutará la técnica y la elección de participantes de diferentes áreas, por ejemplo, personal de calidad o del equipo encargado de las pruebas.

2. **Selección del Equipo y Entrenamiento:**
   La conformación del equipo incluye tanto la selección del personal del proyecto como la incorporación de personal perteneciente a la organización cliente. Una vez conformado el equipo es preciso entrenar al grupo de trabajo en la técnica a utilizar, las responsabilidades de cada uno de los roles, el protocolo de entrevistas a emplear, etc.

3. **Identificación de Riesgos:**
   La etapa de identificación comienza con una sesión de introducción tanto del equipo como de la metodología a emplear y continúa con una serie de entrevistas al personal que ha sido seleccionado como miembro del grupo de participantes. Cada una de las entrevistas antes mencionadas se divide en dos etapas:
   - *Preguntas y respuestas:* Implica el uso explícito de TBQ.
   - *Clarificación:* Implica la aclaración de cualquier situación, comentario o expresión recogida en la etapa anterior y que pudiera presentar algún tipo de duda.

4. **Conclusión de la Identificación:**
   Una vez finalizada la identificación se presenta a los participantes los resultados obtenidos.

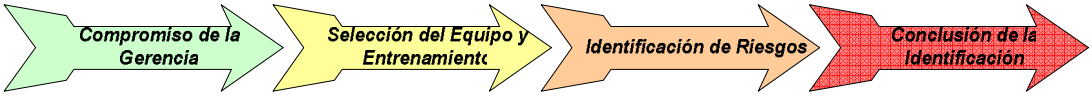


Figura 9 - El proceso de TBQ

Esta metodología es un instrumento que posibilita obtener una amplia gama de riesgos a nivel del sistema y que, adicionalmente, permite destacar aquellas áreas que requieran de mayor atención por parte del equipo de proyecto.

[Hantos, 2000] afirma que una serie de experiencias en el uso de taxonomías para las actividades de identificación de riesgos han demostrado que una adaptación previa a la realidad organizacional de las taxonomías estándar posibilita obtener mejores resultados con una mayor eficiencia y eficacia.

Para concluir, puede afirmarse que las taxonomías pueden usarse con diferentes propósitos a lo largo de todo el proceso de gestión de riesgos: durante la identificación, pueden utilizarse para estimular el pensamiento sobre los inconvenientes que pueden producirse en las distintas áreas del proyecto; durante la puesta en común de ideas aligeran la complejidad de trabajar con muchas cantidades de riesgos porque los riesgos similares pueden agruparse; también pueden emplearse para proporcionar una terminología unificada que el equipo de proyecto puede utilizar para supervisar y notificar el estado de los riesgos a lo largo del proyecto y, finalmente, son muy útiles para establecer las bases de conocimiento de riesgo en las organizaciones [Microsoft, 2002].

2.6. La Administración de Proyectos y CMMI

Con el propósito de realizar mejoras respecto al SW-CMM y al IPD-CMM, en el año 2000, el SEI publicó la versión 1.0 su nuevo modelo llamado **CMMI**.

CMMI incluye cuatro disciplinas en función de la amplitud de los procesos que cubre:

- **CMMI-SE (Ingeniería de Sistemas):** Cubre el desarrollo de sistemas integrales, los cuales pueden o no incluir software. Se focaliza en transformar necesidades, expectativas y restricciones de clientes en productos y en respaldar esos productos a lo largo de su vida.

- **CMMI-SW (Ingeniería del Software):** Cubre el desarrollo de sistemas de software. Se focaliza en aplicar un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software.

- **CMMI- IPPD (Desarrollo de Procesos y Productos Integrados):** Es un enfoque sistemático que logra una colaboración oportuna de las partes interesadas (stakeholders) relevantes en la vida de un producto que satisface necesidades, expectativas y requerimientos de clientes. Si un proyecto u organización elige IPPD, debe ejecutar sus mejores prácticas concurrentemente con las mejores prácticas usadas para producir un determinado producto.

- **CMMI-SS** (**Fuente Proveedora):** Cubre la adquisición de productos de proveedores. En forma similar a IPPD, las mejores prácticas relacionadas, se deben ejecutar concurrentemente con las mejores prácticas usadas para producir un determinado producto.

Dentro del modelo, se presentan dos posibles representaciones:

- **Por Niveles o Escalonado:**
  Permite evaluar el nivel de madurez de una organización en todas las áreas de proceso. El modelo de representación por niveles es similar al tradicionalmente aplicado para SW-CMM.

- **Continua:**
  Permite evaluar el nivel en cada área independientemente. El modelo de representación continua permite seleccionar el orden de mejoramiento que más se adapte y corresponda a los objetivos de la organización.

Considerando la aplicación de un enfoque escalonado, en CMMI se establecen cinco niveles de madurez con una interpretación similar a la empleada en SW-CMM:

1. Inicial.
2. Gestionado.
3. Definido.
4. Gestión Cuantitativa.
5. En Optimización.

Asimismo, en el caso de realizar la aplicación de un enfoque de representación continua, CMMI presenta diferentes niveles de capacidad, a saber:

0. Incompleto.
1. Desempeñado.
2. Gestionado.

3. Definido.
4. Gestión Cuantitativa.
5. En Optimización.

La importancia adquirida por las actividades de gestión de riesgo en proyectos de software puede observarse claramente al destacar que CMMI la considera como una de sus Áreas de Proceso de Nivel 3, estableciendo un marco y fijando pautas respecto a la forma en la cual deberían realizarse las actividades asociadas a dicha práctica.

El siguiente diagrama presenta un resumen de los aspectos más importantes de la organización por niveles de CMMI y destaca la inclusión de la gestión de riesgos en el modelo (véase **Figura 10 - La Gestión de Riesgos y CMMI**).

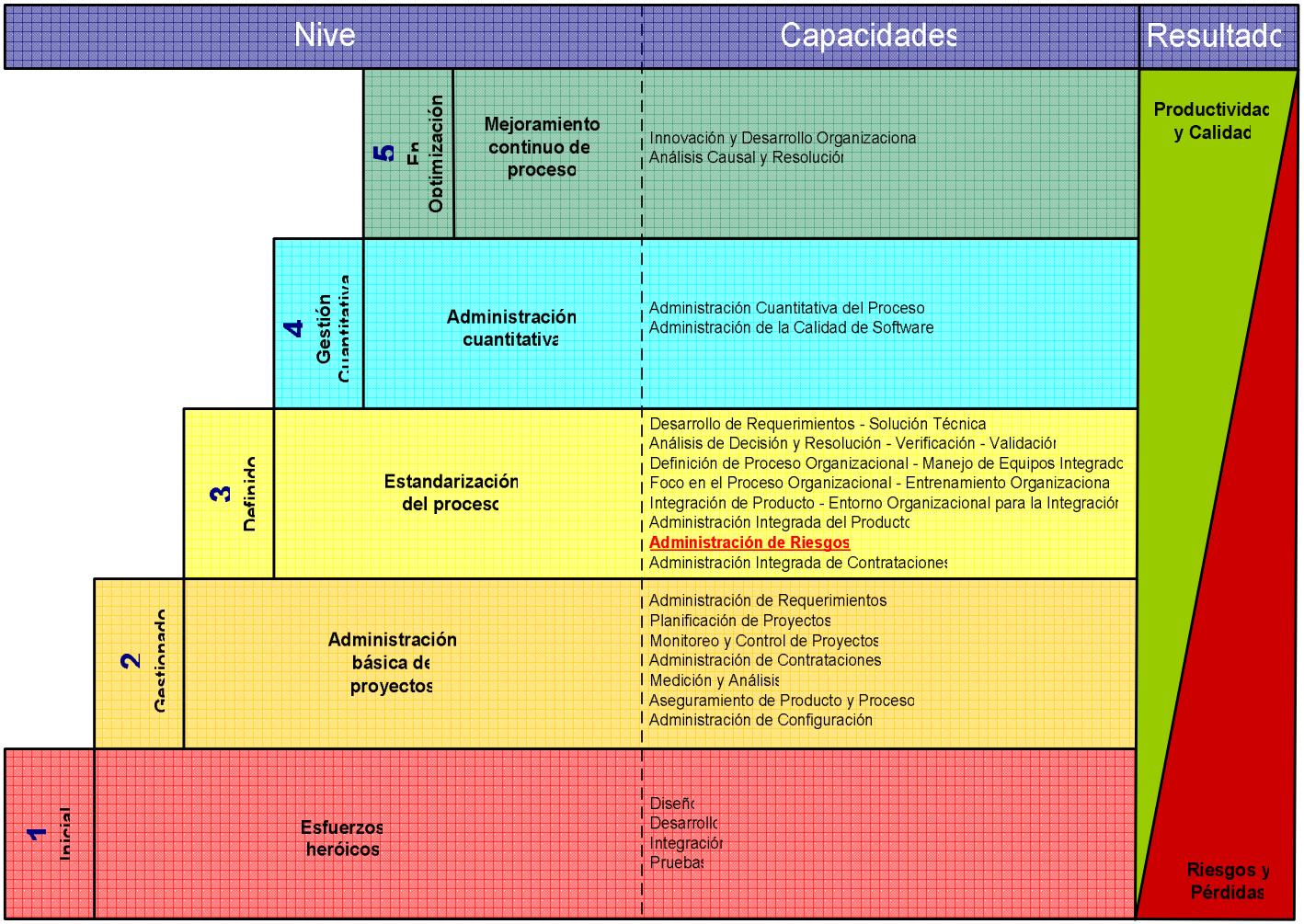


Figura 10 - La Gestión de Riesgos y CMMI

2.6.1. Gestión de Riesgos (CMMI)

Se presenta a continuación un extracto con algunos de los aspectos destacables del PA (*Process Area – Área de procesos*) de Gestión de Riesgos en CMMI [CMMI, 2002] basado en la identificación de SG (Specific Goal – Objetivos Específicos) y SP (Specific Practice – Prácticas Específicas) como así también en sus referencias al uso de taxonomías como herramienta recomendable durante la etapa de identificación de riesgos de proyectos de software (véase
Figura 2 - PA de Gestión de Riesgos en CMMI):

Propósito:

El propósito de la gestión de riesgos es identificar problemas potenciales antes de que ocurran, de forma tal que las actividades de manejo de riesgo pueden ser planificadas y utilizadas a lo largo de la vida del producto o del proyecto para mitigar un impacto adverso a la consecución de los objetivos.

Áreas de Proceso Relacionadas:

- *Planificación de Proyectos*
  Donde se definen algunos aspectos relacionados con la identificación de riesgos y la planificación tendientes a lograr el involucramiento de los participantes relevantes.

- *Monitorización y Control de Proyectos*
  Donde se define información adicional respecto a la manera de hacer monitorización de los riesgos del proyecto.

- *Análisis de Decisión y Resolución*
  Donde se define información sobre el uso de un proceso formal de evaluación para analizar alternativas para la selección y mitigación de los riesgos identificados.

Objetivos Específicos:

SG1	**Preparación para la Gestión de Riesgos**
	Se lleva a cabo la preparación para la gestión de riesgos.
SG2	**Identificar y Analizar Riesgos**
	Los riesgos son identificados y analizados para determinar su importancia relativa.
SG3	**Mitigación de Riesgos**
	Los riesgos son manejados y mitigados, cuando es apropiado, con el propósito de reducir su impacto adverso sobre la consecución de objetivos.

Relación entre Prácticas y Objetivos:

- SG1, Preparación para la Gestión de Riesgos:

- Determinar fuentes de riesgos y categorías.
- Definir parámetros de riesgos.
- Establecer y mantener una estrategia de gestión de riesgos.

- SG2, Identificar y Analizar Riesgos:
  - Identificar riesgos.
  - Evaluar, categorizar y priorizar riesgos.

- SG3, Mitigación de Riesgos:
  - Desarrollar planes de mitigación de riesgos.
  - Implementar planes de mitigación de riesgos.

Prácticas Específicas por Objetivos:

SG1	**Preparación para la Gestión de Riesgos**
	Se lleva a cabo la preparación para la gestión de riesgos.

La preparación se lleva a cabo mediante el establecimiento y mantenimiento de una estrategia para la identificación, el análisis y la mitigación de los riesgos lo cual se documenta típicamente en un plan de gestión de riesgos.

SP1.1-1	**Determinar fuentes de riesgos y categorías**
	Determinar fuentes de riesgos y categorías.

La identificación de las fuentes de riesgo provee de una base para la examinación sistemática de situaciones cambiantes a lo largo del tiempo. Las fuentes de un riesgo pueden ser tanto internas como externas al mismo y a medida que el desarrollo avanza pueden surgir nuevas fuentes. El establecimiento de categorías de riesgos provee de un mecanismo para recolectar y organizar los riesgos como así también para asegurar una búsqueda apropiada y exhaustiva de aquellos riesgos que puedan tener mayores consecuencias sobre el logro de las metas del proyecto.

Productos típicos de la práctica:

- Lista de fuentes de riesgos.
- Lista de categorías de riesgos.

Subprácticas:

- *Determinar las fuentes de riesgos:*
  Las fuentes de riesgo son los productores de riesgos en los proyectos y organizaciones. Existen muchas fuentes de riesgos, tanto internas como externas al proyecto. Las fuentes de riesgos permiten identificar áreas comunes donde los riesgos suelen originarse. Algunas fuentes de riesgo típicas son:
  - Requerimientos inciertos.

- Estimaciones erróneas.
- Diseño inapropiado.
- Tecnología no disponible.
- Asignación irrealista de recursos.
- Conformación del equipo inapropiada.
- Capacidad incierta de los proveedores.

Muchas de estas fuentes de riesgos suelen ser aceptadas frecuentemente sin una planificación apropiada. La temprana identificación de las fuentes internas y externas deriva en una temprana identificación de riesgos lo cual permite implementar anticipadamente planes que lleven a evitar o disminuir la acción del riesgo.

- *Determinar categorías de riesgos:*
  Las categorías de riesgos constituyen los "receptáculos" para la recolección y organización de los riesgos. Una de las razones para identificar categorías de riesgos es ayudar en la futura consolidación de actividades en los planes de mitigación de riesgos.
  Cualquiera de los siguientes factores puede considerarse al momento de determinar las categorías de riesgos:
  - Las fases del modelo de ciclo de vida elegido para el proyecto: requerimientos, diseño, etc.
  - Los tipos de procesos empleados.
  - Los tipos de productos usados.
  - Los inconvenientes a nivel de programa: contrataciones, presupuesto, calendario, recursos, productividad, etc.

Pueden emplearse taxonomías de riesgos para proveer un marco para las actividades de determinación de fuentes y categorías de riesgos.

SG2	**Identificar y Analizar Riesgos**
	Los riesgos son identificados y analizados para determinar su importancia relativa.

El grado de importancia impacta en los recursos asignados para manejar un riesgo identificado. El análisis de riesgos se vincula a la identificación de riesgos tanto de fuentes internas como externas y a la evaluación de cada uno de los riesgos para determinar su probabilidad de ocurrencia e impacto posibles. La categorización de los riesgos en base a su evaluación respecto a las categorías ya definidas y a los criterios desarrollados por la estrategia de gestión de riesgo, provee de la información necesaria para manejar un riesgo. Los riesgos relacionados pueden ser agrupados con el propósito de lograr una administración más eficiente.

SP2.1-1	**Identificar riesgos**
	Identificar y documentar los riesgos.

La identificación de posibles inconvenientes y vulnerabilidades que pueden afectar negativamente los esfuerzos comprometidos para cumplir con el trabajo y los planes es la base sobre la cual descansa una gestión de riesgos exitosa. Los riesgos deben ser identificados y descriptos de una manera entendible antes de poder ser analizados para manejarlos apropiadamente.

La identificación de riesgos debe ser realizada mediante el empleo de un enfoque organizado con el propósito de determinar riesgos probables y realistas que impidan el logro de los objetivos. Para ser efectivo, el proceso de identificación de riesgos no debería intentar referenciar todo posible evento independientemente de la poca probabilidad que pueda estar asociada al mismo. La lista de los riesgos identificados debe revisarse periódicamente para reexaminar posibles fuentes y condiciones cambiantes relacionadas a fuentes aún no consideradas.

Existen muchos métodos para realizar la identificación de riesgos. Entre los métodos típicos se incluye el llevar a cabo una evaluación de riesgos usando taxonomías.

Productos típicos de la práctica:

- Lista de los riesgos identificados, incluyendo el contexto, condiciones y consecuencias de la ocurrencia del riesgo.

SP2.2-1	**Evaluar, categorizar y priorizar riesgos**
	Evaluar y categorizar cada uno de los riesgos identificados mediante el empleo de las categorías y parámetros ya definidos, determinando además su prioridad relativa.

La evaluación de riesgos se realiza con el propósito de asignar una importancia relativa a cada uno de los riesgos identificados, logrando de este modo la determinación de cuándo se requiere una atención por parte de los responsables de la administración. Frecuentemente, suele ser útil el agrupar riesgos en base a su interrelación y desarrollar alternativas a nivel de dichas agrupaciones.

Colectivamente, las actividades de evaluación, categorización y priorización suelen ser llamadas “análisis de riesgos”.

Productos típicos de la práctica:

- Lista de los riesgos, con una prioridad asignad a cada uno de ellos.

Subprácticas:

- *Evaluar los riesgos identificados usando los parámetros definidos.*

- *Categorizar y agrupar los riesgos de acuerdo a las categorías definidas: Los riesgos se categorizan en grupos definidos, proveyendo de este modo la posibilidad de observarlos según su fuente, taxonomía o componente de proyecto.*

- Priorizar los riesgos para su mitigación.

Capítulo

3

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Estudio de Viabilidad del Sistema

3. Estudio de Viabilidad del Sistema (EVS)

En el presente capítulo se describe y desarrolla el estudio de viabilidad propuesto por la metodología utilizada.

3.1. Introducción

Todo proyecto es realizable considerando la disponibilidad de recursos ilimitados y tiempos infinitos pero, desafortunadamente, todo tipo de desarrollo, en general, y de un sistema de software, en particular, se caracteriza por la escasez de recursos y la dificultad de cumplir con los plazos exigidos y, por lo tanto, es necesario y prudente evaluar la viabilidad de un proyecto tan pronto como sea posible.

Un estudio de viabilidad determinará si el proyecto en el cuál se espera trabajar es factible de realizarse en el contexto actual, considerando una serie de cuestiones tanto económicas como funcionales, técnicas, comerciales y legales entre otras.

En el presente contexto, el estudio de viabilidad permitirá determinar si la solución de un problema es viable con sistemas informáticos basados en conocimiento, o no.

3.2. El Estudio de Viabilidad en MÉTRICA Versión 3

El proceso de Desarrollo de Sistemas de Información de MÉTRICA Versión 3 [Métrica V3, 2000] contiene todas las actividades y tareas que se deben llevar

a cabo para desarrollar un sistema, cubriendo desde el análisis de requisitos hasta la instalación del software. Este proceso es, sin duda, el más importante de los identificados en el ciclo de vida de un sistema y se relaciona con todos los demás.

Uno de los procesos que conforman el de Desarrollo de Sistemas de Información es el **Estudio de Viabilidad del Sistema** (**EVS**), cuyo propósito es analizar un conjunto concreto de necesidades, con la idea de proponer una solución a corto plazo. Los criterios con los que se hace esta propuesta no serán estratégicos sino tácticos y relacionados con aspectos económicos, técnicos, legales y operativos.

Los resultados del Estudio de Viabilidad del Sistema constituirán la base para tomar la decisión de seguir adelante o abandonar. Si se decide seguir adelante pueden surgir uno o varios proyectos que afecten a uno o varios sistemas; los cuales se desarrollarán según el resultado obtenido en el estudio de viabilidad y teniendo en cuenta la cartera de proyectos para la estrategia de implantación del sistema global.

Se ha considerado que este proceso es obligatorio, aunque el nivel de profundidad con el que se lleve a cabo dependerá de cada caso.

Mientras que el Plan de Desarrollo de Sistemas tiene como objetivo proporcionar un marco estratégico que sirva de referencia para las aplicaciones de un ámbito concreto de una organización, el objetivo del Estudio de Viabilidad del Sistema es el análisis de un conjunto concreto de necesidades para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas. La solución obtenida como resultado del estudio puede ser la definición de uno o varios proyectos que afecten a uno o varios sistemas ya existentes o nuevos. Para ello, se identifican los requisitos que se ha de satisfacer y se estudia, si procede, la situación actual.

A partir del estado inicial, la situación actual y los requisitos planteados, se estudian las alternativas de solución. Dichas alternativas pueden incluir soluciones que impliquen desarrollos a medida, soluciones basadas en la adquisición de productos software del mercado o soluciones mixtas. Se describe cada una de las alternativas, indicando los requisitos que cubre. Una vez descritas cada una de las alternativas planteadas, se valora su impacto en la organización, la inversión a realizar en cada caso y los riesgos asociados. Esta información se analiza con el fin de evaluar las distintas alternativas y seleccionar la más adecuada, definiendo y estableciendo su planificación.

Si en la organización se ha realizado con anterioridad un Plan de Desarrollo de Sistemas que afecte al sistema objeto de este estudio, se dispondrá de un conjunto de productos que proporcionarán información a tener en cuenta en todo el proceso.

Las actividades que engloba este proceso se recogen en la siguiente figura (véase **Figura 11 - Tareas del EVS**), en la que se indican las tareas que

pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores:

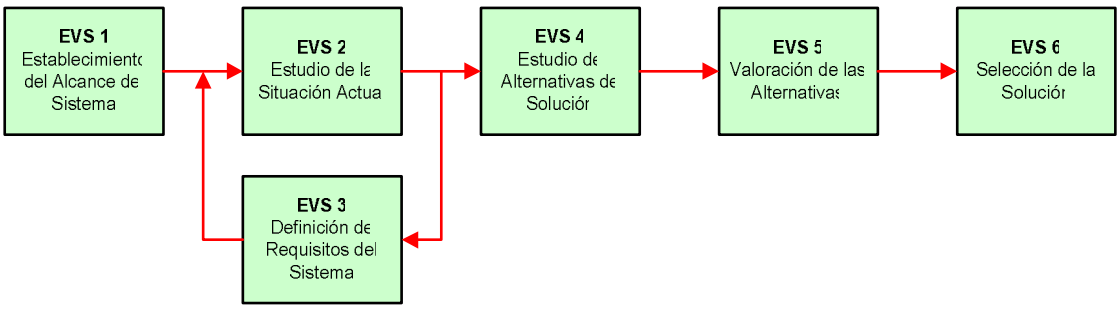


Figura 11 - Tareas del EVS

En lo que respecta a las interfaces, se llevarán a cabo las siguientes actividades en paralelo con el estudio de viabilidad:

- **Gestión de proyectos:** A realizar una vez finalizado el estudio de viabilidad:
  - *GPI1:* Estimación del esfuerzo
  - *GPI2:* Planificación
- **Gestión de la configuración:** A realizar durante el estudio de la viabilidad y una vez finalizado el mismo:
  - *EVS-GC1:* Definición de los requisitos de gestión de configuración.
  - *EVS-GC2:* Establecimiento del plan de gestión de la configuración.
- **Aseguramiento de la calidad:** A realizar durante el estudio de la viabilidad y una vez finalizado el mismo:
  - *EVS-CAL1:* Identificación de las propiedades de calidad para el sistema.
  - *EVS-CAL2:* Establecimiento del plan de aseguramiento de la calidad.
- **Seguridad:** Debido a las características del sistema (independiente y monousuario), y a las características del proyecto (tesis llevada a cabo por una única persona), las actividades de seguridad se limitan a lo mínimo indispensable.
  - *EVS-SEG5:* Evaluación detallada de la seguridad de la solución propuesta.

3.3. Establecimiento del Alcance del Sistema

La gestión de riesgos es un proceso que debería llevarse a cabo como parte de las actividades habituales de proyecto de toda organización dedicada a la generación y/o mantenimiento de software [Futrell, *et al.*, 2002]; esta afirmación se hace aún mas evidente en las compañías que han adoptado o que pretenden adoptar un modelo de calidad de software tal como CMM o CMMI.

La necesidad principal planteada por el grupo de usuarios constituye la de contar con una herramienta que posibilite la aplicación consistente de un proceso de identificación de riesgos basado en taxonomías estándares de la industria y que no requiera de una base de datos propietaria de forma tal de alentar su utilización en proyectos y organizaciones con diferentes culturas y niveles de madurez.

Además de cubrir este objetivo fundamental, se pretende que la herramienta cuente con una interfaz gráfica que facilite su utilización y con la posibilidad de generar un reporte de los resultados obtenidos luego de una sesión de identificación de riesgos para un proyecto.

Adicionalmente, se espera lograr cierta independencia de la aplicación respecto al entorno organizacional y autonomía de uso para quienes la emplean por lo cual se procura evitar la necesidad de conexiones con repositorios externos o remotos.

Considerando el alcance antes detallado, existen dos alternativas de solución, a saber:

1. Emplear una herramienta disponible en el mercado que cubra las necesidades de los usuarios.
2. Desarrollar una nueva herramienta en base a dichas necesidades.

Inicialmente, se ha planteado como restricción económica el intentar evitar la adquisición de software comercial de terceras partes, alentando así el uso de aplicaciones de tipo libre. Desde el punto de vista técnico, no se han planteado restricciones de ningún tipo, aún cuando se pretende emplear una herramienta que pueda operar en diferentes entornos convencionales sin requerir mayores adaptaciones y que no demande la adquisición de hardware o software especiales. Finalmente, y desde el punto de vista legal, se espera emplear taxonomías que sean libres y cuya utilización no implique el incurrir en problemas relacionados con derechos de propiedad intelectual.

Para el desarrollo del estudio de viabilidad, se realizará un análisis comparativo de las herramientas actualmente existentes en el mercado, para finalmente contrastarlo con la posibilidad de creación de un nuevo sistema. En función de los resultados obtenidos luego de dicho proceso, se seleccionará o no la opción de creación de una nueva aplicación para resolver el problema antes explicado.

En caso de que luego de completarse el estudio de viabilidad se seleccione la opción de desarrollo de una nueva herramienta, la única decisión a tomar estaría relacionada con el entorno de programación y el lenguaje a utilizar (tendiendo siempre a la obtención de una herramienta de escritorio desarrollada como un sistema informático convencional). Atendiendo a esta consideración, no se evaluará cuál es el lenguaje o el conjunto de herramientas más adecuado para el desarrollo sino que simplemente se seleccionará la opción mas ampliamente aceptada y difundida dentro de las alternativas libres que cubra las exigencias planteadas.

El alcance del sistema se limitará exclusivamente al soporte de las tareas de identificación de riesgos dentro del modelo de gestión de riesgos. Se ha seleccionado esta alternativa debido a que:

- Este constituye el más importante de los pasos del modelo [Futrell, *et al.*, 2002].
- Las taxonomías brindan la posibilidad de lograr una apropiada estandarización durante esta etapa del proceso.
- Las restantes fases del modelo están fuertemente caracterizadas e influenciadas por las políticas, enfoques, costumbres y realidades de las diferentes organizaciones debido a que son dichos componentes los determinantes de: las posibles acciones a tomar, el modo de implementarlas, la manera de controlar su cumplimiento y la forma de aplicar las posibles mejoras sugeridas a las mismas.

Finalmente, los objetivos principales planteados para el estudio de viabilidad son los siguientes:

- Analizar la situación actual
- Describir las posibles mejoras
- Analizar las alternativas de solución
- Establecer los requerimientos del nuevo sistema
- Servir de base para planificar el desarrollo del nuevo sistema

3.4. Estudio de la Situación Actual

Es posible identificar una tendencia creciente hacia el logro de la estandarización de procesos en las organizaciones; el empleo de taxonomías alienta dicho objetivo durante las actividades de identificación de riesgos debido a que posibilitan organizar la diversidad de conocimientos en forma de clasificaciones que pueden emplearse de manera consistente a lo largo de todos los proyectos de una organización.

Existen una gran cantidad de herramientas de software provistas en el mercado y relacionadas con las actividades de gestión de riesgos, específicamente, algunas de ellas orientan su funcionalidad al soporte de la identificación de riesgos y, parcialmente, algunas pocas lo hacen empleando taxonomías (véase **Tabla 2 - Herramientas para la Identificación de Riesgos**).

El siguiente cuadro resume las características principales de las herramientas de software que se encuentran disponibles actualmente y que están relacionadas con la identificación de riesgos en proyectos:

PRODUCTO	PROVEEDOR	DESCRIPCIÓN	PLATAFORMA
Active Risk Manager (ARM)	Strategic Thought	Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación de Riesgos mediante la	Web Based

PRODUCTO	PROVEEDOR	DESCRIPCIÓN	PLATAFORMA
		utilización de la información contenida en la WBS de proyecto.	
Technical Risk Identification and Mitigation System (TRIMS)	Navy's Best Manufacturing Practices	Herramienta integrada de Gestión de Riesgos que emplea ingeniería de conocimientos y que se enfocada en la identificación, medición y seguimiento de riesgos técnicos de proyectos.	Win32
RiskTrak	Risk Services & Technology	Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación de Riesgos mediante el empleo de base de datos.	Web Based
WelcomRisk	Welcom	Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación Sistemática de Riesgos mediante la utilización de bibliotecas configurables de categorías de riesgos.	Web Based

Tabla 2 - Herramientas para la Identificación de Riesgos

3.5. Definición de Requisitos del Sistema

El estándar de Especificación de Requerimientos del Software de la IEEE [IEEE 830,1998] guía la presente definición inicial de requisitos del sistema.

3.5.1. Descripción General

El producto a desarrollar como parte del presente trabajo final de Tesis constituye el primer prototipo de una herramienta que brinde la posibilidad de efectuar tareas de Identificación de Riesgos en base a Taxonomías estándar en organizaciones de software de cualquier tipo.

Perspectiva del Producto

Se espera la ejecución del sistema de forma “standalone” o independiente, aún cuando será necesario cierto soporte de software para lograr la persistencia de los datos y de hardware para alcanzar los objetivos computacionales y de impresión de reportes.

Funciones del Producto

La siguiente lista enumera las principales funciones requeridas para la herramienta:

a) Asistencia en la determinación de los riesgos que pueden presentarse en un proyecto de software.
b) Generación de un reporte final de los riesgos identificados para un proyecto.
c) Almacenamiento y recuperación de los resultados correspondientes a diferentes sesiones de identificación.
d) Generación de un reporte conteniendo información respecto a la taxonomía empleada durante el proceso de identificación.

Características de los Usuarios

El sistema a desarrollar esta orientado a ser utilizado principalmente por administradores de proyectos de software; por este motivo, se espera de los mismos un elevado nivel de conocimiento tanto del uso de los recursos tecnológicos y computacionales a emplear como del área de aplicación del sistema, esto deriva en que no se detecten requerimientos especiales ligados a ellos.

Asunciones y Dependencias

Se detallan a continuación las asunciones y dependencias técnicas consideradas para el desarrollo de la aplicación:

Asunciones:

- Algunos de los componentes a utilizar para la construcción del prototipo de la herramienta ya se encuentran desarrollados y funcionan correctamente por lo cual no serán probados.
- No existen incompatibilidades entre los componentes preexistentes y el entorno de software y hardware que será requerido para el desarrollo y la utilización de la herramienta.
- No existen incompatibilidades entre los componentes preexistentes entre sí.
- Soporte por parte de los proveedores de los componentes preexistentes seleccionados en caso de detectarse errores durante su incoporación o utilización.

Dependencias:

- Definición de la taxonomía base para la aplicación por parte de los expertos del dominio.
- Definición de las reglas de identificación de riesgos para el sistema por parte de los expertos del dominio.
- Documentación, manuales y ejemplos de uso de los componetes preexistentes seleccionados por parte de sus desarrolladores y proveedores.

3.5.2. Requerimientos de Ingeniería

Requerimientos de Interface Externa

Interfaces de Usuarios

El sistema a desarrollar deberá proveer de una interfaz gráfica tanto para posibilitar al usuario realizar las operaciones de uso del sistema como para consultar, almacenar y recuperar los resultados obtenidos en diferentes sesiones de identificación de riesgos.

Menús de opciones, ventanas, “tooltips” y diálogos son componentes deseables en la interfaz de usuario a definir para la aplicación.

Interafaces de Hardware

No existen requerimientos para las interfaces de hardware; todo el acceso al hardware será realizado mediante la invocación de las interfaces de software definidas por los componentes de software preexistentes que se hayan seleccionado.

Interfaces de Software

No existen requerimientos predefinidos para las interfaces de sofware; el sistema deberá cumplir con las interfaces definidas por los componentes de software de soporte seleccionados con el propósito de invocar sus servicios.

Interfaces de Comunicación

No existen requerimientos para las interfaces de comunicación.

Requerimientos Funcionales

Se amplian a continuación las funciones descriptas en la sección **3.5.1**.**Descripción General** (mayores detalles respecto a los requisitos del sistema pueden encontrase en la sección **4.7**.**Definición del Sistema y Establecimiento de Requisitos**):

a) Asistencia en la determinación de los riesgos que pueden presentarse en un proyecto de software.

   Tomando como entradas una taxonomía base, un conjunto de reglas de relaciones entre riesgos y las respuestas del usuario a una serie de preguntas que el sistema deberá realizar con el propósito de cubrir todos los riesgos de la taxonomía; la herramienta deberá generar como resultado un grupo de los riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio y su posible grado de exposición.

b) Generación de un reporte final de los riesgos identificados para un proyecto.

Partiendo de las entradas provistas y de los resultados obtenidos de una sesión de identificación de riesgos y empleando tanto recursos visuales como de impresión; la herramienta deberá mostrar al usuario toda la información pertinente de la misma, esto es:

- Proyecto Analizado.
- Responsable del Proyecto.
- Organización del Proyecto.
- Categoría de los Riesgos Potenciales.
- Descripción de los Riesgos Potenciales.
- Nivel de Exposición de los Riesgos Potenciales.

c) Almacenamiento y recuperación de los resultados correspondientes a diferentes sesiones de identificación.

Partiendo de las entradas provistas y de los resultados obtenidos de una sesión de identificación de riesgos y empleando el medio de soporte que se determine como mas apropiado en futuras etapas del desarrollo; el sistema deberá permitir almacenar y recuperar la información pertinente de una sesión de identificación de riesgos, esto es:

- Proyecto Analizado.
- Responsable del Proyecto.
- Organización del Proyecto.
- Categoría de los Riesgos Potenciales.
- Descripción de los Riesgos Potenciales.
- Nivel de Exposición de los Riesgos Potenciales.

d) Generación de un reporte conteniendo información respecto a la taxonomía empleada durante el proceso de identificación.

Partiendo de la taxonomía base empleada por el sistema en el proceso de identificación de riesgos y empleando tanto recursos visuales como de impresión; la herramienta deberá mostrar al usuario toda la información pertinente de la misma, esto es:

- Categorías de los Riesgos.
- Descripciones de los Riesgos.
- Posibles Nivel de Exposición de los Riesgos.

Requerimientos de Rendimiento

No existen requerimientos relacionados con el rendimiento a esperar de la aplicación.

Requerimientos de Tamaño

No existen requerimientos relacionados con el tamaño a esperar para la aplicación.

Dicha afirmación es válida y se hace extensiva a dos aspectos diferentes:
- Tamaño estático del producto y su instalador en disco.
- Tamaño dinámico de la aplicación y sus componentes en memoria.

Restricciones de Diseño

No existen restricciones de diseño previamente establecidas para el sistema, aún cuando se considera deseable tanto el uso de componentes preexistentes como de patrones de diseño.

Requerimientos de Seguridad

No existen requerimientos de seguridad de ningún tipo definidos para la herramienta.

Dicha afirmación es válida y se hace extensiva a dos aspectos diferentes:
- Seguridad de acceso de los usuarios a las funciones de la herramienta o a los datos generados para y por el uso de la misma.
- Seguridad de las personas por el uso de la aplicación.

Requerimientos de Adaptación

El sistema debe brindar soporte de internacionalización, es decir, que el sistema deberá ser fácilmente portable a diferentes idiomas.

Requerimientos de Portabilidad

El sistema debe brindar soporte multiplataforma, es decir, que el sistema deberá tener la capacidad de ejecutarse en diferentes equipos con distintos sistemas operativos.

Requerimientos de Mantenimiento

Las actividades de mantenimiento se encuentran fuera del alcance del desarrollo a realizar.

3.5.3. Preparación para la Distribución

Se deberá generar un instalador para la aplicación, el mismo debe cumplir con los mismos requerimientos definidos para el sistema tanto en lo que refiere a la interfaz de usuario como a los aspectos de adaptación y portabilidad.

3.6. Estudio de Alternativas de Solución

3.6.1. Active Risk Manager (ARM)

Descripción: Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación de Riesgos mediante la utilización de la información contenida en la WBS de proyecto.
Proveedor: Strategic Thought.
Plataforma: Web Based (requiere instalación de servidor).
Sistemas Operativos: Windows 98, ME, NT, 2000, y XP.
Idioma: Inglés.
Licenciamiento: Comercial
Características Destacables:

- erenciamiento integrado de riesgos basado en estándares predefinidos.
- Alto grado de integración con herramientas estándares del mercado (VISIO, MS Office, Primavera, etc).
- Generación de reportes.

Se muestra a continuación una pantalla correspondiente a la herramienta analizada (véase **Figura 12 - Pantalla Principal de Gestión de Riesgos en ARM**):

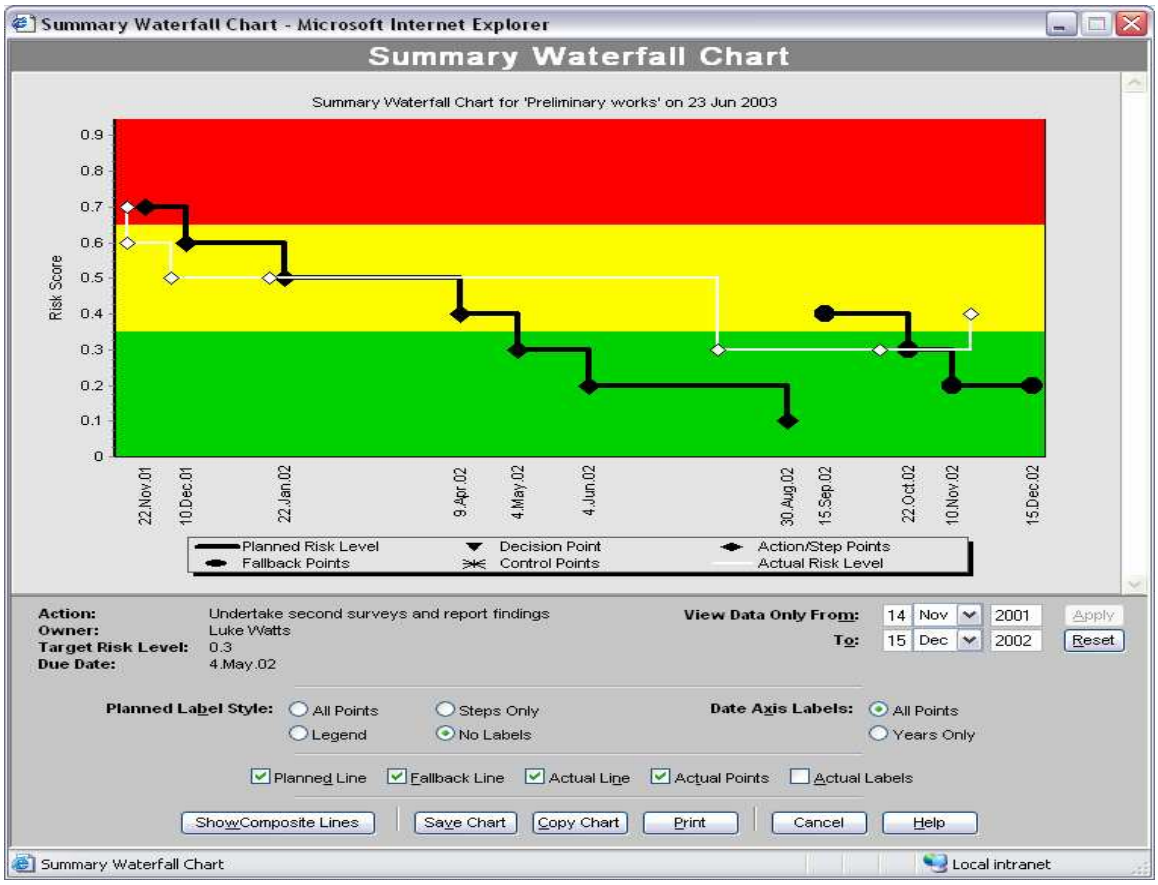


Figura 12 - Pantalla Principal de Gestión de Riesgos en ARM

3.6.2. Technical Risk Identification and Mitigation System (TRIMS)

Descripción: Herramienta integrada de Gestión de Riesgos que emplea ingeniería de conocimientos y que se enfocada en la identificación, medición y seguimiento de riesgos técnicos de proyectos.
Proveedor: Navy's Best Manufacturing Practices.
Plataforma: Win32.
Sistemas Operativos: Windows 98, ME, NT, 2000, y XP.
Idioma: Inglés.
Licenciamiento: Libre.
Características Destacables:

- Gerenciamiento integrado de riesgos.
- Generación de reportes.
- Orientada a categorías de riesgos predefinidas para sectores específicos.

Se muestran a continuación algunas pantallas correspondientes a la herramienta analizada (véase **Figura 13 - Pantalla Principal de TRIMS**, **Figura 14 - Carga de Riesgo en TRIMS** y **Figura 15 - Reporte en TRIMS**):

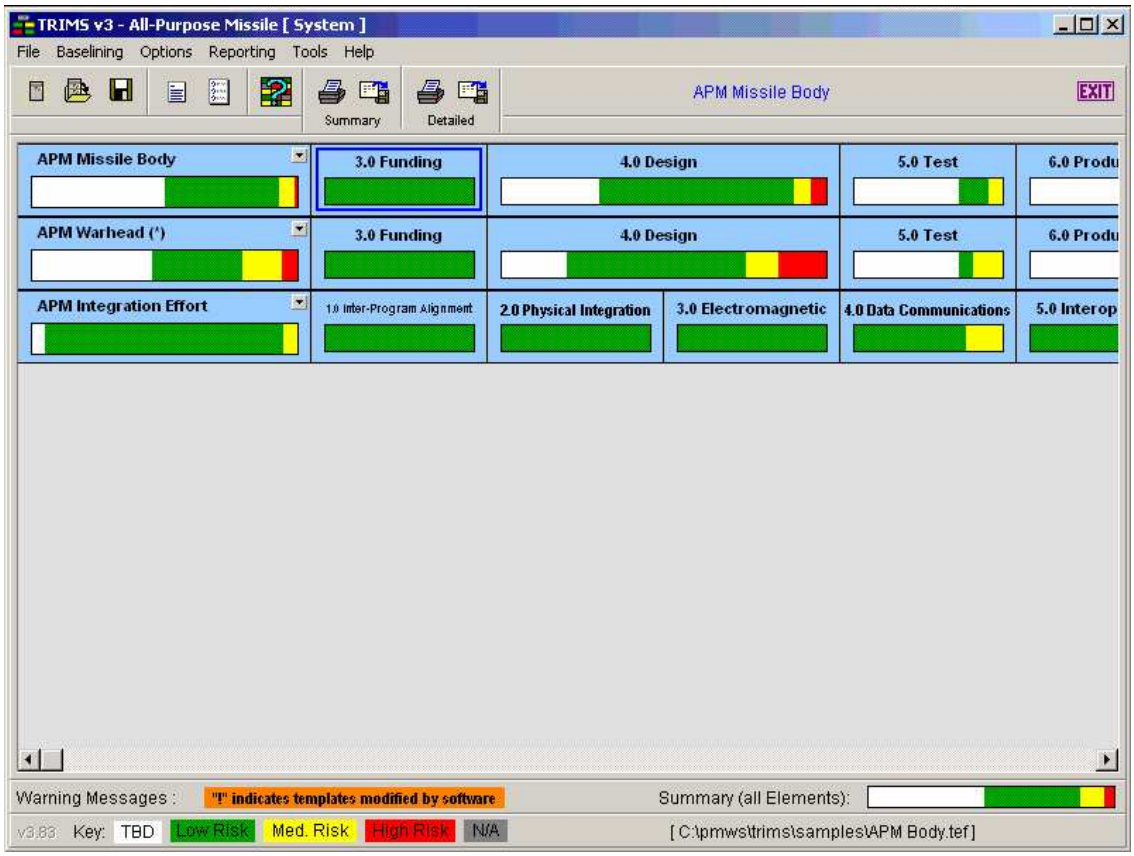


Figura 13 - Pantalla Principal de TRIMS

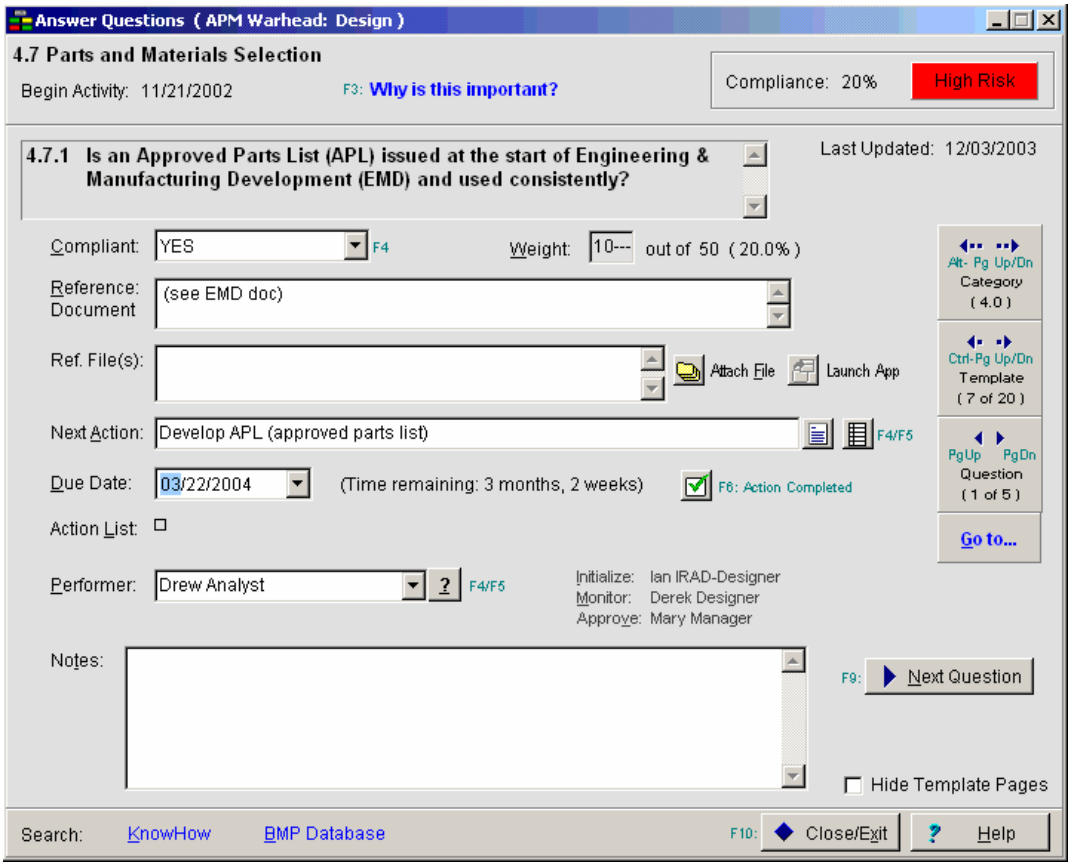


Figura 14 - Carga de Riesgo en TRIMS

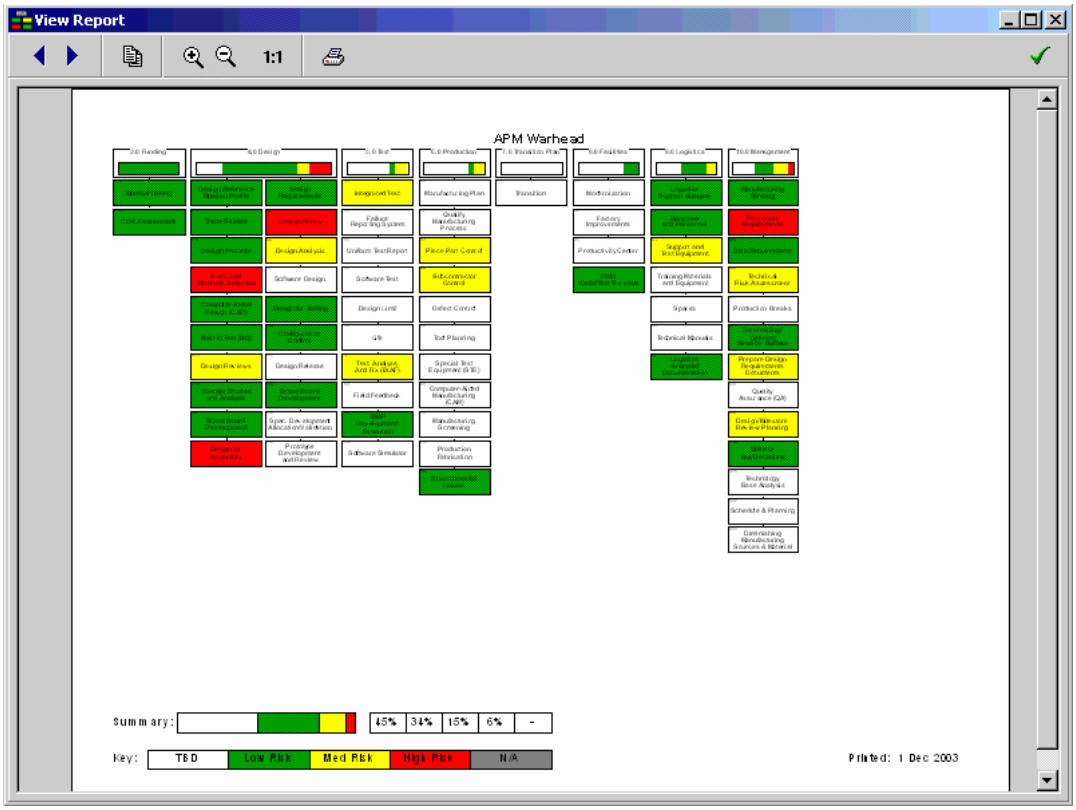


Figura 15 - Reporte en TRIMS

3.6.3. RiskTrak

Descripción: Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación de Riesgos mediante el empleo de base de datos.
Proveedor: Risk Services & Technology.
Plataforma: Web Based (requiere instalación de servidor).
Sistemas Operativos: Windows 98, ME, NT, 2000, y XP.
Idioma: Inglés.
Licenciamiento: Comercial
Características Destacables:

- Generación de reportes.
- Administración centralizada del gerenciamiento de riesgo a nivel organizacional.

Se muestran a continuación algunas pantallas correspondientes a la herramienta analizada (véase **Figura 16 - Categorías de Riesgos en RiskTrack** y **Figura 17 - Análisis de Riesgos en RiskTrack**):

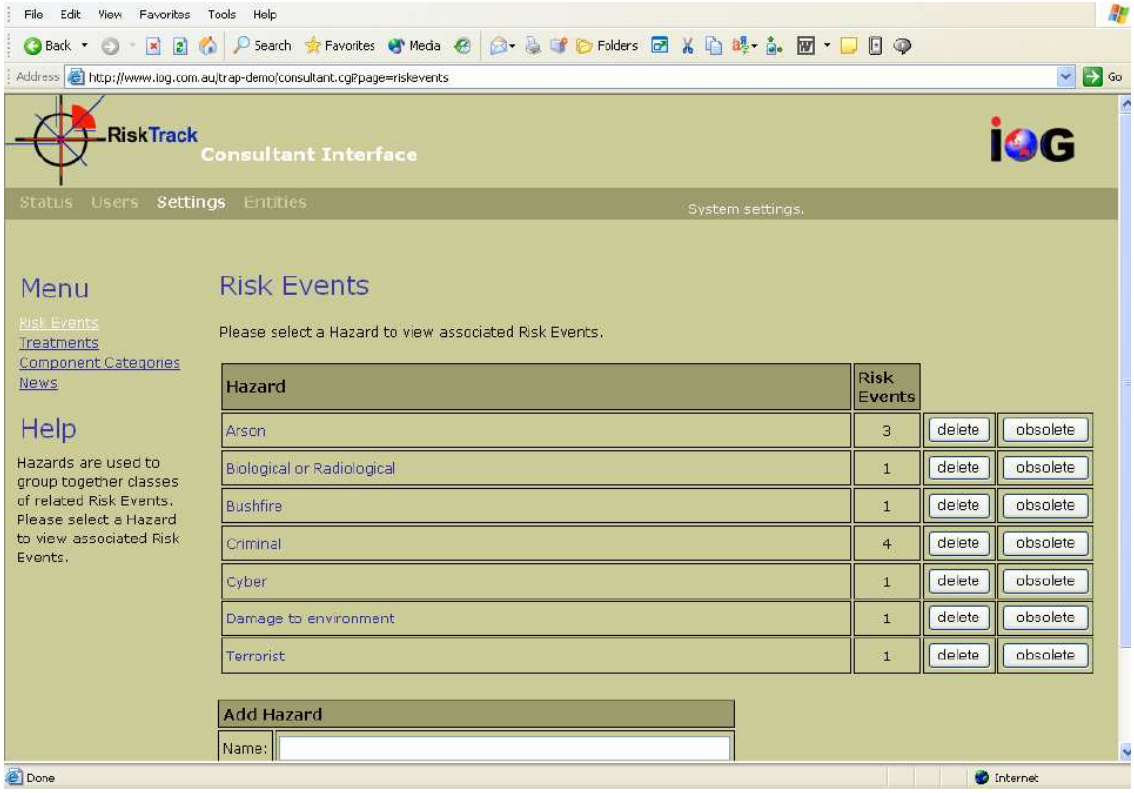


Figura 16 - Categorías de Riesgos en RiskTrack

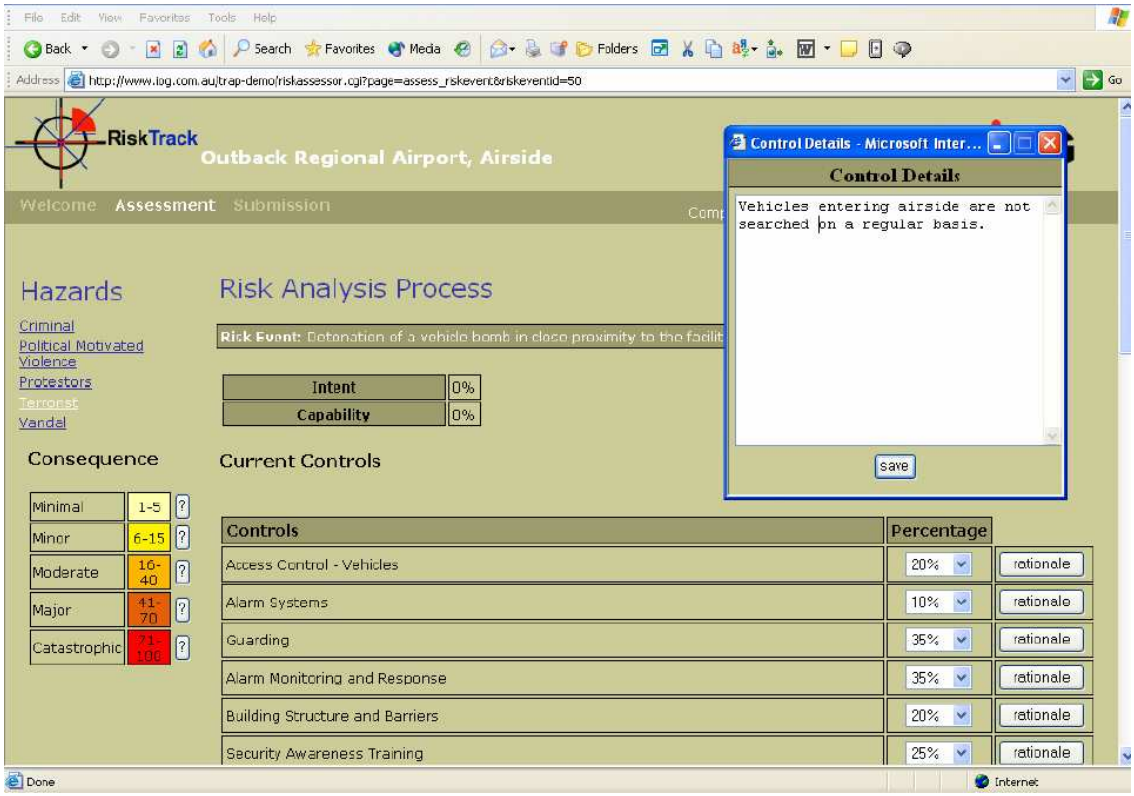


Figura 17 - Análisis de Riesgos en RiskTrack

3.6.4. WelcomRisk

Descripción: Herramienta integrada de Gestión de Riesgos que brinda una solución para la Identificación Sistemática de Riesgos mediante la utilización de bibliotecas configurables de categorías de riesgos.
Proveedor: Welcom
Plataforma: Web Based (requiere instalación de servidor).
Sistemas Operativos: Windows 98, ME, NT, 2000, y XP.
Idioma: Inglés.
Licenciamiento: Comercial
Características Destacables:

- Generación de reportes.
- Integración con herramientas estándares del mercado (MS Project, Primavera, etc).
- Alto grado de seguridad tanto a nivel de sistema como de aplicación

Se muestran a continuación algunas pantallas correspondientes a la herramienta analizada (véase **Figura 18 - Pantalla Principal de WelcomRisk**, **Figura 19 - Resumen de Riesgos** y **Figura 20 - Reporte en WelcomRisk**):

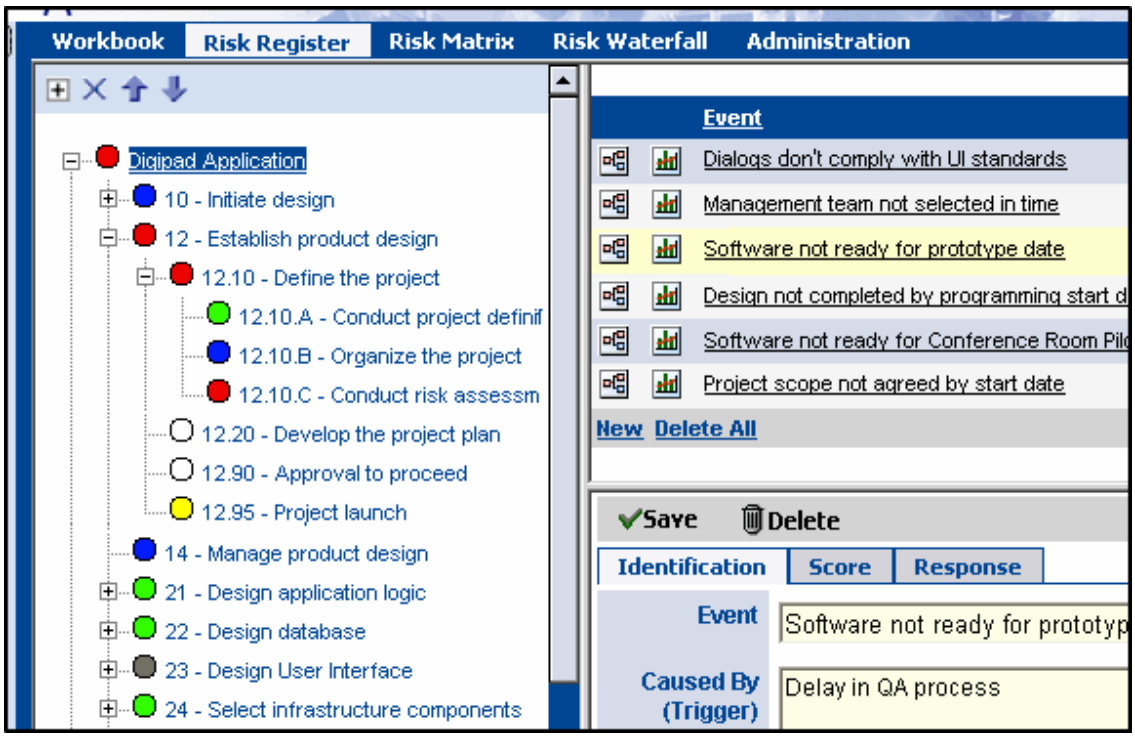


Figura 18 - Pantalla Principal de WelcomRisk

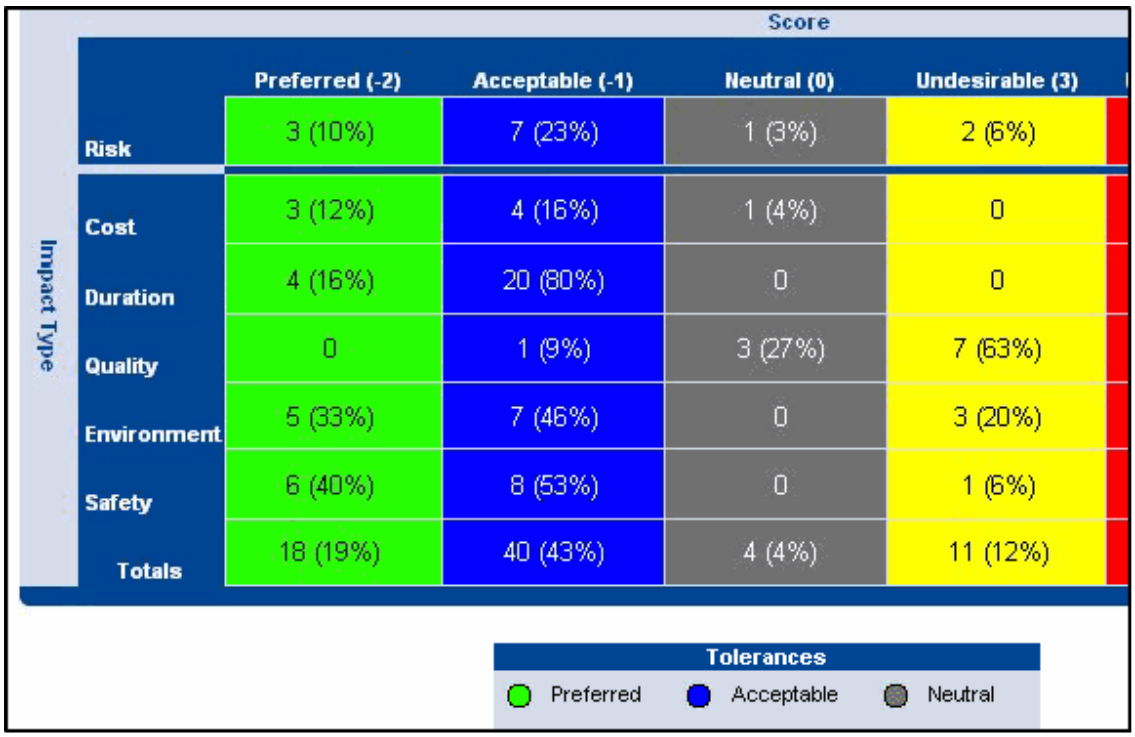


Figura 19 - Resumen de Riesgos

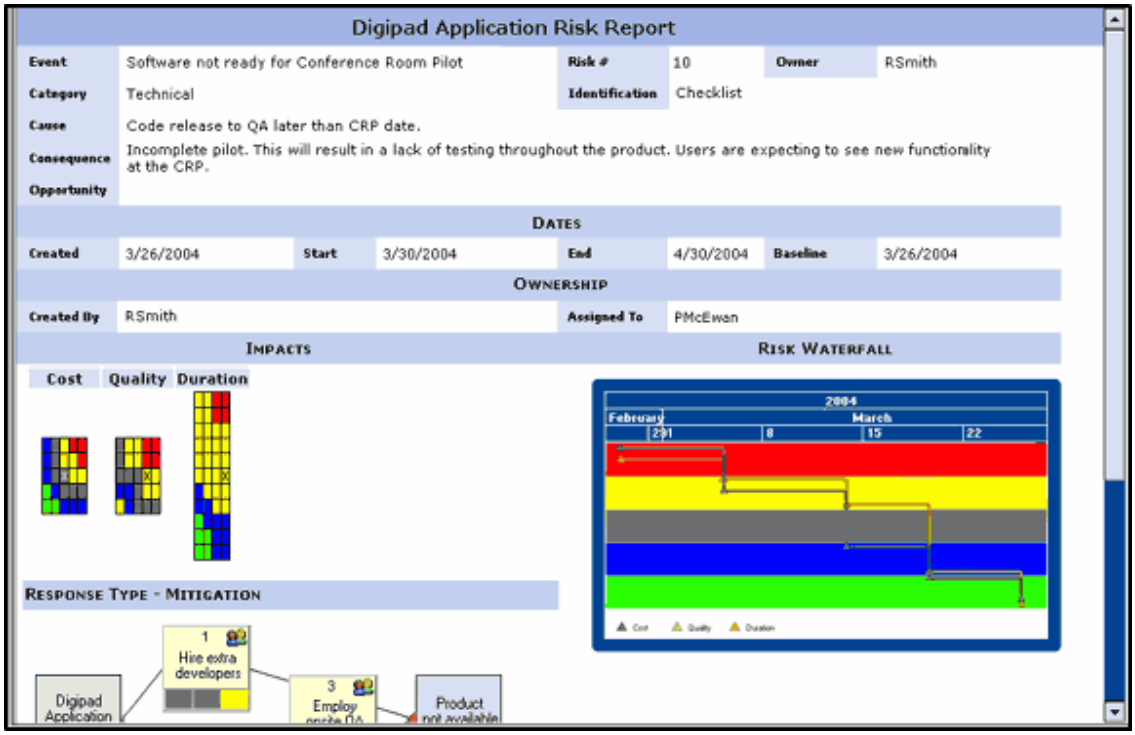


Figura 20 - Reporte en WelcomRisk

Finalmente, se realiza a continuación un estudio detallado respecto a la posibilidad de llevar a cabo el desarrollo de una nueva herramienta para cubrir las necesidades planteadas por los usuarios.

3.6.5. Estudio de Viabilidad para el Proyecto de Desarrollo de Tesis de Magíster

Introducción

No resulta fácil describir las características que hacen que un problema sea adecuado para resolverlo creando un sistema informático. La ingeniería de sistemas debe considerarse para la resolución de un problema particular únicamente si el desarrollo de un sistema informático es posible, está justificado, es apropiado, y su construcción va a tener éxito. Las tres primeras premisas son directas mientras que la última, o de éxito, es indirecta o colateral, pero no por ello menos importante.

Para realizar el estudio de viabilidad, las características a tener en cuenta antes de acometer el desarrollo de un sistema informático han sido divididas en cuatro grandes tipos [Pazos, 2003]:

- Aquellas que hacen que sea posible construir un sistema informático para automatizar una tarea.
- Las que permiten justificar el desarrollo de un sistema informático.
- Aquellas que aconsejan que un sistema informático es la solución más adecuada.
- Y, finalmente, las que aseguran el éxito del sistema.

El Estudio de Viabilidad

Se detallan a continuación una serie de aspectos que han sido considerados y analizados para determinar la viabilidad de la construcción de un nuevo sistema para satisfacer las necesidades planteada por los usuarios.

En primer lugar, la identificación de riesgos en base a taxonomías es una técnica desarrollada en 1993 y las actividades de administración de riesgos cuentan con estándares desarrollados en la década de los años 70, por lo cual se espera que la tarea a modelar permanezca muy estable por un considerable período de tiempo lo cual alienta a la construcción de sistemas tendientes a modelarla.

Un importante aliciente para la construcción de un nuevo sistema lo constituye el hecho de que sus objetivos están claramente establecidos y pueden evaluarse tanto al final del desarrollo como durante el proceso (aprovechando las ventajas y características del enfoque de prototipado incremental que ha sido propuesto). Aún cuando ya se ha desarrollado una tarea previa de conciliación de objetivos con directivos y usuarios, los detalles mas específicos no han sido definidos y la funcionalidad será definida con mayor precisión a lo largo del desarrollo del sistema.

Cabe destacar que la planificación inicial para el proyecto fue aceptada y consensuada con todos los involucrados y que el proyecto no forma parte de un camino crítico con otros sistemas o de otros sistemas lo cual es importante considerando las limitaciones de tiempo con las que cuenta el desarrollador del presente trabajo de Tesis de Magíster.

Adicionalmente, la dirección apoya firmemente el proyecto de construcción de un nuevo sistema por el hecho de que los resultados del mismo brindarán soporte y producirán mejores resultados en los proyectos en los cuales sean aplicados. Asimismo, el proyecto posibilitará iniciar las actividades tendientes a la implementación del PA de Gestión de Riesgos del estándar de calidad CMMI. Por último, los usuarios también están interesados en el proyecto porque el sistema introduciría una estructura ordenada que facilitará las tareas de identificación de riesgos.

Entre otros aspectos que alientan a la construcción de un nuevo sistema y que permiten determinar la factibilidad del mismo pueden destacarse los siguientes:

- No se identifican impedimentos éticos o legales que impidan el desarrollo del sistema.
- La tarea no requiere la realización de investigación básica.
- El sistema no trabajará en tiempo real con otros programas y/o dispositivos físicos.
- No se estima que la continuidad del proyecto esté influenciada por cambios políticos tanto a nivel empresarial como académico.
- Se cuenta con recursos de hardware y software adecuados para el desarrollo del trabajo como así también con material bibliográfico, con el

apoyo y asesoramiento de los tutores y la disponibilidad de expertos administradores de proyectos a quienes consultar.
- Se cuenta con el respaldo de Motorola Argentina SA para la ejecución del proyecto, tanto mediante la provisión de elementos como de colaboradores en determinadas etapas del desarrollo.
- Existen antecedentes previos tanto a nivel industrial como organizacional de resolución exitosa de problemas de similares características de tamaño y tecnología mediante el empleo de una metodología de trabajo similar a MÉTRICA Versión 3 [Métrica V3, 2000].
- Tanto el conjunto de administradores de proyecto a consultar como la organización para la cual se desarrollará el software cuentan con suficientes casos de prueba basados en la historia personal y organizacional (que se encuentra organizada y estructurada en forma de base de datos conteniendo toda la información relacionada a los proyectos desarrollados a través de varios años de desempeño).

En sentido similar, las características propias de la tarea a modelar estimulan la posibilidad de construcción de un sistema ya que la misma se encuentra estructurada y definida a través de una serie de acciones (véase **Figura 21 - Actividades de la Tarea de Identificación de Riesgos**), a saber:

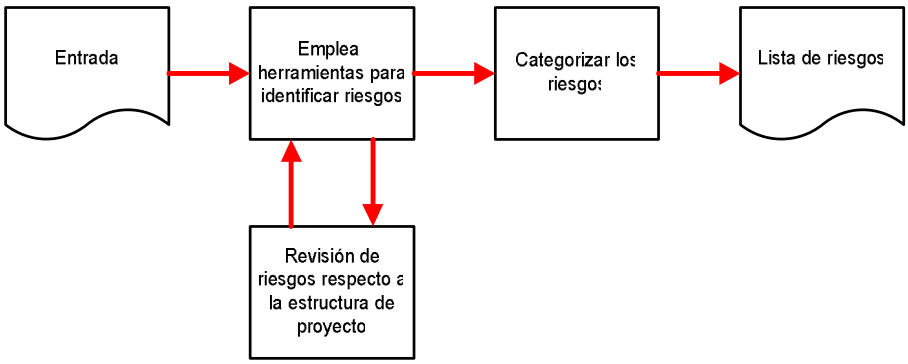


Figura 21 - Actividades de la Tarea de Identificación de Riesgos

Como puede observarse, el proceso a seguir implica el analizar, considerar y evaluar cada situación de proyecto en particular; cada proyecto es diferente a los demás y los resultados obtenidos serán consecuencia de un análisis diferente. Debe considerarse también que el empleo de taxonomías garantiza, necesariamente, un ordenamiento, entendimiento y organización de las herramientas y técnicas a utilizar pero que la gran cantidad de posibles riesgos plantean al mismo tiempo un muy amplio espacio de búsqueda que debe acotarse al momento de realizar el análisis mediante el empleo de heurísticas.

Es importante mencionar que, aún cuando a nivel de mercado no es frecuente el contar con personas con experiencia en las tareas de identificación de riesgos en base a una metodología estructurada, en la organización elegida y en los ámbitos académicos a consultar, la práctica se ha estado desarrollando de forma continua y reiterada aunque no se distinguen fácilmente personas que hayan trabajado en reiteradas ocasiones como administradores de

proyecto y que puedan estructurar con precisión las características a considerar durante el proceso a modelar.

Es posible afirmar también que no es necesario considerar varios niveles de abstracción ya que los conocimientos requeridos no presenten una considerable complejidad y emplean una serie de técnicas que pueden ser comprensibles, incluso, para usuarios no técnicos. Asimismo, el problema puede ser claramente descompuesto en diferentes subproblemas, por ejemplo, considerando diferentes clases o elementos de una taxonomía. Igualmente, la tarea acepta la técnica del prototipado gradual debido a que se pueden determinar diferentes subproblemas y generar un prototipo de herramienta en base a algún grupo seleccionado; mas aún, el alcance inicialmente acordado es muy apto para la adopción de un proceso de prototipación incremental en el cual se realicen incorporaciones de diferentes características (interfaz de usuario, resolución de situaciones triviales, resolución de alternativas, etc.) en distintas etapas del desarrollo.

A nivel organizacional, la solución del problema planteado es absolutamente prioritaria debido a que algunas deficiencias durante la identificación de riesgos han sido catalogadas como claves para problemas posteriores que han surgido en proyectos (estos resultados han sido obtenidos durante las sesiones realizadas para detectar causas raíces de problemas como parte de las actividades de Prevención de Defectos de CMM que Motorola Argentina SA lleva a cabo regularmente). Adicionalmente, la proximidad de la adopción de CMMI como estándar de calidad de desarrollo posibilita prever la necesidad del sistema.

A fines de facilitar la implementación de la nueva herramienta y disminuir la posibilidad de posibles inconvenientes, está previsto el realizar una capacitación, la elaboración futura de los manuales para el uso del sistema, e inicialmente un período de seguimiento en los primeros casos de uso de la herramienta en la organización. Así, se prevé que la introducción del sistema en las actividades organizacionales no depare efectos no deseados. Mas aún, debido a su función de asistencia se espera una buena recepción hacia el mismo, aunque siempre es preciso considerar la posibilidad de la existencia de algún tipo de resistencia y la necesidad de adaptaciones menores. Los administradores de proyecto han comprometido su participación en el proyecto y esperan con un alto grado de entusiasmo los resultados debido a que consideran que esto puede significar una mejora en el desarrollo de sus tareas y una disminución en sus esfuerzos actuales.

Debe destacarse que se espera que el sistema final pueda ser empleado tanto por la comunidad informática a nivel empresarial como académico por lo cual son de suma importancia consideraciones relacionadas al licenciamiento de la herramienta a utilizar. El sistema podrá ser utilizado por todos los integrantes de un equipo de desarrollo de software, aún cuando el usuario principal sea el administrador del proyecto. Adicionalmente, en un ambiente académico, podría ser empleado en cátedras relacionadas con las tareas de gestión de proyectos tanto por docentes como por alumnos.

Finalmente, la recuperación de la inversión estará asociada al logro de una mayor productividad y efectividad, dado que realizando una correcta y completa identificación de riesgos, se contribuye al logro de una acabada administración de proyectos y por lo tanto a la consecución de un mas completo plan de desarrollo y a un ajuste mas adecuado tanto a los plazos de entrega como a los esfuerzos y alcances previstos, derivando así en una mayor satisfacción de los clientes.

A futuro, puede considerarse como una posible evolución del sistema el incluir el aprendizaje basado en la retroalimentación de los usuarios respecto a la correctitud de los resultados alcanzados y los fundamentos de la conclusión alcanzada.

Resumen

Luego de finalizado el estudio de viabilidad y al considerar los resultados obtenidos en el mismo, las necesidades de la organización y las alternativas existentes, puede afirmarse que el desarrollo del sistema se presenta como ampliamente practicable ya que ha sido demostrado que el mismo será posible, es ampliamente justificable, se muestra como adecuado y se le pronostica una alta probabilidad de éxito.

3.7. Valoración de las Alternativas

El siguiente cuadro permite realizar una comparación entre las principales características de las herramientas analizadas y el asistente propuesto como producto del proyecto de Tesis de Magíster:

	ARM	TRIMS	RISKTRAK	WELCOMRISK	**ASISTENTE**
Taxonomías estándar	No	Si	No	No	Si
Riesgos de todo tipo	Si	No	Si	Si	Si
Multiplataforma	Si	No	No	No	Si
Internacionalización	No	No	No	No	Si
Modelo completo	Si	Si	Si	Si	No
Interfaz gráfica	Si	Si	Si	Si	Si
Reportes	Si	Si	Si	Si	Si
Libre	No	Si	No	No	Si

Tabla 3 - Comparación de Herramientas de soporte de Identificación de Riesgos

Como puede observarse, actualmente, no existen herramientas que brinden un soporte adecuado y estandarizado a los administradores de proyectos específicamente durante la fase de identificación de riesgos en organizaciones que comienzan con la implementación formal de las tareas de gestión de riesgos; las herramientas existentes o bien se enfocan solo en una categoría de riesgos [TRIMS, 2004], o bien están orientadas a compañías que poseen

una amplia base de datos organizacional que les permite generar información de categorías propias de riesgos [RiskTrak, 2004; WelcomRisk, 2004], o bien emplean un mecanismo que no se orienta al uso de taxonomías [Active Risk Manager, 2004] (véase **Tabla 3 - Comparación de Herramientas de soporte de Identificación de Riesgos**).

3.8. Selección de la Solución

Finalmente, luego del análisis realizado, es posible determinar que existe una región no soportada completamente por las herramientas existentes; la misma esta caracterizada por las necesidades de colaboración que poseen los administradores de proyectos respecto a la posibilidad de realizar una identificación de riesgos asistida y basada en una gran cantidad de categorías estándar de riesgos (véase **Tabla 3 - Comparación de Herramientas de soporte de Identificación de Riesgos**).

En base a lo antes expuesto, se considera como única alternativa válida de solución al problema planteado (y considerando las restricciones impuestas) el desarrollo de un sistema a medida.

3.8.1. El Sistema a Desarrollar

Se pretende desarrollar una herramienta que brinde la posibilidad de efectuar tareas de identificación de riesgos en base a taxonomías estándar en organizaciones de software de cualquier tipo.

El siguiente gráfico muestra la arquitectura de alto nivel para el sistema a desarrollar como parte del trabajo final de Tesis de Magíster:

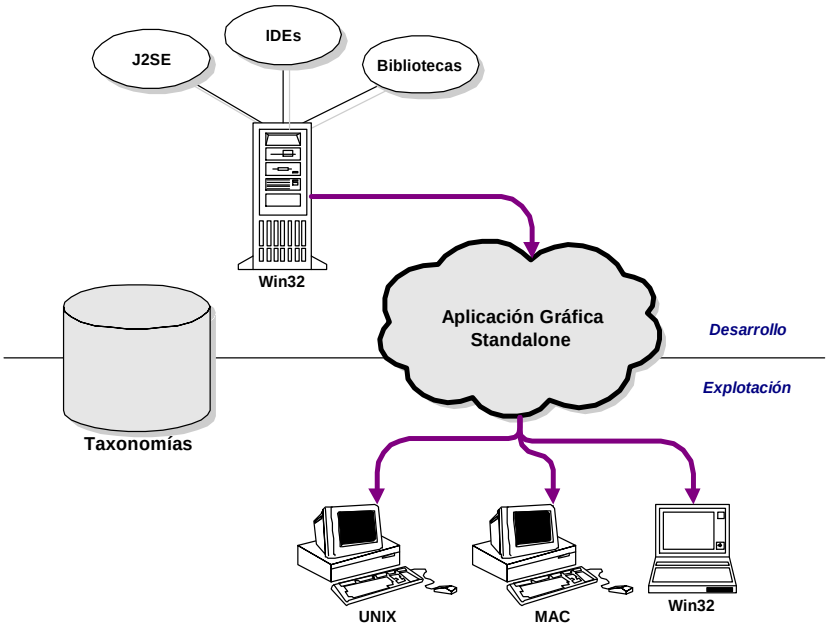


Figura 22 - Arquitectura de Alto Nivel

La herramienta a desarrollar constituirá una aplicación del tipo “standalone” o independiente, ejecutable en cualquier sistema operativo con soporte para Java, con una interfaz gráfica de navegación orientada a facilitar su empleo a los usuarios finales y con conexión a un repositorio de datos donde serán almacenadas las Taxonomías a consultar y el resto la información que permitirá el correcto funcionamiento del sistema (véase **Figura 22 - Arquitectura de Alto Nivel**).

Para la determinación de los riesgos, el asistente preguntará al usuario la información necesaria y producirá como resultado final un reporte conteniendo los riesgos identificados para un determinado proyecto.

En los siguientes capítulos del presente trabajo de Tesis de Magíster se documentan los diferentes modelos generados durante la construcción de la herramienta propuesta (Requerimientos, Análisis, Diseño, Construcción y Pruebas).

Capítulo

4

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Análisis del Sistema de Información

4. Análisis del Sistema de Información (ASI)

En el presente capítulo se describe y desarrolla el análisis del sistema según las actividades propuestas por la metodología utilizada.

4.1. Introducción

El Análisis de Sistemas trata básicamente de determinar los objetivos y límites del sistema objeto de análisis, caracterizar su estructura y funcionamiento, marcar las directrices que permitan alcanzar los objetivos propuestos y evaluar sus posibles consecuencias.

Es posible agrupar formalmente las tareas que constituyen el análisis en una serie de etapas que se suceden de forma iterativa hasta producir y validar el proceso completo.

4.2. El análisis del sistema en MÉTRICA Versión 3

El propósito de este proceso es conseguir la especificación detallada del sistema, a través de un catálogo de requisitos y una serie de modelos que cubran las necesidades de los usuarios para los que se desarrollará el sistema y que serán la entrada para el proceso de Diseño del Sistema de Información [Métrica V3, 2000].

En primer lugar se describe inicialmente el sistema, a partir de los productos generados en el proceso Estudio de Viabilidad del Sistema (EVS). Se delimita su alcance, se genera un catálogo de requisitos generales y se describe el

sistema mediante unos modelos iniciales de alto nivel. Se recogen de forma detallada los requisitos funcionales que el sistema debe cubrir, catalogándolos, lo que permite hacer la traza a lo largo de los procesos de desarrollo. Además, se identifican los requisitos no funcionales del sistema, es decir, las facilidades que ha de proporcionar el sistema, y las restricciones a que estará sometido, en cuanto a rendimiento, frecuencia de tratamiento, seguridad, etc.

Para facilitar el análisis del sistema se identifican los subsistemas de análisis, y se elaboran los modelos apropiados.

Se ha incorporado una actividad específica para la definición de Interfaces de Usuario al tiempo que se van obteniendo y depurando los requisitos y los anteriores modelos. Se especificarán todas las interfaces entre el sistema y el usuario, como formatos de pantallas, diálogos, formatos de informes y formularios de entrada.

Finalizados los modelos, se realiza un análisis de consistencia, mediante una verificación y validación, lo que puede forzar la modificación de algunos de los modelos obtenidos.

Una vez realizado dicho análisis de consistencia se elabora el producto Especificación de Requisitos Software, que constituye un punto de referencia en el desarrollo del software y la línea base de referencia para las peticiones de cambio sobre los requisitos inicialmente especificados.

En este proceso se inicia también la especificación del Plan de Pruebas, que se completará en el proceso Diseño del Sistema de Información (DSI).

Los productos resultantes del Análisis del Sistema de Información, dependen del tipo de desarrollo de que se trate.

La participación activa de los usuarios es una condición imprescindible para el análisis de la aplicación, ya que dicha participación constituye una garantía de que los requisitos identificados son comprendidos e incorporados al sistema y, por tanto, de que éste será aceptado. Para facilitar la colaboración de los usuarios, se pueden utilizar técnicas interactivas que permiten al usuario familiarizarse con el nuevo sistema y colaborar en la construcción y perfeccionamiento del mismo.

Las actividades que engloba este proceso se recogen en la siguiente figura (véase **Figura 23 - Tareas del ASI**), en la que se indican las tareas que pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores:

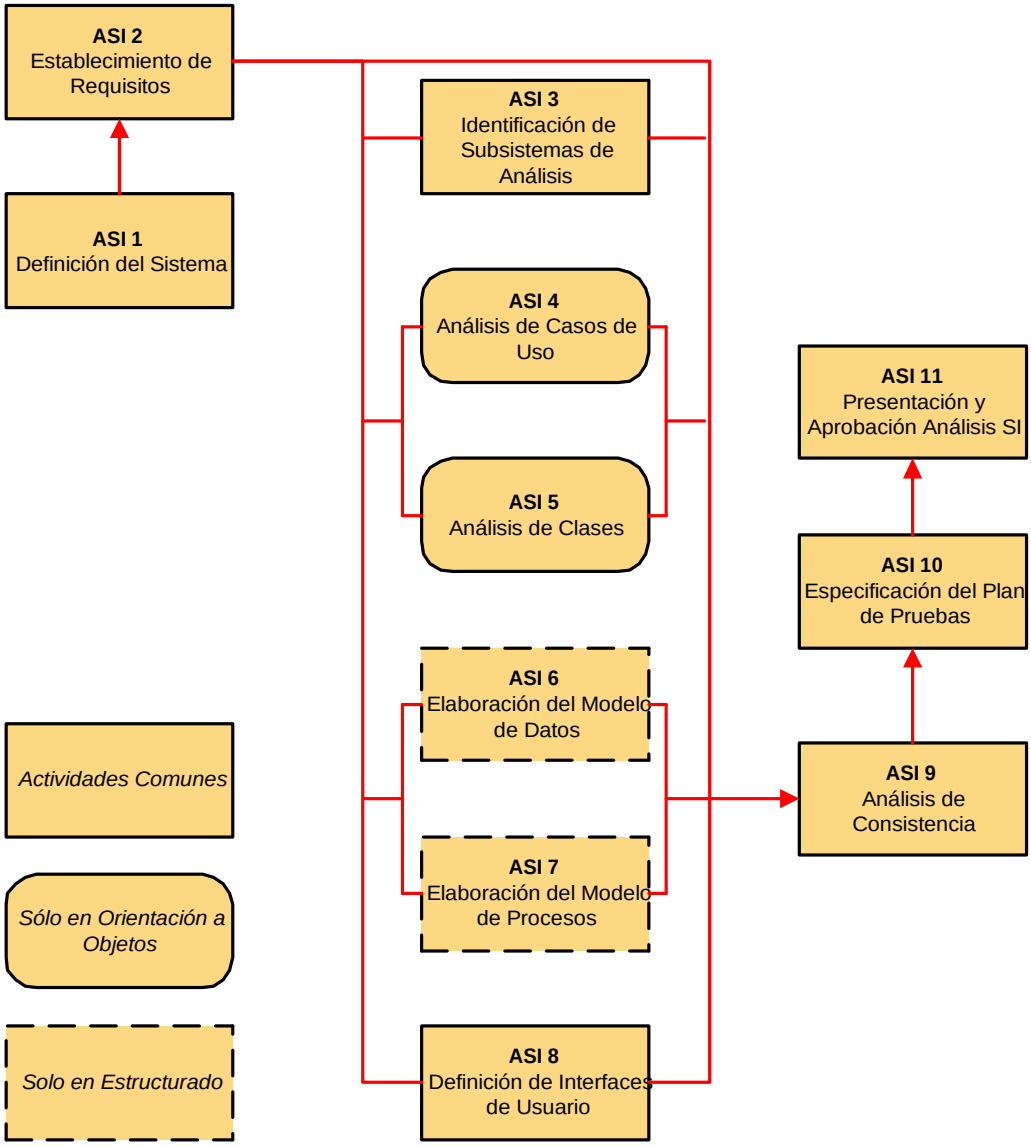


Figura 23 - Tareas del ASI

Dada las características del sistema a realizar, y considerando que la metodología centra su proceso en desarrollos de tipo Estructurado y Orientado a Objetos, se realizarán como parte del presente proyecto las actividades ASI4 y ASI5.

4.3. Un Nuevo Modelo para la Identificación de Riesgos en Base a Taxonomías

[Hantos, 2000] afirma que una serie de experiencias en el uso de taxonomías para las actividades de identificación de riesgos han demostrado que una adaptación previa a la realidad organizacional de las taxonomías estándar posibilita obtener mejores resultados con una mayor eficacia y eficiencia.

Asimismo, [CMMI, 2002] indique que, para ser efectivo, el proceso de identificación de riesgos no debería intentar referenciar todo posible evento

independientemente de la probabilidad que pueda estar asociada al mismo sino que debe centrar su atención en aquellos de mayor relevancia y posibilidad de ocurrencia, en otras palabras, debe validarse la relevancia y aplicabilidad de la información consultada al hacer uso de taxonomías.

En [Microsoft, 2002] se indica que el aprendizaje sólo puede ayudar a mejorar los resultados y el conocimiento obtenido en un proyecto puede reducir la incertidumbre de la toma de decisiones en otros proyectos cuando la información es poco fiable. El análisis directo de los resultados de proyectos anteriores fomenta el aprendizaje dentro de una organización y de un equipo.

Adicionalmente, considerando que la identificación de riesgos es una tarea que se realiza frecuentemente es de esperar un determinado retorno de la inversión en términos monetarios y de tiempo de uso de las técnicas empleadas para llevarla a cabo.

Puede afirmarse que las taxonomías pueden usarse con diferentes propósitos a lo largo de todo el proceso de gestión de riesgos: durante la identificación, para estimular el pensamiento sobre los inconvenientes que pueden producirse en las distintas áreas del proyecto; durante la puesta en común de ideas aligeran la complejidad de trabajar con grandes cantidades de riesgos porque los riesgos similares pueden agruparse; también pueden emplearse para proporcionar una terminología unificada que el equipo de proyecto puede utilizar para supervisar y notificar el estado de los riesgos a lo largo del proyecto y, finalmente, son muy útiles para establecer las bases de conocimiento de riesgo en las organizaciones [Microsoft, 2002].

Finalmente, sería deseable contar con herramientas que soporten el modelo de gestión elegido con el propósito de facilitar su utilización y lograr así un mayor grado de aprovechamiento de sus ventajas.

Los modelos de gestión de riesgos en base a taxonomías que se encuentran mas difundidos en la actualidad no proveen de un soporte ni contemplan explícitamente la necesidad de adecuar y ajustar las clasificaciones empleadas a la realidad presente y a los resultados previamente obtenidos por la organización y los proyectos en las cuales han sido y serán empleados; surge así la necesidad de plantear un nuevo modelo que promueva la explotación de los beneficios derivados de dichas adaptaciones.

4.3.1. MIRT - Modelo para la Identificación de Riesgos en base a Taxonomías

MIRT tiene como objetivo principal el establecer, estandarizar y sistematizar prácticas para la identificación de riesgos entre los encargados de la gestión de proyectos.

Aún cuando el estudio realizado se centra en proyectos de software, MIRT es aplicable a proyectos de cualquier tipo en diferentes áreas de conocimiento y especialidad.

MIRT guía a los gestores de proyecto por el proceso de identificación inicial de riesgos y puede ser utilizado en organizaciones de cualquier tipo por ser parametrizable y por promover el uso de taxonomías estándar.

Los beneficios mas tangibles de MIRT respecto a los modelos tradicionales de soporte para el empleo de taxonomías están evidenciados por los siguientes aspectos:

- Posibilitar el logro de una importante reducción de tiempos y esfuerzos durante las tareas de identificación temprana de riesgos como consecuencia de evitar, en cada sesión de uso, el análisis completo de una taxonomía de riesgos.

- Incrementar la confiabilidad de los riesgos detectados como consecuencia de la utilización de datos históricos y opiniones de expertos tanto de la gestión de proyectos en general como del dominio específico en particular.

Entre otros beneficios apreciables de MIRT se encuentran también el favorecer la estandarización de actividades en una organización, el promover la mejora continua y el uso de lecciones aprendidas y el facilitar la implementación de un modelo de calidad como CMMI.

Hasta el momento no se habían planteado modelos utilizables en organizaciones de cualquier tipo que guiaran a los gestores de proyectos a lo largo del ciclo de adaptación, utilización y reajuste de taxonomías en una actividad de tan vital importancia para los proyectos como lo es la identificación de riesgos.

El presente trabajo final de Tesis de Magíster constituye, entonces, el desarrollado una implementación de MIRT a nivel de prototipo con un doble propósito:

1. Validar el modelo planteado.

2. Brindar soporte a Motorola Argentina SA en las actividades de implementación del PA de Gestión de Riesgos del modelo de calidad CMMI.

Estructura de MIRT

MIRT se organiza en tres fases divididas en actividades (véase **Figura 24 - Estructura de MIRT**):

- **Fase 1 – Parametrización**
  - *Actividad 1* – Selección de una Taxonomía Base.
  - *Actividad 2* – Selección de una Clasificación de Tipos de Proyectos.
  - *Actividad 3* – Generación de una Taxonomía Ajustada de Riesgos.
  - *Actividad 4* – Generación de Reglas de Identificación de Riesgos.

- **Fase 2 – Ejecución**
  - *Actividad 1* – Especificación de las Características de Proyecto.
  - *Actividad 2* – Ejecución de las Reglas de Identificación de Riesgos.
  - *Actividad 3* – Respuesta del Gestor de Proyecto a consultas sobre Riesgos.
  - *Actividad 4* – Presentación de los Riesgos Identificados.

- **Fase 3 – Seguimiento**
  - *Actividad 1* – Monitoreo de los Riesgos Identificados.
  - *Actividad 2* – Detección de Oportunidades de Mejora.

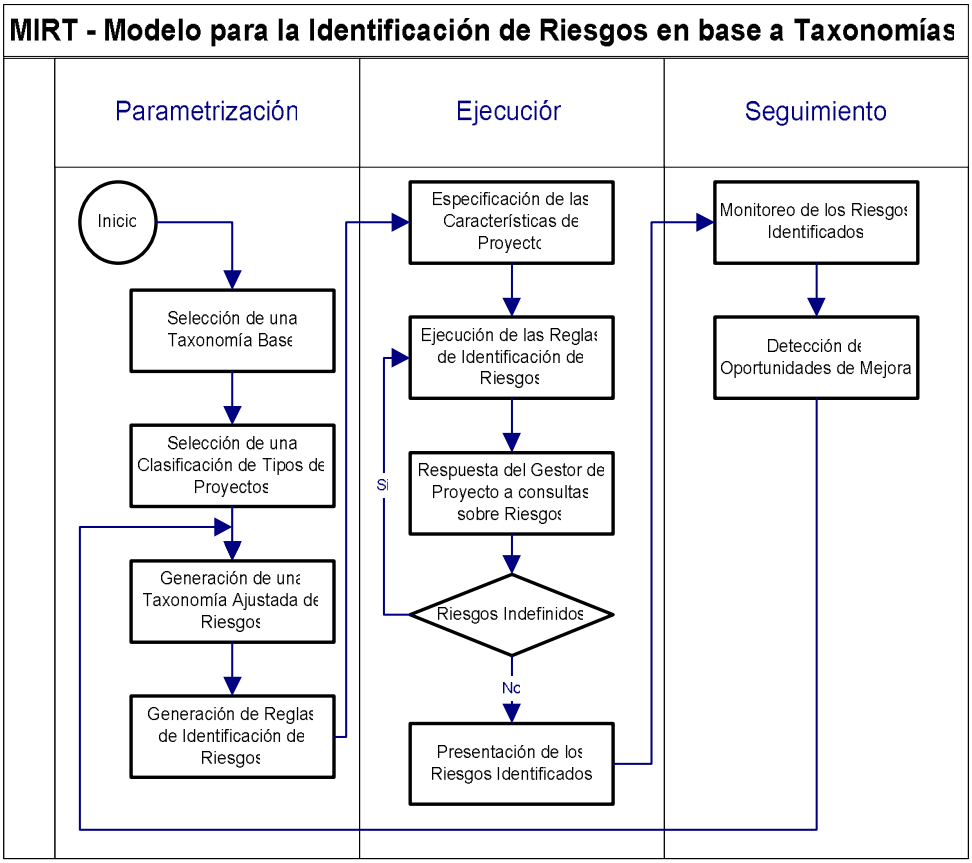


Figura 24 - Estructura de MIRT

Descripción de MIRT

Se describen a continuación cada una de las fases y actividades que componen el modelo presentado.

Fase 1 – Parametrización

La primera de las fases de MIRT es la de **Parametrización**, y su objetivo es el adaptar y ajustar los componentes fundamentales del modelo (taxonomías, riesgos y tipificación de proyectos) a las características particulares de la organización en la cual será utilizado.

Esta primera fase consta de cuatro actividades dos de las cuales conforman la subfase de Inicialización del modelo (*Selección de una Taxonomía Base y Selección de una Clasificación de Tipos de Proyectos*) y las restantes que

conforman la subfase de Mejoramiento Continuo (*Generación de una Taxonomía Ajustada de Riesgos y Generación de Reglas de Identificación de Riesgos*).

La actividad de *Selección de una Taxonomía Base* implica elegir (crear y/o combinar) algunas de las taxonomías de riesgos estándar existentes en la industria para cada una de las diferentes áreas de conocimiento.

La *Selección de una Clasificación de Tipos de Proyectos* involucra el optar (crear y/o combinar) por algunas de las tipificaciones de proyectos existentes en la industria para el área de conocimiento en la cual se trabaje.

La *Generación de una Taxonomía Ajustada de Riesgos* se realiza luego de finalizada la subfase de Inicialización. Esta actividad toma como elementos de partida tanto la Taxonomía Base elegida como la Clasificación de Tipos de Proyectos seleccionada y, mediante una serie de sesiones de tipo Wideband Delphi entre expertos del dominio, produce como resultado una modificación de la Taxonomía Base que, por cada riesgo y tipo de proyecto, contiene información de la aplicabilidad y del grado final de exposición típicamente esperable.

Finalmente, en la actividad de *Generación de Reglas de Identificación de Riesgos* se consulta a un grupo de expertos (tanto en el dominio como en las actividades de gestión de proyectos) con el propósito de identificar y producir un conjunto de reglas que reflejen relaciones causales y de exclusión entre riesgos, es decir, para establecer criterios por los cuales la presencia (o ausencia) de uno o un conjunto de riesgos alienta (o dificulta) la aparición de otro u otros riesgos.

Fase 2 – Ejecución

La segunda de las fases de MIRT es la de **Ejecución**, y su objetivo es el determinar un listado de riesgos candidatos en un proyecto particular considerando sus características y empleando como bases la Taxonomía Ajustada de Riesgos y las Reglas de Identificación surgidas en la fase previa.

Esta segunda fase consta de cuatro actividades dos de las cuales se realizan una vez solamente (*Especificación de las Características de Proyecto y Presentación de los Riesgos Identificados*) y las restantes que se llevan a cabo de manera repetitiva (*Ejecución de las Reglas de Identificación de Riesgos y Respuesta del Gestor de Proyecto a consultas sobre Riesgos*).

La *Especificación de las Características de Proyecto* implica el determinar diferentes valuaciones a varios aspectos que definirán al proyecto que se desea evaluar (tipo, existencia de organización cliente y entorno organizacional, por ejemplo), de formal tal de determinar un primer grupo de riesgos candidatos en base a los criterios de aplicabilidad definidos previamente.

A continuación, y hasta lograr determinar la exposición en el proyecto de cada uno de los riesgos de la taxonomía, se realiza la *Ejecución de las Reglas de Identificación de Riesgos* y se solicita *Respuesta del Gestor de Proyecto a consultas sobre Riesgos*. En base al valor de la exposición de cada riesgo (predefinido en la Taxonomía Ajustada o determinado en una iteración previa) se ejecutan las reglas de identificación con el propósito de establecer, de forma automática, la importancia final esperada para cada uno de los riesgos de valor aún indeterminado; luego, para todos aquellos riesgos para los cuales no es factible aún fijar una valor de exposición se consulta al gestor de proyecto respecto a cuál utilizar.

Una vez finalizado este proceso repetitivo, se realiza la *Presentación de los Riesgos Identificados* que consiste en listar aquellos riesgos de la taxonomía que sean aplicables al proyecto con el valor de exposición determinado a lo largo de la fase de Ejecución mediante cualquiera de los métodos posibles (predeterminación en la taxonomía ajustada, identificación mediante reglas o consulta al gestor de proyectos).

Fase 3 – Seguimiento

La última de las fases de MIRT es la de **Seguimiento**, y su objetivo es detectar oportunidades de mejora tanto para la Taxonomía Ajustada de Riesgos como para las Reglas de Identificación generadas durante la fase de Parametrización.

Esta tercera fase consta de dos actividades recurrentes: *Monitoreo de los Riesgos Identificados y Detección de Oportunidades de Mejora*.

El *Monitoreo de los Riesgos Identificados* implica el validar la correctitud de los riesgos seleccionados para el proyecto al utilizar el modelo propuesto una vez que las actividades de desarrollo se han puesto en funcionamiento; de esta forma será posible realizar la *Detección de Oportunidades de Mejora* tanto para la Taxonomía Ajustada de Riesgos como para las Reglas de Identificación generadas mediante el empleo de información histórica que posibilite realizar ajustes, agregados, correcciones y nuevas adaptaciones cuyos beneficios serán verificados y aprovechados en posteriores aplicaciones de MIRT.

4.4. Taxonomía de Riesgos del Magíster

Dado que el objetivo principal del trabajo de tesis es el de desarrollar una herramienta que brinde la posibilidad de efectuar tareas de Identificación de Riesgos en base a taxonomías estándar en organizaciones de software de cualquier tipo, y considerando que existen diferentes taxonomías en la industria aplicables a dicho propósito, se emprendió, como parte del trabajo de Tesis de Magíster, el desarrollo de una taxonomía de riesgos que constituya un compendio de la información contenida en muchas de las diferentes clasificaciones provistas y usadas a nivel industrial.

La siguiente figura (véase **Figura 25 - Proceso de Generación de Taxonomía de Riesgos del Magíster**) describe el proceso empleado para llevar a cabo la generación de la taxonomía de riesgos a emplear durante el desarrollo de la Tesis de Magíster.

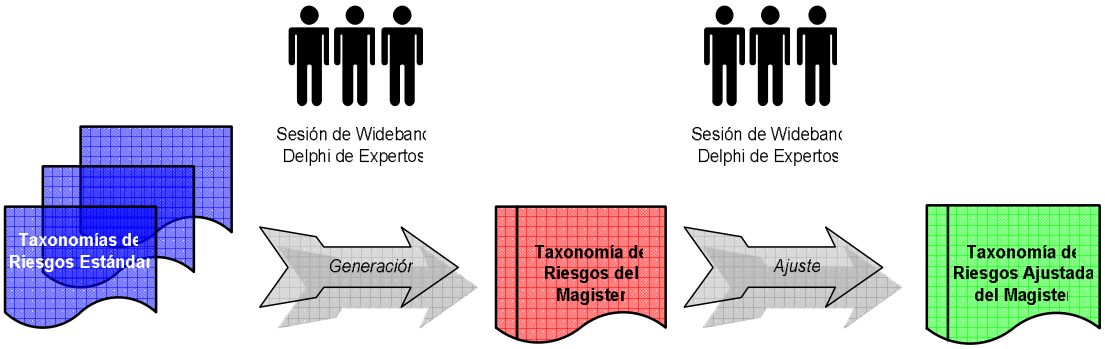


Figura 25 - Proceso de Generación de Taxonomía de Riesgos del Magíster

En base a una serie de taxonomías estándar de la industria ([Marvin J. Carr et al., 1993], [Caper Jones, 1994], [TeraQuest, 1998], [Microsoft, 2002] y [TBS, 2002]) y luego de una serie de sesiones de tipo Wideband Delphi entre expertos del dominio, se generó una Taxonomía de Riesgos del Magíster; dicha taxonomía contiene un compendio de la información y categorías propuestas por diferentes autores y entidades.

Empleando la Taxonomía de Riesgos del Magíster generada, y después de una nueva serie de sesiones de tipo Wideband Delphi entre expertos del dominio, se desarrolló una Taxonomía de Riesgos Ajustada del Magíster; dicha taxonomía agrega a la anterior una serie de criterios de aplicabilidad, relaciones causales y probabilidades de ocurrencia en base a tipos de proyectos que han sido determinados y validados en base a la experiencia y conocimiento de los expertos y que serán empleados durante el desarrollo de la herramienta producto de la Tesis de Magíster.

El resultado del proceso antes descrito puede ser consultado en detalle en **Apéndice F: Taxonomía de Riesgos Ajustada**.

4.5. Entorno tecnológico del sistema

El sistema será construido como una herramienta de software monousuario, ejecutable en computadoras personales y con amplias capacidades gráficas de interfaz con el usuario; asimismo, no poseerá componentes distribuidos ni necesitará infraestructura de red. Teniendo en cuenta estas restricciones, la plataforma tecnológica sobre la cual se construirá la herramienta será J2SE (Edición Estándar de Java versión 2, *Java 2 Standard Edition* en inglés) [J2SE, 2004].

Esta plataforma posibilitará la ejecución de la aplicación sin cambios bajo distintos sistemas operativos. Además de estas facilidades multiplataforma,

Java y la plataforma J2SE cuentan con amplias facilidades de construcción de interfaces gráficas de usuario, con una serie de clases que brindan soporte de internacionalización y con un alto nivel de madurez y aceptación dentro de la industria del software.

El entorno tecnológico mínimo requerido para el sistema es el siguiente:

- Computadora Personal Pentium III con 64 MB de memoria RAM.
- Ambiente de ejecución y/o desarrollo de la plataforma J2SE versión 1.4.2.

4.6. Catálogo de Involucrados

La siguiente tabla (véase **Tabla 4 - Catálogo de Involucrados en el Desarrollo**) enumera los distintos involucrados en el desarrollo de la aplicación final del trabajo de Tesis de Magíster.

ROL	RESPONSABILIDADES	RESPONSABLES
Usuario	Ayudar a entender el problema a resolver y a definir los requisitos del sistema. Validar los requisitos del sistema, las interfaces de usuario, y las funcionalidades en general. Verificar el correcto funcionamiento del sistema una vez finalizado.	Gerencia de Motorola Argentina SA
Tesista	Desarrollar la herramienta siguiendo paso a paso la metodología Métrica V3, definiendo la herramienta con el usuario y planificando el desarrollo con el director de tesis. Documentar todos los productos obtenidos durante el desarrollo. Escribir el documento de tesis.	Lic. Sebastián Maniasi
Director	Controlar el desarrollo de la herramienta de acuerdo a la metodología Métrica V3. Definir la planificación conjuntamente con el tesista. Supervisar los productos obtenidos durante el desarrollo y el documento de tesis.	M. Ing. Paola Britos M. Ing. Eduardo Diez
Equipo de Proyecto	Colaborar en el desarrollo de la aplicación según las guías y especificaciones indicadas por el Tesista.	Personal asignado de Motorola Argentina SA

Tabla 4 - Catálogo de Involucrados en el Desarrollo

4.7. Definición del Sistema y Establecimiento de Requisitos

En esta sección se efectúa una descripción del sistema, delimitando su alcance, estableciendo sus interfaces con otros sistemas e identificando los

usuarios representativos; además se lleva a cabo la definición, análisis y validación de los requisitos a partir de la información facilitada por el usuario.

Los requisitos han sido descriptos y detallados de acuerdo al formato especificado en el estándar de Especificación de Requerimientos del Software de la IEEE [IEEE 830,1998].

4.7.1. Introducción

Objetivos

El propósito del sistema es asistir a los Administradores de Proyectos de Software en la identificación de los riesgos de sus proyectos de desarrollo empleando como técnica base el uso de taxonomías.

Alcance

Esta especificación abarca solamente al sistema denominado Seldon: Asistente para la Identificación de Riesgos de Proyectos de Software en base a Taxonomías. No se incluye a ningún otro sistema existente ni relacionado con este, como asi también a ningún otro hardware o software que pueda ser empleado como soporte para el sistema.

Adicionalmente, el alcance del sistema se limita exclusivamente al soporte de las tareas de identificación de riesgos dentro del modelo de administración de riesgos.

4.7.2. Descripción general

Funciones del producto

El sistema a desarrollar deberá proveer de las siguientes funciones:

- Asistencia en la identificación de riesgos:
  El sistema asistirá al usuario en la identicación de los riesgos de un proyecto de desarrollo de software empleando como base la técnica del uso de taxonomías.

- Reporte de resultados:
  El sistema permitirá que el usuario obtenga un reporte con los riesgos candidatos identificados para un proyecto.

- Administración de los resultados de las sesiones de identificación:
  El sistema brindará al usuario la posibilidad de almacenar y consultar mediante archivos los resultados de las sesiones de identificación de riesgos efectuadas.

- Reporte de taxonomía base:
  El sistema posibilitará al usuario consultar la taxonomía empleada como base para las tareas de identificación de riesgos y la información asociada a cada uno de los elementos de la misma.

Características de los usuarios

Los usuarios del sistema son básicamente todos aquellos administradores de proyectos de desarrollo de software interesados en la realización de la identificación de los posibles riesgos de sus proyectos.

Desde el punto de vista del sistema no se distinguen los usuarios ni por su nivel de experiencia, ni por el tipo de organización en la que se desempeñan ni por el tipo de proyecto de desarrollo que gerencian, existiendo así un único rol de usuario para la aplicación cuyas tareas abarcan toda la funcionalidad detallada previamente.

Restricciones generales

La utilización del sistema está restringida a un único usuario por vez ya que no existe la necesidad de atender a varios usuarios concurrentemente.

La taxonomía empleada como base del proceso por parte del sistema no puede ser reemplazada.

Intefaces externas

El sistema a desarrollar no posee interfaces externas de ningún tipo con otro sistema de software mas allá de las relacionadas con las utilidades brindadas por el sistema operativo (impresión de documentos y guardado o recuperación de archivos, por ejemplo).

4.7.3. Requerimientos específicos

En esta sección se listan los requerimientos específicos del sistema agrupados como requerimientos funcionales y no funcionales.

Todos los requerimientos serán modelados empleando Rational Rose [Rational, 2002], de manera de facilitar la tareas a realizar durante las subsiguientes etapas de de análisis, diseño y codificación.

Requerimientos funcionales

La siguiente figura (véase **Figura 26 - Requerimientos Funcionales**) muestra los requerimientos funcionales del sistema, agrupados de acuerdo a las funciones detalladas previamente.

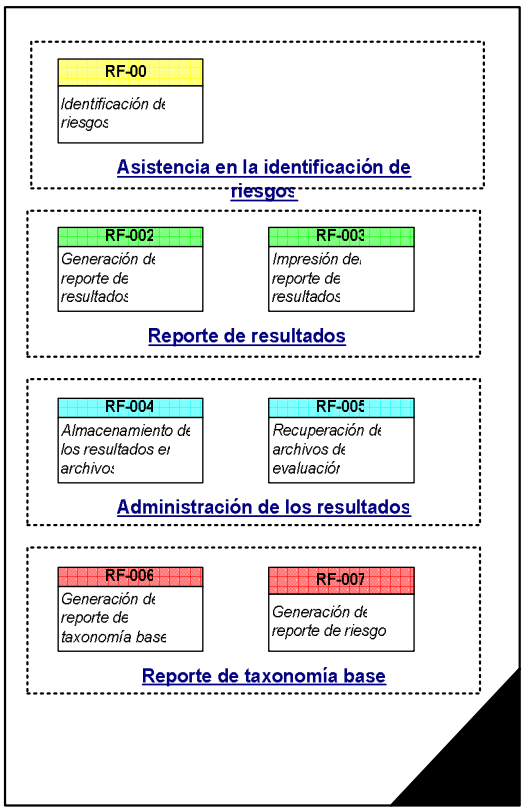


Figura 26 - Requerimientos Funcionales

Asistencia en la identificación de riesgos

El sistema asistirá al usuario en la identificación de los riesgos de un proyecto de desarrollo de software empleando como base la técnica del uso de taxonomías.

RF-001: *Identificación de riesgos*

En base a una serie de datos iniciales relacionados con las características de la organización y el proyecto que el usuario deberá ingresar y luego de una serie de preguntas relacionadas con particularidades de la situación actual del desarrollo en diferentes áreas que el usuario deberá responder, el sistema identificará, a solicitud del usuario, los riesgos candidatos para el proyecto en el cual se trabaja y, en caso de contar con la información suficiente, una estimación inicial del grado de su probabilidad de ocurrencia.

Entradas:

- Taxonomía base ajustada.
- Reglas de identificación de riesgos.
- Características definidas para la organización y el proyecto.
- Respuestas del usuario a preguntas que permitan cubrir todos los riesgos de la taxonomía.

Proceso:

- Se realiza la especificación de las características del proyecto, esto es, determinar diferentes valuaciones a varios aspectos que definirán al proyecto que se desea evaluar (tipo, existencia de organización cliente y entorno organizacional, por ejemplo), de formal tal de determinar un

primer grupo de riesgos candidatos en base a los criterios de aplicabilidad definidos previamente en la taxonomía base ajustada. A continuación, y hasta lograr determinar la exposición en el proyecto de cada uno de los riesgos de la taxonomía, se realiza la ejecución de las reglas de identificación de riesgos y se solicita respuesta del gestor de proyecto a las consultas sobre riesgos. En base al valor de la exposición de cada riesgo (predefinido en la taxonomía base ajustada o determinado mediante cualquiera de los mecanismos antes descriptos en una iteración previa) se ejecutan las reglas de identificación con el propósito de establecer, de forma automática, la importancia final esperada para cada uno de los riesgos de valor aún indeterminado; luego, para todos aquellos riesgos para los cuales no es factible aún fijar una valor de exposición se consulta al gestor de proyecto respecto a cuál utilizar.

Salidas:

- Conjunto de riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio y su posible grado de exposición.

Reporte de resultados

El sistema permitirá que el usuario obtenga un reporte con los riesgos candidatos identificados para un proyecto.

RF-002: *Generación de reporte de resultados*

El sistema deberá permitir que el usuario genere un reporte con los resultados de la sesión de identificación de riesgos. El reporte mostrará los riesgos identificados, su categoría y, en caso de ser posible, una estimación inicial del grado de su probabilidad de ocurrencia.

Entradas:

- Conjunto de riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio y su posible grado de exposición.

Proceso:

- Se recorre el conjunto de todos los riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio con el propósito de generar un árbol jerárquico mediante las relaciones entre los mismos y sus categorías y, a su vez, de obtener toda la información referente a estos riesgos, que incluye: descripción, categoría y probable grado de exposición.

Salidas:

- Pantalla conteniendo un listado en forma de arbol jerárquico de los riesgos identificados, su descripción, categoría y su posible grado de exposición.

RF-003: *Impresión del reporte de resultados*

El sistema deberá permitir que el usuario imprima el reporte con los resultados de la sesión de identificación de riesgos.

Entradas:

- Conjunto de riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio y su posible grado de exposición.

Proceso:

- Se recorre el conjunto de riesgos de la taxonomía candidatos a presentarse en el proyecto objeto de estudio con el propósito de obtener toda la información referente a los mismos: descripción, categoría y probable grado de exposición. Una vez conseguido, se invoca una interfaz de software estándar de impresión y se envían los datos recolectados.

Salidas:

- Reporte de impresión de riesgos según el formato definido en **4.10.4**.**Formatos de Impresión**.

Administración de los resultados

El sistema brindará al usuario la posibilidad de almacenar y consultar mediante archivos los resultados de las sesiones de identificación de riesgos efectuadas.

RF-004: *Almacenamiento de los resultados en archivos*

El sistema deberá permitir que el usuario almacene en un archivo los resultados de una sesión de identificación. Para ello, brindará una interfaz donde el usuario podrá indicar el archivo donde se almacenará la información.

Entradas:

- Una sesión de identificación de riesgos finalizada.

Proceso:

- Se obtiene toda la información relacionada con la sesión de identificación de riesgos, esto incluye: características particulares definidas tanto para la organización y como para el proyecto y, además, el conjunto de los riesgos identificados. Una vez conseguido, se invoca una interfaz de software estándar de almacenamiento de archivos en disco y se envían los datos recolectados en el formato a definir en etapas posteriores del proceso de desarrollo del sistema.

Salidas:

- Un archivo en la ubicación seleccionada conteniendo toda la información de la sesión de identificación de riesgos que fuera almacenada.

RF-005: *Recuperación de archivos de evaluación*

El sistema deberá permitir que el usuario abra el resultado de una sesión de identificación guardado en archivo para su consulta.

Entradas:

- Indicación de la ubicación y el archivo que contiene la información de la sesión de identificación de riesgos que se pretende recuperar.

Proceso:

- Se invoca una interfaz de software estándar de lectura y recuperación de archivos en disco y se muestran los datos recolectados en forma del reporte de resultados (siguiendo la descripción realizada para **RF-002:** *Generación de reporte de resultados*).

Salidas:

- Reporte de resultados para la sesión de riesgos recuperada desde el archivo.

Reporte de taxonomía base
El sistema posibilitará al usuario consultar la taxonomía empleada como base para las tareas de identificación de riesgos y la información asociada a cada uno de los elementos de la misma.

RF-006: *Generación de reporte de taxonomía base*
El sistema deberá permitir que el usuario genere un reporte con la información de la taxonomía que sirve de base al proceso de identificación de riesgos. El reporte mostrará las categorías y los riesgos comprendidos en cada una de ellas.

Entradas:
- Taxonomía base ajustada.

Proceso:
- Se recorre la taxonomía base ajustada con el propósito de generar un árbol jerárquico de categorías y, a su vez, de identificar los riesgos comprendidos en cada una de ellas.

Salidas:
- Pantalla conteniendo un listado en forma de arbol jerárquico de las categorías de riesgos contenidas en la taxonomía y los riesgos comprendidos en la misma.

RF-007: *Generación de reporte de riesgo*
El sistema deberá permitir que el usuario genere un reporte con la información específica de un riesgo de la taxonomía. El reporte mostrará la categoría a la cual pertenece el riesgo y el resto de la información relevante respecto al mismo.

Entradas:
- Taxonomía base ajustada.

Proceso:
- Se recorre el conjunto de riesgos de la taxonomía base ajustada con el propósito de obtener, para cada uno de ellos, toda la información referente a los mismos: descripción, categoría y probables grados de exposición.

Salidas:
- Pantalla conteniendo, para cada uno de los riesgos, su descripción, categoría y posibles grados de exposición.

En la sección **4.9.3**.**Definición de Casos de Uso** puede encontrarse una descripción de los requerimientos antes detallados desde el punto de vista del usuario de la aplicación.

Requerimientos no funcionales

La siguiente figura (véase **Figura 27 - Requerimientos No Funcionales**) muestra los requerimientos no funcionales del sistema, agrupados de acuerdo a características de usabilidad y de portabilidad. No se han definido requerimientos para otros aspectos no funcionales.

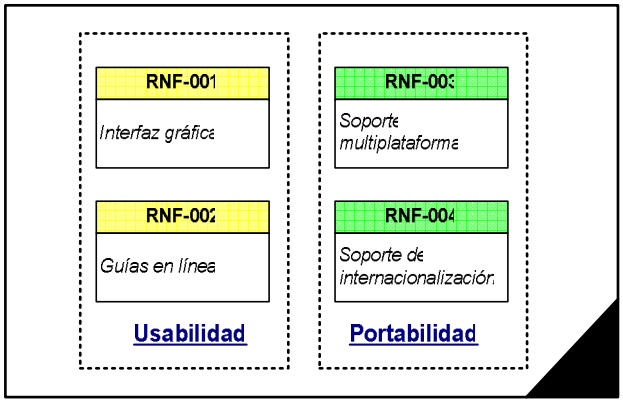


Figura 27 - Requerimientos No Funcionales

Usabilidad
Se detallan a continuación los requerimientos no funcionales de usabilidad del sistema.

RNF-001: *Interfaz gráfica*
La herramienta a desarrollar constituirá una aplicación del tipo “standalone” o independiente con una interfaz gráfica de navegación orientada a facilitar su empleo a los usuarios finales. El usuario podrá navegar y utilizar las funciones del sistema mediante el uso de menús y ventanas.

RNF-002: *Guías en línea*
El sistema deberá proveer al usuario guías en línea de fácil acceso mediante la interfaz gráfica; dichas guías deberán contener la documentación completa tanto de la taxonomía empleada para realizar las tareas de identificación como de cada uno de los riesgos componentes de la misma.

Portabilidad
Se detallan a continuación los requerimientos no funcionales de portabilidad del sistema.

RNF-003: *Soporte multiplataforma*
El sistema deberá tener la capacidad de ejecutarse en los equipos y sistemas operativos que posean la capacidad de soportar aplicaciones de tipo J2SE (Edición Estándar de Java versión 2, *Java 2 Standard Edition* en inglés) [J2SE, 2004].

RNF-004: *Soporte de internacionalización*
El sistema deberá ser portable a diferentes idiomas mediante el reemplazo de archivos de configuración creados a tal efecto y desarrollados siguiendo las restricciones de formato y contenido que serán definidas durante la construcción de la aplicación.

4.8. Identificación de Subsistemas de Análisis

Teniendo en cuenta la complejidad del sistema y considerando el tamaño de la aplicación a desarrollar, no ha sido necesaria ni resultaría provechosa la división del sistema en subsistemas para llevar a cabo las tareas de análisis.

Debe considerarse que las funciones relacionadas con manejo de interfaz gráfica, impresión, internacionalización y soporte de reglas serán provistas por componentes preexistentes del lenguaje y que, por lo tanto, no se consideran como subsistemas del producto del proyecto de Tesis de Magíster.

4.9. Análisis de Casos de Uso y Clases

En esta sección se pretende clarificar los requerimientos mediante la descripción de los casos de uso que los caracterizan e identificar y describir, además, las clases cuyos objetos son necesarios para realizarlos.

4.9.1. Diagrama de Casos de Uso

La siguiente figura (véase **Figura 28 - Diagrama de Casos de Uso**) representa el diagrama de casos de uso del sistema a construir como resultado de la Tesis de Magíster.

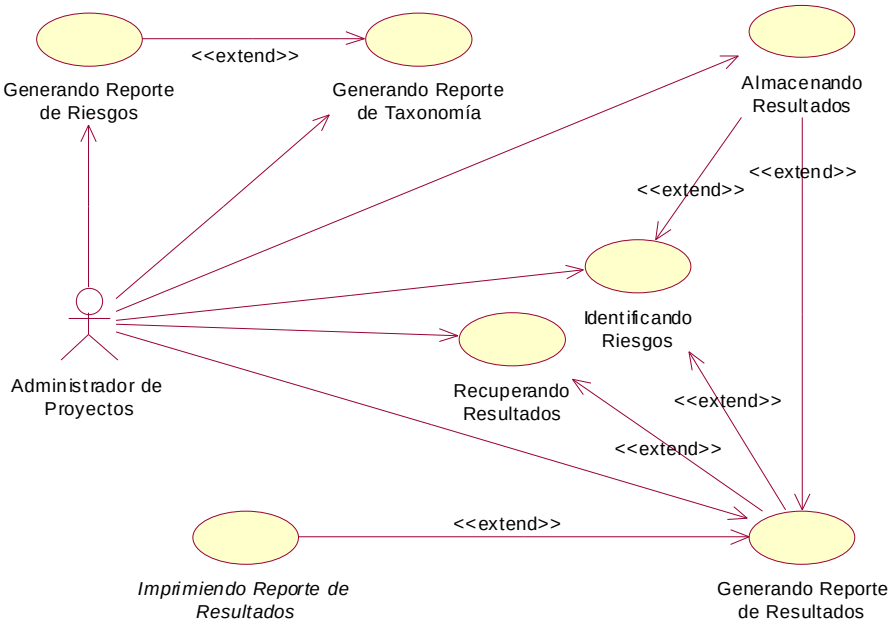


Figura 28 - Diagrama de Casos de Uso

4.9.2. Definición de Actores

Administrador de Proyectos

Cualquier ingeniero de software o profesional de la industria del software interesado en la realización de la identificación de los posibles riesgos de sus proyectos de desarrollo.

4.9.3. Definición de Casos de Uso

Nivel: ☐ Negocio ☒ Aplicación		
Nombre: Identificando Riesgos		**Identificador**: CU-001
Prioridad: ☒ Alta ☐ Media ☐ Baja	**Complejidad**: ☒ Alta ☐ Media ☐ Baja	
Actor Principal: Administrador de Proyectos	**Actor Secundario**: N/A	
Tipo: ☒ Concreto ☐ Abstracto		
Objetivo: Asistir al usuario en la identificación de los riesgos de un proyecto de desarrollo de software empleando como base la técnica del uso de taxonomías.		
Pre-Condiciones: ▪ Sistema configurado apropiadamente.		
Post-Condiciones: Éxito: ▪ Identificación de riesgos finalizada. ▪ Mensaje de error reportado. Fracaso: ▪ La identificación de riesgos no pudo ser realizada.		
Flujo Normal	**Alternativas**	
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita realizar una identificación de riesgos.		
2.- El sistema solicita el ingreso de datos relacionados con la identificación del proyecto (organización a la que pertenece, nombre del proyecto, responsable, etc.).	2.1.a.- Los datos ingresados son validados correctamente. 2.1.b.- Los datos ingresados no son validados correctamente. 2.1.b.1.- El usuario corrige los datos. 2.1.b.2.- El usuario no corrige los datos. 2.1.b.2.a.- Cancelación del caso de uso reportando el error.	
3.- El sistema solicita el ingreso de datos relacionados con las características del proyecto (tipo de	3.1.a.- Los datos ingresados son validados correctamente. 3.1.b.- Los datos ingresados no son	

proyecto, existencia de organización cliente, entorno organizacional, etc.).	validados correctamente. 3.1.b.1.- El usuario corrige los datos. 3.1.b.2.- El usuario no corrige los datos. 3.1.b.2.a.- Cancelación del caso de uso reportando el error.
4.- El sistema solicita el ingreso de datos relacionados con las características de los posibles tipos de riesgos de las taxonomías (situación actual, precedentes, etc.).	4.1.a.- Los datos ingresados son validados correctamente. 4.1.b.- Los datos ingresados no son validados correctamente. 4.1.b.1.- El usuario corrige los datos. 4.1.b.2.- El usuario no corrige los datos. 4.1.b.2.a.- Cancelación del caso de uso reportando el error.
5.- La identificación de riesgos se realiza de forma exitosa empleando la taxonomía base.	5.1.- Invocar al caso de uso “Generando Reporte de Resultados”. 5.2.- Invocar al caso de uso “Almacenando Resultados”. 5.3.- La identificación de riesgos no se realiza de forma exitosa. 5.3.a.- Cancelación del caso de uso reportando el error.
6.- Fin del caso de uso.	
Asociaciones de Extensión: ▪ Generando Reporte de Resultados ▪ Almacenando Resultados	**Casos de Uso Extendidos:** No Aplicable
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel: ☐ Negocio ☒ Aplicación		
Nombre: Generando Reporte de Resultados		**Identificador**: CU-002
Prioridad: ☒ Alta ☐ Media ☐ Baja	**Complejidad**: ☐ Alta ☒ Media ☐ Baja	
Actor Principal: Administrador de Proyectos	**Actor Secundario**: N/A	
Tipo: ☒ Concreto ☐ Abstracto		
Objetivo: Mostrar los resultados de una sesión de identificación de riesgos.		
Pre-Condiciones: ▪ Resultados de una sesión de identificación de riesgos disponibles.		

Post-Condiciones: Éxito: ▪ Reporte generado. ▪ Mensaje de error reportado. Fracaso: ▪ La generación del reporte no pudo ser realizada.	
Flujo Normal	**Alternativas**
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita la generación de un reporte de resultados.	
2.- Tomando como base los resultados de una sesión de identificación de riesgos, el sistema genera exitosamente un reporte conteniendo los riesgos identificados, su categoría y, en caso de ser posible, una estimación inicial del grado de su probabilidad de ocurrencia.	2.1.- Invocar al caso de uso "Almacenando Resultados". 2.2.- Invocar al caso de uso "Imprimiendo Reporte de Resultados". 2.3.- La generación del reporte de resultados no se realiza de forma exitosa. 2.3.a.- Cancelación del caso de uso reportando el error.
3.- Fin del caso de uso.	
Asociaciones de Extensión: ▪ Almacenando Resultados ▪ Imprimiendo Reporte de Resultados	**Casos de Uso Extendidos:** ▪ Identificando Riesgos
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel:	☐ Negocio		☒ Aplicación		
Nombre: Imprimiendo Reporte de Resultados			**Identificador**: CU-003		
Prioridad: ☐ Alta	☒ Media	☐ Baja	**Complejidad**: ☐ Alta	☒ Media	☐ Baja
Actor Principal: Administrador de Proyectos			**Actor Secundario**: N/A		
Tipo:	☐ Concreto		☒ Abstracto		
Objetivo: Imprimir los resultados de una sesión de identificación de riesgos.					
Pre-Condiciones: ▪ Reporte de resultados de una sesión de identificación de riesgos disponibles.					
Post-Condiciones: Éxito:					

▪ Impresión generada. ▪ Mensaje de error reportado. Fracaso: ▪ La impresión no pudo ser realizada.	
Flujo Normal	**Alternativas**
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita la impresión de un reporte de resultados.	
2.- Tomando como base el reporte de resultados de una sesión de identificación de riesgos, el sistema lo imprime exitosamente.	2.1.- La impresión del reporte de resultados no se realiza de forma exitosa. 2.1.a.- Cancelación del caso de uso reportando el error.
3.- Fin del caso de uso.	
Asociaciones de Extensión: No Aplicable	**Casos de Uso Extendidos:** ▪ Generando Reporte de Resultados
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel: ☐ Negocio ☒ Aplicación		
Nombre: Almacenando Resultados		**Identificador**: CU-004
Prioridad: ☐ Alta ☒ Media ☐ Baja	**Complejidad**: ☐ Alta ☒ Media ☐ Baja	
Actor Principal: Administrador de Proyectos	**Actor Secundario**: N/A	
Tipo: ☒ Concreto ☐ Abstracto		
Objetivo: Almacenar en un archivo los resultados de una sesión de identificación.		
Pre-Condiciones: ▪ Resultados de una sesión de identificación de riesgos disponibles.		
Post-Condiciones: Éxito: ▪ Almacenamiento finalizado. ▪ Mensaje de error reportado. Fracaso: ▪ El almacenamiento no pudo ser realizado.		
Flujo Normal	**Alternativas**	
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita el almacenamiento de los resultados		

de una sesión de identificación.	
2.- El usuario selecciona la localización y el nombre del archivo donde se almacenarán los resultados.	2.1.- El usuario no selecciona la localización y el nombre del archivo donde se almacenarán los resultados. 2.1.a.- Cancelación del caso de uso reportando el error.
3.- Los resultados se almacenan correctamente en el archivo.	3.1.- Los resultados no se almacenan correctamente en el archivo. 3.1.a.- Cancelación del caso de uso reportando el error.
4.- Fin del caso de uso.	
Asociaciones de Extensión: No Aplicable	**Casos de Uso Extendidos:** ▪ Identificando Riesgos ▪ Generando Reporte de Resultados
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel:	☐ Negocio		☒ Aplicación	
Nombre: Recuperando Resultados			**Identificador**: CU-005	
Prioridad: ☐ Alta	☒ Media	☐ Baja	**Complejidad**: ☐ Alta	☒ Media ☐ Baja
Actor Principal: Administrador de Proyectos			**Actor Secundario**: N/A	
Tipo:	☒ Concreto		☐ Abstracto	
Objetivo: Recuperar de un archivo los resultados de una sesión de identificación.				
Pre-Condiciones: No Aplicable.				
Post-Condiciones: Éxito: ▪ Recuperación finalizada. ▪ Mensaje de error reportado. Fracaso: ▪ La recuperación no pudo ser realizada.				
Flujo Normal			**Alternativas**	
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita la recuperación de los resultados de una sesión de identificación.				
2.- El usuario selecciona la localización y el nombre del archivo			2.1.- El usuario no selecciona la localización y el nombre del archivo	

desde donde se recuperarán los resultados.	desde donde se recuperarán los resultados. 2.1.a.- Cancelación del caso de uso reportando el error.
3.- Los resultados se recuperan correctamente del archivo.	3.1.- Invocar al caso de uso "Generando Reporte de Resultados". 3.2.- Los resultados no se recuperan correctamente desde el archivo. 3.2.a.- Cancelación del caso de uso reportando el error.
4.- Fin del caso de uso.	
Asociaciones de Extensión: ▪ Generando Reporte de Resultados	**Casos de Uso Extendidos:** No Aplicable
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel:	☐ Negocio		☒ Aplicación	
Nombre: Generando Reporte de Taxonomía			**Identificador**: CU-006	
Prioridad: ☐ Alta	☐ Media	☒ Baja	**Complejidad**: ☐ Alta	☒ Media ☐ Baja
Actor Principal: Administrador de Proyectos			**Actor Secundario**: N/A	
Tipo:	☒ Concreto		☐ Abstracto	

Objetivo:
Mostrar la información de la taxonomía que sirve de base al proceso de identificación de riesgos.

Pre-Condiciones:
No Aplicable.

Post-Condiciones:
Éxito:
- Reporte generado.
- Mensaje de error reportado.

Fracaso:
- La generación del reporte no pudo ser realizada.

Flujo Normal	**Alternativas**
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita la generación de un reporte de taxonomía base.	
2.- El sistema genera exitosamente un reporte conteniendo las categorías de	2.1.- Invocar al caso de uso "Generando Reporte de Riesgos".

la taxonomía y los riesgos comprendidos en cada una de ellas.	2.2.- La generación del reporte de taxonomía base no se realiza de forma exitosa. 2.2.a.- Cancelación del caso de uso reportando el error.
3.- Fin del caso de uso.	
Asociaciones de Extensión: ▪ Generando Reporte de Riesgos	**Casos de Uso Extendidos:** No Aplicable
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

Nivel:	☐ Negocio	☒ Aplicación		
Nombre: Generando Reporte de Riesgos			**Identificador**: CU-007	
Prioridad: ☐ Alta	☐ Media	☒ Baja	**Complejidad**: ☐ Alta	☒ Media ☐ Baja
Actor Principal: Administrador de Proyectos			**Actor Secundario**: N/A	
Tipo:	☒ Concreto	☐ Abstracto		

Objetivo:
Mostrar la información específica de un riesgo de la taxonomía que sirve de base al proceso de identificación de riesgos.

Pre-Condiciones:
No Aplicable.

Post-Condiciones:
Éxito:
- Reporte generado.
- Mensaje de error reportado.

Fracaso:
- La generación del reporte no pudo ser realizada.

Flujo Normal	**Alternativas**
1.- El caso de uso comienza cuando el Administrador de Proyectos solicita la generación de un reporte de riesgo.	
2.- El usuario selecciona el riesgo respecto al cual desea obtener información.	2.1.- El usuario no selecciona el riesgo respecto al cual desea obtener información. 2.1.a.- Cancelación del caso de uso reportando el error.
3.- El sistema genera exitosamente un reporte conteniendo la categoría a la cual pertenece el riesgo y el resto de	3.1.- La generación del reporte de riesgo no se realiza de forma exitosa. 3.1.a.- Cancelación del caso de uso

la información relevante respecto al mismo.	reportando el error.
4.- Fin del caso de uso.	
Asociaciones de Extensión: No Aplicable	**Casos de Uso Extendidos:** ▪ Generando Reporte de Taxonomía
Asociaciones de Inclusión: No Aplicable	**Casos de Uso Incluidos:** No Aplicable
Casos de Uso de Generalización: No Aplicable	

4.9.4. Diagrama de Clases

La siguiente figura (véase **Figura 29 - Diagrama de Clases**) representa el diagrama de clases que permiten solucionar los casos de uso identificados para el sistema a construir como resultado de la Tesis de Magíster.

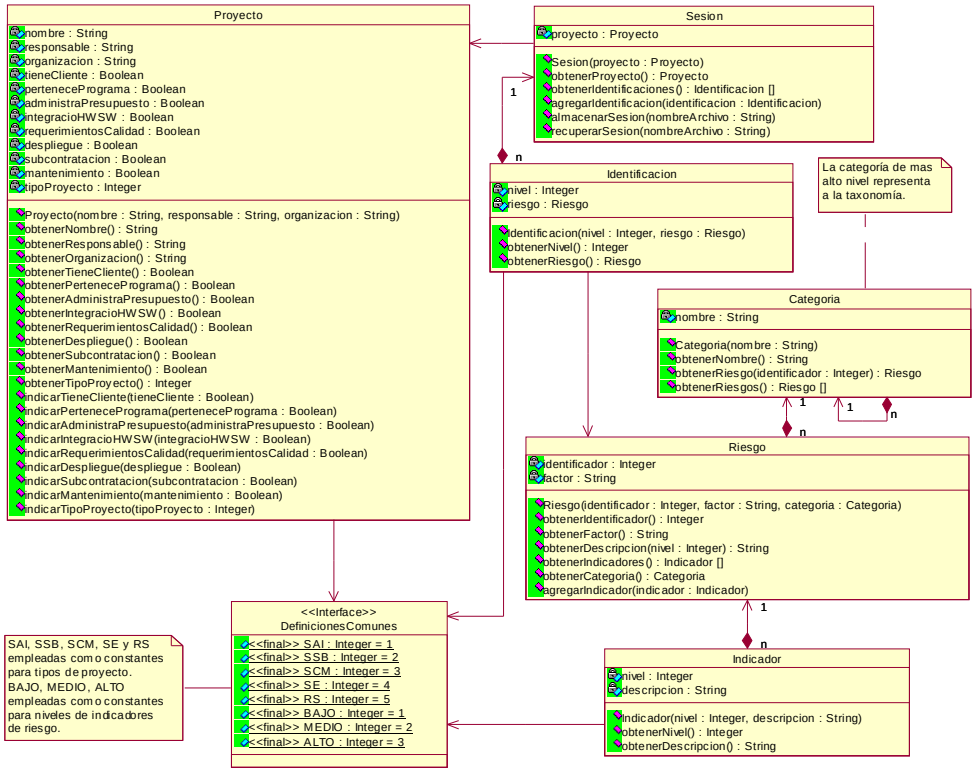


Figura 29 - Diagrama de Clases

Descripción de Atributos y Operaciones

Se describe a continuación cada una de las clases identificadas mediante la caracterización de sus atributos y operaciones.

Categoria

Clase que representa una categoría de la Taxonomía Base para la identificación de riesgos. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 5 - Operaciones de la Clase Categoria** y **Tabla 6 - Atributos de la Clase Categoria**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
`Categoria`	`Categoria (String nombre)`	Constructor de la clase.
`obtenerNombre`	`String obtenerNombre ()`	Permite conocer el nombre de una categoría.
`obtenerRiesgo`	`Riesgo obtenerRiesgo (Integer identificador)`	Permite conocer un riesgo de la categoría.
`obtenerRiesgos`	`Riesgo [] obtenerRiesgos ()`	Permite conocer todos los riesgos de la categoría.

Tabla 5 - Operaciones de la Clase Categoria

ATRIBUTO	TIPO	DESCRIPCIÓN
`Nombre`	`String`	Atributo que representa el nombre de la categoría.

Tabla 6 - Atributos de la Clase Categoria

DefinicionesComunes

Interface que contiene las definiciones comunes empleadas por las clases del sistema. La caracterización de la interface puede observarse en el cuadro siguiente (véase **Tabla 7 - Atributos de la Interfaz DefinicionesComunes**).

ATRIBUTO	TIPO	VALOR	DESCRIPCIÓN
SAI	Integer	1	Constante para identificar los Sistemas de Administración de Información.
SSB	Integer	2	Constante para identificar los emprendimientos de Software de Sistemas Base.
SCM	Integer	3	Constante para identificar los sistemas de Software Comercial de Mercado.
SE	Integer	4	Constante para identificar los emprendimientos de Sistemas Especiales.
RS	Integer	5	Constante para identificar los emprendimientos de reingeniería de sistemas.
BAJO	Integer	1	Nivel de indicador de riesgo bajo.
MEDIO	Integer	2	Nivel de indicador de riesgo medio.
ALTO	Integer	3	Nivel de indicador de riesgo alto.

Tabla 7 - Atributos de la Interfaz DefinicionesComunes

Identificacion

Clase que representa cada una de las identificaciones de riesgos realizadas por el sistema. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 8 - Operaciones de la Clase Identificacion** y **Tabla 9 - Atributos de la Clase Identificacion**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
Identificacion	Identificacion (Integer nivel, Riesgo riesgo)	Constructor de la clase.
obtenerNivel	Integer obtenerNivel ()	Permite conocer el nivel del riesgo identificado por el sistema.
obtenerRiesgo	Riesgo obtenerRiesgo ()	Permite conocer el riesgo identificado por la aplicación.

Tabla 8 - Operaciones de la Clase Identificacion

ATRIBUTO	TIPO	DESCRIPCIÓN
nivel	Integer	Atributo que representa el nivel estimado del riesgo identificado por el sistema.
riesgo	Riesgo	Atributo que representa el riesgo identificado.

Tabla 9 - Atributos de la Clase Identificacion

Indicador

Clase que representa los diferentes indicadores para el nivel de un riesgo. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 10 - Operaciones de la Clase Indicador** y **Tabla 11 - Atributos de la Clase Indicador**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
Indicador	Indicador (Integer nivel, String descripcion)	Constructor de la clase.
obtenerNivel	Integer obtenerNivel ()	Permite conocer el nivel asociado a un indicador.
obtenerDescripcion	String obtenerDescripcion ()	Permite conocer la descripción de un identificador.

Tabla 10 - Operaciones de la Clase Indicador

ATRIBUTO	TIPO	DESCRIPCIÓN
nivel	Integer	Atributo que representa el nivel asociado al indicador.
descripcion	String	Atributo que representa la descripción del indicador.

Tabla 11 - Atributos de la Clase Indicador

Proyecto

Clase que representa un proyecto. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 12 - Operaciones de la Clase Proyecto** y **Tabla 13 - Atributos de la Clase Proyecto**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
`Proyecto`	`Proyecto (String nombre, String responsable, String organizacion)`	Constructor de la clase.
`obtenerNombre`	`String obtenerNombre ()`	Permite obtener el nombre del proyecto.
`obtenerResponsable`	`String obtenerResponsable ()`	Permite obtener el nombre del responsable del proyecto.
`obtenerOrganizacion`	`String obtenerOrganizacion ()`	Permite obtener el nombre de la organización a la cual pertenece el proyecto.
`obtenerTieneCliente`	`Boolean obtenerTieneCliente ()`	Permite conocer si el proyecto tiene o no un cliente o usuario directo.
`obtenerPertenecePrograma`	`Boolean obtenerPertenecePrograma ()`	Permite conocer si el proyecto pertenece o no a un programa dentro de la organización.
`obtenerAdministraPresupuesto`	`Boolean obtenerAdministraPresupuesto ()`	Permite conocer si el

OPERACIÓN	FIRMA	DESCRIPCIÓN
		proyecto es responsable o no por la administración del presupuesto.
obtenerIntegracioHWSW	Boolean obtenerIntegracioHWSW ()	Permite conocer si el proyecto requiere o no de integración con hardware y/o software especifico.
obtenerRequerimientosCalidad	Boolean obtenerRequerimientosCalidad ()	Permite conocer si el proyecto incluye o no requerimientos de calidad.
obtenerDespliegue	Boolean obtenerDespliegue ()	Permite conocer si el proyecto incluye o no la realización de un despliegue.
obtenerSubcontratacion	Boolean obtenerSubcontratacion ()	Permite conocer si el proyecto realiza o no subcontratación.
obtenerMantenimiento	Boolean obtenerMantenimiento ()	Permite conocer si el proyecto realiza o no mantenimiento del software.
obtenerTipoProyecto	Integer obtenerTipoProyecto ()	Permite conocer el t ipo del proyecto.
indicarTieneCliente	indicarTieneCliente (Boolean tieneCliente)	Permite indicar si el proyecto tiene o no

OPERACIÓN	FIRMA	DESCRIPCIÓN
		un cliente o usuario directo.
indicarPertenecePrograma	indicarPertenecePrograma (Boolean pertenecePrograma)	Permite indicar si el proyecto pertenece o no a un programa dentro de la organización.
indicarAdministraPresupuesto	indicarAdministraPresupuesto (Boolean administraPresupuesto)	Permite indicar si el proyecto es responsable o no por la administración del presupuesto.
indicarIntegracioHWSW	indicarIntegracioHWSW (Boolean integracioHWSW)	Permite indicar si el proyecto requiere o no de integración con hardware y/o software especifico.
indicarRequerimientosCalidad	indicarRequerimientosCalidad (Boolean requerimientosCalidad)	Permite indicar si el proyecto incluye o no requerimientos de calidad.
indicarDespliegue	indicarDespliegue (Boolean despliegue)	Permite indicar si el proyecto incluye o no la realización de un despliegue.
indicarSubcontratacion	indicarSubcontratacion (Boolean subcontratacion)	Permite indicar si el proyecto realiza o no subcontratación.

OPERACIÓN	FIRMA	DESCRIPCIÓN
indicarMantenimiento	indicarMantenimiento (Boolean mantenimiento)	Permite indicar si el proyecto realiza o no mantenimiento del software.
indicarTipoProyecto	indicarTipoProyecto (Integer tipoProyecto)	Permite indicar el tipo del proyecto.

Tabla 12 - Operaciones de la Clase Proyecto

ATRIBUTO	TIPO	DESCRIPCIÓN
nombre	String	Atributo empleado para almacenar el nombre del proyecto.
responsable	String	Atributo empleado para almacenar el nombre del responsable del proyecto.
organizacion	String	Atributo empleado para almacenar el nombre de la organización a la cual pertenece el proyecto.
tieneCliente	Boolean	Atributo que permite indicar si el proyecto tiene o no un cliente o usuario directo.
pertenecePrograma	Boolean	Atributo que permite indicar si el proyecto pertenece o no a un programa dentro de la organización.
administraPresupuesto	Boolean	Atributo que permite indicar si el proyecto es responsable o no por la administración del presupuesto.
integracioHWSW	Boolean	Atributo que permite indicar si el proyecto requiere o no de integración con hardware y/o software especifico.
requerimientosCalidad	Boolean	Atributo que permite indicar si el proyecto incluye o no requerimientos de calidad.
Despliegue	Boolean	Atributo que permite indicar si el proyecto incluye o no la realización de un despliegue.
Subcontratación	Boolean	Atributo que permite indicar si el proyecto realiza o no subcontratación.
Mantenimiento	Boolean	Atributo que permite indicar si el proyecto realiza o no mantenimiento del software.
tipoProyecto	Integer	Atributo que permite indicar el tipo del proyecto.

Tabla 13 - Atributos de la Clase Proyecto

Riesgo

Clase que representa un riesgo. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 14 - Operaciones de la Clase Riesgo** y **Tabla 15 - Atributos de la Clase Riesgo**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
`Riesgo`	`Riesgo (Integer identificador, String factor, Categoria categoria)`	Constructor de la clase.
`obtenerIdentificador`	`Integer obtenerIdentificador ()`	Permite conocer el identificador único del riesgo en la taxonomía.
`obtenerFactor`	`String obtenerFactor ()`	Permite conocer la descripción del riesgo.
`obtenerDescripcion`	`String obtenerDescripcion (Integer nivel)`	Permite obtener la descripción de un indicador según su nivel.
`obtenerIndicadores`	`Indicador [] obtenerIndicadores ()`	Permite conocer todos los indicadores asociados a un riesgo.
`obtenerCategoria`	`Categoria obtenerCategoria ()`	Permite conocer la categoría a la cual pertenece un riesgo.
`agregarIndicador`	`agregarIndicador (Indicador indicador)`	Permite agregar un indicador al riesgo.

Tabla 14 - Operaciones de la Clase Riesgo

ATRIBUTO	TIPO	DESCRIPCIÓN
identificador	Integer	Identificador único del riesgo en la taxonomía.
factor	String	Descripción del riesgo.

Tabla 15 - Atributos de la Clase Riesgo

Sesion

Clase que representa una sesión de identificación de riesgos. La caracterización de la clase puede observarse en los cuadros siguientes (véase **Tabla 16 - Operaciones de la Clase Sesion** y **Tabla 17 - Atributos de la Clase Sesion**).

OPERACIÓN	FIRMA	DESCRIPCIÓN
Sesion	Sesion (Proyecto proyecto)	Constructor de la clase.
obtenerProyecto	Proyecto obtenerProyecto ()	Permite conocer el proyecto relacionado con una sesión de identificación de riesgos.
obtenerIdentificaciones	Identificacion [] obtenerIdentificaciones ()	Permite conocer todas las identificaciones realizadas para una sesión.
agregarIdentificacion	agregarIdentificacion (Identificacion identificacion)	Permite agregar una nueva identificación de un riesgo a la sesión.
almacenarSesion	almacenarSesion (String nombreArchivo)	Permite almacenar la sesión de identificación de riesgos en un archivo en disco.
recuperarSesion	recuperarSesion (String nombreArchivo)	Permite recuperar una sesión de identificación de riesgos en un archivo en disco.

Tabla 16 - Operaciones de la Clase Sesion

ATRIBUTO	TIPO	DESCRIPCIÓN
Proyecto	Proyecto	Atributo que representa el proyecto para el cual se realiza la sesión de identificación de riesgos.

Tabla 17 - Atributos de la Clase Sesion

4.10. Definición de Interfaces de Usuario

En esta sección se especifican las interfaces entre el sistema y el usuario: formatos de pantallas, diálogos, e informes, principalmente. El objetivo ha sido realizar un análisis de los procesos del sistema en los que se requiere una interacción del usuario, con el fin de crear una interfaz que satisfaga todos los requisitos establecidos.

4.10.1. Principios Generales de la Interfaz de Usuario

La interfaz de usuario será gráfica e interactiva, del tipo estándar utilizado en las aplicaciones basadas en entorno de ventanas.

Los lineamientos principales para la construcción de la interfaz de usuario son los siguientes:

- La activación de las distintas operaciones de la herramienta se produce mediante opciones de una barra de menú y botones de opciones.
- La aplicación muestra la taxonomía de riesgos y los resultados de los procesos de identificación mediante un árbol, donde los diferentes nodos representan las categorías y los riesgos.
- El usuario puede seleccionar cualquier nodo del árbol que representa un riesgo y obtener sus características.
- La identificación de riesgos se efectúa mediante el uso de ventanas con formularios que contienen campos de texto, listas de selección simple y botones de selección simple o múltiple.
- Las ventanas tendrán, en general, un botón para aceptar los datos provistos y un botón para cancelarlos y, dependiendo de la funcionalidad provista, botones auxiliares para realizar otro tipo de operaciones. Todos los botones se ubicarán en la parte inferior del formulario.
- Los mensajes de error se mostrarán mediante ventanas emergentes.
- En todos los formularios, menús y ventanas de la aplicación, sólo estarán activadas las opciones que pueden ser utilizadas. Aquellas opciones que no pueden ser utilizadas en un momento determinado, se encontrarán deshabilitadas.
- Cualquier operación de cancelación o cierre de una ventana exigirá la confirmación del usuario.

4.10.2. Menús del Sistema

La siguiente pantalla (véase **Figura 30 - Menús de Opciones**) muestra la totalidad de los menús previstos para el sistema.

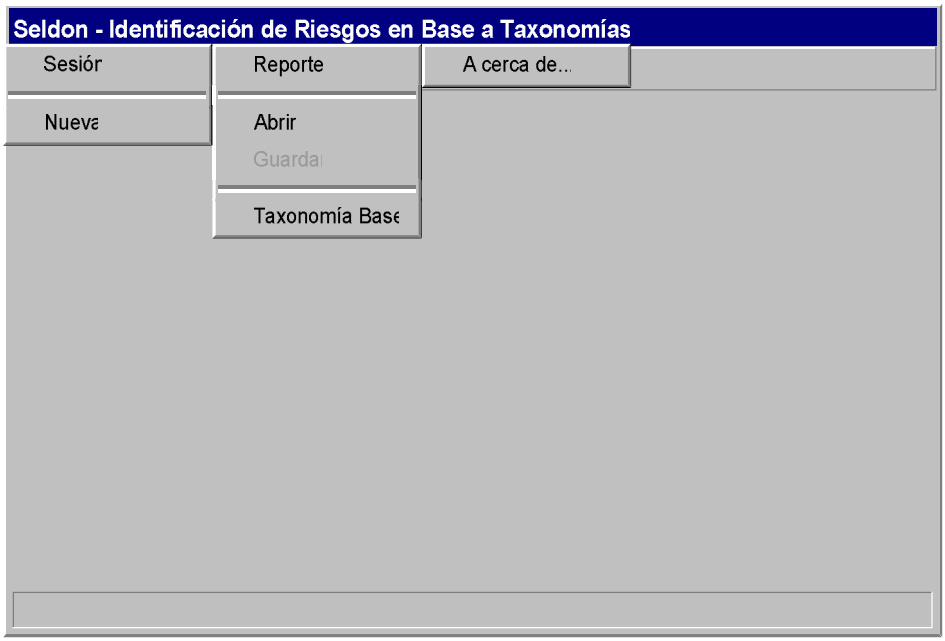


Figura 30 - Menús de Opciones

El siguiente cuadro (véase **Tabla 18 - Asociación Menú-Casos de Uso**) muestra la asociación entre cada una de las opciones de estos menús y los casos de uso definidos para el sistema que se encuentran principalmente relacionados con los mismos.

MENÚ	CASO DE USO
Sesión → Nueva	CU-001
Reporte → Abrir	CU-004
Reporte → Guardar	CU-005
Reporte → Taxonomía Base	CU-006
A cerca de…	No Aplica. Muestra información general del sistema

Tabla 18 - Asociación Menú-Casos de Uso

Aquellos casos de uso que no se encuentran directamente relacionados con alguna de las opciones del menú no se muestran en este cuadro y serán accedidos a través de las facilidades provistas por las pantallas descriptas en la sección siguiente.

4.10.3. Usando el Sistema

Se muestran a continuación una serie de pantallas que permitirán ejemplificar el flujo de operación del sistema a través de los diferentes casos de uso definidos.

Iniciando una nueva sesión de identificación

Al emplear esta pantalla (véase **Figura 31 - Iniciando Nueva Sesión de Identificación**) el usuario solicita realizar una nueva sesión de identificación de riesgos.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso normal.

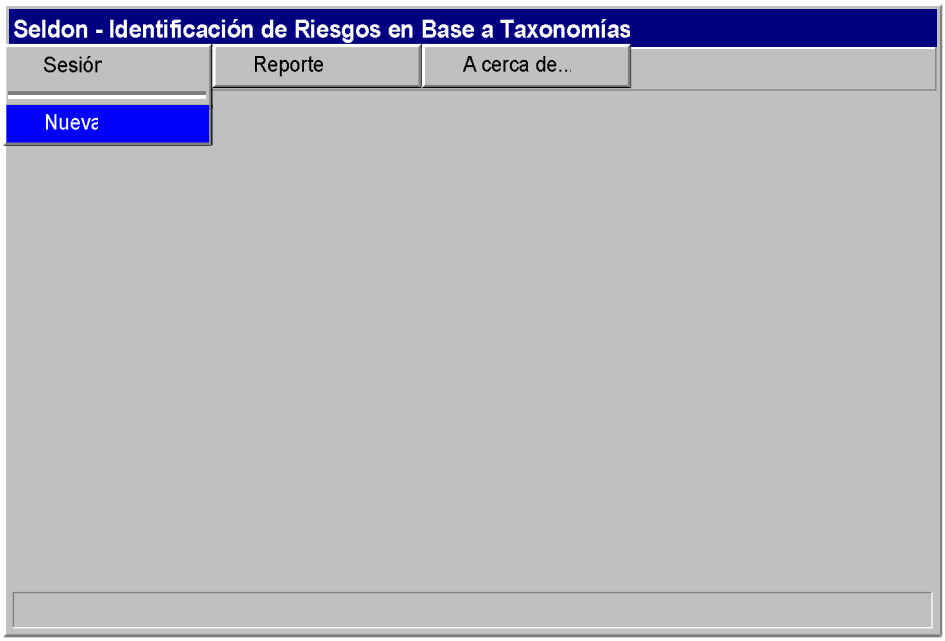


Figura 31 - Iniciando Nueva Sesión de Identificación - INT-001

Identificando el proyecto

Al emplear esta pantalla (véase **Figura 32 - Identificando el Proyecto**) el usuario puede ingresar los datos relacionados con la identificación del proyecto para el cual se realiza la sesión de identificación de riesgos.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso normal.



Figura 32 - Identificando el Proyecto - INT-002

Caracterizando el proyecto

Al emplear esta pantalla (véase **Figura 33 - Caracterizando el Proyecto**) el usuario puede ingresar los datos relacionados con la caracterización del proyecto para el cual se realiza la sesión de identificación de riesgos.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso normal.

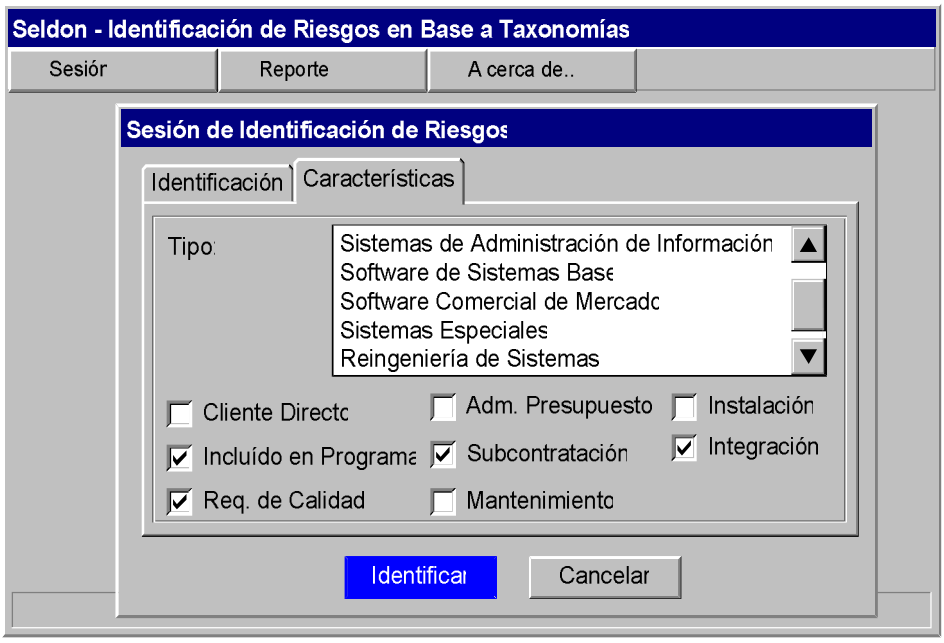


Figura 33 - Caracterizando el Proyecto - INT-003

Caracterizando posibles riesgos

Al emplear esta pantalla (véase **Figura 34 - Caracterizando Posibles Riesgos**) el usuario puede ingresar los datos relacionados con la caracterización de los posibles riesgos del proyecto para el cual se realiza la sesión de identificación.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso normal.

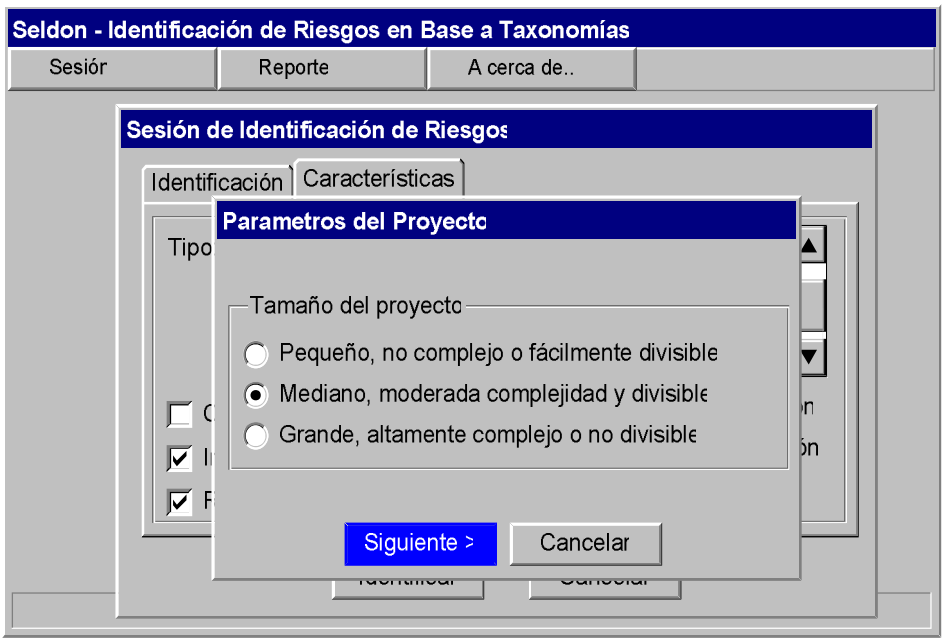


Figura 34 - Caracterizando Posibles Riesgos **-** INT-004

Solicitando la generación del reporte de riesgos

Al emplear esta pantalla (véase **Figura 35 - Solicitando Generación de Reporte de Riesgos**) el usuario puede solicitar al sistema generar el reporte de los riesgos candidatos del proyecto.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso alternativo.
- CU-002, Generando Reporte de Resultados – Curso normal.



Figura 35 - Solicitando Generación de Reporte de Riesgos - INT-005

Revisando y solicitando guardado el reporte de riesgos

Al emplear estas pantallas (véase **Figura 36 - Revisando y Solicitando Guardado del Reporte (1)** y **Figura 37 - Revisando y Solicitando Guardado del Reporte (2)**) el usuario puede observar el reporte de riesgos generado por la aplicación y solicitar el almacenamiento del mismo al sistema.

Los casos de uso relacionados son:

- CU-002, Generando Reporte de Resultados – Curso alternativo.
- CU-004, Almacenando Resultados – Curso normal.

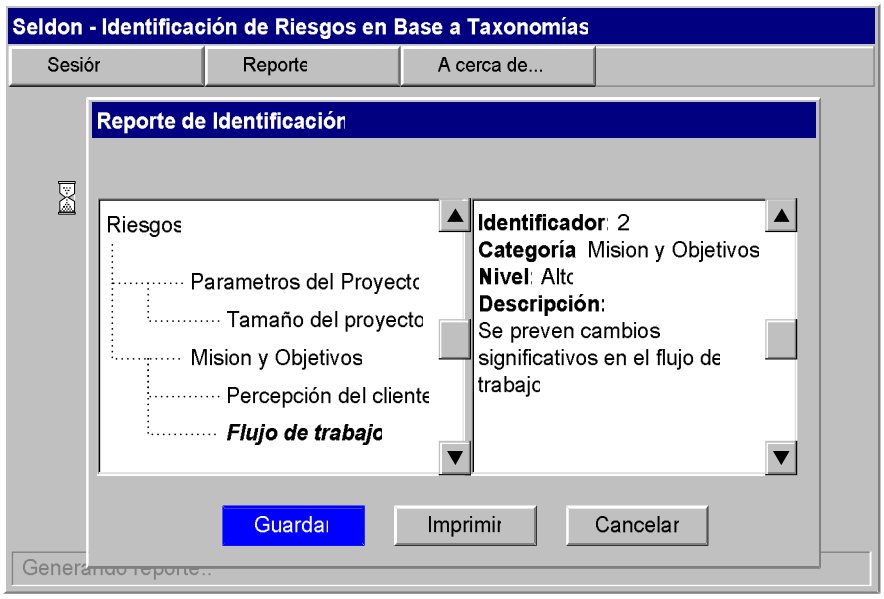


Figura 36 - Revisando y Solicitando Guardado del Reporte (1) - INT-006

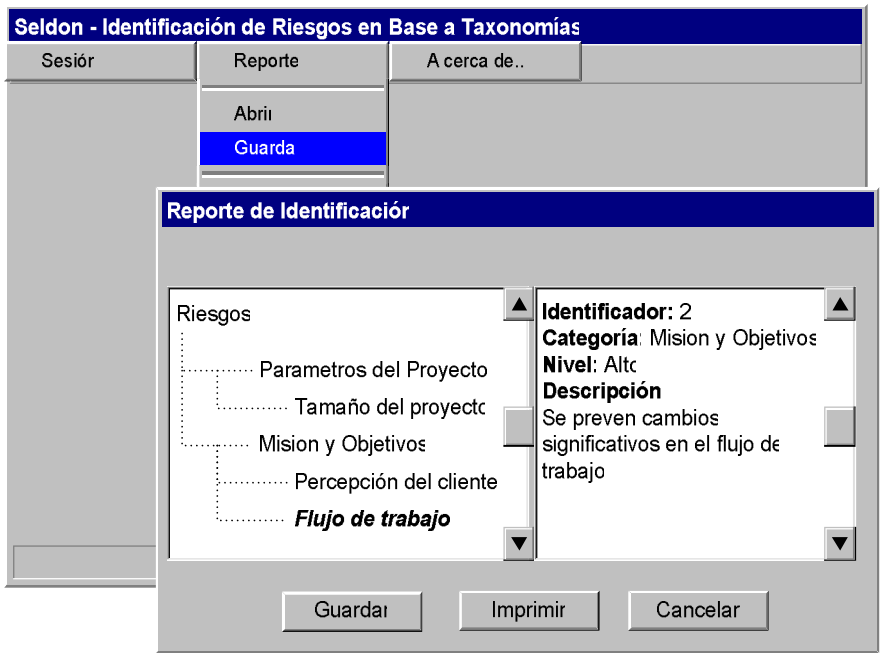


Figura 37 - Revisando y Solicitando Guardado del Reporte (2) - INT-007

Solicitando guardado el reporte de riesgos

Al emplear estas pantallas (véase **Figura 38 - Solicitando Guardado del Reporte de Riesgos**) el usuario puede solicitar el almacenamiento del reporte de riesgos generado por la aplicación.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos – Curso alternativo.
- CU-004, Almacenando Resultados – Curso normal.

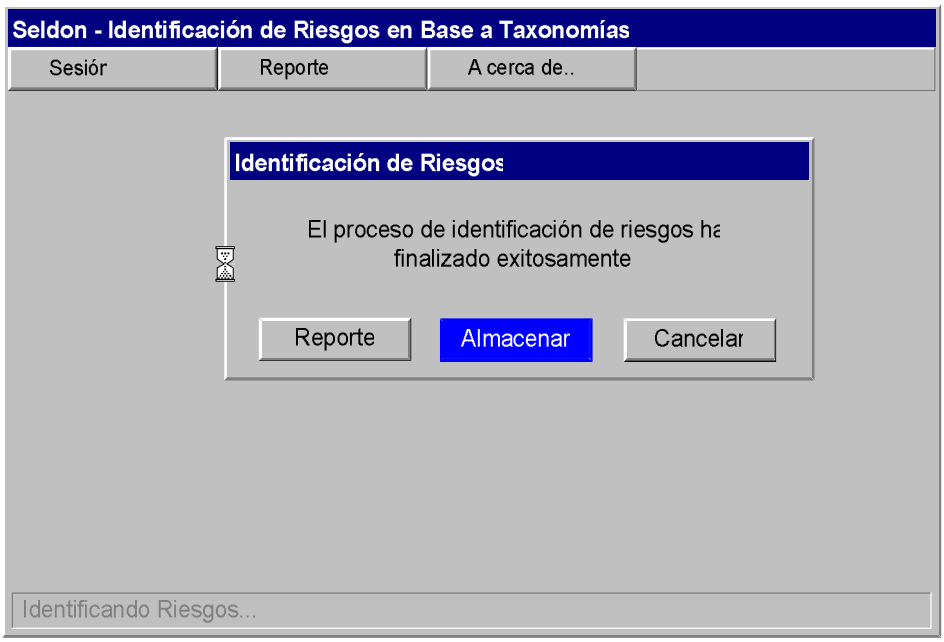


Figura 38 - Solicitando Guardado del Reporte de Riesgos - INT-008

Guardando el reporte de riesgos

Al emplear esta pantalla (véase **Figura 39 - Guardando Reporte de Riesgos**) el usuario puede guardar el reporte de riesgos generado por la aplicación.

Los casos de uso relacionados son:

- CU-004, Almacenando Resultados – Curso normal.

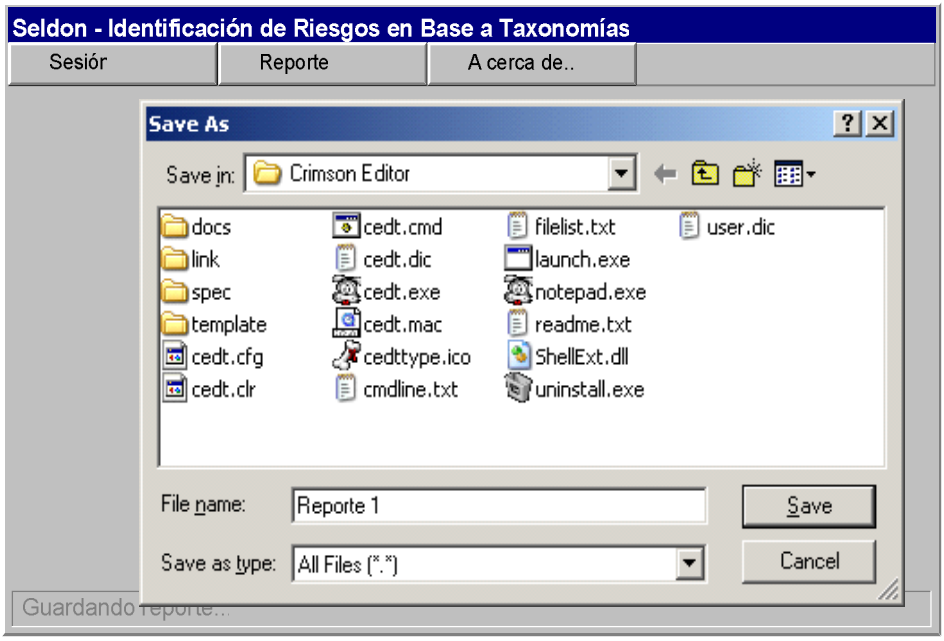


Figura 39 - Guardando Reporte de Riesgos - INT-009

Solicitando apertura de reporte de riesgos

Al emplear esta pantalla (véase **Figura 40 - Solicitando Apertura de Reporte de Riesgos**) el usuario puede solicitar la apertura de un reporte de riesgos al sistema.

Los casos de uso relacionados son:

- CU-005, Recuperando Resultados – Curso normal.

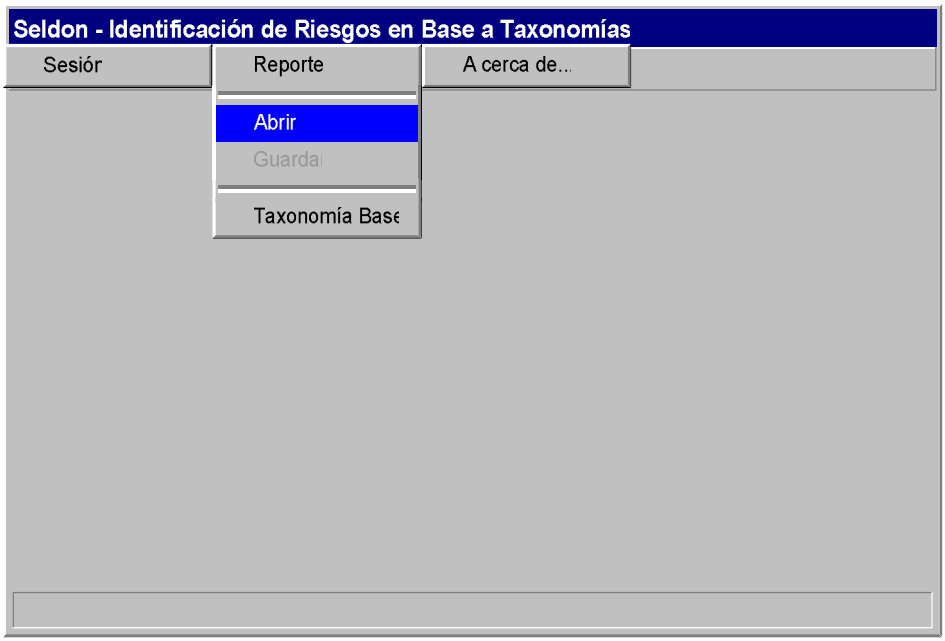


Figura 40 - Solicitando Apertura de Reporte de Riesgos - INT-010

Abriendo el reporte de riesgos

Al emplear esta pantalla (véase **Figura 41 - Abriendo Reporte de Riesgos**) el usuario puede abrir un reporte de riesgos que había sido previamente almacenado por la aplicación.

Los casos de uso relacionados son:

- CU-005, Recuperando Resultados – Curso normal.



Figura 41 - Abriendo Reporte de Riesgos - INT-011

Solicitando impresión del reporte de riesgos

Al emplear esta pantalla (véase **Figura 42 - Solicitando Impresión de Reporte de Riesgos**) el usuario puede solicitar la impresión de un reporte de riesgos al sistema.

Los casos de uso relacionados son:

- CU-002, Generando Reporte de Resultados – Curso alternativo.
- CU-003, Imprimiendo Reporte de Resultados – Curso normal.

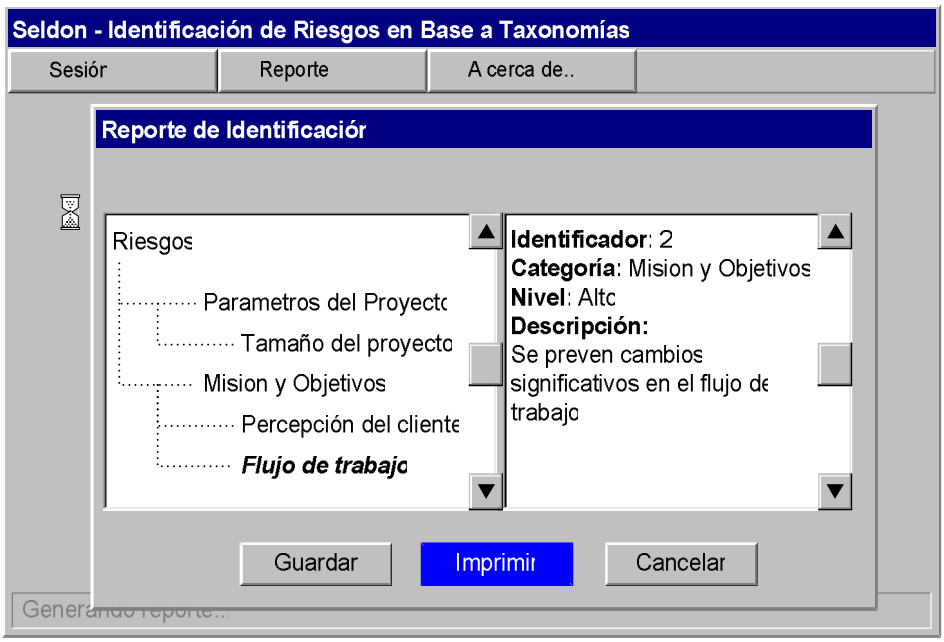


Figura 42 - Solicitando Impresión de Reporte de Riesgos - INT-012

Imprimiendo el reporte de riesgos

Al emplear esta pantalla (véase **Figura 43 - Imprimiendo Reporte de Riesgos**) el usuario puede imprimir el reporte de riesgos generado por la aplicación.

Los casos de uso relacionados son:

- CU-003, Imprimiendo Reporte de Resultados – Curso normal.

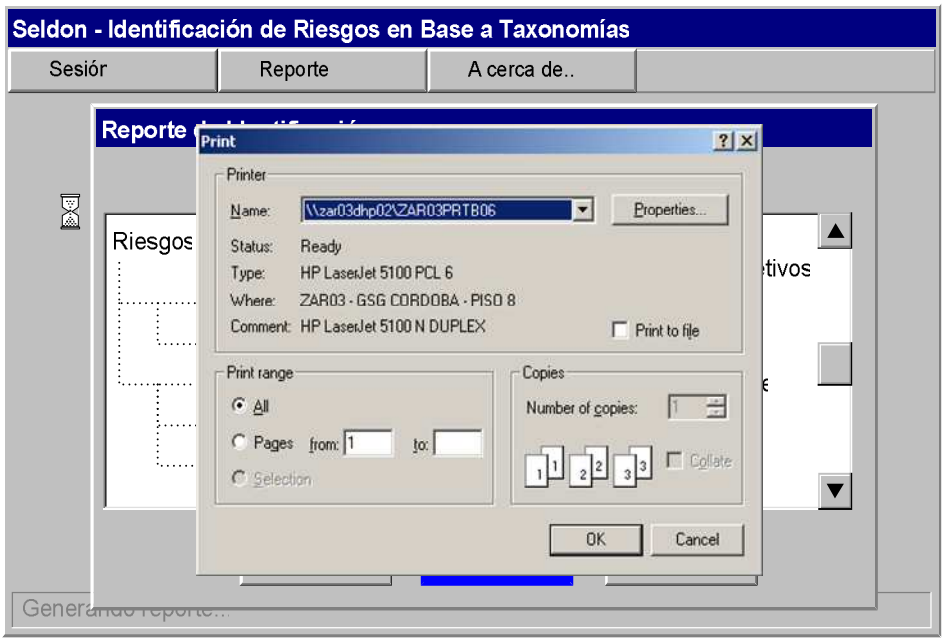


Figura 43 - Imprimiendo Reporte de Riesgos - INT-013

Solicitando generación del reporte de taxonomía base

Al emplear esta pantalla (véase **Figura 44 - Solicitando Generación de Reporte de Taxonomía Base**) el usuario puede solicitar la generación del reporte de la taxonomía empleada como base por el sistema.

Los casos de uso relacionados son:

- CU-006, Generando Reporte de Taxonomía – Curso normal.

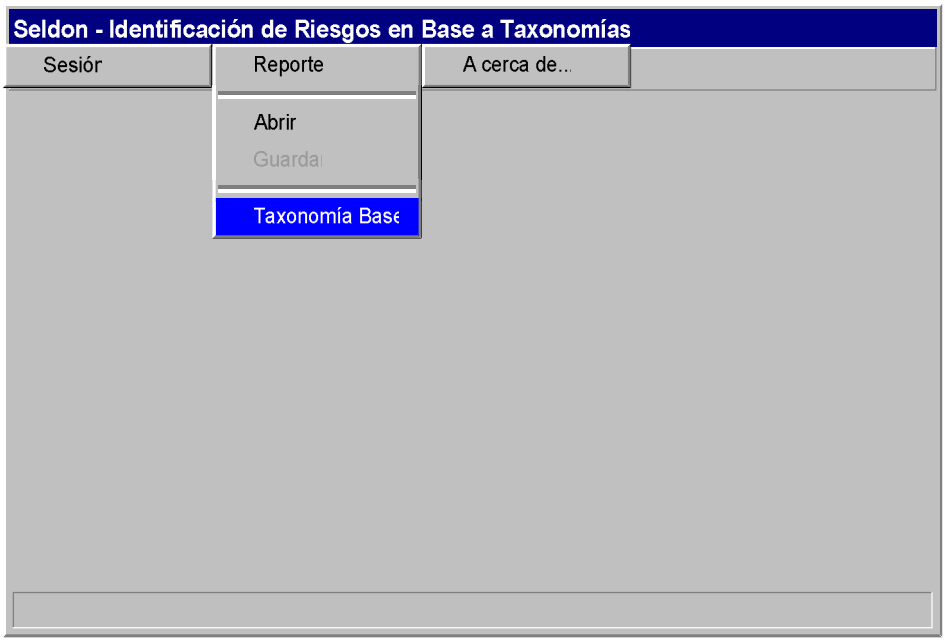


Figura 44 - Solicitando Generación de Reporte de Taxonomía Base - INT-014

Consultando el reporte de taxonomía base y riesgos

Al emplear esta pantalla (véase **Figura 45 - Consultando Reporte de Taxonomía Base y Riesgos**) el usuario puede consultar el reporte de la taxonomía base y los riesgos en ella contenidos.

Los casos de uso relacionados son:

- CU-006, Generando Reporte de Taxonomía – Curso normal.
- CU-007, Generando Reporte de Riesgos – Curso normal.

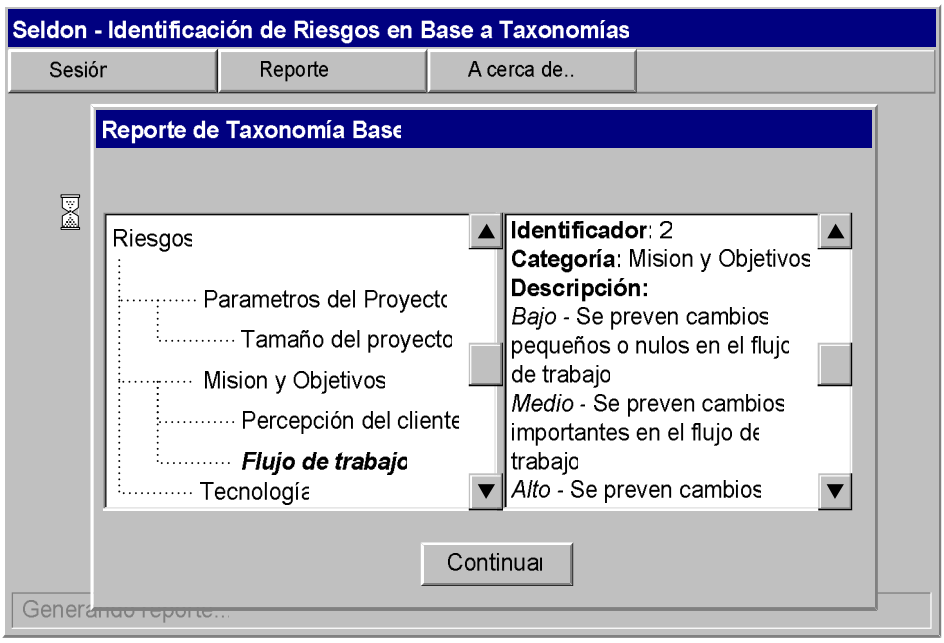


Figura 45 - Consultando Reporte de Taxonomía Base y Riesgos - INT-015

4.10.4. Formatos de Impresión

En esta sección se especifica la descripción del formato que deberán seguir cada uno de los reportes de la aplicación que el sistema permite imprimir.

Reporte de Resultados de Sesión de Identificación de Riesgos

La siguiente figura (véase **Figura 46 - Formato Reporte de Resultados de Sesión de Identificación**) define el formato de impresión para el reporte de los resultados determinados por el sistema para una sesión de identificación de riesgos.

Los casos de uso relacionados son:

- CU-001, Identificando Riesgos– Curso alternativo.
- CU-002, Generando Reporte de Resultados – Curso alternativo.
- CU-003, Imprimiendo Reporte de Resultados – Curso normal.

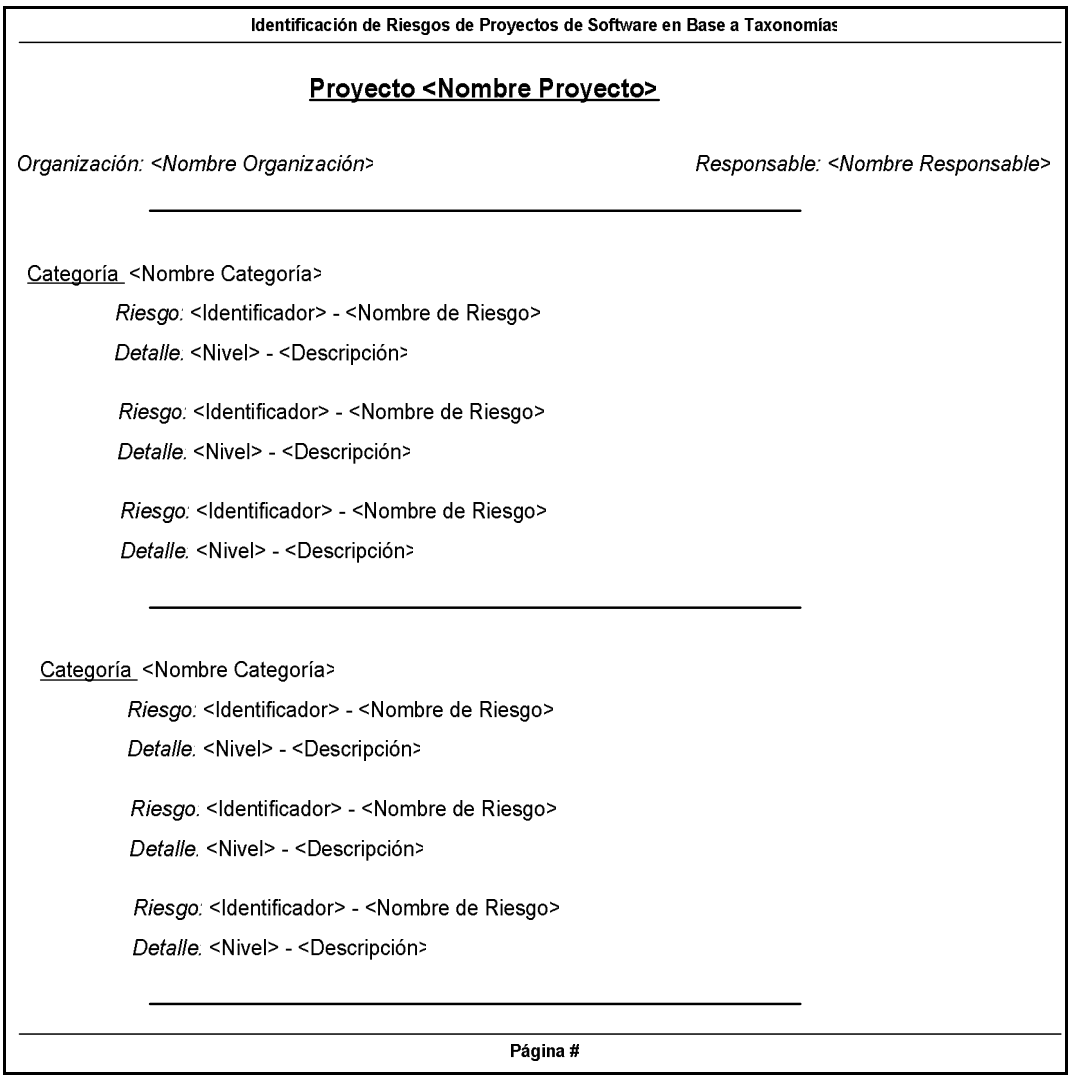


Figura 46 - Formato Reporte de Resultados de Sesión de Identificación - INT-016

4.11. Análisis de Consistencia

4.11.1. Verificación de los Modelos

Se verificó la calidad de los distintos modelos de forma separada con el propósito de garantizar su adecuación al problema a resolver y su seguimiento respecto a la técnica de análisis seleccionada.

Como resultado de la verificación, se efectuaron correcciones que fueron posteriormente comprobadas.

4.11.2. Resultado del análisis de consistencia

Se comprobó la coherencia entre los distintos modelos mediante un conjunto de matrices de trazabilidad. Referirse a Apéndice G: Matrices de Trazabilidad.

4.12. Plan de Pruebas

En esta sección se presenta el plan de pruebas del sistema que sirve como guía para la realización de las pruebas, y permite verificar que el sistema construido cumple las necesidades establecidas, con las debidas garantías de calidad.

El presente plan ha sido documentado de acuerdo al formato especificado en el estándar de Documentación de Pruebas de Software de la IEEE [IEEE 829,1983].

4.12.1. Introducción

El propósito de este plan es describir la estrategia, el alcance, la aproximación, la calendarización y los recursos necesarios para llevar a cabo las actividades de prueba de la aplicación denominada Seldon. Este plan identifica los ítems a ser probados, las características de los mismos, las tareas a ser realizadas y los responsables de las mismas.

4.12.2. Alcance

El alcance de este plan esta limitado a la definición de las actividades generales de prueba a realizarse para la aplicación Seldon.

4.12.3. Audiencia

La audiencia para el presente documento es:
- La administración del proyecto, con el propósito de proveer los recursos requeridos por el presente plan.
- El equipo de proyecto, con el propósito de desarrollar y probar el producto.

4.12.4. Ítems y características a probar

Aplicación Seldon

Se realizarán pruebas unitarias (aplicada a cada funcionalidad para verificar funciones, métodos, clases, etc.) y una prueba de sistema.

4.12.5. Características que no van a ser probadas.

- Aplicaciones y herramientas no desarrolladas por el equipo (por ejemplo, Sistema Operativo, Máquina Virtual de Java, soporte de impresión y almacenamiento, componentes de internacionalización y reglas).
- Cualidades de usabilidad del producto (por ejemplo, tiempo de respuesta de la aplicación e interfaz de usuario).
- Entorno de trabajo (por ejemplo, disponibilidad de red y tiempo de respuesta del equipo).

4.12.6. Aproximación

Actividades principales

Las principales actividades de prueba están asociadas a:
- Pruebas unitarias.
- Pruebas de sistema.
- Análisis y evaluación de la prueba.

Técnicas

Pruebas unitarias

Esta actividad debe cubrir cada una de las clases creadas durante la etapa de codificación. Toda entrada/salida debe ser probada y siempre que sea posible las combinaciones de las mismas. Las pruebas unitarias deberán desarrollarse de forma paralela a la codificación de la aplicación. Una vez que las actividades de pruebas unitarias han sido superadas exitosamente, es posible proseguir con las siguientes actividades de la prueba.

Pruebas de integración

Las pruebas de integración serán orientadas a validar la operatoria entre dos funciones o módulos, centrando el interés en las interfaces que entre ellos se definan. La utilización de patrones de diseño, el código autogenerado y el seguimiento de estándares de definición de interfces previenen errores y facilitan las tareas relacionadas con las pruebas de integración. Una vez que todos los casos de prueba han sido superados exitosamente entre todos los grupos de funciones o módulos seleccionados, la aplicación estará lista para ser probada como un sistema.

Pruebas de sistema

Las pruebas de sistema serán orientadas según la técnica de “caja negra”; con este método los casos de prueba y los resultados se determinan a partir de la especificación funcional del método de una clase. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener en cuenta la estructura lógica interna del software. Una vez que todos los casos de prueba han sido superados exitosamente, la aplicación estará lista para ser entregada.

Pruebas de aceptación

Estas pruebas deberán ser realizadas por la gerencia a cargo de la aplicación y el criterio de aceptación será el de cumplimentar los requisitos especificados en la sección **4.7**.**Definición del Sistema y Establecimiento de Requisitos** durnate la ejecución de proyectos piloto.

Otras Pruebas

Como parte de las tareas de prueba del sistema en diferentes etapas del desarrollo se planea la realización de:

- Pruebas de Tipo Alpha: Serán ejecutadas una vez que el código se encuentre mayormente terminado y contemple gran parte de la funcionalidad finalmente esperada y previo a la intervención de los usuarios, de manera informal y por parte de los desarrolladores.
- Pruebas de Compatibilidad: Se llevarán a cabo en diferente entornos y para validar el requerimiento de portabilidad **RNF-003** de (referirse a **4.7.3**.**Requerimientos específicos**).
- Pruebas de Instalación: Serán realizadas al finalizar las pruebas del sistema y luego de generar el instalador de la aplicación en un equipo que cumpla con los requisitos del sistema.

No se ha considerado en este plan la realización de Pruebas de Carga, Pruebas de Rendimiento ni Pruebas de Tensión debido a que no existen requerimientos específicos respecto a ninguno de estos aspectos, el producto a desarrollar constituye un prototipo de una herramienta, se espera la ejecución del sistema de forma “standalone” o independiente y se han fijado claramente los requisitos de hardware y software mínimos requeridos por el producto (garantizándose su ejecución en un ambiente capaz de cumplirlos).

Herramientas

Pruebas unitarias

Los resultados de las pruebas unitarias no serán almacenados en el documento de registro estándar y la ejecución de los mismos será manual. En caso de ser posible, dichas pruebas podrían ejecutarse con JUnit.

Pruebas de integración

Los resultados de las pruebas de integración no serán almacenados en el documento de registro estándar y la ejecución de los mismos será manual. Se incluirán pruebas de tipo estáticas (revisiones formales de pares e inspecciones) a las interfaces definidas entre los componentes (tarea que será iniciada durante el desarrollo del diseño).

Pruebas de sistema

Los resultados de las pruebas deberán ser almacenados en el documento de registro estándar y la ejecución de las mismas será manual.

Amplitud de las pruebas

La amplitud y criterio de completitud a emplear se basará en la cobertura realizada sobre la funcionalidad requerida.

Reporte de fallas de las pruebas

Las fallas serán identificadas durante el análisis y evaluación de los resultados de la ejecución de las pruebas. El reporte de pruebas será entregado como parte del presente documento.

Trazabilidad de requerimientos

Referirse a **Apéndice G: Matrices de Trazabilidad**.

Disponibilidad de ítems de prueba

Las actividades de prueba no podrán comenzar hasta que todas las unidades de prueba se encuentren disponibles.

Disponibilidad de recursos para las pruebas

El siguiente listado corresponde a los recursos necesarios para llevar a cabo las actividades de prueba:

- 1 PC.
- 1 Impresora.

Fechas límites

Referirse a **Apéndice B: Estimaciones y Calendario**.

4.12.7. Criterio de Paso/Falla

Ítems

El criterio a emplear sobre cada uno de los ítems es el siguiente:
- Paso: Todas las pruebas realizadas sobre el ítem fueron exitosas.
- Fallo: Al menos una de las pruebas realizadas no fue exitosa.

Aplicación

El criterio a emplear sobre las pruebas de la aplicación es el siguiente:
- Exitosa: Todas las pruebas fueron realizados y no se encontraron defectos de severidad 1, 2 o 3. (Véase **Tabla 19 - Severidad de Defectos**).
- Fallida: Al menos un defecto de severidad 1, 2 o 3 fue encontrado. (Véase **Tabla 19 - Severidad de Defectos**).

SEVERIDAD	SIGNIFICADO
1	Sistema detenido
2	Fallas de funcionalidad
3	Una solución alternativa puede aplicarse
4	Error de documentación/ayuda
5	Cambios y mejoras

Tabla 19 - Severidad de Defectos

4.12.8. Criterio de suspensión y reiniciación de pruebas

Las actividades de prueba deberían ser suspendidas si ocurre alguna de las situaciones:
- Se encuentra un problema en una prueba que impide la realización de más pruebas.
- Se encuentra un problema en un ítem que impide la realización de más pruebas.

Cuando se encuentre un problema y el mismo no impida la realización de más pruebas, las mismas deben continuar.

Una vez solucionados los problemas encontrados, las pruebas deben realizarse nuevamente.

4.12.9. Artefactos de las pruebas

La siguiente lista resume los artefactos a generarse durante la realización de las pruebas:

- Plan de pruebas y Documento de diseño de la prueba.
- Especificación de los casos de prueba y Especificación del procedimiento de prueba.
- Informe de los casos de prueba.
- Informe de la prueba.

4.12.10. Actividades de prueba

Para la realización de cada uno de los ciclos de prueba las siguientes actividades pueden ser requeridas:

- Actualizar el plan de pruebas y documento de diseño.
- Crear o actualizar casos y procedimientos de prueba.
- Ejecutar las pruebas y realizar el análisis, evaluación e informe de las mismas.
- Llevar a cabo la prueba de aceptación.

Procedimiento de pruebas

El siguiente procedimiento (véase **Figura 47 - Procedimiento de Pruebas**) será seguido durante la ejecución de los casos de prueba para la aplicación Seldon:

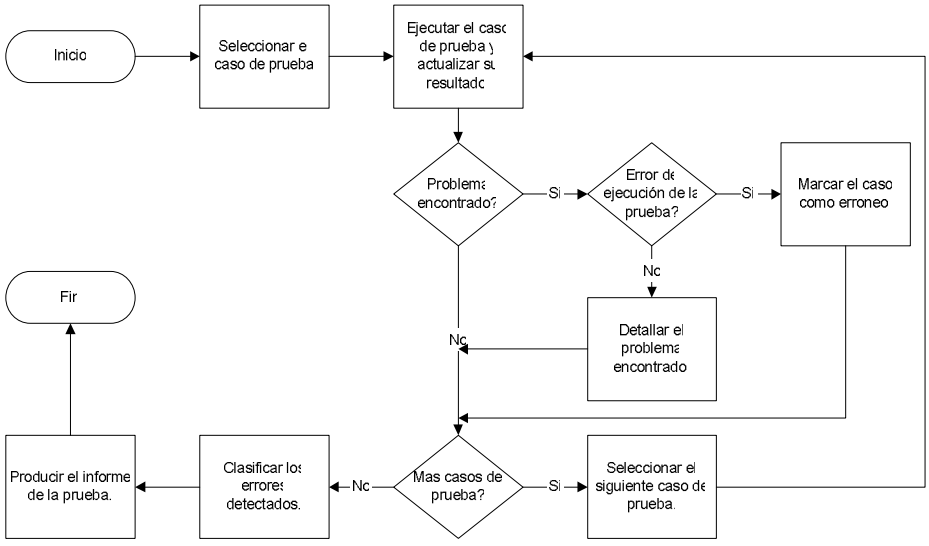


Figura 47 - Procedimiento de Pruebas

4.12.11. Necesidades de ambiente

Existen dos entornos principales relacionados con las pruebas:

- Entorno local de pruebas.

- Entorno del cliente.

Ambos deben ser coincidentes y las propiedades mínimas requeridas para los mismos se especifican a continuación.

Hardware

PC Pentium 166 MHz o superior 32 MB de RAM (se recomienda 64 MB) con al menos 50 MB libres en el disco duro.

Software

JDK 1.4

Seguridad

Permisos de lectura/escritura sobre el directorio donde se ejecutará la aplicación.

Modo de uso

Referirse a Definición del Sistema y Establecimiento de Requisitos.

Certificación de entorno

No existen necesidades concretas de certificación de entorno.

4.12.12. Responsabilidades

Ingeniero de pruebas

- Diseñar, preparar y realizar las actividades de prueba.
- Gestionar el entorno local de prueba.
- Recolectar y analizar los resultados obtenidos de la realización de las actividades de prueba.

Líder de desarrollo

- Proveer de los ítems a probar.
- Revisar el plan de pruebas.
- Revisar las actividades de prueba.
- Revisar los resultados de las pruebas realizadas.

Líder de proyecto

- Revisar el plan de pruebas.
- Revisar los resultados de las pruebas realizadas.
- Coordinar las actividades de prueba.

Especialista en calidad

- Revisar el plan de pruebas.
- Revisar las actividades de prueba.

Cliente

- Realizar las pruebas de aceptación.

4.12.13. Necesidades de personal, calendario y entrenamiento

Para llevar a cabo las actividades de prueba será necesario la asignación de tiempo completo de un ingeniero de pruebas.

Para mayores detalles respecto a calendarización y asignaciones referirse a **Apéndice B: Estimaciones y Calendario**.

No se han detectado necesidades de entrenamiento para llevar a cabo las actividades de prueba.

4.12.14. Riesgos y contingencias de pruebas

- Riesgo: Diferencias entre el entorno de desarrollo y de pruebas pruebas y el entorno del cliente.
- Contingencia: Especificar como parte de la documentación a entregar los requerimientos de software y hardware y colaborar con el usuario en la verificación/instalación de los mismos.

- Riesgo: Falta de conocimiento del usuario de la herramienta para ejecutar la aplicación.
- Contingencia: Especificar como parte de la documentación a entregar los pasos para ejecutar el sistema y colaborar con el usuario en la realización de los mismos.

4.13. Aprobación del Análisis

Se confirmaron y validaron con los usuarios los siguientes modelos:

- Catálogo de requisitos
- Modelo de casos de uso
- Prototipos de interfaz

Los usuarios confirmaron y validaron los requerimientos planteados para el sistema y la manera en la que los mismos se cubren mediante la funcionalidad expresada en los casos de uso como así también con la interacción planeada para el sistema a través de los prototipos de interfaz.

Capítulo

5

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Diseño del Sistema de Información

5. Diseño del Sistema de Información (DSI)

En el presente capítulo se describe y desarrolla el diseño del sistema según las actividades propuestas por la metodología utilizada.

5.1. Introducción

En la fase de Diseño se crea una solución a nivel lógico para satisfacer los requisitos, basándose en el conocimiento reunido en la etapa de Análisis.

Se descompone el sistema en módulos y se definen sus interfaces y sus flujos de datos y de control. Además, se especifican formatos, entradas, salidas, algoritmos, excepciones y cualquier otro componente con un alto nivel de detalle de forma tal de dar soporte a las actividades de codificación y pruebas unitarias.

5.2. El diseño del sistema en MÉTRICA Versión 3

El propósito del Diseño del Sistema de Información (DSI) es obtener la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de software. A partir de dicha información, se generan todas las especificaciones de construcción relativas al propio sistema, así como la especificación técnica del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial, éstos últimos cuando proceda. [Métrica V3, 2000]

El diseño de la arquitectura del sistema dependerá en gran medida de las características de la instalación.

Este proceso consta de un primer bloque de actividades, que se realizan en paralelo, y cuyo objetivo es obtener el diseño de detalle del sistema de software que comprende la partición física de la aplicación, independiente de un entorno tecnológico concreto, la organización en subsistemas de diseño, la especificación del entorno tecnológico sobre el que se despliegan dichos subsistemas y la definición de los requisitos de operación, administración del sistema, seguridad y control de acceso.

Al igual que en el proceso de Análisis del Sistema de Información (ASI), antes de proceder a la especificación de los componentes, se realiza una verificación y validación, con objeto de analizar la consistencia entre los distintos modelos y formalizar la aceptación del diseño de la arquitectura del sistema.

Un segundo bloque de actividades complementa el diseño del sistema de software, en el que se generan todas las especificaciones necesarias para la construcción del sistema.

Las actividades que engloba este proceso se recogen en la siguiente figura (véase **Figura 48 - Tareas del DSI**), en la que se indican las tareas que pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores:

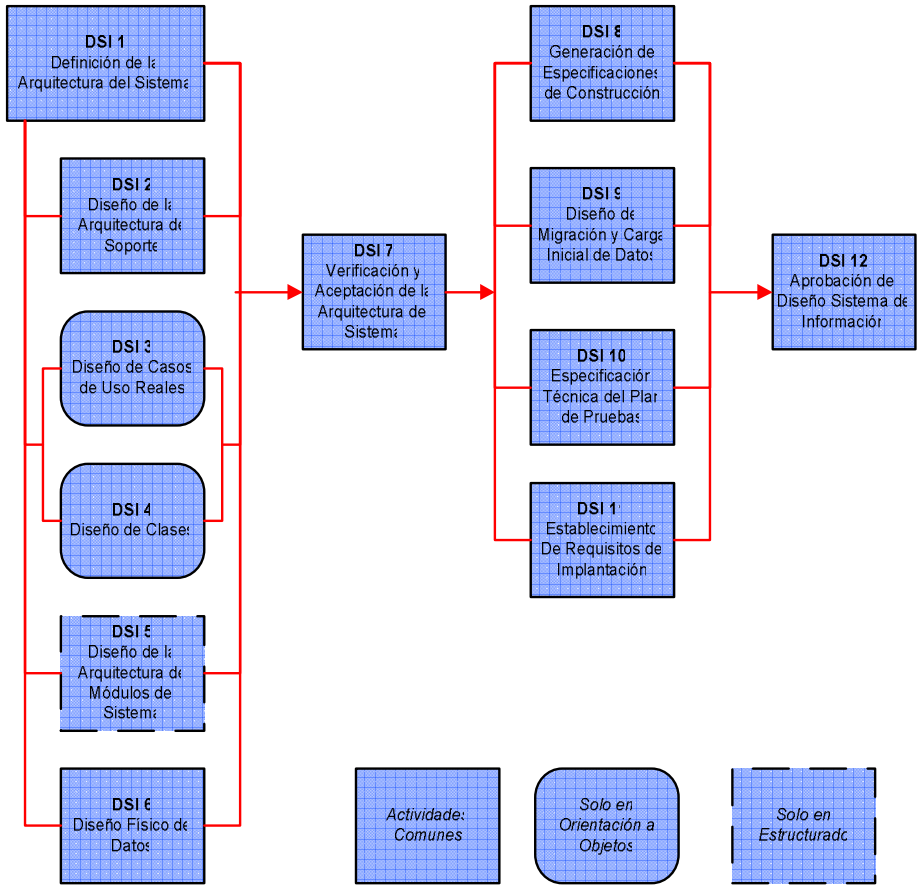


Figura 48 - Tareas del DSI

Dada las características del sistema a realizar, y considerando que la metodología centra su proceso en desarrollos de tipo Estructurado y Orientado a Objetos, se realizarán como parte del presente proyecto las actividades DSI3 y DSI4.

5.3. Definición de la Arquitectura del Sistema

5.3.1. Definición de Niveles de la Arquitectura

Se describen a continuación los niveles de la arquitectura software, mediante la definición de las principales particiones físicas del sistema representadas como nodos y sus comunicaciones (donde cada nodo corresponderá a una partición física o parte significativa del sistema construído, con características propias de ejecución o función, e incluso de diseño y construcción).

La definición de los niveles de la arquitectura del sistema se documenta mediante un Modelo de Despliegue y Componentes de UML [Booch, et al., 2000]. Este modelo muestra la arquitectura del sistema desde el punto de vista físico, por lo cual se contemplan los siguientes elementos:

- Nodos de procesamiento.
- Dispositivos de hardware.
- Comunicaciones entre nodos y con dispositivos.
- Componentes de software empaquetados en unidades instalables.

La siguiente figura (véase **Figura 49 - Niveles de la Arquitectura**) muestra la definición de los niveles de la arquitectura y sus componentes:

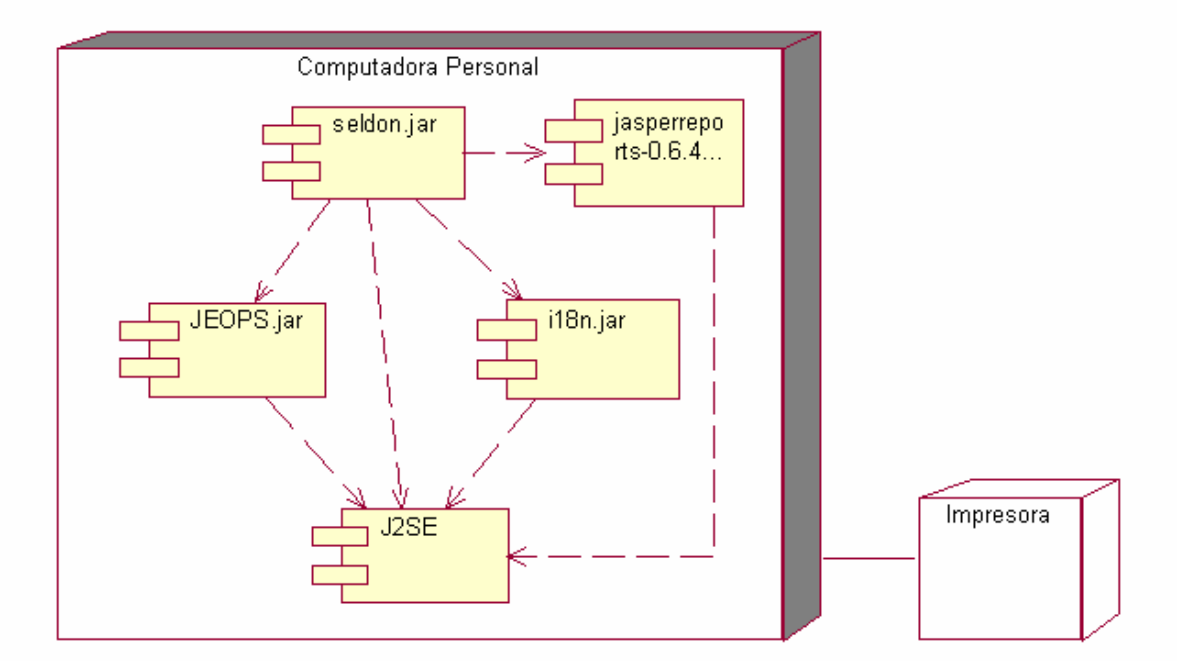


Figura 49 - Niveles de la Arquitectura

Se describen a continuación cada uno de los elementos del diagrama de despliegue de la aplicación:

Computadora Personal
Dada las características del sistema, es el único nodo de procesamiento empleado para hacer posible la ejecución del sistema construido.
Sus requisitos mínimos son:

- *Procesador:* Pentium III o equivalente
- *Memoria:* 64 MBytes
- *Espacio en disco:* 10 MBytes
- *Software:* Ambiente de ejecución de la plataforma J2SE versión 1.4.2.

Impresora
Con el propósito de imprimir los reportes generados por la aplicación, la instalación deberá contar con una impresora. La comunicación entre el sistema y la impresora se realiza mediante el soporte otorgado por la plataforma Java [J2SE, 2004].

Los componentes identificados para el sistema son:

J2SE
Este componente representa el entorno de ejecución de la plataforma Java y constituye la base fundamental para la ejecución del sistema construido ya que brinda un conjunto de librerías estándar utilizadas por la aplicación y el soporte de una máquina virtual que posibilita a la aplicación Java su ejecución en una máquina real (Computadora Personal).

seldon.jar
Este componente representa el sistema completo (clases, archivos de configuración y taxonomía de riesgos).

jasperreports-0.6.4.jar
Este componente representa la utilidad de generación de reportes JasperReports [JasperReports, 2005] empleada para la generación e impresión de los reportes producidos por la aplicación.

i18n.jar
Este componente representa el conjunto de recursos que dan soporte a las características de internacionalización y localización para el sistema.

JEOPS.jar
Este componente representa el conjunto de recursos que brindan el soporte de reglas [JEOPS, 2005] empleado por el sistema construido.

5.3.2. Identificación de Requisitos de Diseño y Construcción

Se indican a continuación (véase **Tabla 20 - Requisitos de Diseño y Construcción**) los requisitos de diseño y construcción, es decir, aquellos que

están directamente relacionados con la adopción o diseño de una arquitectura o infraestructura concreta:

IDENTIFICADOR	DESCRIPCIÓN	VOLATILIDAD	PRIORIDAD
RNF-005	Las impresiones requeridas al sistema serán realizadas por una impresora local.	Baja	Media
RNF-006	Los repositorios sobre los que se basará el sistema para operar serán locales.	Baja	Alta

Tabla 20 - Requisitos de Diseño y Construcción

5.3.3. Especificación de Excepciones

Se indican a continuación (véase **Tabla 21 - EXC-001**, **Tabla 22 - EXC-002**, **Tabla 23 - EXC-003**, **Tabla 24 - EXC-004** y **Tabla 25 - EXC-005**) los comportamientos no habituales en el sistema, que reflejan situaciones anómalas o secundarias en el funcionamiento y ejecución de la aplicación producto del desarrollo:

IDENTIFICADOR	EXC-001
TIPO	Comunicación
DESCRIPCIÓN	El sistema intenta enviar datos a la impresora local pero ésta no responde.
PRECONDICIONES	La computadora personal no está conectada con la impresora.
COMPONENTES	Computadora Personal, Impresora
RESPUESTA	El sistema no puede conectarse con la impresora - Verifique que se encuentre encendida y examine la conexión, luego vuelva a intentarlo.

Tabla 21 - EXC-001

IDENTIFICADOR	EXC-002
TIPO	Comunicación
DESCRIPCIÓN	El sistema intenta acceder a un repositorio no disponible.
PRECONDICIONES	El repositorio de datos no se encuentra disponible.
COMPONENTES	Computadora Personal
RESPUESTA	El sistema no puede acceder a <Recurso> - Verifique que el mismo exista y que pueda ser accedido.

Tabla 22 - EXC-002

IDENTIFICADOR	EXC-003
TIPO	Validación
DESCRIPCIÓN	Se pretende cargar un dato cuyo valor se encuentra fuera del rango establecido en la validación o no es un valor válido para el campo.
PRECONDICIONES	Selección de una opción que requiere del ingreso de un valor o fecha dentro de un rango de valores preestablecidos.
COMPONENTES	Computadora Personal
RESPUESTA	El valor ingresado no es válido.

Tabla 23 - EXC-003

IDENTIFICADOR	EXC-004
TIPO	Almacenamiento
DESCRIPCIÓN	Se pretende almacenar un archivo con el mismo identificador que uno existente.
PRECONDICIONES	Archivo existente y requerimiento de almacenamiento realizado.
COMPONENTES	Computadora Personal
RESPUESTA	El archivo ya existe, desea reemplazarlo?

Tabla 24 - EXC-004

IDENTIFICADOR	EXC-005
TIPO	Almacenamiento
DESCRIPCIÓN	Se pretende almacenar un archivo pero no existe lugar para hacerlo.
PRECONDICIONES	Dispositivo de almacenamiento completo.
COMPONENTES	Computadora Personal
RESPUESTA	No hay lugar disponible para almacenar la información. Libere espacio y vuelva a intentarlo.

Tabla 25 - EXC-005

5.3.4. Especificación de Estándares y Normas de Diseño y Construcción

No se han identificado nuevos estándares técnicos y de nomenclatura, normas ni recomendaciones aplicables al diseño o la construcción de la aplicación.

5.3.5. Identificación de Subsistemas de Diseño

En esta sección se divide de forma lógica el sistema en subsistemas de diseño, con el fin de reducir la complejidad y facilitar el mantenimiento.

El siguiente diagrama (véase **Figura 50 - Subsistemas de Diseño**) muestra la división lógica realizada para el sistema a construir como resultado final de la Tesis de Magíster.

Los subsistemas resultantes han sido calificados como *específicos* o *de soporte*. Los subsistemas específicos contemplan las funcionalidades propias de la herramienta, mientras que los de soporte cubren servicios comunes, proporcionando un acceso transparente a los distintos recursos.

Es importante considerar que el patrón de diseño empleado como base para la construcción de la aplicación será el denominado Modelo-Vista-Controladores o MVC (en inglés, *Model View Controller*) [Hunt, 2001].

Las características principales de MVC son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. El Modelo es el objeto que representa los datos del programa; maneja los datos y controla todas sus transformaciones y no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos; es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo. La Vista es el conjunto de objetos que maneja la presentación visual de los datos representados por el Modelo; genera una representación visual del Modelo, muestra los datos al usuario e interactúa con el Modelo a través de una referencia al propio Modelo. El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo; cuando se realiza algún cambio en la información del Modelo o por alteraciones de la Vista, se acciona e interactúa con el Modelo a través de una referencia al propio Modelo.

Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos separadamente.
- Hay una interfaz muy bien definida por lo cual podrá reemplazarse el Modelo, la Vista o el Controlador sin dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución y no en tiempo de compilación lo cual brinda mayor flexibilidad al sistema.

Se encuentran identificados de manera diferente aquellos paquetes existentes y empleados por la aplicación (i18n, JasperReports, JEOPS) de aquellos que serán desarrollados como parte de la Tesis de Magíster (mvc, model, view, controller y persistence).

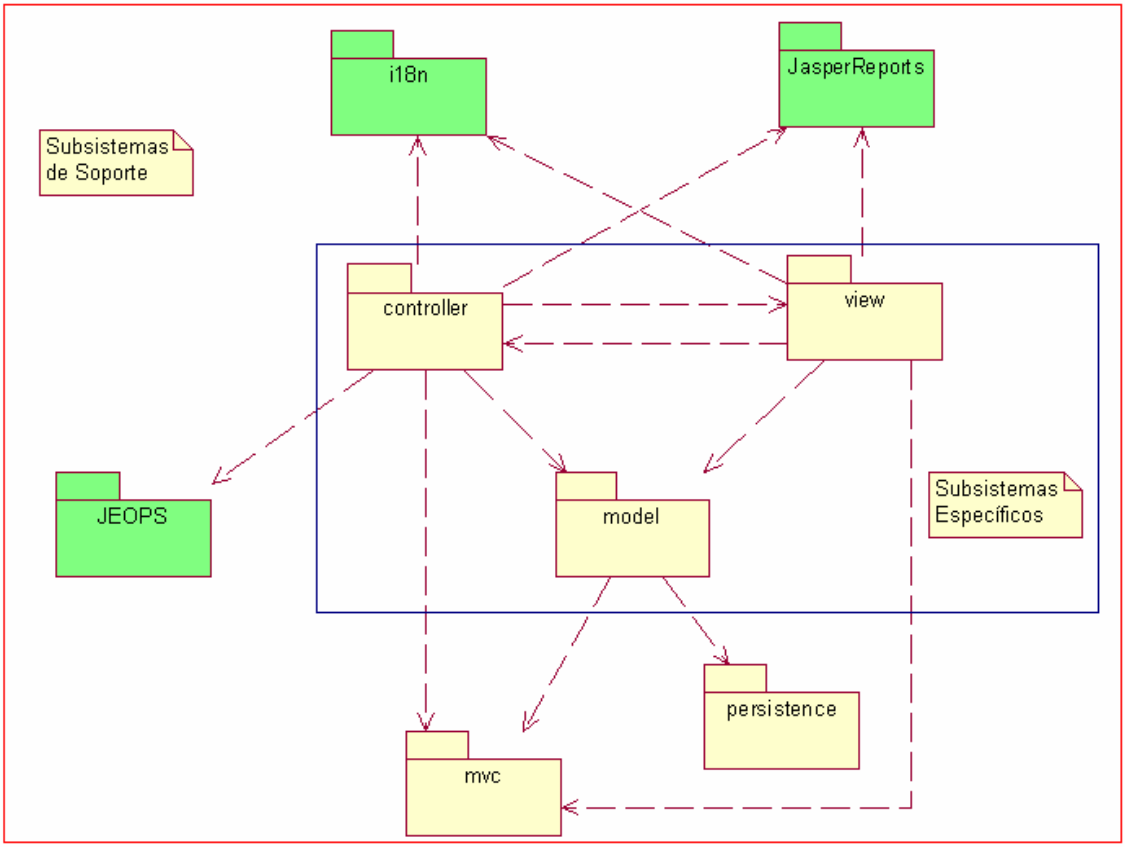


Figura 50 - Subsistemas de Diseño

Se describe a continuación brevemente el propósito de cada uno de los paquetes identificados:

De Soporte

i18n
Contiene el conjunto de clases e interfaces que brindan el soporte a las características de internacionalización y localización para el sistema. Brinda una interfaz mediante la cual es posible recuperar mensajes de texto a partir de códigos preestablecidos y formatear fechas y números según preferencias localmente configuradas.

JasperReports
Contiene el conjunto de clases e interfaces que posibilitan la generación e impresión de los reportes producidos por la aplicación.

JEOPS
Contiene el conjunto de clases e interfaces que brindan el soporte de gestión de reglas empleado por el sistema construido durante el proceso de identificación de riesgos.

persistente
Contiene el conjunto de clases e interfaces que permiten lograr el almacenamiento y la recuperación de elementos del sistema empleando el sistema de archivos.

mvc
Contiene el conjunto de clases e interfaces que dan soporte al patrón de diseño MVC (que será efectivamente implementado por las clases de los paquetes model, view y controller).

Específicos

model
Contiene un conjunto de clases e interfaces del modelo definidas según el patrón de diseño MVC.

view
Contiene un conjunto de clases e interfaces de vista definidas según el patrón de diseño MVC.

controller
Contiene un conjunto de clases e interfaces de control definidas según el patrón de diseño MVC.

5.3.6. Especificación del Entorno Tecnológico

Los elementos de infraestructura técnica que dan soporte al sistema ha construir son los siguientes:

Hardware:
- Computadora personal Pentium III o superior.
- 64 Mbytes de memoria RAM mínimo.
- Espacio mínimo en disco de 10 MBytes.

Software:
- Plataforma J2SE 1.4.2.
- Sistema operativo con soporte para la plataforma J2SE.

El sistema no requiere de una infraestructura de comunicaciones, ya que se trata de una herramienta del tipo independiente sin necesidad de acceder a recursos remotos para operar.

5.3.7. Especificación de Requisitos de Operación y Seguridad

No se han identificado requisitos no comprometer el correcto funcionamiento del sistema y garantizar el cumplimiento de los niveles de servicios que exigirá el sistema en cuanto a la gestión de operaciones (procesos por lotes, seguridad, comunicaciones, etc.).

5.4. Diseño Físico de Datos

En esta sección se definen las estructuras físicas de datos que utilizará el sistema y que ha sido definida considerando el modelo de clases desarrollado y en busca de lograr una mayor eficiencia en el tratamiento de los datos considerando las características específicas del sistema, los requisitos establecidos para el mismo y las particularidades del entorno tecnológico a emplear.

Se ha identificado la necesidad de generar un diseño físico de datos para la taxonomía de riesgos del sistema y para los resultados obtenidos de las sesiones de identificación de riesgos realizadas con la herramienta. Considerando los requisitos del sistema, la tecnología a utilizar y la existencia de componentes preexistentes, se ha optado por utilizar un archivo en formato XML con tal propósito; es decir, que la persistencia de los datos será lograda a través de este mecanismo y no de la utilización de bases de datos.

XML es un lenguaje que contiene un conjunto de reglas que permiten definir etiquetas semánticas para organizar un documento en diferentes partes; es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

Al emplear XML como tecnología, es necesario realizar una definición formal de un tipo de documento y, a la vez, especificar la estructura lógica de cada documento como así también la definición tanto de los elementos de una página como de sus atributos. Dicha definición se realiza a través de uno o más archivos denominados DTD (definición del tipo de documento, en inglés, *Document Type Definition*).

Se detalla a continuación el DTD empleado por la aplicación Seldon para representar tanto la taxonomía de riesgos como los resultados obtenidos durante las sesiones de identificación (que constituirán, en la práctica, un subconjunto de la información contenida en la taxonomía base antes mencionada):

```
<!-- DTD for Seldon     - S. Maniasi - 2005.03.04 -->

<!ELEMENT TAXO     (TITLE, CAT+)>
<!ELEMENT TITLE    (#PCDATA)>
<!ELEMENT CAT      (TITLE, (CAT | RISK)+)>
<!ELEMENT RISK     (ID, TITLE, (APPLY)*, (DESC)+, NOTE, (RANK)+)>
<!ELEMENT APPLY    (#PCDATA)>
<!ELEMENT DESC     ((NOTE, STITLE)+, (LINE)+)>
<!ELEMENT ID       (#PCDATA)>
<!ELEMENT STITLE   (#PCDATA)>
<!ELEMENT LINE     (#PCDATA)*>
<!ELEMENT NOTE     (#PCDATA)>
<!ELEMENT RANK     (TYPE, VALUE)>
<!ELEMENT TYPE     (#PCDATA)>
<!ELEMENT VALUE    (#PCDATA)>
```

Donde:

- APPLY puede tomar los valores:
  - CUS = Riesgos válidos sólo cuando el proyecto cuenta con un "cliente/usuario" directo.
  - PRG = Riesgos válidos sólo cuando el proyecto forma parte de un programa.
  - BUD = Riesgos válidos sólo cuando el proyecto realiza administración de presupuesto.
  - INT = Riesgos válidos sólo cuando el producto del proyecto será integrado con HW/SW.
  - QUA = Riesgos válidos sólo cuando el producto del proyecto tiene requerimientos de calidad.
  - INS = Riesgos válidos sólo cuando el proyecto realiza instalación del producto.
  - CON = Riesgos válidos sólo cuando el proyecto realiza subcontratacion.
  - MAN = Riesgos válidos sólo cuando el proyecto realiza mantenimiento.
- TYPE puede tomar los valores:
  - SAI - Sistemas de Administración de Información
  - SSB - Software de Sistemas Base
  - SCM - Software Comercial de Mercado
  - SE - Sistemas Especiales
  - RS - Reingeniería de Sistemas
- VALUE puede tomar los valores:
  - HIGH
  - MEDIUM
  - LOW
  - NA (Not Applicable)
  - ND (Not Determined)

El siguiente es un ejemplo de la instanciación de la estructura de datos antes mencionada:

```
<?xml version="1.0" ?>
<!DOCTYPE TAXO (View Source for full doctype...)>
<TAXO>
    <TITLE>Taxonomina de Riesgos</TITLE>
    <CAT>
      <TITLE>Mision y Objetivos</TITLE>
      <RISK>
          <ID>1</ID>
          <TITLE>El proyecto se adecua a la organizacion
              proveedora.</TITLE>
          <DESC>
            <NOTE>Indicador de Alto Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>El proyecto no se relaciona con la mision y objetivos
            de la organizacion proveedora.</LINE>
          </DESC>
          <DESC>
```

```
            <NOTE>Indicador de Mediano Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>El proyecto impacta indirectamente en uno o mas
            objetivos definidos por la organizacion proveedora.</LINE>
          </DESC>
          <DESC>
            <NOTE>Indicador de Bajo Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>El proyecto soporta directamente la mision y objetivos
            de la organizacion proveedora.</LINE>
          </DESC>
          <NOTE>Categoria: Mision y Objetivos</NOTE>
          <RANK>
            <TYPE>SAI</TYPE>
            <VALUE>MEDIUM</VALUE>
          </RANK>
          <RANK>
            <TYPE>SSB</TYPE>
            <VALUE>LOW</VALUE>
          </RANK>
      </RISK>
      <RISK>
          <ID>2</ID>
          <TITLE>Flujo de trabajo.</TITLE>
          <DESC>
            <NOTE>Indicador de Alto Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Cambios basicos o nulos en el flujo de trabajo.</LINE>
          </DESC>
          <DESC>
            <NOTE>Indicador de Mediano Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Cambios importantes en el flujo de trabajo.</LINE>
          </DESC>
          <DESC>
            <NOTE>Indicador de Bajo Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Cambios significativos en el flujo de trabajo.</LINE>
          </DESC>
          <NOTE>Categoria: Mision y Objetivos</NOTE>
          <RANK>
            <TYPE>SAI</TYPE>
            <VALUE>MEDIUM</VALUE>
          </RANK>
          <RANK>
            <TYPE>SSB</TYPE>
            <VALUE>LOW</VALUE>
          </RANK>
      </RISK>
  </CAT>
  <CAT>
      <TITLE>Ingenieria del Producto</TITLE>
          <CAT>
            <TITLE>Requerimientos</TITLE>
            <RISK>
                <ID>38</ID>
                <TITLE>Estabilidad.</TITLE>
```

```
        <DESC>
            <NOTE>Indicador de Alto Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Muy pocos o ninguno de los requerimientos se
            modifican durante el desarrollo del sistema.</LINE>
        </DESC>
        <DESC>
            <NOTE>Indicador de Mediano Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Algunos de los requerimientos se modifican
            durante el desarrollo del sistema.</LINE>
        </DESC>
        <DESC>
            <NOTE>Indicador de Bajo Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Muchos de los requerimientos se modifican
            durante el desarrollo del sistema.</LINE>
        </DESC>
        <NOTE>Categoria: Requerimientos</NOTE>
        <RANK>
            <TYPE>SAI</TYPE>
            <VALUE>MEDIUM</VALUE>
        </RANK>
        <RANK>
            <TYPE>SSB</TYPE>
            <VALUE>LOW</VALUE>
        </RANK>
    </RISK>
</CAT>
<CAT>
    <TITLE>Codigo y Pruebas Unitarias</TITLE>
        <RISK>
        <ID>50</ID>
        <TITLE>Pruebas unitarias.</TITLE>
            <DESC>
            <NOTE>Indicador de Alto Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Las pruebas unitarias han sido estimadas y
            planificadas pero no se ha definido el nivel de detalle
            para las mismas.</LINE>
        </DESC>
        <DESC>
            <NOTE>Indicador de Mediano Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Las pruebas unitarias han sido estimadas y
            planificadas pero no seran documentadas.</LINE>
        </DESC>
        <DESC>
            <NOTE>Indicador de Bajo Riesgo</NOTE>
            <STITLE>Descripcion</STITLE>
            <LINE>Las pruebas unitarias no han sido estimadas ni
            planificadas.</LINE>
        </DESC>
        <NOTE>Categoria: Codigo y Pruebas Unitarias</NOTE>
        <RANK>
            <TYPE>SAI</TYPE>
            <VALUE>MEDIUM</VALUE>
```

```
                </RANK>
                <RANK>
                    <TYPE>SSB</TYPE>
                    <VALUE>LOW</VALUE>
                </RANK>
            </RISK>
        </CAT>
    </CAT>
</TAXO>
```

La siguiente pantalla (véase **Figura 51 - Instanciación de la Estructura de Datos para la Taxonomía de Riesgos**) ejemplifica la manera en la cual la aplicación mostrará la instanciación antes detallada:

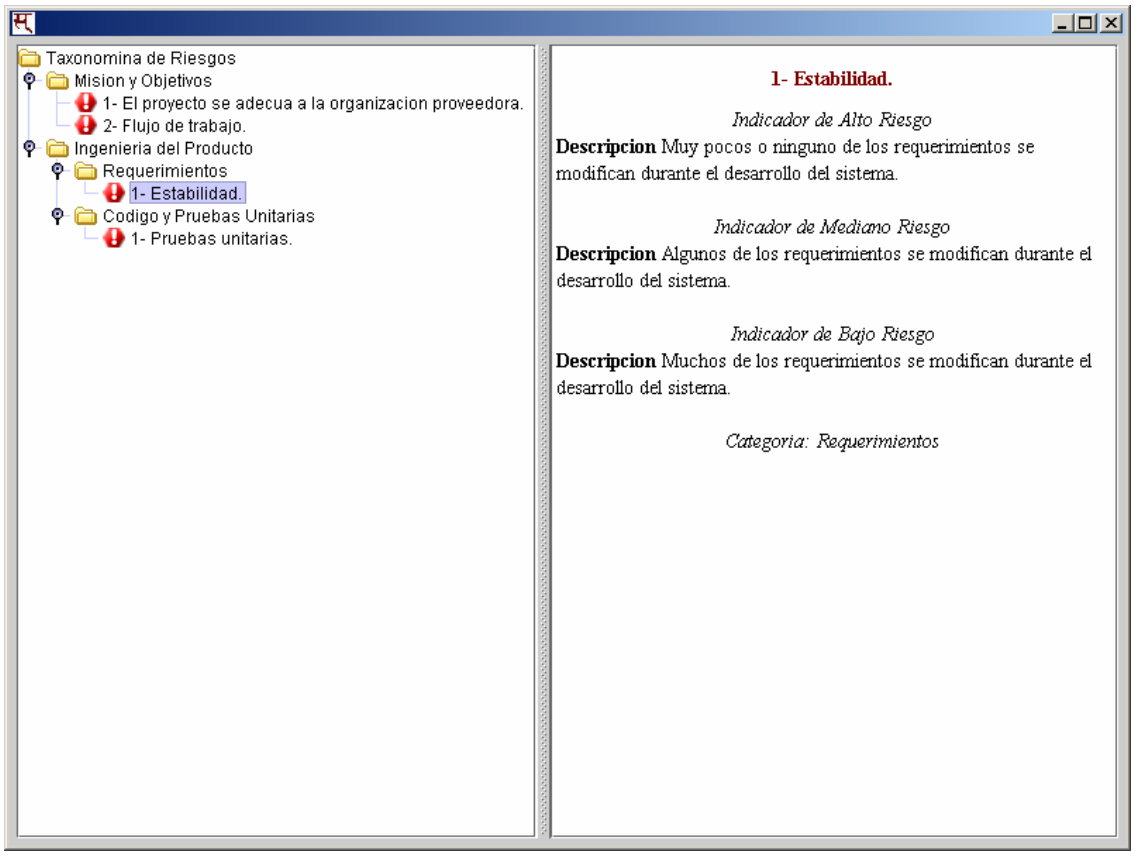


Figura 51 - Instanciación de la Estructura de Datos para la Taxonomía de Riesgos

Se detalla a continuación el DTD empleado por la aplicación Seldon para representar los tipos de proyectos:

```
<!-- DTD for Tipo Proyecto    - S. Maniasi - 2005.06.11 -->

<!ELEMENT TYPES   (TYPE+)>
<!ELEMENT TYPE    (ID, NAME)>
<!ELEMENT ID      (#PCDATA)>
<!ELEMENT NAME    (#PCDATA)>
```

El siguiente es un ejemplo de la instanciación de la estructura de datos antes mencionada:

```
<?xml version="1.0" ?>
<!DOCTYPE TAXO (View Source for full doctype...)>
<TYPES>
    <TYPE>
      <ID>SAI</ID>
      <NAME>Sistemas de Administracion de Informacion</NAME>
    </TYPE>
</TYPES>
```

5.5. Diseño de la Arquitectura de Soporte

En esta sección se realiza la especificación y diseño de los módulos/clases que forman parte de los subsistemas de soporte. Considerando que todos los módulos/clases a utilizar como parte de la arquitectura de soporte constituyen implementaciones de patrones de diseño o paquetes de software previamente existentes, no se realiza una descripción o diseño detallado de los mismos sino que se destacan y enfatizan aquellos aspectos principales que influyen sobre el modo en el cual deben ser entendidos y a través de los cuales estos módulos/clases serán utilizados por Seldon.

5.5.1. mvc

La arquitectura de la aplicación Seldon se basa en el patrón de diseño Model-View-Controller que ha sido ubicado en la capa de soporte y que se encuentra contenido en el paquete mvc. Los paquetes de la capa específica del negocio contendrán clases que implementarán la funcionalidad del patrón.

El siguiente diagrama de clases (véase **Figura 52 - Modelo Estático de mvc**) detalla la estructura estática de clases e interfaces del paquete *mvc*:

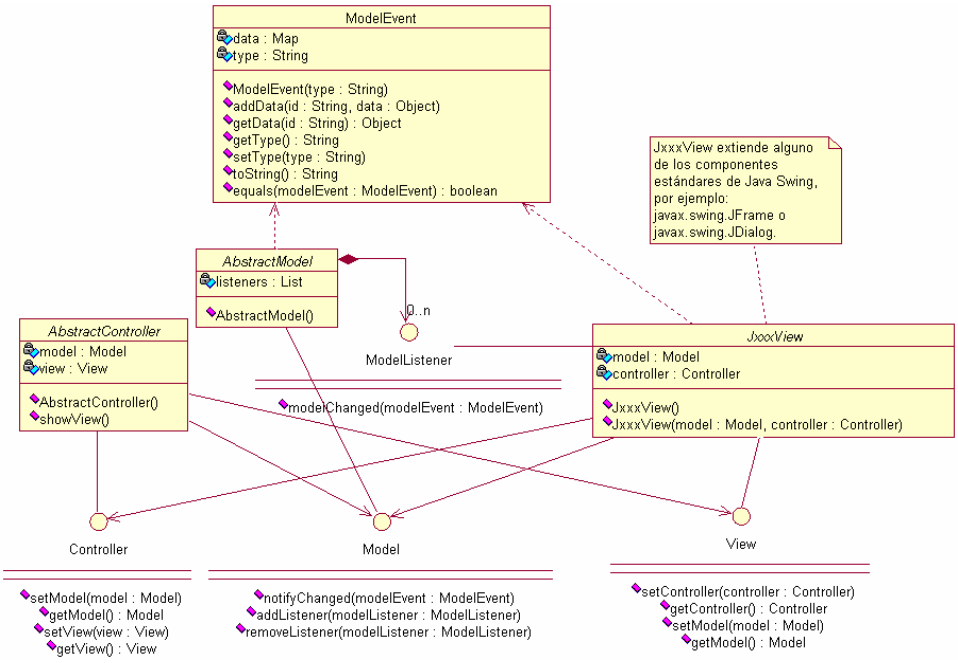


Figura 52 - Modelo Estático de mvc

AbstractController
Implementa los métodos definidos en la interfaz Controller para asociar una vista y un modelo al controlador y obtenerlos.
Atributos Privados
model : Model → Representa el modelo asociado al controlador. view : View → Representa la vista asociada al controlador.
Operaciones Publicas
AbstractController () → Constructor de la clase. showView () → Permite mostrar la vista asociada al controlador.

AbstractModel
Implementa los métodos definidos en la interfaz Model para registrar y eliminar suscripctores del modelo y para notificar los cambios producidos en el mismo.
Atributos Privados
model : Model → Representa el modelo asociado al controlador.
Operaciones Publicas
AbstractModel () → Constructor de la clase.

Controller
Interfaz que define el comportamiento común esperado para todos los controladores en el patrón MVC. El comportamiento común implica asociar un modelo y una vista al controlador y posibilitar el obtenerlos.
Atributos Privados
No Aplicable.
Operaciones Publicas
setModel (model : Model) → Asocia un modelo al controlador. getModel () : Model → Obtiene el modelo asociado al controlador. setView (view : View) → Asocia una vista al controlador. getView () : View → Obtiene la vista asociada al controlador.

JxxxView
Implementa los métodos definidos en la interfaz View para asociar un modelo y un controlador a la vista y obtenerlos. Implementa los métodos definidos en la interfaz ModelListener para suscribirse a los cambios en un modelo. Extiende un componente del paquete Swing (estándar de Java).
Atributos Privados
model : Model → Representa el modelo asociado a la vista. controller : Controller → Representa el controlador asociado a la vista.
Operaciones Publicas
JxxxView () → Constructor de la clase. JxxxView (model : Model, controller : Controller) → Constructor de la clase.

Model
Interfaz que define el comportamiento común esperado para todos los modelos en el patrón MVC. El comportamiento común implica brindar un mecanismo para notificar a los objetos suscriptos los cambios producidos sobre el modelo.
Atributos Privados
No Aplicable.
Operaciones Publicas
notifyChanged (modelEvent : ModelEvent) → Permite Notificar a los objetos suscriptos los cambios producidos en el modelo. addListener (modelListener : ModelListener) → Permite agregar un objeto a la lista suscriptos al modelo. removeListener (modelListener : ModelListener) → Permite eliminar un objeto de la lista de suscriptos al modelo.

ModelEvent
Representa los eventos de notificación mediante los que los objetos suscriptos son notificados sobre los cambios en los modelos.
Atributos Privados
data : Map → Colección con pares identificador-valor donde se almacenan los cambios efectuados en un modelo. type : String → Indicador de tipo de evento.
Operaciones Publicas
ModelEvent (type : String) → Constructor de la clase. addData (id : String, data : Object) → Permite agregar un dato. getData (id : String) : Object → Permite obtener el valor asociado a un dato almacenado en el mapa. getType () : String → Permite retornar el tipo del evento. setType (type : String) → Permite indicar el tipo del evento. toString () : String → Contienen la reimplementación del método toString de Java para mostrar los contenidos del evento. equals (modelEvent : ModelEvent) : boolean → Contiene la reimplementación del método equals de Java para comparar dos eventos.

ModelListener
Interfaz que define el comportamiento común esperado para todos los objetos que deseen suscribirse como interesados en los cambios de un modelo. El comportamiento común implica aceptar las notificaciones de cambio generadas desde un modelo.
Atributos Privados
No Aplicable.
Operaciones Publicas
modelChanged (modelEvent : ModelEvent) → Permite realizar la notificación de los cambios en el modelo.

View
Interfaz que define el comportamiento común esperado para todos las vistas en el patrón MVC. El comportamiento común implica asociar un modelo y un controlador a la vista y posibilitar el obtenerlos.
Atributos Privados
No Aplicable.
Operaciones Publicas
setController (controller : Controller) → Asocia un controlador a la vista. getController () : Controller → Obtiene el controlador asociado a la vista. setModel (model : Model) → Asocia un modelo a la vista. getModel () : Model →Obtiene el modelo asociado a la vista.

Durante la inicialización del sistema se crea una instancia de algún controlador, el cual crea una instancia del modelo al cual transmitirá las acciones del usuario y una instancia de la vista a presentar al usuario. La vista que se ha asociado al modelo y al controlador se registra como observador (en inglés, *listener*) del modelo recibido como argumento, ante lo cual, éste envía una notificación de bienvenida (*ModelEvent: tipo="Welcome"*).

Una acción del usuario será traducida en una notificación de evento Swing (evento estándar de Java e identificado como *ActionEvent*). La vista que ha recibido este evento delega el procesamiento al controlador al cual se la ha asociado mediante la invocación de algún método provisto por el mismo para procesar las acciones recibidas de los usuario (por ejemplo, *doAction*). El controlador produce una invocación a un método de negocios perteneciente a su modelo asociado (por ejemplo, *doBusiness*) el cual genera los cambios correspondientes en el modelo, que crea un evento de notificación (*ModelEvent*) a través del cual informa a todos los objetos registrados los cambios producidos. La notificación provoca una actualización en todas las vistas relacionadas al modelo que ha sido modificado.

Los siguientes diagramas de colaboración (véase **Figura 53 - Modelo Dinámico de mvc – Inicialización** y **Figura 54 - Modelo Dinámico de mvc – Utilización**) detallan el comportamiento dinámico del paquete *mvc*:

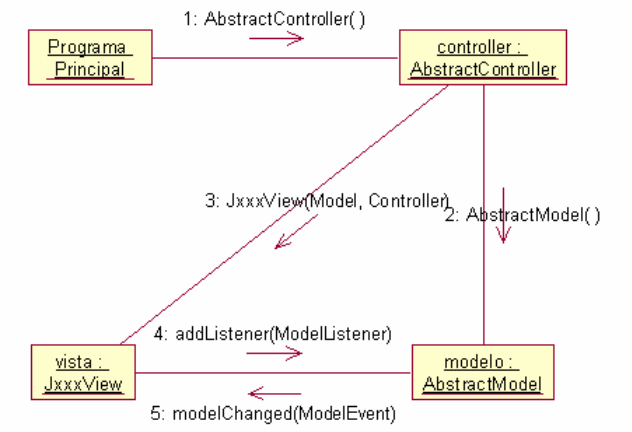


Figura 53 - Modelo Dinámico de mvc – Inicialización

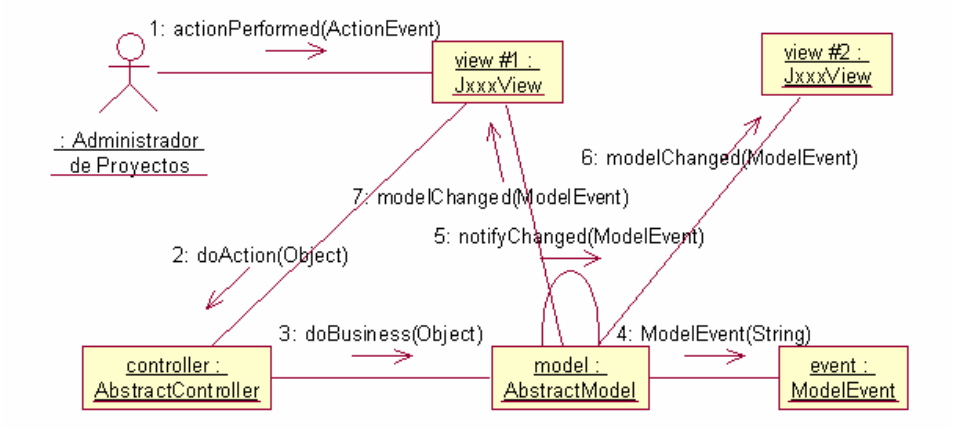


Figura 54 - Modelo Dinámico de mvc – Utilización

5.5.2. JasperReports

El manejo de reportes del sistema Seldon se realizará a través de este subsistema que permitirá realizar la generación, previsualización e impresión de los informes producidos por el sistema.

El siguiente diagrama de clases (véase **Figura 55 - Modelo Estático de JasperReports**) detalla la estructura estática de clases e interfaces que componen el paquete *JasperReports*:

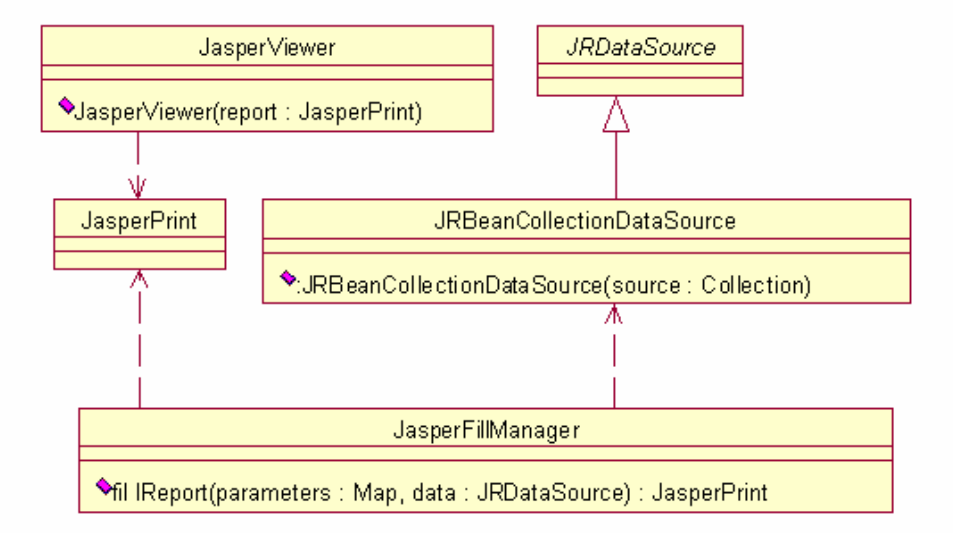


Figura 55 - Modelo Estático de JasperReports

JasperFillManager
Posibilita completar un reporte empleando una fuente de datos.
Atributos Privados
No Aplicable

Operaciones Publicas
fil lReport (parameters : Map, data : JRDataSource) : JasperPrint → Posibilita llenar un reporte en base a los parámetros y datos enviados como parámetros.

JasperPrint
Se emplea para representar una versión imprimible de un reporte.
Atributos Privados
No Aplicable.
Operaciones Publicas
No Aplicable.

JasperViewer
Implementa la interfaz de previsualización de reportes y brinda facilidades de avance y retroceso de páginas.
Atributos Privados
No Aplicable
Operaciones Publicas
JasperViewer (report : JasperPrint) → Constructor de la clase.

JRBeanCollectionDataSource
Representa la fuente de datos a emplear para producir el reporte. Se emplea mediante su inicialización con una colección de datos.
Atributos Privados
No Aplicable
Operaciones Publicas
JRBeanCollectionDataSource (source : Collection) → Constructor de la clase.

JRDataSource
Representa la abstracción de las fuentes de datos a emplear para producir el reporte.
Atributos Privados
No Aplicable
Operaciones Publicas
No Aplicable

La clase cliente del paquete completa una colección con los datos a mostrar en el reporte y la envía como argumento al constructor de JRBeanCollectionDataSource. Además de esos datos, también se completa un mapa con los parámetros genéricos para el reporte (encabezado, pie de página y títulos por ejemplo) y luego se solicita a JasperFillManager la generación de una versión de previsualización del reporte a través del envío de la información de la plantilla del reporte, el mapa con los parámetros y la fuente de datos. Finalmente se produce la previsualización del reporte a través del empleo de un objeto de tipo JasperViewer.

El siguiente diagrama de colaboración (véase **Figura 56 - Modelo Dinámico de JasperReports – Generación de Reporte**) detalla el comportamiento dinámico correspondiente al paquete *JasperReports*:

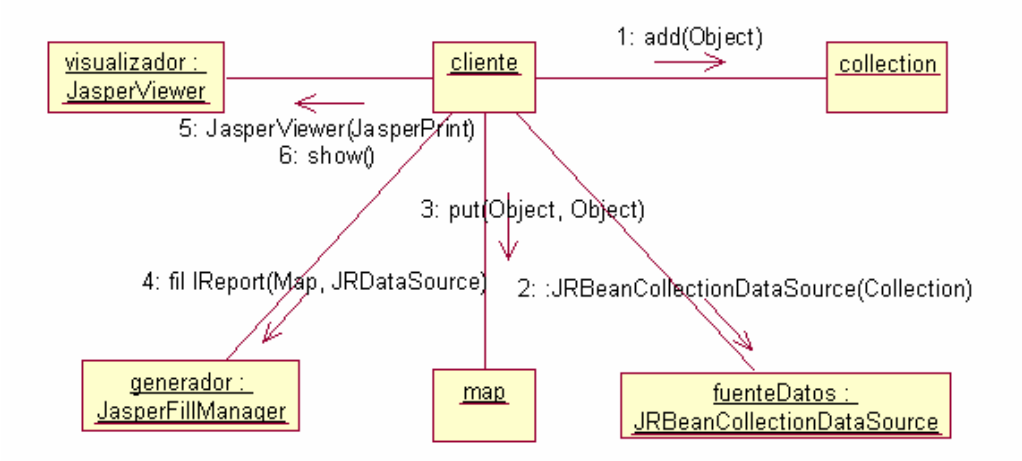


Figura 56 - Modelo Dinámico de JasperReports – Generación de Reporte

5.5.3. persistence

El manejo de persistencia del sistema Seldon se realizará a través de este subsistema que permitirá almacenar y recuperar objetos del sistema.

El siguiente diagrama de clases (véase **Figura 57 - Modelo Estático de persistence**) detalla la estructura estática de clases e interfaces que componen el paquete *persistence*:

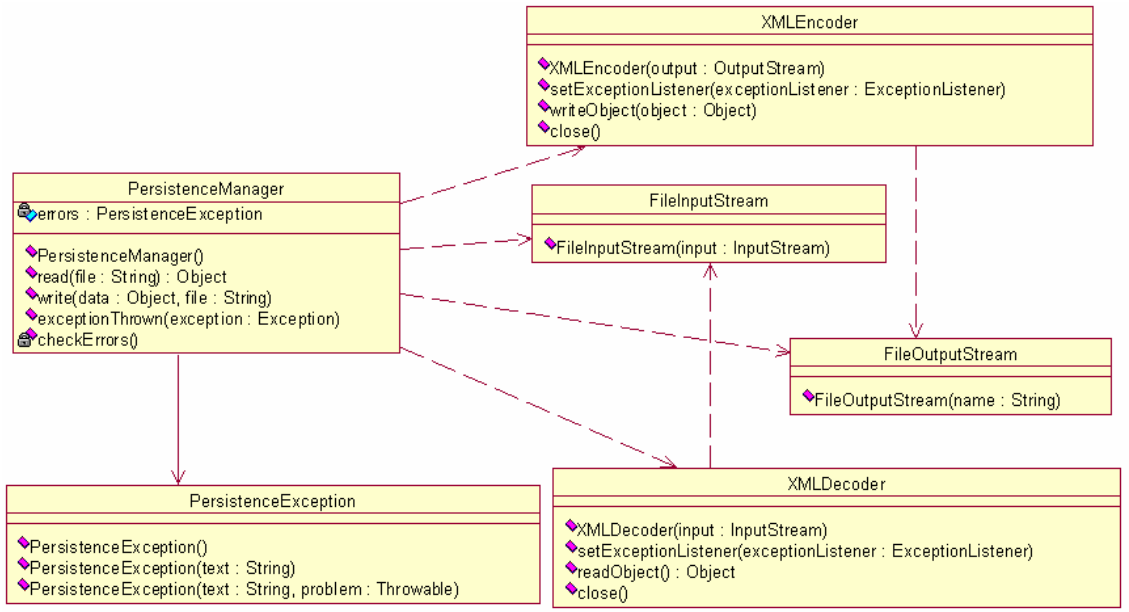


Figura 57 - Modelo Estático de persistence

FileInputStream
Permite leer información desde un archivo. Pertenece a las librerías de la plataforma J2SE.
Atributos Privados
No Aplicable.
Operaciones Publicas
FileInputStream (input : InputStream) → Constructor de la clase.

FileOutputStream
Permite escribir información en un archivo. Pertenece a las librerías de la plataforma J2SE.
Atributos Privados
No Aplicable.
Operaciones Publicas
FileOutputStream (name : String) → Constructor de la clase.

PersistenceException
Representa una excepción arrojada por el subsistema cuando no es posible leer o escribir los objetos en los archivos.
Atributos Privados
No Aplicable
Operaciones Publicas
PersistenceException () → Constructor de la clase. PersistenceException (text : String) → Constructor de la clase. PersistenceException (text : String, problem : Throwable) → Constructor de la clase.

PersistenceManager
Permite manejar el almacenamiento y recuperación de objetos. Implementa la interfaz ExceptionListener para escuchar las exceptiones arrojadas por el XMLEncoder y el XMLDecoder. Finalmente, implementa el patrón de diseño Singleton por lo cual existe un único objeto de esta clase en toda la aplicación.
Atributos Privados
errors : PersistenceException → Representa la excepción a arrojar en caso de existir errores.
Operaciones Publicas
PersistenceManager () → Constructor de la clase. read (file : String) : Object → Permite leer un objeto desde un archivo. write (data : Object, file : String) → Permite escribir un objeto en un archivo. exceptionThrown (exception : Exception) → Este método es invocado cada vez que se produce una excepción sobre un objeto sobre el que se está escuchando.
Operaciones Privadas
checkErrors () → Permite verificar el estado del indicador de errores y arroja una excepción en caso existir problemas.

XMLDecoder
Permite convertir documentos XML en objetos. Pertenece a las librerías de la plataforma J2SE.
Atributos Privados
No Aplicable.
Operaciones Publicas
XMLDecoder (input : InputStream) → Constructor de la clase. setExceptionListener (exceptionListener : ExceptionListener) → Permite registrar un objeto como escucha de excepciones. readObject () : Object → Permite leer un documento XML de la fuente recibida en el constructor y convertirlo en un objeto. close ()→ Cierra el decodificador.

XMLEncoder
Permite convertir objetos en documentos XML. Pertenece a las librerías de la plataforma J2SE.
Atributos Privados
No Aplicable.
Operaciones Publicas
XMLEncoder (output : OutputStream) → Constructor de la clase. setExceptionListener (exceptionListener : ExceptionListener) → Permite registrar un objeto como escucha de excepciones. writeObject (object : Object) → Permite conviertir el objeto recibido como parámetro a XML para luego escribirlo en la fuente recibida en el constructor. close () → Cierra el codificador.

La clase cliente envía un mensaje de escritura PersistenceManager indicando el objeto a almacenar y el archivo de destino a utilizar, quién crea un flujo de salida a un archivo y, en caso de producirse una excepción, instancia una PersistenceException con el mensaje a reportar y la envía al cliente; en caso de no haber excepciones, se crea una instancia de XMLEncoder con el flujo de salida y se registra como interesado en las excepciones del mismo y finalmente envía el mensaje writeObject al XMLEncoder con el objeto a almacenar. Si se produce una excepción en alguno de los métodos del XMLEncoder, éste invoca al método exceptionThrown del objeto PersistenceManager.

La clase cliente envía un mensaje de lectura al PersistenceManager indicando el archivo fuente a utilizar, quién crea un flujo de entrada al archivo y, en caso de producirse una excepción, instancia una PersistenceException con el mensaje a reportar y la envía al cliente; en caso de no haber excepciones, se crea una instancia de XMLDecoder con el flujo de entrada y se registra como interesado en las excepciones del mismo y finalmente envía el mensaje readObject al XMLDecoder. Si se produce una excepción en alguno de los métodos del XMLDecoder, éste invoca al método exceptionThrown del objeto PersistenceManager.

Los siguientes diagramas de colaboración (véase **Figura 58 - Modelo Dinámico de mvc – Escritura** y **Figura 59 - Modelo Dinámico de mvc – Lectura**) detallan el comportamiento dinámico correspondiente al paquete *mvc*:

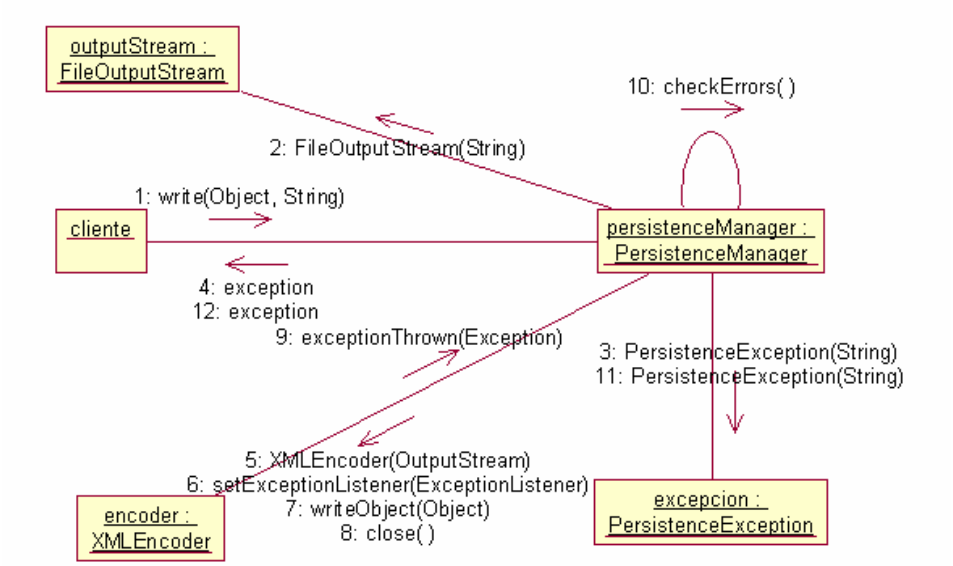


Figura 58 - Modelo Dinámico de mvc – Escritura

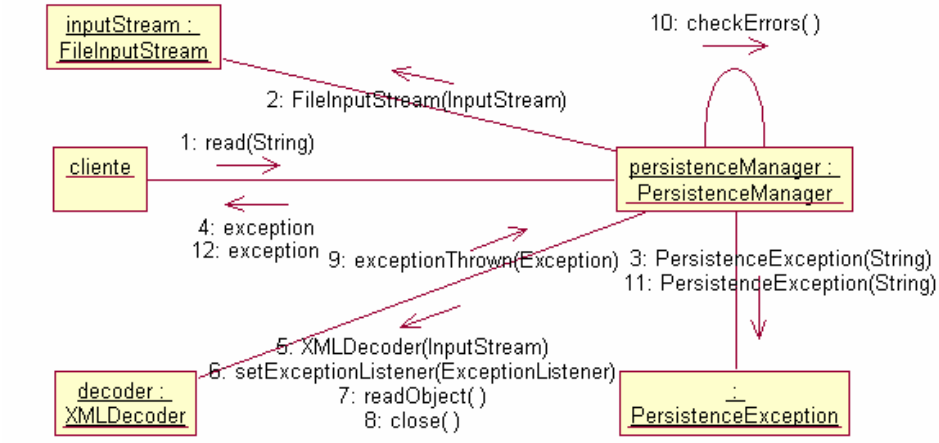


Figura 59 - Modelo Dinámico de mvc – Lectura

5.5.4. i18n

El siguiente paquete da soporte a la característica de internacionalización para la aplicación Seldon tanto para mensajes, números, fechas e imágenes.

El siguiente diagrama de clases (véase **Figura 60 - Modelo Estático de i18n**) detalla la estructura estática de clases e interfaces que componen el paquete *i18n*:

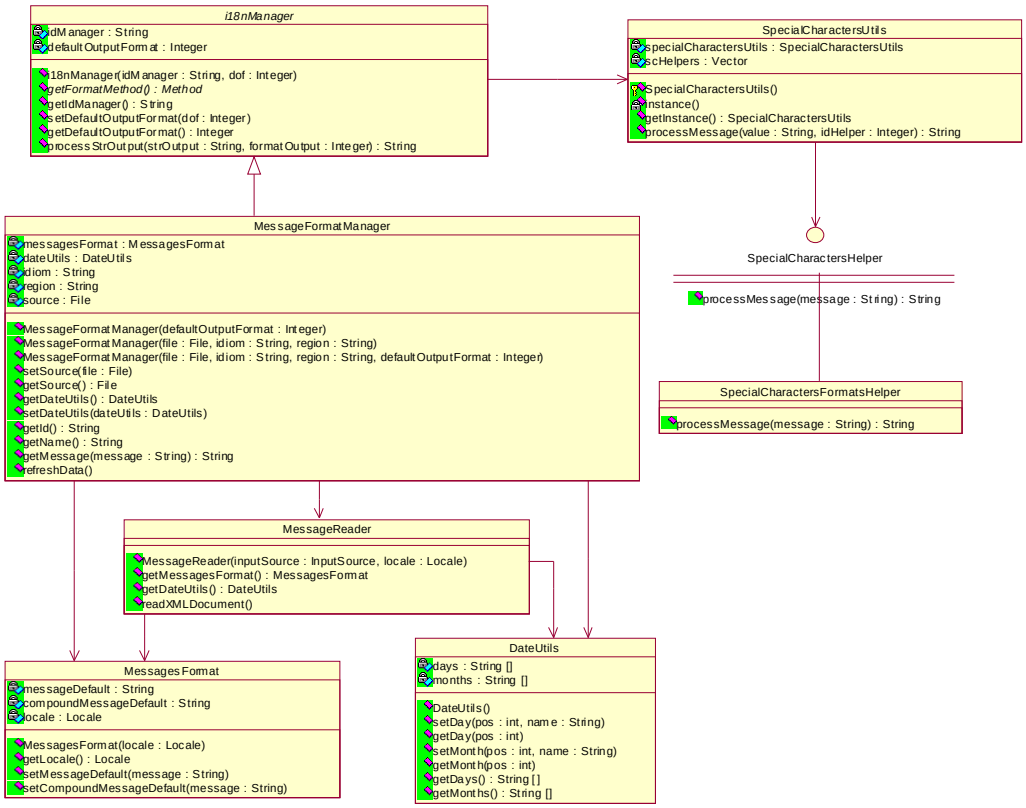


Figura 60 - Modelo Estático de i18n

DateUtils
Clase de utilidad para formateos de fechas.
Atributos Privados
days : String [] → Arreglo de los días de la semana. months : String [] → Arreglo de los meses del año.
Operaciones Publicas
DateUtils () → Constructor de la clase. setDay (pos : int, name : String) → Permite indicar el nombre de un día de la semana. getDay (pos : int) → Permite obtener el nombre de un día de la semana. setMonth (pos : int, name : String) → Permite indicar el nombre de un mes del año. getMonth (pos : int) → Permite obtener el nombre de un mes del año. getDays () : String [] → Permite conocer todos los nombres de los días de la semana. getMonths () : String [] → Permite conocer todos los nombres de los meses del año.

MessageFormatManager
Clase que representa el gestor de internacionalización para mensajes.

Atributos Privados
messagesFormat : MessagesFormat → Formateador de mensajes. dateUtils : DateUtils → Utilidad de formateo de fechas. idiom : String → Idioma del gestor de internacionalización. region : String → Región del gestor de internacionalización. source : File → Fuente del gestor de internacionalización.
Operaciones Publicas
MessageFormatManager (defaultOutputFormat : Integer) → Constructor de la clase. MessageFormatManager (file : File, idiom : String, region : String) → Constructor de la clase. MessageFormatManager (file : File, idiom : String, region : String, defaultOutputFormat : Integer) → Constructor de la clase. setSource (file : File) → Permite indicar la fuente de datos para el gestor de internacionalización. getSource () : File → Permite conocer la fuente de datos para el gestor de internacionalización. getDateUtils () : DateUtils → Permite conocer la utilidad de formateo de fechas asociada al gestor de internacionalización. setDateUtils (dateUtils : DateUtils) → Permite asociar la utilidad de formateo de fechas asociada al gestor de internacionalización. getId () : String → Permite conocer el identificador del formateador de mensajes. getName () : String → Permite conocer el nombre del formateador de mensajes. getMessage (message : String) : String → Permite dar formato a un mensaje. refreshData () → Permite crear o refrescar la información contenida en el gestor de internacionalización para dar formato a los mensajes.

MessageReader
Permite leer el archivo XML conteniendo la información para dar formato a mensajes.
Atributos Privados
No Aplicable.
Operaciones Publicas
MessageReader (inputSource : InputSource, locale : Locale) → Constructor de la clase. getMessagesFormat () : MessagesFormat → Permite obtener el formateador de mensajes para la configuración indicada en la fuente de datos. getDateUtils () : DateUtils → Permite obtener la utilidad de fechas para la configuración indicada en la fuente de datos. readXMLDocument () → Permite leer la configuración de formateo de mensajes contenida en un archivo XML.

MessagesFormat
Clase que permite dar formato a los mensajes de un gestor de internacionalización.

Atributos Privados
messageDefault : String → Mensaje simple por defecto. compoundMessageDefault : String → Mensaje compuesto por defecto. locale : Locale → Localización a emplear para dar formato a los mensajes.
Operaciones Publicas
MessagesFormat (locale : Locale) → Constructor de la clase. getLocale () : Locale → Permite obtener la localización del formateador de mensajes. setMessageDefault (message : String) → Permite indicar el mensaje simple por defecto. setCompoundMessageDefault (message : String) → Permite indicar el mensaje compuesto por defecto.

SpecialCharactersFormatsHelper
Clase que implementa la interfaz de colaboración para el formateo de mensajes con caracteres especiales.
Atributos Privados
No Aplicable.
Operaciones Publicas
processMessage (message : String) : String → Permite procesar un mensaje con caracteres especiales.

SpecialCharactersHelper
Interfaz que representa un procesador de caracteres especiales.
Atributos Privados
No Aplicable.
Operaciones Publicas
processMessage (message : String) : String → Permite procesar un mensaje con caracteres especiales.

SpecialCharactersUtils
Clase que brinda utilidades para manejo de caracteres especiales. Se emplea en la misma el patrón de diseño “Singleton”.
Atributos Privados
specialCharactersUtils : SpecialCharactersUtils → Instancia única de la clase. scHelpers : Vector → Conjunto de colaboradores registrados.
Operaciones Publicas
getInstance () : SpecialCharactersUtils → Permite implementar el patrón de diseño “Singleton”. Obtiene la instancia única. processMessage (value : String, idHelper : Integer) : String → Permite procesar un mensaje con caracteres especiales.
Operaciones Protegidas
SpecialCharactersUtils () → Constructor de la clase.
Operaciones Privadas
instance () → Permite implementar el patrón de diseño “Singleton”. Crea la

instancia única.

i18nManager
Clase que define el comportamiento común esperado para todos los gestores de internacionalización.
Atributos Privados
idManager : String → Identificador del gestor de internacionalización. defaultOutputFormat : Integer → Indicador de formato de salida por defecto.
Operaciones Publicas
i18nManager (idManager : String, dof : Integer) → Constructor de la clase. getFormatMethod () : Method → Permite retornar el método que dará formato a un elemento (mensaje, número, fecha, etc.) getIdManager () : String → Permite conocer el identificador del gestor de internacionalización. setDefaultOutputFormat (dof : Integer) → Permite indicar el formato de salida por defecto para el manejador. getDefaultOutputFormat () : Integer → Permite conocer el formato de salida por defecto del gestor de internacionalización. processStrOutput (strOutput : String, formatOutput : Integer) : String → Permite dar formato especial a una cadena mediante el uso de caracteres especiales.

El programa principal crea un gestor de internacionalización de mensajes que, como parte de su inicialización y en base a la configuración definida para el sistema, crea una utilidad de manejo de fechas y de formateo de mensajes que servirán de colaboradores durante el proceso de procesamiento de mensajes. Ante cada solicitud de un mensaje por parte del programa principal, el gestor de internacionalización define el mensaje a ser mostrado y su formato final mediante la colaboración de las clases de manejo de caracteres especiales.

El siguiente diagrama de colaboración (véase **Figura 61 - Modelo Dinámico de i18n – Utilización**) detalla el comportamiento dinámico correspondiente al paquete *i18n*:

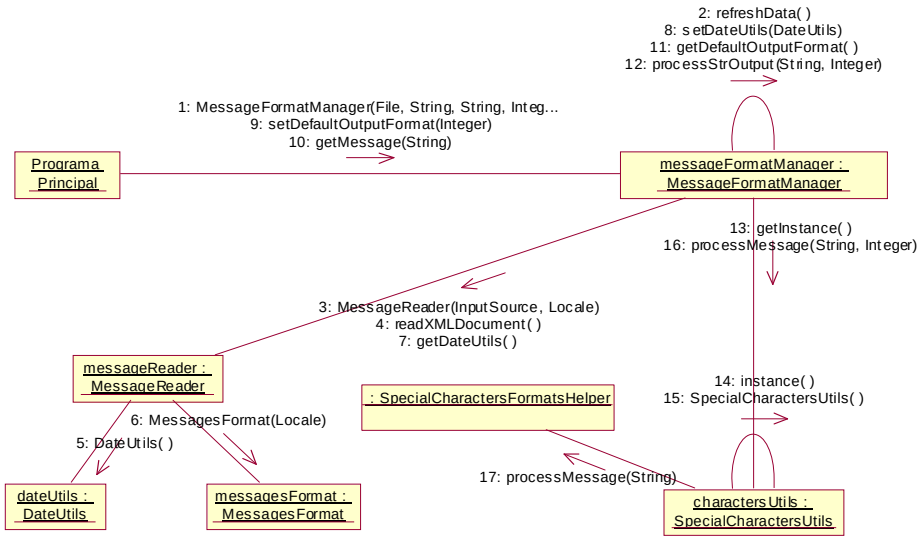


Figura 61 - Modelo Dinámico de i18n – Utilización

Debe destacarse que una estructura estática y un comportamiento similar puede ser observado dentro del mismo paquete para el conjunto de clases que dan soporte a la internacionalización de:

- Números.
- Fechas.
- Imágenes.

El siguiente es un ejemplo de los archivos de configuración que deben ser creados para hacer uso del soporte de internacionalización del paquete *i18n*:

Archivo de configuración del paquete

```
- <parameters>
  - <parameter key="MessageFilePath">
      <value>../</value>
    </parameter>
  - <parameter key="MessageFileName">
      <value>message_format</value>
    </parameter>
  - <parameter key="DefaultIdiom">
      <value>ES</value>
    </parameter>
  - <parameter key="DefaultRegion">
      <value>AR</value>
    </parameter>
  </parameters>
```

Archivo de configuración de mensajes

```
- <messagesFormat>
  - <messagesFormatFunctionality KEY="default">
    - <message key="OK">
        <value>Bien</value>
      </message>
    - <message key="SAVE">
        <value>Guardar</value>
      </message>
    - <message key="CD">
        <value>Disco Compacto®</value>
      </message>
    - <message key="WRITE">
        <value>Escribir</value>
      </message>
    - <months>
      <month1>Enero</month1>
      <month2>Febrero</month2>
      <month3>Marzo</month3>
      <month4>Abril</month4>
      <month5>Mayo</month5>
      <month6>Junio</month6>
      <month7>Julio</month7>
```

```
        <month8>Agosto</month8>
        <month9>Septiembre</month9>
        <month10>Octubre</month10>
        <month11>Noviembre</month11>
        <month12>Diciembre</month12>
      </months>
    - <daysOfWeek>
        <dayOfWeek1>Domingo</dayOfWeek1>
        <dayOfWeek2>Lunes</dayOfWeek2>
        <dayOfWeek3>Martes</dayOfWeek3>
        <dayOfWeek4>Miercoles</dayOfWeek4>
        <dayOfWeek5>Jueves</dayOfWeek5>
        <dayOfWeek6>Viernes</dayOfWeek6>
        <dayOfWeek7>Sabado</dayOfWeek7>
      </daysOfWeek>
  </messagesFormat>
```

Archivo de configuración de números

```
- <numbersFormat>
  - <defaultRegion>
    - <numbersFormatRegion key="ARG">
      - <numberFormat key="NUMBER">
          <integerFormater>#</integerFormater>
          <integerPrecision>6</integerPrecision>
          <groupSize>3</groupSize>
          <decimalFormater>#</decimalFormater>
          <groupSeparator>,</groupSeparator>
          <decimalSeparator>.</decimalSeparator>
          <decimalPrecision>3</decimalPrecision>
        </numberFormat>
      - <numberFormat key="CURRENCY">
          <integerFormater>#</integerFormater>
          <integerPrecision>6</integerPrecision>
          <groupSize>3</groupSize>
          <decimalFormater>0</decimalFormater>
          <groupSeparator>,</groupSeparator>
          <decimalSeparator>.</decimalSeparator>
          <decimalPrecision>2</decimalPrecision>
          <prefix>$</prefix>
        </numberFormat>
      - <numberFormat key="VERSION">
          <integerFormater>#</integerFormater>
          <integerPrecision>3</integerPrecision>
          <groupSize>3</groupSize>
          <decimalFormater>0</decimalFormater>
          <groupSeparator>,</groupSeparator>
          <decimalSeparator>.</decimalSeparator>
          <decimalPrecision>3</decimalPrecision>
        </numberFormat>
      </numbersFormatRegion>
    </defaultRegion>
  - <numbersFormatRegion key="BRA">
```

```
- <numberFormat key="NUMBER">
    <integerFormater>#</integerFormater>
    <integerPrecision>6</integerPrecision>
    <groupSize>3</groupSize>
    <decimalFormater>0</decimalFormater>
    <groupSeparator>.</groupSeparator>
    <decimalSeparator>,</decimalSeparator>
    <decimalPrecision>8</decimalPrecision>
  </numberFormat>
- <numberFormat key="CURRENCY">
    <integerFormater>#</integerFormater>
    <integerPrecision>6</integerPrecision>
    <groupSize>3</groupSize>
    <decimalFormater>0</decimalFormater>
    <groupSeparator>.</groupSeparator>
    <decimalSeparator>º</decimalSeparator>
    <decimalPrecision>2</decimalPrecision>
    <prefix>¥</prefix>
    <suffix>®</suffix>
  </numberFormat>
- <numberFormat key="VERSION">
    <integerFormater>#</integerFormater>
    <integerPrecision>3</integerPrecision>
    <groupSize>3</groupSize>
    <decimalFormater>0</decimalFormater>
    <groupSeparator>.</groupSeparator>
    <decimalSeparator>,</decimalSeparator>
    <decimalPrecision>3</decimalPrecision>
  </numberFormat>
 </numbersFormatRegion>
</numbersFormat>
```

Archivo de configuración de fechas

```
- <datesFormat>
  - <defaultRegion>
    - <datesFormatRegion key="ARG">
      - <dateFormat key="LONG_DATE">
          <year>yyyy</year>
          <mounth>MMM</mounth>
          <day>dd</day>
          <hour>hh</hour>
          <minute>mm</minute>
          <second>ss</second>
          <dayOfWeek />
          <dateSeparator>de</dateSeparator>
          <timeSeparator>:</timeSeparator>
          <elementOrder>3217456</elementOrder>
        </dateFormat>
      - <dateFormat key="SHORT_DATE">
          <year>yyyy</year>
          <mounth>MM</mounth>
          <day>dd</day>
```

```
            <dateSeparator>x</dateSeparator>
            <elementOrder>321</elementOrder>
          </dateFormat>
        </datesFormatRegion>
      </defaultRegion>
    - <datesFormatRegion key="USA">
      - <dateFormat key="LONG_DATE">
          <year>yyyy</year>
          <mounth>MMM</mounth>
          <day>dd</day>
          <hour>hh</hour>
          <minute>mm</minute>
          <second>ss</second>
          <dayOfWeek />
          <dateSeparator>of</dateSeparator>
          <timeSeparator>:</timeSeparator>
          <elementOrder>2317456</elementOrder>
        </dateFormat>
      - <dateFormat key="SHORT_DATE">
          <hour>a hh</hour>
          <minute>mm</minute>
          <timeSeparator>:</timeSeparator>
          <elementOrder>45</elementOrder>
        </dateFormat>
      </datesFormatRegion>
    </datesFormat>
```

Archivo de configuración de imágenes

```
- <imagesFormat>
  - <imagesFormatFunctionality KEY="default">
    - <image key="Img1">
        <value>./images/argentina1.gif</value>
      </image>
    - <image key="Img2">
        <value>./images/argentina2.gif</value>
      </image>
    </imagesFormatFunctionality>
  - <imagesFormatFunctionality KEY="BRA">
    - <image key=" Img1">
        <value>./images/brasil1.gif</value>
      </image>
    - <image key=" Img1">
        <value>./images/brasil2.gif</value>
      </image>
    </imagesFormatFunctionality>
  </imagesFormat>
```

5.5.5. JEOPS

El siguiente paquete brinda el soporte de gestión de reglas empleado por el sistema construido durante el proceso de identificación de riesgos.

JEOPS es una biblioteca de uso libre de Java empleada para el uso de reglas de deducción en sistemas que provee una infraestructura para definir, manejar y realizar consultas basadas en reglas. Se encuentra basado en el algoritmo Rete: un eficiente algoritmo de correspondencia de patrones específicamente diseñado para la implementación de sistemas basados en reglas que fuera desarrollado por el Dr. Charles L. Forgy de Carnegie Mellon University en el año1979.

JEOPS se encuentra dividido en dos grupos de funcionalidades:

- JEOPS Compiler:
  Compuesto por los paquetes:
  - *jeops.compiler*: Que representa el compilador que traduce archivos de reglas en clases Java.
  - *jeops.compiler.parser*: Que contiene clases auxiliaries empleadas durante el proceso de compilación y traducción de archivos de reglas.
- JEOPS Runtime Environment:
  Compuesto por los paquetes:
  - *jeops*: El entorno de ejecución para JEOPS.
  - *jeops.conflict*: El conjunto de políticas de resolución de conflictos estándar que pueden ser empleadas par alas bases de conocimientos generadas por JEOPS.
  - *jeops.rete*: Las clases que implementan los nodos de una red Rete empleada como el mecanismo seleccionado para la gestión de JEOPS.

Los siguientes diagramas de clases (véase **Figura 62 - Modelo Estático de JEOPS (paquete jeops), Figura 63 - Modelo Estático de JEOPS (paquete compiler), Figura 64 - Modelo Estático de JEOPS (paquete conflict) y Figura 65 - Modelo Estático de JEOPS (paquete rete)**) detallan la estructura estática de clases e interfaces que componen el paquete *JEOPS*:

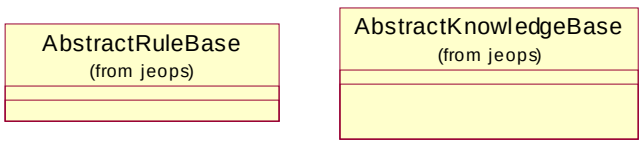


Figura 62 - Modelo Estático de JEOPS (paquete jeops)

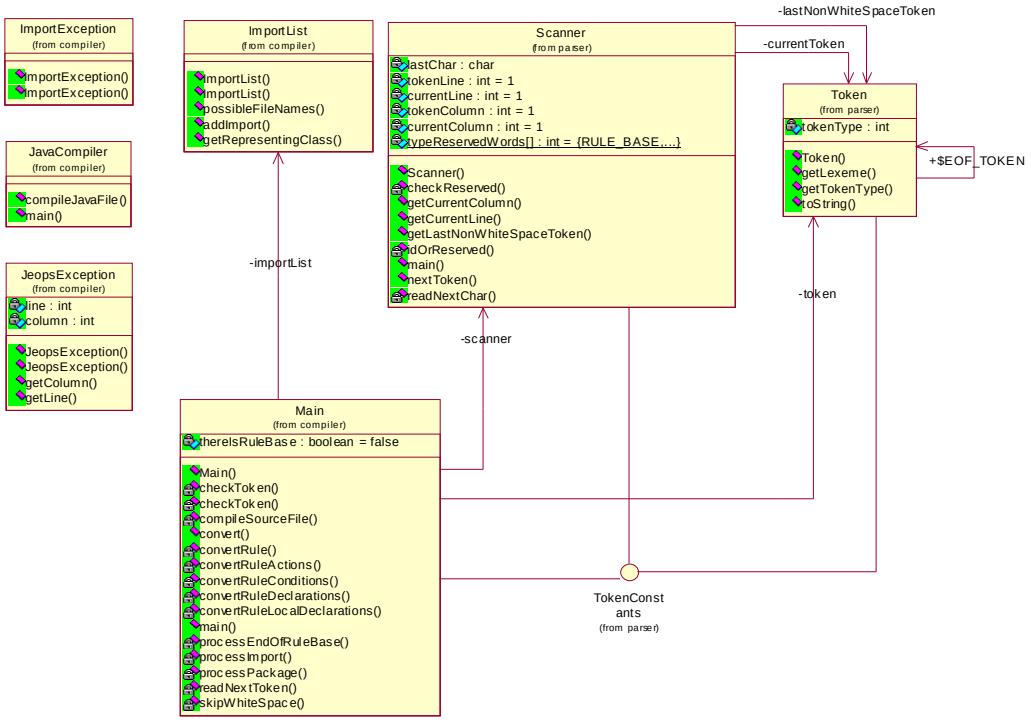


Figura 63 - Modelo Estático de JEOPS (paquete compiler)

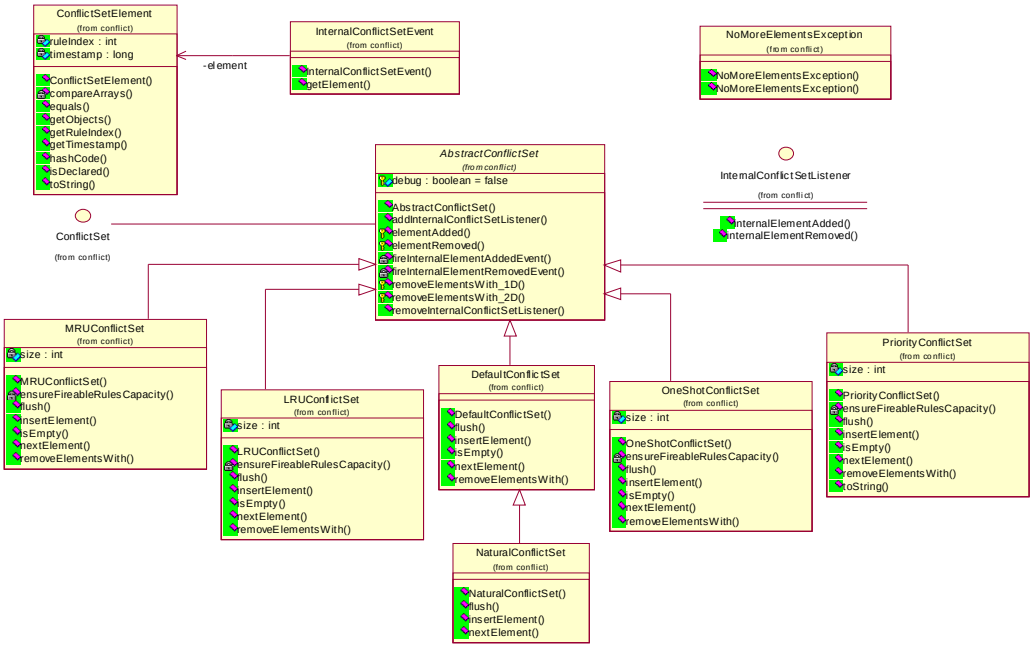


Figura 64 - Modelo Estático de JEOPS (paquete conflict)

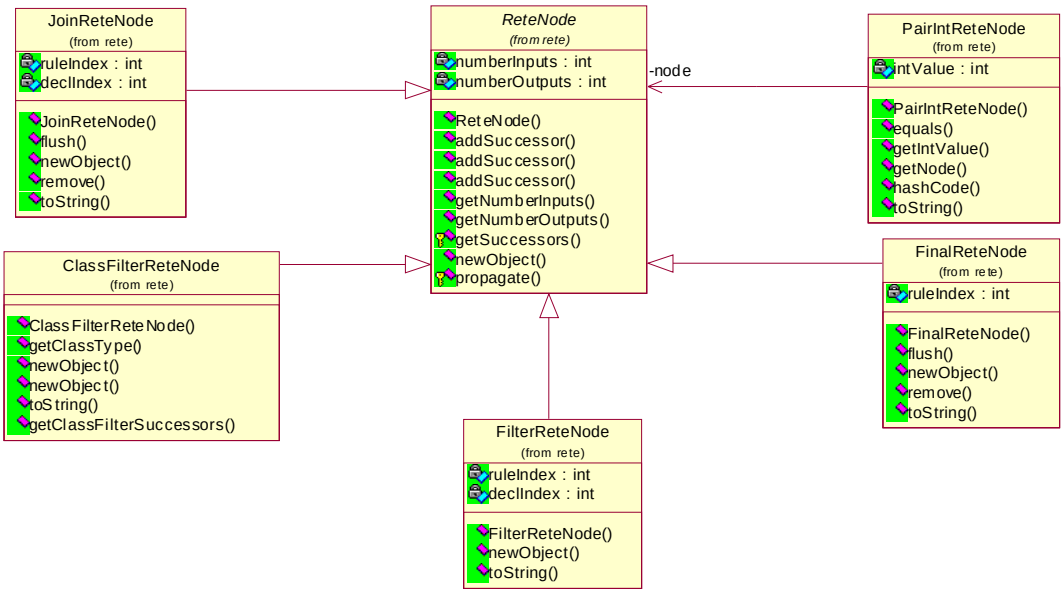


Figura 65 - Modelo Estático de JEOPS (paquete rete)

El siguiente cuadro (véase **Tabla 26 - Clases e Interfaces de**) describe las principales características de cada una de las clases e interfaces identificadas dentro del producto:

ConflictSet	Define las operaciones necesarias que deben ser implementadas en un conjunto conflicto (que representa el repositorio donde se almacenan las reglas candidatas a ser disparadas). Cada una de las implementaciones determinará la política usada para seleccionar la regal a ejecutar.
AbstractConflictSet	Provee una implementación vacía de la interfaz ConflictSet.
Main	Clase principal de la fase de JEOPS encargada de converter el archive de reglas en clases Java.
FilterReteNode	Un nodo con una entrada y una salida en la red de nodos de JEOPS.
NaturalConflictSet	Un conjunto conflicto cuya política de resolución indica que no se puede disparar una regla mas de una vez para el mismo grupo de objetos.
FinalReteNode	Un nodo final en la red de nodos de JEOPS.
JoinReteNode	Un nodo con más de una entrada y una salida en la red de nodos de JEOPS.
InternalConflictSetEvent	Un evento semántico que indica que se agregó o eliminó un elemento del conjunto conflicto.
ClassFilterReteNode	Un nodo en la red de nodos de JEOPS que se activa si el objeto referido pertenece a la clase que dicho nodo pertenece.
PriorityConflictSet	Un conjunto conflicto cuya política de resolución indica que la prioridad de las reglas coincide con

	el orden en el que fueron definidas.
ReteNode	Un nodo en la red de nodos de JEOPS.
DefaultConflictSet	Un conjunto conflicto cuya política de resolución no ha sido indicada.
LRUConflictSet	Un conjunto conflicto cuya política de resolución indica que el nodo mas prioritario es el menos recientemente utilizado.
MRUConflictSet	Un conjunto conflicto cuya política de resolución indica que el nodo mas prioritario es el menos recientemente utilizado.
InternalConflictSetListener	La interfaz escucha de eventos sobre conjuntos conflicto.
Scanner	Clase empleada para el parseo de un archive de reglas.
Token	Clase empleada para almacenar los elementos de un archivo de reglas.
TokenConstants	Interfaz empleada para definir las constantes utilizadas durante el proceso de parseo de un archivo de reglas.
NoMoreElementsException	Excepción disparada cuando no existen más clases a disparar en un conjunto conflicto.
ImportException	Excepción disparada cuando un identificador puede representar a más de una clase (ambigüedad de definición).
OneShotConflictSet	Un conjunto conflicto cuya política de resolución indica que una regla no puede ser disparada mas de una vez.
JavaCompiler	Clase auxiliar para compilar archivos Java y genera bytecodes (archivos .class).
PairIntReteNode	Clase auxliar para agrupar pares de enteros/nodos.
ConflictSetElement	Un elemento presente en un conjunto conflicto.
JeopsException	Una excepción disparada por JEOPS.

Tabla 26 - Clases e Interfaces de JEOPS

La base de reglas de JEOPS puede ser escrita en un archivo de texto siguiendo la siguiente sintaxis:

```
Rule ::= "rule" "{" <Rule Body> "}"
Rule Body ::= <Declarations> <Local Declarations>? <Conditions>
<Actions>
Declarations ::= "declarations" (<class name> <ident> ("," <ident>)*
)*
Local Declarations ::= "localdecl" (<class name> <ident> "="
<expression>)*
Conditions ::= "conditions" (<expression>)*
Actions ::= (Action)+
Action ::= "assert" "(" <expression> ")"
         | "retract" "(" <expression> ")"
         | "modified" "(" <expression> ")"
         | <block>
```

Donde `<expression>` puede ser cualquier expresión válida en Java y `<block>` cualquier bloque válido en Java.

El proceso de utilización de JEOPS consta de dos etapas:

1. Transcripción de las reglas escritas en un archivo de texto a una serie de clases Java.
2. Compilación, utilización y ejecución de las clases Java derivadas del archivo de texto de reglas.

5.6. Diseño de Casos de Uso Reales

En esta sección se especifica el comportamiento del sistema para cado caso de uso mediante objetos o subsistemas de diseño que interactúan y determinan y validan las operaciones de las clases e interfaces de los distintos subsistemas de diseño.

5.6.1. Enfoque Seleccionado

Considerando las características y el nivel de profundidad del análisis realizado previamente, se ha optado por cubrir las tareas propuestas por Métrica V3 (Identificación de Clases Asociadas a un Caso de Uso, Diseño de la Realización de los Casos de Uso y Revisión de Subsistemas de Diseño e Interfaces) mediante la creación de Diagramas de Robustez que posibilitarán:

- Identificar las clases que intervienen en cada caso de uso.
- Definir cómo interactúan entre sí los objetos identificados.
- Determinar los subsistemas que participan en cada caso de uso y las interfaces requeridas.

Adicionalmente, se actualizarán las matrices de trazabilidad (véase **Apéndice G: Matrices de Trazabilidad**) con el propósito de reflejar las interacciones detectadas.

5.6.2. Diagramas de Robustez

Las siguientes secciones contienen los Diagramas de Robustez que permiten describir cada uno de los Casos de Uso identificados y definidos durante la fase de Análisis del Sistema.

Identificando Riesgos – DR-001

La siguiente figura (véase **Figura 66 - Identificando Riesgos - DR-001**) representa el diagrama de robustez para el caso de uso “Identificando Riesgos – CU-001”.

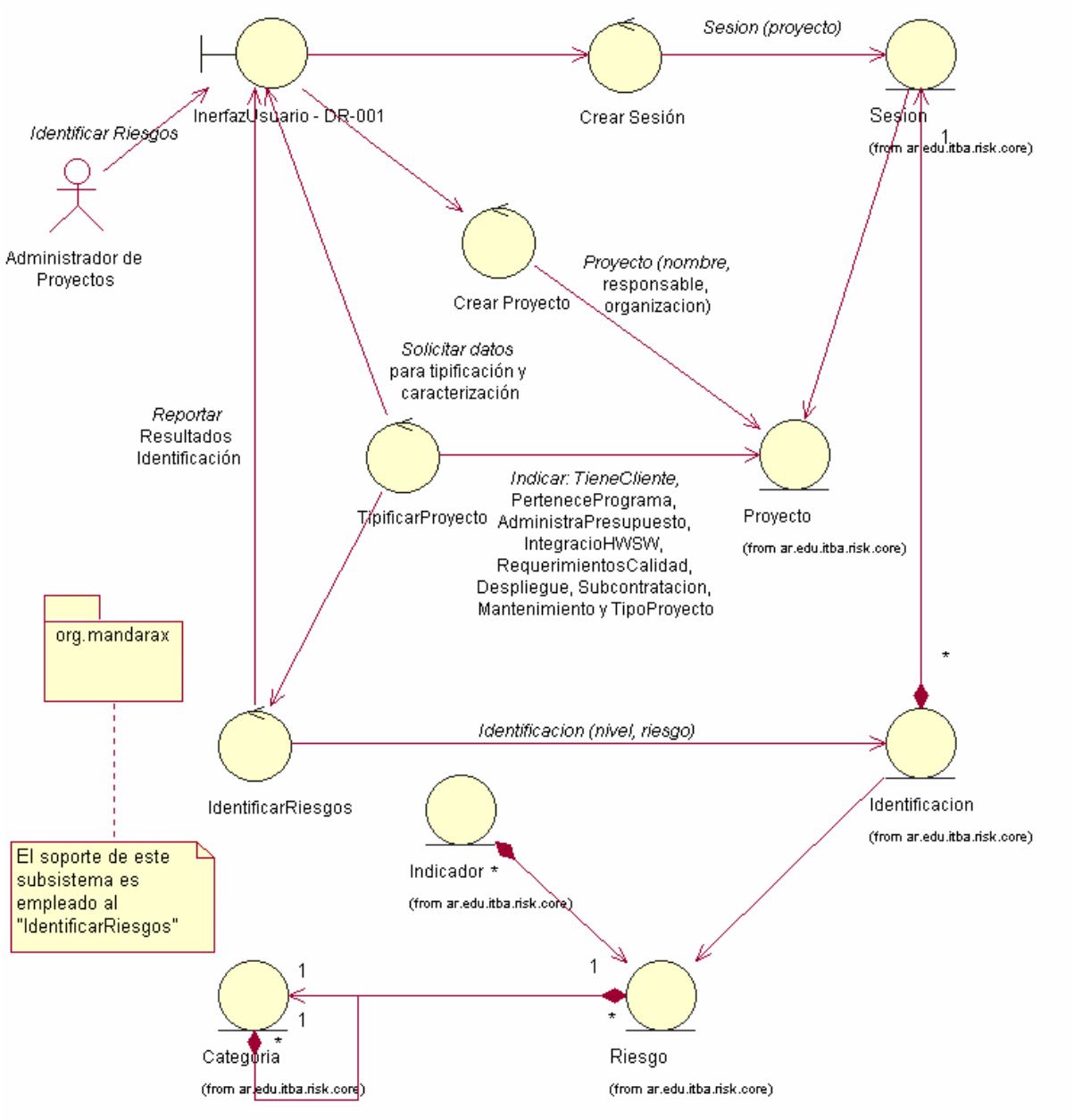


Figura 66 - Identificando Riesgos - DR-001

Inicialmente, el Administrador de Proyectos solicita realizar una identificación de riesgos a través de la interfaz de usuario ingresando los siguientes datos iniciales:

- Nombre del proyecto.
- Responsable del proyecto.
- Organización a la que pertenece el proyecto.

Luego se produce la creación del proyecto y de la sesión asociados al pedido de identificación de riesgos y se solicita la tipificación y caracterización del proyecto.

Una vez finalizadas las actividades de creación e inicialización se produce a la efectiva identificación de los riesgos del proyecto empleando el soporte brindado por el subsistema de gestión de reglas.

Generando Reporte de Resultados – DR-002

La siguiente figura (véase **Figura 67 - Generando Reporte de Resultados - DR-002**) representa el diagrama de robustez para el caso de uso “Generando Reporte de Resultados – CU-002”.

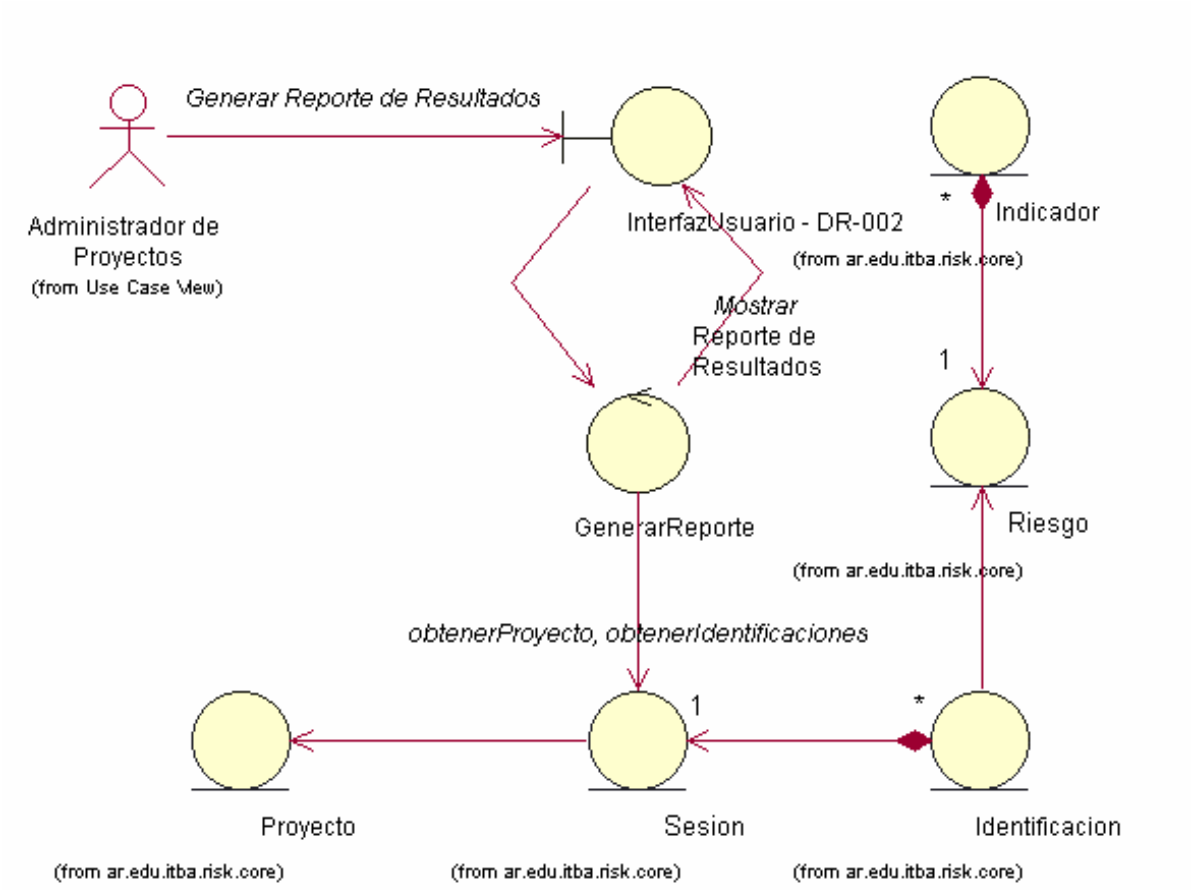


Figura 67 - Generando Reporte de Resultados - DR-002

El Administrador de Proyectos solicita la generación de un reporte de los resultados de un proceso de identificación de riesgos que es presentado al usuario mediante la interfaz gráfica definida.

A través de la sesión es posible acceder tanto al proyecto como a las identificaciones realizadas, mediante las cuales se puede determinar, para cada riesgo, el indicador asociado luego de la identificación.

Imprimiendo Reporte de Resultados – DR-003

La siguiente figura (véase **Figura 68 - Imprimiendo Reporte de Resultados - DR-003**) representa el diagrama de robustez para el caso de uso “Imprimiendo Reporte de Resultados – CU-003”.

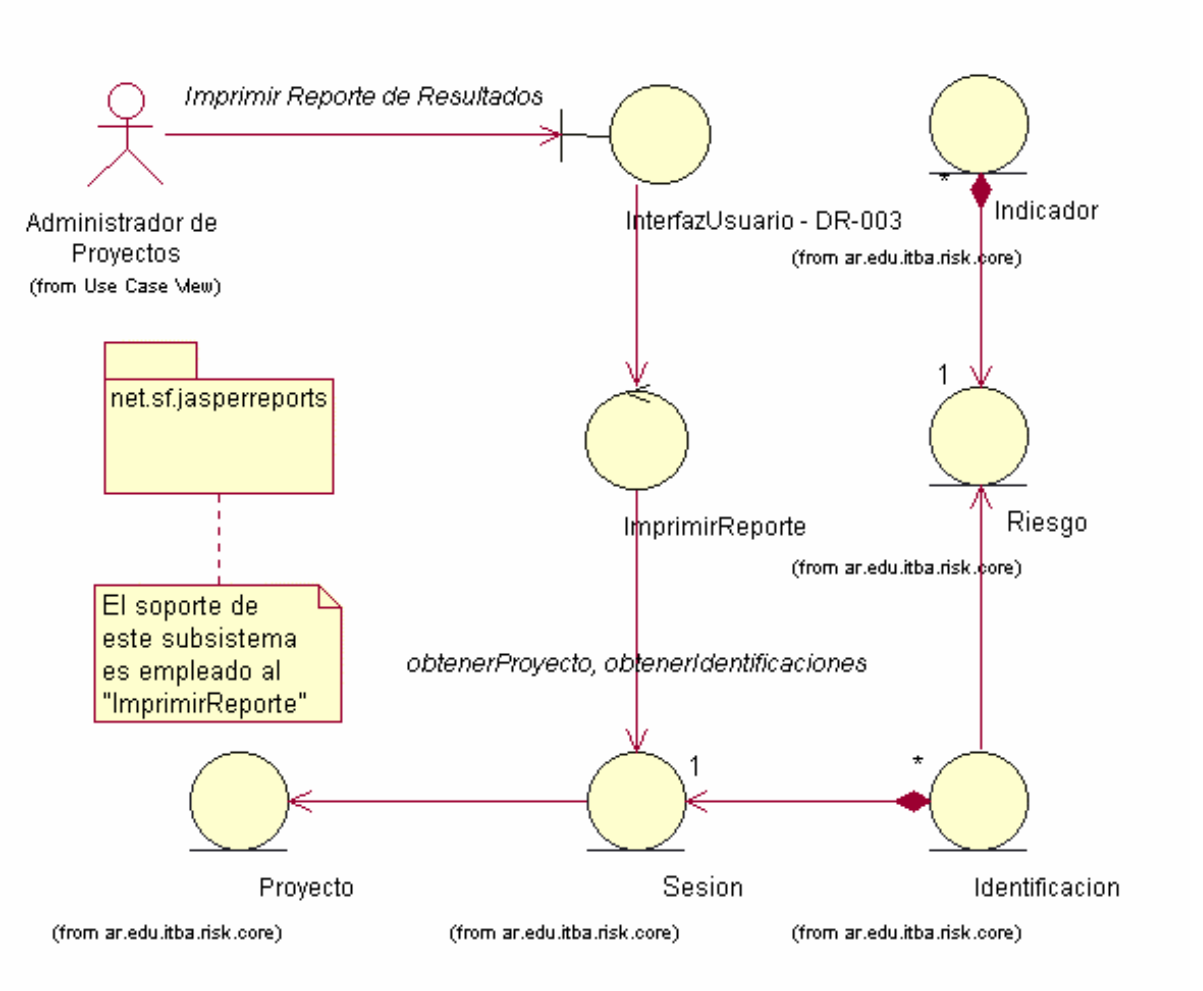


Figura 68 - Imprimiendo Reporte de Resultados - DR-003

El Administrador de Proyectos solicita la impresión de un reporte de los resultados de un proceso de identificación de riesgos lo cual se realiza empleando el soporte brindado por el subsistema de generación de reportes.

A través de la sesión es posible acceder tanto al proyecto como a las identificaciones realizadas, mediante las cuales se puede determinar, para cada riesgo, el indicador asociado luego de la identificación.

Almacenando Resultados – DR-004

La siguiente figura (véase **Figura 69 - Almacenando Resultados - DR-004**) representa el diagrama de robustez para el caso de uso “Almacenando Resultados – CU-004”.

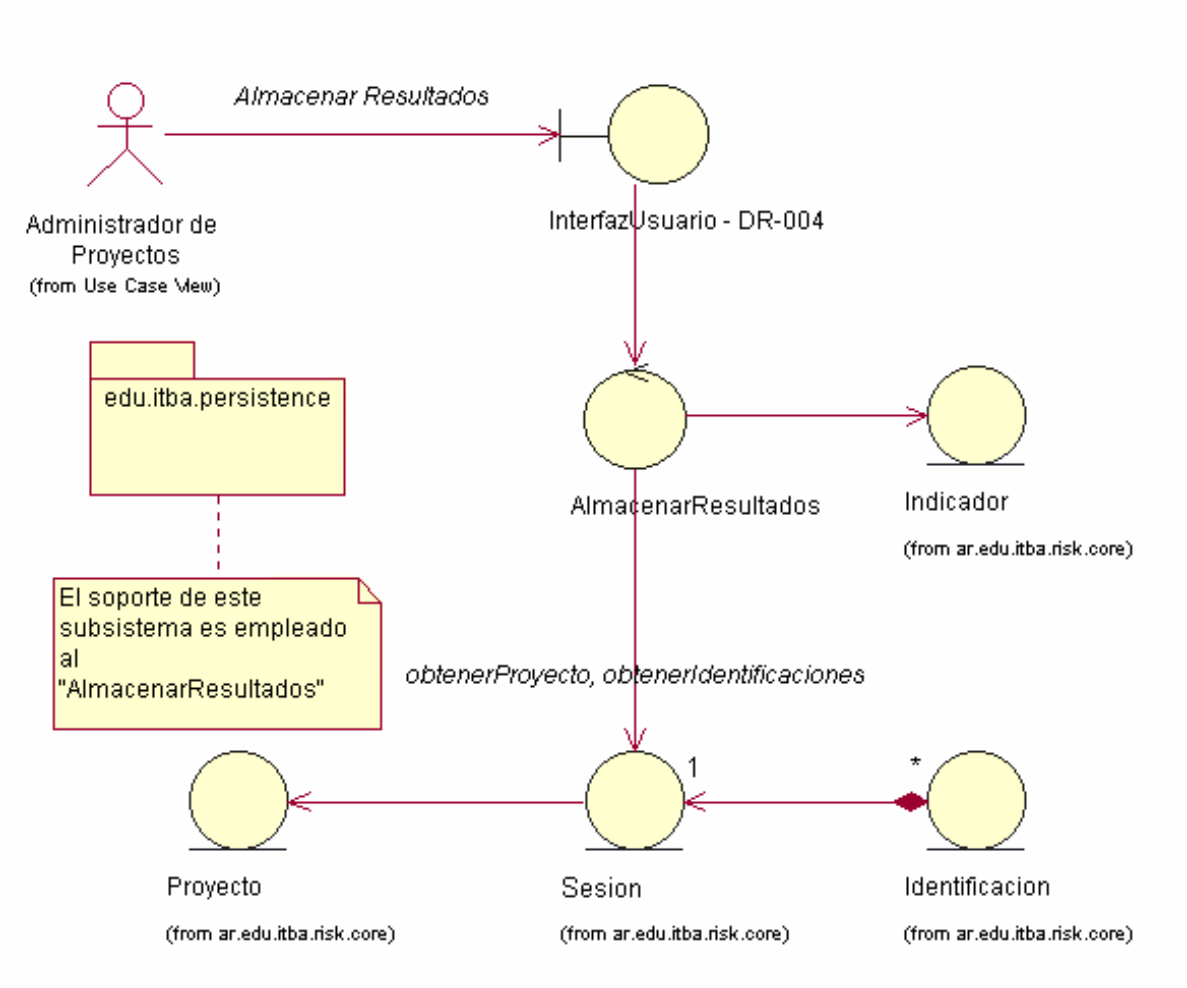


Figura 69 - Almacenando Resultados - DR-004

El Administrador de Proyectos solicita el almacenamiento de los resultados de un proceso de identificación de riesgos lo cual se realiza empleando el soporte brindado por el subsistema de persistencia.

A través de la sesión es posible acceder tanto al proyecto como a las identificaciones realizadas, mediante las cuales se puede determinar, para cada riesgo, el indicador asociado luego de la identificación.

Recuperando Resultados – DR-005

La siguiente figura (véase **Figura 70 - Recuperando Resultados - DR-005**) representa el diagrama de robustez para el caso de uso “Recuperando Resultados – CU-005”.

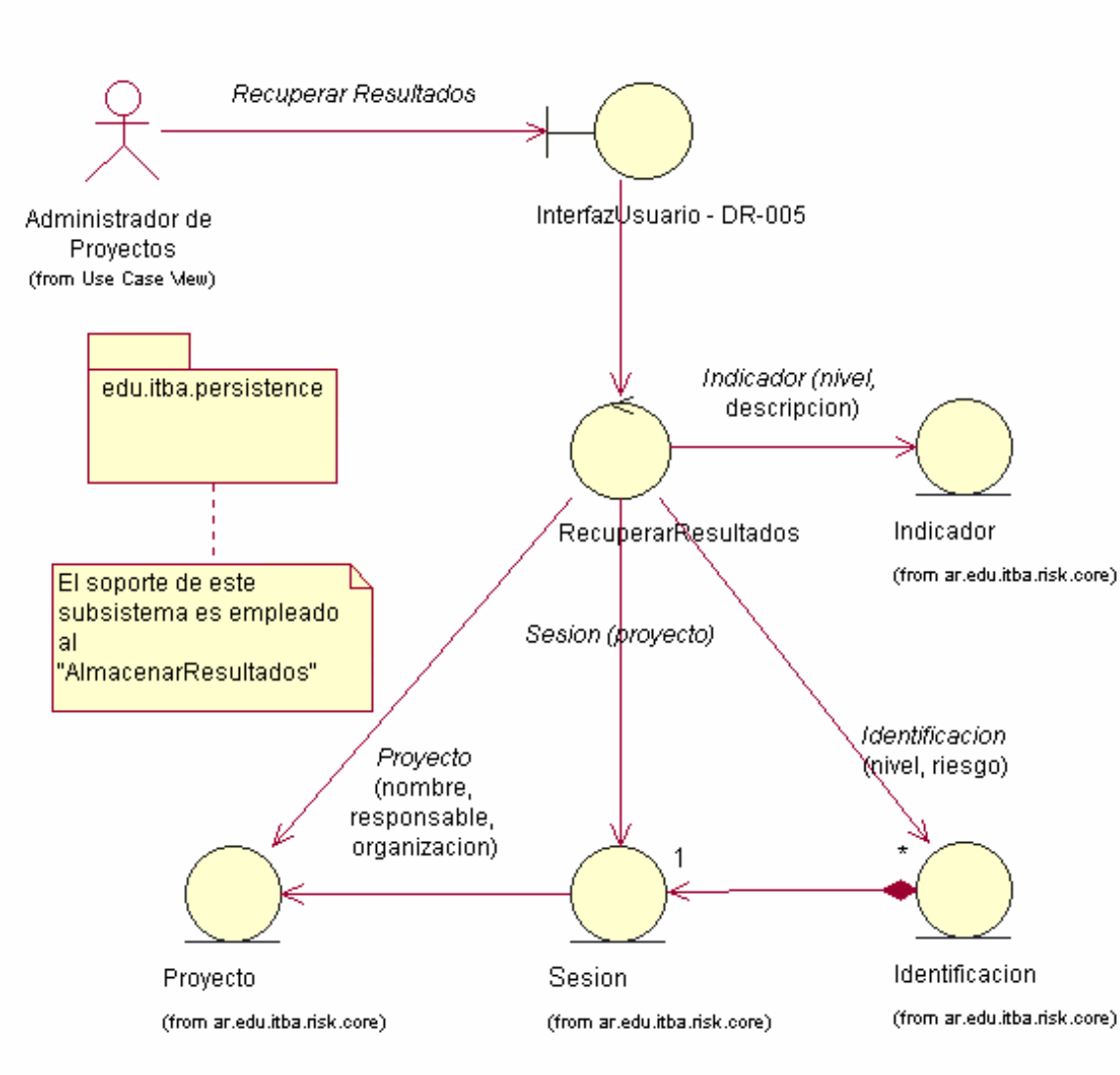


Figura 70 - Recuperando Resultados - DR-005

El Administrador de Proyectos solicita la recuperacióm de los resultados de un proceso de identificación de riesgos lo cual se realiza empleando el soporte brindado por el subsistema de persistencia.

Se deben crear nuevamente tanto la sesión, como el proyecto, las identificaciones realizadas y, para cada riesgo, el indicador asociado luego de la identificación.

Generando Reporte de Taxonomía – DR-006

La siguiente figura (véase **Figura 71 - Generando Reporte de Taxonomía - DR-006**) representa el diagrama de robustez para el caso de uso “Generando Reporte de Taxonomía – CU-006”.

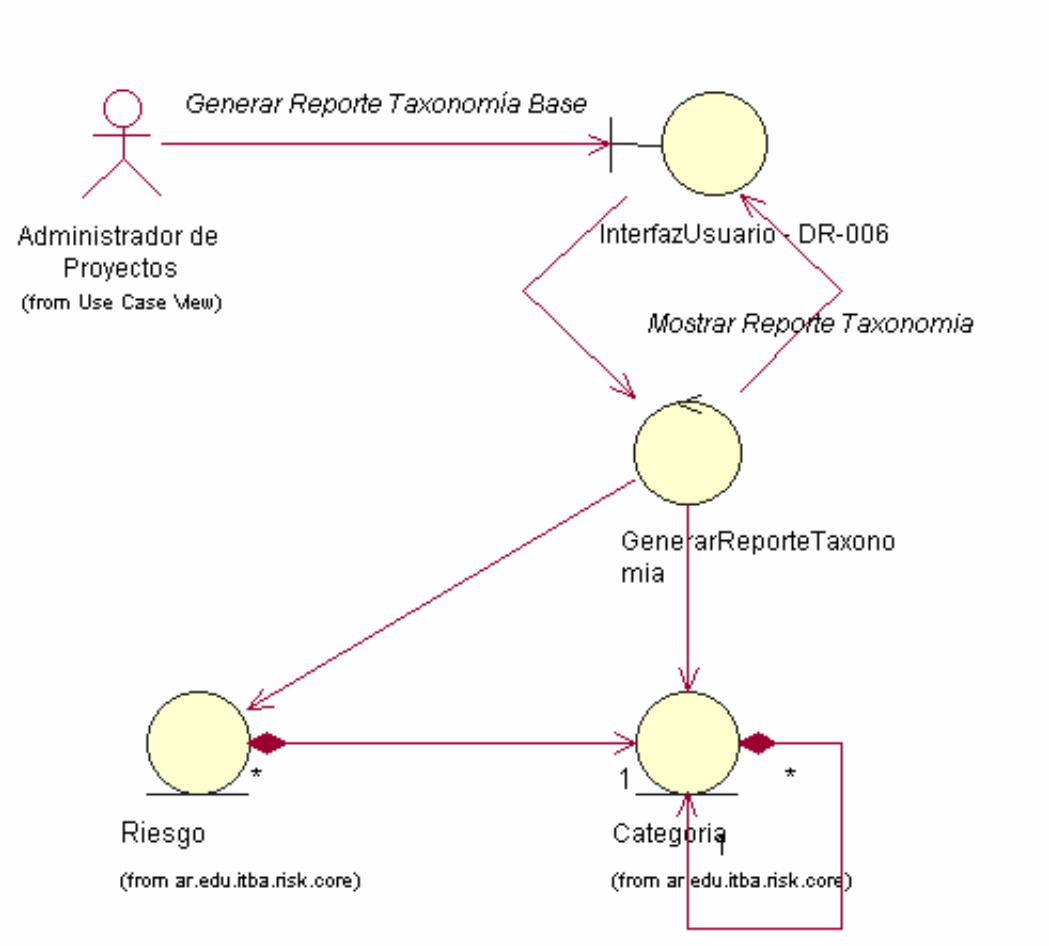


Figura 71 - Generando Reporte de Taxonomía - DR-006

El Administrador de Proyectos solicita la generación de un reporte de taxonomía base que es presentado al usuario mediante la interfaz gráfica definida.

Generando Reporte de Riesgos – DR-007

La siguiente figura (véase **Figura 72 - Generando Reporte de Riesgos - DR-007**) representa el diagrama de robustez para el caso de uso “Generando Reporte de Riesgos – CU-007”.

El Administrador de Proyectos solicita la generación de un reporte de taxonomía base que es presentado al usuario mediante la interfaz gráfica definida.

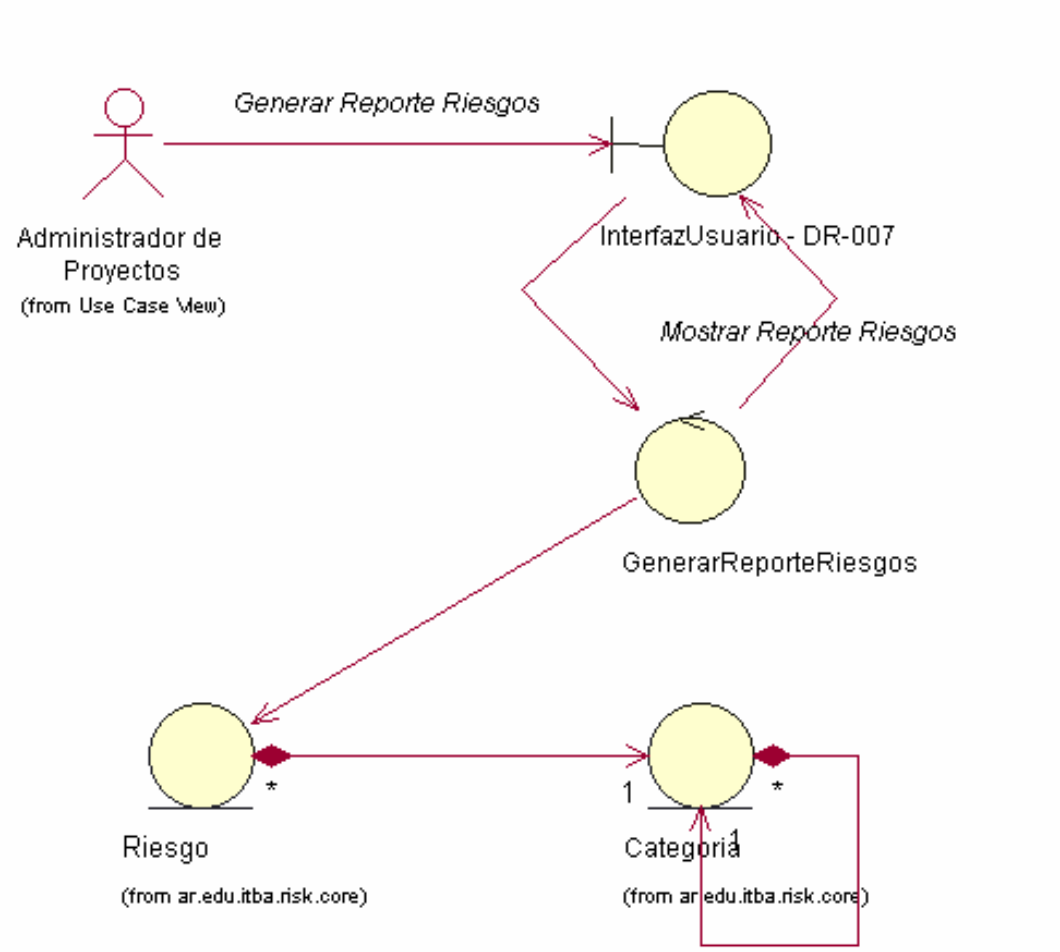


Figura 72 - Generando Reporte de Riesgos - DR-007

5.6.3. Revisión de la Interfaz de Usuario

Luego de concluidas las actividades antes descriptas, fueron revisadas y corregidas las interfaces de usuario considerando toda la información de diseño hasta aquí generada y el entorno tecnológico definido.

La siguiente lista enumera los aspectos que fueron revisados:

- La interfaz de usuario.
- La navegación entre ventanas.
- Los elementos que forman cada interfaz y sus características.
- La disposición y gestión de eventos relacionados con cada uno de los elementos de las ventanas.

Debido al hecho de que no se realizaron modificaciones significativas sobre la interfaz de usuario no fue necesario realizar una nueva validación de la misma ni la construcción de nuevos prototipos.

5.7. Reglas de Identificación de Riesgos

Empleando como base la Taxonomía Ajustada de Riesgos que fue generada durante la fase de análisis (véase **Apéndice F: Taxonomía de Riesgos Ajustada**), y utilizando nuevamente a un grupo de expertos como colaboradores, se llevó a cabo la generación de un conjunto de reglas que serán empleadas en el proceso de identificación de riesgos de Seldon.

El objetivo de esta actividad consistió en determinar de que forma podría ser deducible el valor aún desconocido de un riesgo a partir de valores ya otorgados a otros riesgos de la taxonomía, logrando de esta manera incrementar la automatización de las primeras actividades de identificación en base a los conocimientos ya adquiridos por un grupo de expertos.

Considerando lo antes expresado y revisando lo indicado en las fases anteriores del proceso de desarrollo del sistema, es posible afirmar que las actividades de identificación de riesgos pueden ser divididas en tres grupos:

1. Determinación del valor de un riesgo en base al Tipo de Proyecto que se evalúa.
2. Determinación del valor de un riesgo en base a las Reglas de Identificación de Riesgos.
3. Determinación del valor de un riesgo en base a las respuestas otorgadas por el usuario a las preguntas formuladas por la aplicación.

El siguiente gráfico (véase **Figura 73 - Actividades de Identificación de Riesgos en Seldon**) resume lo antes expresado:

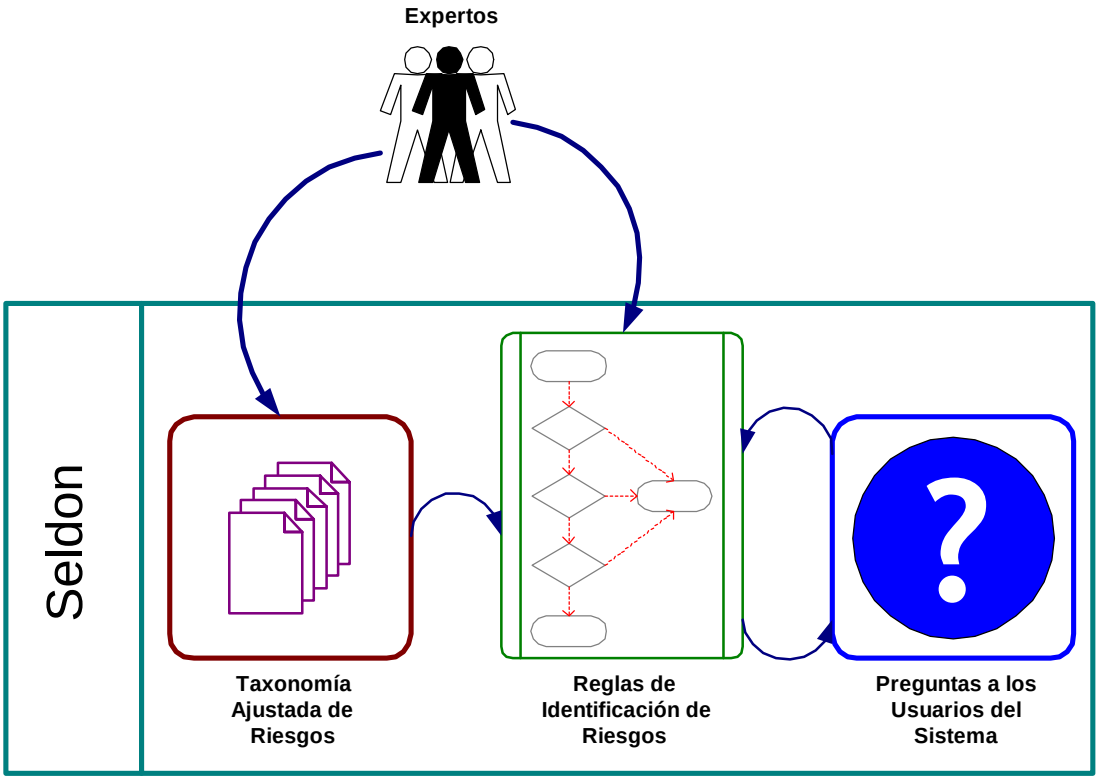


Figura 73 - Actividades de Identificación de Riesgos en Seldon

5.7.1. Seudoreglas:

Como parte de los componentes del sistema se empleará un conjunto de utilidades que brindan las facilidades típicamente provistas por los sistemas de producción con encadenamiento hacia delante; con el propósito de emplear y aplicar dichas facilidades, se detallan a continuación las reglas surgidas luego del análisis de dependencias entre riesgos realizada por los expertos:

Estado de la regla	Texto de la regla
Palabras de los expertos	Si el proyecto soporta directamente la misión y objetivos de la organización proveedora y hay cambios pequeños o nulos esperados en la estructura organizacional o hay cambios importantes esperados en la estructura organizacional y las condiciones del contrato son favorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo) y claramente se comunican los objetivos y el estado entre el equipo y el resto de la organización entonces suele suceder que la gerencia se encuentre fuertemente comprometida con el éxito del proyecto.
Formulación en la herramienta	**Si** *Elemento1* (*Probabilidad*) = “Baja” y (*Elemento18* (*Probabilidad*) = “Baja” o *Elemento18* (*Probabilidad*) = “Media”) y *Elemento85* (*Probabilidad*) = “Baja” y *Elemento92* (*Probabilidad*) = “Baja” **Entonces** *Elemento21* (*Probabilidad*) = “Baja”
Nombre	#1 – Elemento 21 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto no se relaciona con la misión y objetivos de la organización proveedora y hay cambios significativos esperados en la estructura organizacional y las condiciones del contrato son algo desfavorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo) o las condiciones del contrato son desfavorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo) y esporádicamente se comunican los objetivos y el estado entre el equipo y el resto de la organización entonces suele suceder que la gerencia no se encuentra comprometida con el éxito del proyecto.
Formulación en la herramienta	**Si** *Elemento1* (*Probabilidad*) = “Alta” y *Elemento18* (*Probabilidad*) = “Alta” y (*Elemento25* (*Probabilidad*) = “Media” o *Elemento25* (*Probabilidad*) = “Alta”) y *Elemento92* (*Probabilidad*) = “Alta” **Entonces** *Elemento21* (*Probabilidad*) = “Alta”
Nombre	#2 – Elemento 21 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando las estimaciones de tamaño, esfuerzo y costos se han realizado y validado en base a un proceso definido y apropiado que involucra el uso de diferentes técnicas o las estimaciones de tamaño, esfuerzo y costos se han realizado en base a un proceso definido pero no han sido validadas y los datos históricos tanto a nivel organizacional como industrial que se emplean para realizar estimaciones y determinar niveles esperados de productividad y calidad son confiables y se encuentran disponibles o los datos históricos tanto a nivel organizacional como industrial que se emplean para realizar estimaciones y determinar niveles esperados de productividad y calidad no son absolutamente confiables o no se encuentran totalmente disponibles y la fecha de entrega ha sido fijada en base a un proceso adecuado y apropiado para el proyecto o la fecha de entrega ha sido fijada parcialmente en base a la necesidad de alcanzar algún objetivo de negocio no necesariamente relacionado con las estimaciones técnicas y los individuos en la organización comprenden sus roles y responsabilidades y la de los demás y hay suficiente presupuesto asignado o hay un moderado presupuesto asignado y las fechas comprometidas de entrega estables o existen algunas fechas comprometidas de entrega inciertas y el proceso de desarrollo se adecua a las necesidades de proyecto y esta soportado por los métodos y herramientas seleccionados o el proceso de desarrollo se adecua a las necesidades de proyecto pero requiere de ciertas adaptaciones y esta soportado parcialmente por los métodos y herramientas seleccionados y claramente se comunican los objetivos y el estado entre el equipo y el resto de la organización y el administrador de proyectos posee formalmente la autoridad para ejercer un liderazgo efectivo y hay un alto grado de conocimiento en el dominio de la aplicación o hay alguna experiencia en el dominio de la aplicación o capacidad para contactar a los expertos entonces suele suceder que el equipo acuerda que el calendario es aceptable y puede ser cumplido.
Formulación en la herramienta	**Si** (*Elemento12* (*Probabilidad*) = “Baja” o *Elemento12* (*Probabilidad*) = “Media”) y (*Elemento13* (*Probabilidad*) = “Baja” o *Elemento13* (*Probabilidad*) = “Media”) y (*Elemento15* (*Probabilidad*) = “Baja” o *Elemento15* (*Probabilidad*) = “Media”) y *Elemento19* (*Probabilidad*) = “Baja” y (*Elemento33* (*Probabilidad*) = “Baja” o *Elemento33* (*Probabilidad*) = “Media”) y (*Elemento36* (*Probabilidad*) = “Baja” o *Elemento36* (*Probabilidad*) = “Media”) y (*Elemento68* (*Probabilidad*) = “Baja” o *Elemento68* (*Probabilidad*) = “Media”) y *Elemento92* (*Probabilidad*) = “Baja” y *Elemento95* (*Probabilidad*) = “Baja” y (*Elemento107* (*Probabilidad*) = “Baja” o *Elemento107*

	(*Probabilidad*) = “Media”) **Entonces** *Elemento37* (*Probabilidad*) = “Baja”
Nombre	#3 – Elemento 37 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando las estimaciones de tamaño, esfuerzo y costos han sido realizadas sin seguir un proceso formal y sin una validación final y no se emplean datos históricos para realizar estimaciones y determinar niveles esperados de productividad y calidad y la fecha de entrega ha sido fijada en base a la necesidad de alcanzar algún objetivo de negocio sin considerar las estimaciones técnicas y los individuos en la organización comprenden sus roles y responsabilidades pero no conocen claramente los roles y responsabilidades fuera de su grupo de trabajo inmediato o los individuos en la organización no comprenden sus roles y responsabilidades ni la de los demás y hay un escaso presupuesto asignado y las fechas de compromiso son inestables y fluctuantes y el proceso de desarrollo se adecua a las necesidades de proyecto pero requiere de ciertas adaptaciones y esta soportado parcialmente por los métodos y herramientas seleccionados o el proceso de desarrollo no se adecua a las necesidades de proyecto o no esta soportado por los métodos y herramientas seleccionados y esporádicamente se comunican los objetivos y el estado entre el equipo y el resto de la organización y el administrador de proyectos es capaz de ejercer ciertas influencias en la organización en base a sus relaciones personales o el administrador de proyectos no posee una autoridad formal ni cuenta con relaciones personales con los responsables por la toma de decisiones que le permitan ejercer sus funciones y hay nula experiencia en el dominio de la aplicación o sin capacidad para contactar a los expertos entonces suele suceder que el equipo encuentra el calendario de algunas de las fases del plan difícilmente alcanzable.
Formulación en la herramienta	**Si** *Elemento12* (*Probabilidad*) = “Alta” y *Elemento13* (*Probabilidad*) = “Alta” y *Elemento15* (*Probabilidad*) = “Alta” y (*Elemento19* (*Probabilidad*) = “Media” o *Elemento19* (*Probabilidad*) = “Alta”) y *Elemento33* (*Probabilidad*) = “Alta” y *Elemento36* (*Probabilidad*) = “Alta” y (*Elemento68* (*Probabilidad*) = “Media” o *Elemento68* (*Probabilidad*) = “Alta”) y *Elemento92* (*Probabilidad*) = “Alta” y (*Elemento95* (*Probabilidad*) = “Media” o *Elemento95* (*Probabilidad*) = “Alta”) y *Elemento107* (*Probabilidad*) = “Alta” **Entonces** *Elemento37* (*Probabilidad*) = “Alta”
Nombre	#4 – Elemento 37 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto soporta directamente la misión y objetivos de la organización proveedora y hay cambios pequeños o nulos esperados en la estructura organizacional y las políticas y estándares se encuentran cuidadosamente definidos y son seguidos de forma conveniente o las políticas y estándares se encuentran definidos pero no son seguidos de forma cuidadosa y hay un efectivo proceso de desarrollo establecido y respetado y no se han identificado interdependencias respecto a los procesos de la organización cliente entonces suele suceder que el proceso de desarrollo es definido, probado y con alto nivel de madurez.
Formulación en la herramienta	**Si** *Elemento2* (*Probabilidad*) = “Baja” y *Elemento18* (*Probabilidad*) = “Baja” y (*Elemento20* (*Probabilidad*) = “Baja” o *Elemento20* (*Probabilidad*) = “Media”) y *Elemento66* (*Probabilidad*) = “Baja” y *Elemento77* (*Probabilidad*) = “Baja” **Entonces** *Elemento67* (*Probabilidad*) = “Baja”
Nombre	#5 – Elemento 67 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando se prevén cambios importantes en el flujo de trabajo o se prevén cambios significativos en el flujo de trabajo y hay cambios importantes esperados en la estructura organizacional o hay cambios significativos esperados en la estructura organizacional y las políticas y estándares se encuentran definidos pero no son seguidos de forma cuidadosa o las políticas y estándares no se encuentran definidos o no son seguidos y hay un proceso de desarrollo establecido pero no empleado efectivamente o no hay un proceso de desarrollo establecido y se han identificado importantes interdependencias respecto a los procesos de la organización cliente entonces suele suceder que hay un nuevo proceso de desarrollo sin experiencias previas de aplicación o un proceso no definido.
Formulación en la herramienta	**Si** (*Elemento2* (*Probabilidad*) = “Media” o *Elemento2* (*Probabilidad*) = “Alta”) y (*Elemento18* (*Probabilidad*) = “Media” o *Elemento18* (*Probabilidad*) = “Alta”) y (*Elemento20* (*Probabilidad*) = “Media” o *Elemento20* (*Probabilidad*) = “Alta”) y (*Elemento66* (*Probabilidad*) = “Media” o *Elemento66* (*Probabilidad*) = “Alta”) y *Elemento77* (*Probabilidad*) = “Alta” **Entonces** *Elemento67* (*Probabilidad*) = “Alta”
Nombre	#6 – Elemento 67 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando los individuos en la organización comprenden sus roles y responsabilidades y la de los demás y las políticas y estándares se encuentran cuidadosamente definidos y son

	seguidos de forma conveniente y hay un efectivo proceso de desarrollo establecido y respetado y se ha definido un efectivo y consistente proceso de control de cambios entonces suele suceder que los cambios a los compromisos en cuanto a alcance, contenidos y calendario son revisados y aprobados por todos los involucrados.
Formulación en la herramienta	**Si** *Elemento19* (*Probabilidad*) = "Baja" y *Elemento20* (*Probabilidad*) = "Baja" y *Elemento66* (*Probabilidad*) = "Baja" y *Elemento75* (*Probabilidad*) = "Baja" **Entonces** *Elemento70* (*Probabilidad*) = "Baja"
Nombre	#7 – Elemento 70 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando los individuos en la organización comprenden sus roles y responsabilidades pero no conocen claramente los roles y responsabilidades fuera de su grupo de trabajo inmediato o los individuos en la organización no comprenden sus roles y responsabilidades ni la de los demás y las políticas y estándares se encuentran definidos pero no son seguidos de forma cuidadosa o las políticas y estándares no se encuentran definidos o no son seguidos y hay un proceso de desarrollo establecido pero no empleado efectivamente o no hay un proceso de desarrollo establecido y se ha definido un proceso de control de cambios pero es usado de forma inconsistente o no se ha definido un proceso de control de cambios entonces suele suceder que los cambios a los compromisos en cuanto a alcance, contenidos y calendario son realizados sin participar a los involucrados.
Formulación en la herramienta	**Si** (*Elemento19* (*Probabilidad*) = "Media" o *Elemento19* (*Probabilidad*) = "Alta") y (*Elemento20* (*Probabilidad*) = "Media" o *Elemento20* (*Probabilidad*) = "Alta") y (*Elemento66* (*Probabilidad*) = "Media" o *Elemento66* (*Probabilidad*) = "Alta") y (*Elemento75* (*Probabilidad*) = "Media" o *Elemento75* (*Probabilidad*) = "Alta") **Entonces** *Elemento70* (*Probabilidad*) = "Alta"
Nombre	#8 – Elemento 70 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando las políticas y estándares se encuentran cuidadosamente definidos y son seguidos de forma conveniente y hay un efectivo proceso de desarrollo establecido y respetado y hay un proceso de desarrollo definido, probado y con alto nivel de madurez y los cambios a los compromisos en cuanto a alcance, contenidos y calendario son revisados y aprobados por todos los involucrados y se produce una incorporación permanente de un proceso de revisión de pares o se produce una incorporación esporádica y selectiva de un proceso de

	revisión de pares y se ha definido un efectivo y consistente proceso de seguimiento de defectos o se ha definido un proceso de seguimiento de defectos pero es usado de forma inconsistente y se ha definido un efectivo y consistente proceso de control de cambios y las métricas a colectar se encuentran claramente definidas y son fácilmente colectables en base a los procesos y herramientas con las que se cuenta o las métricas a colectar se definidas y en su mayoría son colectables en base a los procesos y herramientas con las que se cuenta y hay un alto grado de experiencia con el proceso o hay alguna experiencia con el proceso entonces suele suceder que existe un efectivo sistema de aseguramiento de la calidad establecido y respetado.
Formulación en la herramienta	**Si** *Elemento20* (*Probabilidad*) = "Baja" y *Elemento66* (*Probabilidad*) = "Baja" y *Elemento67* (*Probabilidad*) = "Baja" y *Elemento70* (*Probabilidad*) = "Baja" y (*Elemento73* (*Probabilidad*) = "Baja" o *Elemento73* (*Probabilidad*) = "Media") y (*Elemento74* (*Probabilidad*) = "Baja" o *Elemento74* (*Probabilidad*) = "Media") y *Elemento75* (*Probabilidad*) = "Baja" y **(***Elemento76* (*Probabilidad*) = "Baja" o *Elemento76* (*Probabilidad*) = "Media") y (*Elemento103* (*Probabilidad*) = "Baja" o *Elemento103* (*Probabilidad*) = "Media") **Entonces** *Elemento71* (*Probabilidad*) = "Baja"
Nombre	#9 – Elemento 71 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando las políticas y estándares se encuentran definidos pero no son seguidos de forma cuidadosa o las políticas y estándares no se encuentran definidos o no son seguidos y no hay un proceso de desarrollo establecido y hay un proceso de desarrollo definido, probado pero con importantes mejoras requeridas o hay un nuevo proceso de desarrollo sin experiencias previas de aplicación o proceso no definido y los cambios a los compromisos en cuanto a alcance, contenidos y calendario son realizados sin participar a los involucrados y el proceso de revisión de pares no es incorporado y no se ha definido un proceso para realizar el seguimiento de defectos y no se ha definido un proceso de control de cambios y las métricas a colectar no se encuentran correctamente definidas o no son colectables en base a los procesos y herramientas con las que se cuenta y hay escasa o ninguna experiencia con el proceso entonces suele suceder que el sistema de aseguramiento de la calidad o procesos no este establecido.
Formulación en la herramienta	**Si** (*Elemento20* (*Probabilidad*) = "Media" o *Elemento20* (*Probabilidad*) = "Alta") y *Elemento66* (*Probabilidad*) = "Alta" y **(***Elemento67* (*Probabilidad*) = "Media" o *Elemento67* (*Probabilidad*) = "Alta") y *Elemento70* (*Probabilidad*) = "Alta" y *Elemento73* (*Probabilidad*) = "Alta" y *Elemento74* (*Probabilidad*) = "Alta" y *Elemento75* (*Probabilidad*) = "Alta" y

	Elemento76 (*Probabilidad*) = “Alta” y *Elemento103* (*Probabilidad*) = “Alta” **Entonces** *Elemento71* (*Probabilidad*) = “Alta”
Nombre	#10 – Elemento 71 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto soporta directamente la misión y objetivos de la organización proveedora o el proyecto impacta indirectamente en uno o mas de los objetivos definidos por la organización proveedora y el cliente esperaba que esta organización sea la proveedora del producto del proyecto y el proyecto soporta directamente la misión y objetivos de la organización cliente o el proyecto impacta indirectamente en uno o mas de los objetivos definidos por la organización cliente y no se han tomado para el proyecto decisiones motivadas por influencias políticas y la gerencia se encuentra fuertemente comprometida con el éxito del proyecto y el soporte ejecutivo es visible y fuerte y los objetivos de proyecto son verificables para requerimientos razonables y hay fondos asignados sin restricciones o hay algunas dudas respecto a la disponibilidad de fondos y el administrador de proyectos posee formalmente la autoridad para ejercer un liderazgo efectivo entonces suele suceder que el administrador de proyectos esta fuertemente comprometido con éxito del proyecto.
Formulación en la herramienta	**Si** (*Elemento1* (*Probabilidad*) = “Baja” o *Elemento1* (*Probabilidad*) = “Media”) y *Elemento3* (*Probabilidad*) = “Baja” y (*Elemento4* (*Probabilidad*) = “Baja” o *Elemento4* (*Probabilidad*) = “Media”) y *Elemento11* (*Probabilidad*) = “Baja” y *Elemento21* (*Probabilidad*) = “Baja” y *Elemento22* (*Probabilidad*) = “Baja” y *Elemento23* (*Probabilidad*) = “Baja” y (*Elemento34* (*Probabilidad*) = “Baja” o *Elemento34* (*Probabilidad*) = “Media”) y *Elemento95* (*Probabilidad*) = “Baja” **Entonces** *Elemento94* (*Probabilidad*) = “Baja”
Nombre	#11 – Elemento 94 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto no se relaciona con la misión y objetivos de la organización proveedora y el proyecto no coincide con productos o servicios previos de esta organización y el proyecto no se relaciona con la misión y objetivos de la organización cliente y se han tomado para el proyecto algunas decisiones motivadas por influencias políticas o se han tomado para el proyecto una gran cantidad de decisiones motivadas por influencias políticas y la gerencia no se encuentra comprometida con el éxito del proyecto y el soporte ejecutivo es ocasional y ante situaciones especiales o no existe soporte ejecutivo aún en situaciones especiales y

	los objetivos de proyecto no han sido definidos o no son medibles y las asignaciones presupuestarias en duda o sujetas a cambiar sin notificación previa y el administrador de proyectos es capaz de ejercer ciertas influencias en la organización en base a sus relaciones personales o el administrador de proyectos no posee una autoridad formal ni cuenta con relaciones personales con los responsables por la toma de decisiones que le permitan ejercer sus funciones entonces suele suceder que el administrador de proyectos esta escasamente comprometido con éxito del proyecto.
Formulación en la herramienta	**Si** *Elemento1* (*Probabilidad*) = "Alta" y *Elemento3* (*Probabilidad*) = "Alta" y *Elemento4* (*Probabilidad*) = "Alta" y (*Elemento11* (*Probabilidad*) = "Media" o *Elemento11* (*Probabilidad*) = "Alta") y *Elemento21* (*Probabilidad*) = "Alta" y (*Elemento22* (*Probabilidad*) = "Media" o *Elemento22* (*Probabilidad*) = "Alta") y *Elemento23* (*Probabilidad*) = "Alta" y *Elemento34* (*Probabilidad*) = "Alta" y (*Elemento95* (*Probabilidad*) = "Media" o *Elemento95* (*Probabilidad*) = "Alta") **Entonces** *Elemento94* (*Probabilidad*) = "Baja"
Nombre	#12 – Elemento 94 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando los individuos en la organización comprenden sus roles y responsabilidades y la de los demás y la gerencia se encuentra fuertemente comprometida con el éxito del proyecto y el equipo acuerda que el calendario es aceptable y puede ser cumplido y claramente se comunican los objetivos y el estado entre el equipo y el resto de la organización o se comunica parte de la información solo en algunas situaciones especiales y el administrador de proyectos esta fuertemente comprometido con éxito del proyecto y el administrador de proyectos posee formalmente la autoridad para ejercer un liderazgo efectivo y la comunicación interna del equipo es clara y fluida entonces suele suceder que el administrador de proyecto posee un completo soporte del equipo de proyecto.
Formulación en la herramienta	**Si** *Elemento19* (*Probabilidad*) = "Baja" y *Elemento21* (*Probabilidad*) = "Baja" y *Elemento37* (*Probabilidad*) = "Baja" y (*Elemento92* (*Probabilidad*) = "Baja" o *Elemento92* (*Probabilidad*) = "Media") y *Elemento94* (*Probabilidad*) = "Baja" y *Elemento95* (*Probabilidad*) = "Baja" y *Elemento100* (*Probabilidad*) = "Baja" **Entonces** *Elemento96* (*Probabilidad*) = "Baja"
Nombre	#13 – Elemento 96 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando los individuos en la organización comprenden sus roles y responsabilidades pero no conocen claramente los roles y responsabilidades fuera de su grupo de trabajo

	inmediato o los individuos en la organización no comprenden sus roles y responsabilidades ni la de los demás y la gerencia se encuentra comprometida con el éxito del proyecto pero no totalmente o la gerencia no se encuentra comprometida con el éxito del proyecto y el equipo encuentra el calendario de algunas de las fases del plan difícilmente alcanzable y esporádicamente se comunican los objetivos y el estado entre el equipo y el resto de la organización y el administrador de proyectos esta limitadamente comprometido con éxito del proyecto o el administrador de proyectos esta escasamente comprometido con éxito del proyecto y el administrador de proyectos no posee una autoridad formal ni cuenta con relaciones personales con los responsables por la toma de decisiones que le permitan ejercer sus funciones y la comunicación interna del equipo es adecuada pero escasa en ciertas situaciones o la comunicación interna del equipo es confusa o inexistente entonces suele suceder que el administrador de proyecto esta sin soporte del equipo de proyecto.
Formulación en la herramienta	**Si** (*Elemento19* (*Probabilidad*) = "Media" o *Elemento19* (*Probabilidad*) = "Alta") y (*Elemento21* (*Probabilidad*) = "Media" o *Elemento21* (*Probabilidad*) = "Alta") y *Elemento37* (*Probabilidad*) = "Alta" y *Elemento92* (*Probabilidad*) = "Alta" y (*Elemento94* (*Probabilidad*) = "Media" o *Elemento94* (*Probabilidad*) = "Alta") y *Elemento95* (*Probabilidad*) = "Alta" y (*Elemento100* (*Probabilidad*) = "Media" o *Elemento100* (*Probabilidad*) = "Alta") **Entonces** *Elemento96* (*Probabilidad*) = "Alta"
Nombre	#14 – Elemento 96 – Caracterización Alta

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto soporta directamente la misión y objetivos de la organización proveedora y los proyectos del programa comparten recursos sin ningún conflicto y los individuos en la organización comprenden sus roles y responsabilidades y la de los demás y la gerencia se encuentra fuertemente comprometida con el éxito del proyecto y el equipo acuerda que el calendario es aceptable y puede ser cumplido y claramente se comunican los objetivos y el estado entre el equipo y el resto de la organización o se comunica parte de la información solo en algunas situaciones especiales y el administrador de proyectos esta fuertemente comprometido con éxito del proyecto y la comunicación interna del equipo es clara y fluida entonces suele suceder que el equipo esta fuertemente comprometido con éxito del proyecto y es cooperativo.
Formulación en la herramienta	**Si** *Elemento1* (*Probabilidad*) = "Baja" y *Elemento6* (*Probabilidad*) = "Baja" y *Elemento19* (*Probabilidad*) = "Baja" y *Elemento21* (*Probabilidad*) = "Baja" y *Elemento37*

	(*Probabilidad*) = “Baja” y (*Elemento92* (*Probabilidad*) = “Baja” o *Elemento92* (*Probabilidad*) = “Media”) y *Elemento94* (*Probabilidad*) = “Baja” y *Elemento100* (*Probabilidad*) = “Baja” **Entonces** *Elemento105* (*Probabilidad*) = “Baja”
Nombre	#15 – Elemento 105 – Caracterización Baja

Estado de la regla	Texto de la regla
Palabras de los expertos	Cuando el proyecto impacta indirectamente en uno o mas de los objetivos definidos por la organización proveedora o el proyecto no se relaciona con la misión y objetivos de la organización proveedora y los proyectos del programa calendarizan el uso de los recursos de forma tal de evitar conflictos o los proyectos del programa frecuentemente necesitan los mismos recursos al mismo momento y los individuos en la organización no comprenden sus roles y responsabilidades ni la de los demás y la gerencia se encuentra comprometida con el éxito del proyecto pero no totalmente o la gerencia no se encuentra comprometida con el éxito del proyecto y el equipo encuentra el calendario de alguna de las fases del plan demasiado agresivo o el equipo encuentra el calendario de algunas de las fases del plan difícilmente alcanzable y esporádicamente se comunican los objetivos y el estado entre el equipo y el resto de la organización y el administrador de proyectos esta limitadamente comprometido con éxito del proyecto o el administrador de proyectos esta escasamente comprometido con éxito del proyecto y la comunicación interna del equipo es adecuada pero escasa en ciertas situaciones o la comunicación interna del equipo es confusa o inexistente entonces suele suceder que el equipo de proyecto esta escasamente comprometido con éxito del proyecto y poco cohesivo.
Formulación en la herramienta	**Si** (*Elemento1* (*Probabilidad*) = “Media” o *Elemento1* (*Probabilidad*) = “Alta”) y (*Elemento6* (*Probabilidad*) = “Media” o *Elemento6* (*Probabilidad*) = “Alta”) y *Elemento19* (*Probabilidad*) = “Alta” y (*Elemento21* (*Probabilidad*) = “Media” o *Elemento21* (*Probabilidad*) = “Alta”) y (*Elemento37* (*Probabilidad*) = “Media” o *Elemento37* (*Probabilidad*) = “Alta”) y *Elemento92* (*Probabilidad*) = “Alta” y (*Elemento94* (*Probabilidad*) = “Media” o *Elemento94* (*Probabilidad*) = “Alta”) y (*Elemento100* (*Probabilidad*) = “Media” o *Elemento100* (*Probabilidad*) = “Alta”) **Entonces** *Elemento105* (*Probabilidad*) = “Alta”
Nombre	#16 – Elemento 105 – Caracterización Alta

Se presenta a continuación la representación de las reglas antes descriptas en la sintaxis requerida por JEOPS:

```
package controller;

import model.Riesgo;

/**
 * Seudoreglas para el análisis de dependencia entre riesgos
 * utilizando JEOPS.
 *
*/
public ruleBase Seudoreglas {

      rule _1_elemento21_baja {
            declarations
                  FiltroRiesgos f;
            conditions
                  f.getIdentificacion( 21 ) != null;
                  f.getIdentificacion(     21     ).getNivel()     ==
                  Riesgo.NO_DETERMINADO;

                  f.getIdentificacion( 1 ) != null;
                  f.getIdentificacion( 18 ) != null;
                  f.getIdentificacion( 85 ) != null;
                  f.getIdentificacion( 92 ) != null;

                  f.getIdentificacion( 1 ).getNivel() == Riesgo.BAJO;
                  f.getIdentificacion( 18 ).getNivel() == Riesgo.BAJO
                  ||    f.getIdentificacion(    18    ).getNivel()    ==
                  Riesgo.MEDIO;
                  f.getIdentificacion( 85 ).getNivel() == Riesgo.BAJO;
                  f.getIdentificacion( 92 ).getNivel() == Riesgo.BAJO;
            actions
                  f.getIdentificacion( 21 ).setNivel( Riesgo.BAJO );
                  modified(f);
      }

      rule _2_elemento21_alta {
            declarations
                  FiltroRiesgos f;
            conditions
                  f.getIdentificacion( 21 ) != null;
                  f.getIdentificacion(     21     ).getNivel()     ==
                  Riesgo.NO_DETERMINADO;

                  f.getIdentificacion( 1 ) != null;
                  f.getIdentificacion( 18 ) != null;
                  f.getIdentificacion( 25 ) != null;
                  f.getIdentificacion( 92 ) != null;

                  f.getIdentificacion( 1 ).getNivel() == Riesgo.ALTO;
                  f.getIdentificacion( 18 ).getNivel() == Riesgo.ALTO;
                  f.getIdentificacion( 25 ).getNivel() == Riesgo.ALTO
                  ||    f.getIdentificacion(    25    ).getNivel()    ==
                  Riesgo.MEDIO;
                  f.getIdentificacion( 92 ).getNivel() == Riesgo.ALTO;
            actions
                  f.getIdentificacion( 21 ).setNivel( Riesgo.ALTO );
                  modified(f);
      }

      rule _3_elemento37_baja {
            declarations
```

```
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 37 ) != null;
            f.getIdentificacion(     37     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 12 ) != null;
            f.getIdentificacion( 13 ) != null;
            f.getIdentificacion( 15 ) != null;
            f.getIdentificacion( 19 ) != null;
            f.getIdentificacion( 33 ) != null;
            f.getIdentificacion( 36 ) != null;
            f.getIdentificacion( 68 ) != null;
            f.getIdentificacion( 92 ) != null;
            f.getIdentificacion( 95 ) != null;
            f.getIdentificacion( 107 ) != null;

            f.getIdentificacion( 12 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   12   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 13 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   13   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 15 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   15   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 19 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 33 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   33   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 36 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   36   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 68 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   68   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 92 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 95 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 107 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   107   ).getNivel()   ==
            Riesgo.MEDIO;
      actions
            f.getIdentificacion( 37 ).setNivel( Riesgo.BAJO );
            modified(f);
}

rule _4_elemento37_alta {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 37 ) != null;
            f.getIdentificacion(     37     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 12 ) != null;
            f.getIdentificacion( 13 ) != null;
            f.getIdentificacion( 15 ) != null;
            f.getIdentificacion( 19 ) != null;
            f.getIdentificacion( 33 ) != null;
            f.getIdentificacion( 36 ) != null;
            f.getIdentificacion( 68 ) != null;
```

```
            f.getIdentificacion( 92 ) != null;
            f.getIdentificacion( 95 ) != null;
            f.getIdentificacion( 107 ) != null;

            f.getIdentificacion( 12 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 13 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 15 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 19 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   19   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 33 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 36 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 68 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   68   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 92 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 95 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   95   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion(    107    ).getNivel()    ==
            Riesgo.ALTO;
      actions
            f.getIdentificacion( 37 ).setNivel( Riesgo.ALTO );
            modified(f);
}

rule _5_elemento67_baja {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 67 ) != null;
            f.getIdentificacion(    67    ).getNivel()    ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 2 ) != null;
            f.getIdentificacion( 18 ) != null;
            f.getIdentificacion( 20 ) != null;
            f.getIdentificacion( 66 ) != null;
            f.getIdentificacion( 77 ) != null;

            f.getIdentificacion( 2 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 18 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 20 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   20   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 66 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 77 ).getNivel() == Riesgo.BAJO;
      actions
            f.getIdentificacion( 67 ).setNivel( Riesgo.BAJO );
            modified(f);
}

rule _6_elemento67_alta {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 67 ) != null;
            f.getIdentificacion(    67    ).getNivel()    ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 2 ) != null;
```

```
            f.getIdentificacion( 18 ) != null;
            f.getIdentificacion( 20 ) != null;
            f.getIdentificacion( 66 ) != null;
            f.getIdentificacion( 77 ) != null;

            f.getIdentificacion( 2 ).getNivel() == Riesgo.ALTO
            || f.getIdentificacion( 2 ).getNivel() ==
            Riesgo.MEDIO;
            f.getIdentificacion( 18 ).getNivel() == Riesgo.ALTO
            || f.getIdentificacion( 18 ).getNivel() ==
            Riesgo.MEDIO;
            f.getIdentificacion( 20 ).getNivel() == Riesgo.ALTO
            || f.getIdentificacion( 20 ).getNivel() ==
            Riesgo.MEDIO;
            f.getIdentificacion( 66 ).getNivel() == Riesgo.ALTO
            || f.getIdentificacion( 66 ).getNivel() ==
            Riesgo.MEDIO;
            f.getIdentificacion( 77 ).getNivel() == Riesgo.ALTO;
      actions
            f.getIdentificacion( 67 ).setNivel( Riesgo.ALTO );
            modified(f);
}

rule _7_elemento70_baja {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 70 ) != null;
            f.getIdentificacion( 70 ).getNivel() ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 19 ) != null;
            f.getIdentificacion( 20 ) != null;
            f.getIdentificacion( 66 ) != null;
            f.getIdentificacion( 75 ) != null;

            f.getIdentificacion( 19 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 20 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 66 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 75 ).getNivel() == Riesgo.BAJO;
      actions
            f.getIdentificacion( 70 ).setNivel( Riesgo.BAJO );
            modified(f);
}

rule _8_elemento70_alta {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 70 ) != null;
            f.getIdentificacion( 70 ).getNivel() ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 19 ) != null;
            f.getIdentificacion( 20 ) != null;
            f.getIdentificacion( 66 ) != null;
            f.getIdentificacion( 75 ) != null;

            f.getIdentificacion( 19 ).getNivel() == Riesgo.ALTO
            || f.getIdentificacion( 19 ).getNivel() ==
            Riesgo.MEDIO;
```

```
            f.getIdentificacion( 20 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   20   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 66 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   66   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 75 ).getNivel() == Riesgo.ALTO
            ||   f.getIdentificacion(   75   ).getNivel()   ==
            Riesgo.MEDIO;
      actions
            f.getIdentificacion( 70 ).setNivel( Riesgo.ALTO );
            modified(f);
}

rule _9_elemento71_baja {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 71 ) != null;
            f.getIdentificacion(     71     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 20 ) != null;
            f.getIdentificacion( 66 ) != null;
            f.getIdentificacion( 67 ) != null;
            f.getIdentificacion( 70 ) != null;
            f.getIdentificacion( 73 ) != null;
            f.getIdentificacion( 74 ) != null;
            f.getIdentificacion( 75 ) != null;
            f.getIdentificacion( 76 ) != null;
            f.getIdentificacion( 103 ) != null;

            f.getIdentificacion( 20 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 66 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 67 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 70 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 73 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   73   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 74 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   74   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 75 ).getNivel() == Riesgo.BAJO;
            f.getIdentificacion( 76 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   76   ).getNivel()   ==
            Riesgo.MEDIO;
            f.getIdentificacion( 103 ).getNivel() == Riesgo.BAJO
            ||   f.getIdentificacion(   103   ).getNivel()   ==
            Riesgo.MEDIO;
      actions
            f.getIdentificacion( 71 ).setNivel( Riesgo.BAJO );
            modified(f);
}

rule _10_elemento71_alta {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 71 ) != null;
            f.getIdentificacion(     71     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;
```

```
                f.getIdentificacion( 20 ) != null;
                f.getIdentificacion( 66 ) != null;
                f.getIdentificacion( 67 ) != null;
                f.getIdentificacion( 70 ) != null;
                f.getIdentificacion( 73 ) != null;
                f.getIdentificacion( 74 ) != null;
                f.getIdentificacion( 75 ) != null;
                f.getIdentificacion( 76 ) != null;
                f.getIdentificacion( 103 ) != null;

                f.getIdentificacion( 20 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   20   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 66 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 67 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   67   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 70 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 73 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 74 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 75 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 76 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion(    103    ).getNivel()    ==
Riesgo.ALTO;
          actions
                f.getIdentificacion( 71 ).setNivel( Riesgo.ALTO );
                modified(f);
     }

     rule _11_elemento94_baja {
          declarations
                FiltroRiesgos f;
          conditions
                f.getIdentificacion( 94 ) != null;
                f.getIdentificacion(    94    ).getNivel()    ==
                Riesgo.NO_DETERMINADO;

                f.getIdentificacion( 1 ) != null;
                f.getIdentificacion( 3 ) != null;
                f.getIdentificacion( 4 ) != null;
                f.getIdentificacion( 11 ) != null;
                f.getIdentificacion( 21 ) != null;
                f.getIdentificacion( 22 ) != null;
                f.getIdentificacion( 23 ) != null;
                f.getIdentificacion( 34 ) != null;
                f.getIdentificacion( 95 ) != null;

                f.getIdentificacion( 1 ).getNivel() == Riesgo.BAJO
                ||   f.getIdentificacion(   1   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 3 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 4 ).getNivel() == Riesgo.BAJO
                ||   f.getIdentificacion(   4   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 11 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 21 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 22 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 23 ).getNivel() == Riesgo.BAJO;
```

```
            f.getIdentificacion( 34 ).getNivel() == Riesgo.BAJO
            ||    f.getIdentificacion(    34    ).getNivel()    ==
            Riesgo.MEDIO;
            f.getIdentificacion( 95 ).getNivel() == Riesgo.BAJO;
      actions
            f.getIdentificacion( 94 ).setNivel( Riesgo.BAJO );
            modified(f);
}

rule _12_elemento94_alta {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 94 ) != null;
            f.getIdentificacion(     94     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 1 ) != null;
            f.getIdentificacion( 3 ) != null;
            f.getIdentificacion( 4 ) != null;
            f.getIdentificacion( 11 ) != null;
            f.getIdentificacion( 21 ) != null;
            f.getIdentificacion( 22 ) != null;
            f.getIdentificacion( 23 ) != null;
            f.getIdentificacion( 34 ) != null;
            f.getIdentificacion( 95 ) != null;

            f.getIdentificacion( 1 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 3 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 4 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 11 ).getNivel() == Riesgo.ALTO
            ||    f.getIdentificacion(    11    ).getNivel()    ==
            Riesgo.MEDIO;
            f.getIdentificacion( 21 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 22 ).getNivel() == Riesgo.ALTO
            ||    f.getIdentificacion(    22    ).getNivel()    ==
            Riesgo.MEDIO;
            f.getIdentificacion( 23 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 34 ).getNivel() == Riesgo.ALTO;
            f.getIdentificacion( 95 ).getNivel() == Riesgo.ALTO
            ||    f.getIdentificacion(    95    ).getNivel()    ==
            Riesgo.MEDIO;
      actions
            f.getIdentificacion( 94 ).setNivel( Riesgo.ALTO );
            modified(f);
}

rule _13_elemento96_baja {
      declarations
            FiltroRiesgos f;
      conditions
            f.getIdentificacion( 96 ) != null;
            f.getIdentificacion(     96     ).getNivel()     ==
            Riesgo.NO_DETERMINADO;

            f.getIdentificacion( 19 ) != null;
            f.getIdentificacion( 21 ) != null;
            f.getIdentificacion( 37 ) != null;
            f.getIdentificacion( 92 ) != null;
            f.getIdentificacion( 94 ) != null;
            f.getIdentificacion( 95 ) != null;
```

```
                f.getIdentificacion( 100 ) != null;

                f.getIdentificacion( 19 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 21 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 37 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 92 ).getNivel() == Riesgo.BAJO
                ||   f.getIdentificacion(   92   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 94 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 95 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion(    100    ).getNivel()    ==
Riesgo.BAJO;
          actions
                f.getIdentificacion( 96 ).setNivel( Riesgo.BAJO );
                modified(f);
     }

     rule _14_elemento96_alta {
          declarations
                FiltroRiesgos f;
          conditions
                f.getIdentificacion( 96 ) != null;
                f.getIdentificacion(    96    ).getNivel()    ==
                Riesgo.NO_DETERMINADO;

                f.getIdentificacion( 19 ) != null;
                f.getIdentificacion( 21 ) != null;
                f.getIdentificacion( 37 ) != null;
                f.getIdentificacion( 92 ) != null;
                f.getIdentificacion( 94 ) != null;
                f.getIdentificacion( 95 ) != null;
                f.getIdentificacion( 100 ) != null;

                f.getIdentificacion( 19 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   19   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 21 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   21   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 37 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 92 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 94 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   94   ).getNivel()   ==
                Riesgo.MEDIO;
                f.getIdentificacion( 95 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 100 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   100   ).getNivel()   ==
                Riesgo.MEDIO;
          actions
                f.getIdentificacion( 96 ).setNivel( Riesgo.ALTO );
                modified(f);
     }

     rule _15_elemento105_baja {
          declarations
                FiltroRiesgos f;
          conditions
                f.getIdentificacion( 105 ) != null;
                f.getIdentificacion(    105    ).getNivel()    ==
                Riesgo.NO_DETERMINADO;
```

```
                f.getIdentificacion( 1 ) != null;
                f.getIdentificacion( 6 ) != null;
                f.getIdentificacion( 19 ) != null;
                f.getIdentificacion( 21 ) != null;
                f.getIdentificacion( 37 ) != null;
                f.getIdentificacion( 92 ) != null;
                f.getIdentificacion( 94 ) != null;
                f.getIdentificacion( 100 ) != null;

                f.getIdentificacion( 1 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 6 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 19 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 21 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 37 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion( 92 ).getNivel() == Riesgo.BAJO
                ||    f.getIdentificacion(    92    ).getNivel()    ==
                Riesgo.MEDIO;
                f.getIdentificacion( 94 ).getNivel() == Riesgo.BAJO;
                f.getIdentificacion(     100     ).getNivel()     ==
Riesgo.BAJO;
          actions
                f.getIdentificacion( 105 ).setNivel( Riesgo.BAJO );
                modified(f);
     }

     rule _16_elemento105_alta {
          declarations
                FiltroRiesgos f;
          conditions
                f.getIdentificacion( 105 ) != null;
                f.getIdentificacion(     105     ).getNivel()     ==
                Riesgo.NO_DETERMINADO;

                f.getIdentificacion( 1 ) != null;
                f.getIdentificacion( 6 ) != null;
                f.getIdentificacion( 19 ) != null;
                f.getIdentificacion( 21 ) != null;
                f.getIdentificacion( 37 ) != null;
                f.getIdentificacion( 92 ) != null;
                f.getIdentificacion( 94 ) != null;
                f.getIdentificacion( 100 ) != null;

                f.getIdentificacion( 1 ).getNivel() == Riesgo.ALTO
                ||    f.getIdentificacion(    1    ).getNivel()    ==
                Riesgo.MEDIO;
                f.getIdentificacion( 6 ).getNivel() == Riesgo.ALTO
                ||    f.getIdentificacion(    6    ).getNivel()    ==
                Riesgo.MEDIO;
                f.getIdentificacion( 19 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 21 ).getNivel() == Riesgo.ALTO
                ||    f.getIdentificacion(    21    ).getNivel()    ==
                Riesgo.MEDIO;
                f.getIdentificacion( 37 ).getNivel() == Riesgo.ALTO
                ||    f.getIdentificacion(    37    ).getNivel()    ==
                Riesgo.MEDIO;
                f.getIdentificacion( 92 ).getNivel() == Riesgo.ALTO;
                f.getIdentificacion( 94 ).getNivel() == Riesgo.ALTO
                ||    f.getIdentificacion(    94    ).getNivel()    ==
                Riesgo.MEDIO;
```

```
                f.getIdentificacion( 100 ).getNivel() == Riesgo.ALTO
                ||   f.getIdentificacion(   100   ).getNivel()   ==
                Riesgo.MEDIO;
            actions
                f.getIdentificacion( 105 ).setNivel( Riesgo.ALTO );
                modified(f);
        }
}
```

5.8. Diseño detallado de Clases

La siguiente figura (véase **Figura 74 - Diagrama Detallado de Clases**) representa el diagrama detallado de clases del sistema.

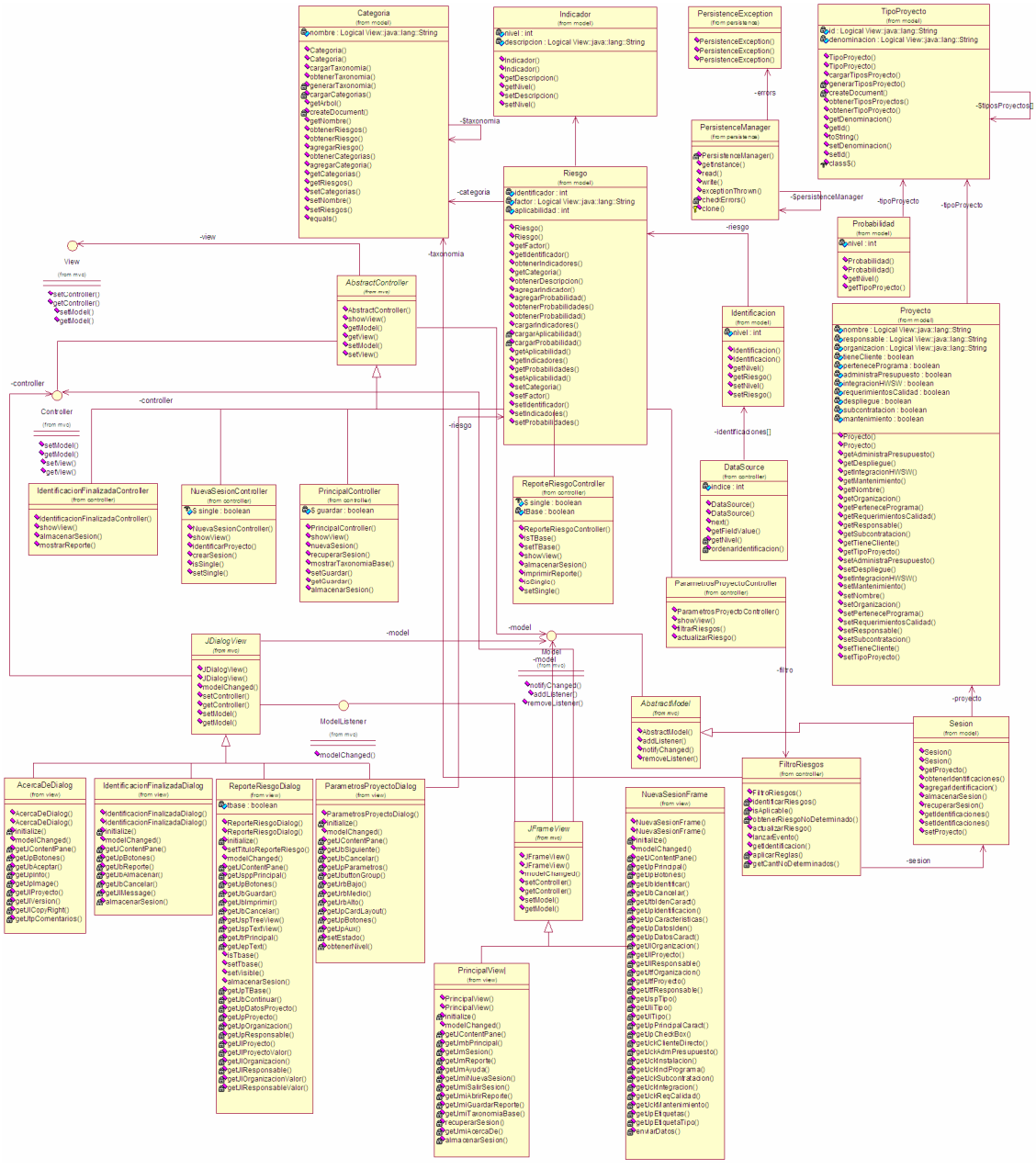


Figura 74 - Diagrama Detallado de Clases

La descripción detallada de los atributos y operaciones de las clases identificadas se encuentra especificada en **Apéndice H: Documentación de Clases**.

5.9. Verificación y Aceptación de la Arquitectura del Sistema

5.9.1. Verificación de las Especificaciones de Diseño

Se verificó la calidad de los distintos modelos de forma separada con el propósito de garantizar su adecuación al problema a resolver y su seguimiento respecto a la técnica de diseño seleccionada.

Como resultado de la verificación, se efectuaron correcciones que fueron posteriormente comprobadas.

5.9.2. Análisis de Consistencia de las Especificaciones de Diseño

Se comprobó la coherencia entre los distintos modelos mediante un conjunto de matrices de trazabilidad (**Apéndice G: Matrices de Trazabilidad)**.

5.9.3. Aceptación de la Arquitectura del Sistema

Se confirmaron y validaron con los usuarios los siguientes modelos:

- Subsistemas de soporte
- Modelo de clases
- Diseño de datos

Los usuarios confirmaron y validaron las soluciones planteados para el sistema y la manera en la que las mismas se implementarán.

5.10. Generación de Especificaciones de Construcción

En esta sección se generan las especificaciones para la construcción de la herramienta, a partir del diseño detallado.

5.10.1. Especificación del Entorno de Construcción

Se realiza a continuación la definición detallada y completa del entorno necesario para la construcción de los diferentes componentes de la aplicación.

Hardware
- Computadora Personal Pentium III o superior, con al menos 256 MB de memoria RAM.

Software de Base
- Ambiente de ejecución y/o desarrollo de la plataforma J2SE versión 1.4.2 o superior.

Herramientas
- Entorno desarrollo integrado Eclipse 3.0.1 [Eclipse, 2005] que se utilizará como herramienta principal para la codificación, compilación y depuración del código del sistema.
- Herramienta CASE Rational Rose [Rational, 2002] que se utilizará para la generación automática de código a partir de los modelos de diseño.
- Utilitarios de Microsoft Office que se utilizarán para la documentación producida durante las actividades de construcción.
- Como herramienta de generación de ejecutables se empleará IzPack 3.7.2 [IzPack, 2005].

5.10.2. Definición de Componentes y Subsistemas de Construcción

Cada módulo o clase y cada formato de interfaz se corresponden con un componente. Asimismo, dichos componentes han sido agrupados e incluidos en módulos contenedores siguiendo alguno de los siguientes criterios:
- Optimización de recursos.
- Características comunes de funcionalidad o de acceso a datos.
- Necesidades especiales de ejecución, por ejemplo: accesos costosos a datos comunes, acceso a recursos computacionales compartidos, etc.

Asimismo, en la definición de los componentes y subsistemas de construcción se han considerado los módulos y subsistemas definidos durante la fase de diseño.

edu.itba.mvc
En este componente se encapsula el patrón de diseño “Model View Controller” que provee el mecanismo base principal de funcionamiento pata toda la aplicación.

edu.itba.mvc.view
En este componente se encapsulan las interfaces de usuario.

edu.itba.mvc.controller
En este componente se encapsulan los controladores de interfaz de usuario responsables por el procesamiento de las acciones del mismo sobre los diferentes elementos de la interfaz gráfica.

edu.itba.mvc.model
En este componente se encapsulan los componentes del modelo que dan soporte a la aplicación.

edu.itba.persistence
En este componente se encapsula el subsistema encargado de mantener la persistencia de los elementos de la aplicación.

ar.edu.ubp.i18n
En este componente se encapsula el subsistema de internacionalización.

net.sf.jasperreports
En este componente se encapsula el producto JasperReports utilizado para la generación de reportes de la aplicación.

jeops
En este componente se encapsula el producto JEOPS utilizado para la gestión de reglas de la aplicación.

XML Files
Este componente representa el repositorio en el cual serán almacenados los datos persistentes de la aplicación.

La siguiente figura (véase **Figura 75 - Modelo de Componentes de Seldon**) muestra el Modelo de Componentes (a nivel de paquetes) que será empleado como base durante la construcción del sistema.

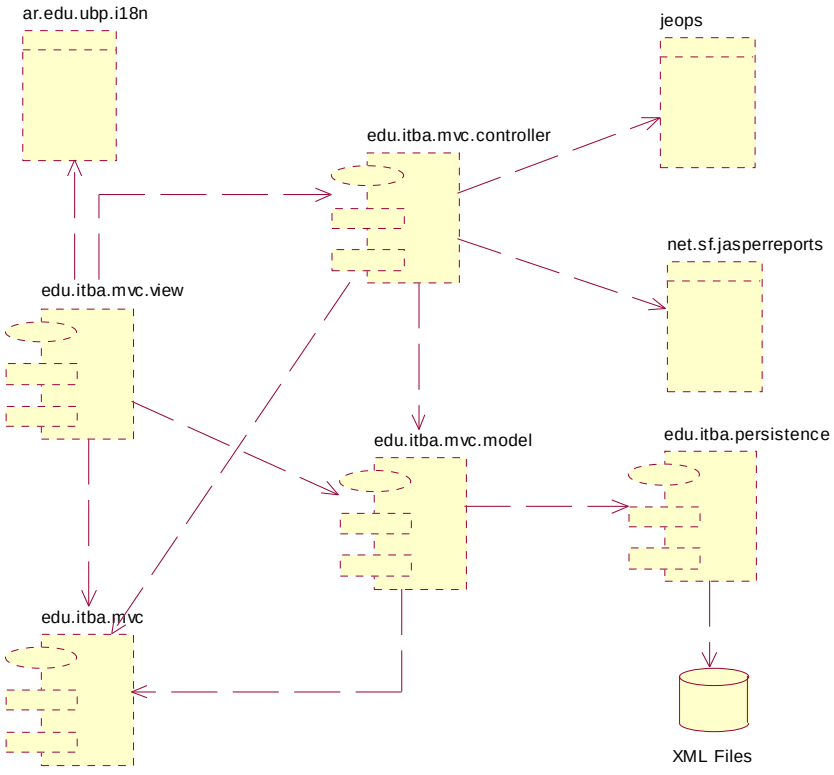


Figura 75 - Modelo de Componentes de Seldon

5.10.3. Plan de Integración del Sistema

Considerando el moderado grado de complejidad del sistema a construir y teniendo en cuenta el hecho de que el presente emprendimiento se trata de un desarrollo individual, la integración de los componentes se realizará a medida que los mismos vayan siendo desarrollados y luego de superar exitosamente las pruebas unitarias planteadas para cada uno de ellos, consiguiéndose una integración gradual a medida que avanza el desarrollo.

En base a los componentes antes detallados, se define el siguiente orden para la construcción y prueba de los componentes del sistema (véase **Tabla 27 - Plan de Integración de Seldon**):

ORDEN	COMPONENTE	DETALLES
1	persistente	Construcción y ejecución de las pruebas unitarias de interfaz
2	mvc	Construcción del marco de trabajo en base al patrón de diseño
3	model	Construcción y ejecución de las pruebas unitarias
4	controller	Construcción y ejecución de las pruebas unitarias
5	view	Construcción y ejecución de las pruebas unitarias

Tabla 27 - Plan de Integración de Seldon

5.11. Diseño de la Migración y Carga Inicial de Datos

Esta actividad sólo se lleva a cabo cuando es necesaria una carga inicial de información o una migración de datos desde otros sistemas.

Dada las características de la aplicación y considerando que se trata de un sistema que intenta automatizar y estandarizar la realización de una serie de nuevas actividades dentro de las organizaciones y conociendo, además, la independencia que se logra mediante el uso de una taxonomía estándar, no se ha identificado la necesidad de realizar una migración de datos desde otros sistemas.

Es importante destacar que el sistema requiere de la carga inicial de la Taxonomía de Riegos (**Apéndice F: Taxonomía de Riesgos Ajustada**) como requisito previo para su operación, pero debe aclararse que la misma ha sido considerada, incluida y calendarizada como parte de las tareas a realizar en la fase de construcción de la aplicación y no como una actividad especial e independiente debido al reducido alcance y extrema facilidad que la mencionada tarea requiere.

5.12. Especificación Técnica del Plan de Pruebas

En esta actividad se realiza la especificación de detalle del plan de pruebas del sistema.

El catálogo de requisitos, el catálogo de excepciones y el diseño detallado de la aplicación, permiten la definición de los casos de pruebas de verificación que deben realizarse para comprobar que el sistema responde a los requisitos planteados. En **4.12.6-Aproximación** se especifican las actividades, técnicas y herramientas a utilizar para generar cada uno de los diferentes casos de prueba antes mencionados.

Se realiza a continuación la definición detallada y completa del entorno necesario para la ejecución de las diferentes pruebas aplicables sobre la herramienta:

Configuración #1:

- Hardware
  - Computadora Personal Pentium III o superior, con al menos 256 MB de memoria RAM.
- Software de Base
  - Sistema Operativo Windows XP.
  - Ambiente de ejecución de la plataforma J2SE versión 1.4.2 o superior.

Configuración #2:

- Hardware
  - Computadora Personal Pentium III o superior, con al menos 256 MB de memoria RAM.
- Software de Base
  - Sistema Operativo Linux 2.6.9.
  - Ambiente de ejecución de la plataforma J2SE versión 1.4.2 o superior.

En caso de no realizarse una indicación diferente, los casos de prueba serán ejecutados bajo la **Configuración #1** antes descripta.

Los siguientes ítems describen la estructura estándar de cada caso de prueba:

- Identificador:
  Representa un identificador único del caso de prueba.
- Propósito:
  Corresponde a la explicación del caso de prueba, su fin y su alcance.
- Preparación:
  Indica las acciones que es necesario realizar previo a la ejecución del caso de prueba.
- Entradas:
  Datos, consideraciones especiales o asunciones que corresponden a criterios de ingreso obligatorios para ejecutar el caso de prueba.
- Procedimiento:

Detalle de cada uno de los pasos que deben seguirse para ejecutar el caso de prueba.

- Resultado Esperado:
  El resultado que hace que el caso de prueba se cumpla. Este resultado debe ser comparado con la salida real del caso de prueba para determinar si el mismo fue exitoso o no.

Se lista a continuación la especificación de los casos de prueba para el sistema Seldon:

Identificador	CP-001
Propósito	Verificar la correcta inicialización de la aplicación.
Preparación	Instalar el sistema Seldon. No realizar cambios sobre la configuración original.
Entradas	- Sistema instalado exitosamente.
Procedimiento	**1.** Ubicarse en el directorio base de la instalación de la aplicación Seldon. **2.** Ejecutar la siguiente instrucción desde línea de comandos: `javaw -classpath .;./lib/commons-beanutils.jar;./lib/commons-beanutils-bean-collections.jar;./lib/commons-beanutils-core.jar;./lib/commons-collections-3.1.jar;./lib/commons-digester-1.7.jar;./lib/commons-logging.jar;./lib/itext-1.3.jar;./lib/jasperreports-0.6.8.jar;./lib/JEOPS.jar;./lib/xml.jar;./lib/i18n.jar edu.itba.mvc.controller.Main seldon.props`  **Nota:** En entornos Windows, el paso **2** equivale a la ejecución de la instrucción `Seldon`.
Resultado Esperado	Se muestra en pantalla la ventana principal de la aplicación Seldon.

Identificador	CP-002
Propósito	Verificar el correcto cierre de la aplicación.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Salir*.
Resultado Esperado	Se muestra una ventana solicitando confirmación de abandono de la aplicación.

Identificador	CP-003
Propósito	Verificar el correcto funcionamiento de la ventana de confirmación.
Preparación	Ejecución de CP-002.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *OK* o *Aceptar*.
Resultado Esperado	Finaliza el sistema y se regresa a la línea de comandos.

Identificador	CP-004
Propósito	Verificar el correcto funcionamiento de la ventana de confirmación.
Preparación	Ejecución de CP-002.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Cancel* o *Cancelar*.
Resultado Esperado	Se muestra en pantalla la ventana principal de la aplicación Seldon.

Identificador	CP-005
Propósito	Verificar el correcto funcionamiento de la opción de información del sistema.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Ayuda* → *Acerca de...* **2.** Seleccionar el botón *Aceptar*.
Resultado Esperado	Se muestra una ventana conteniendo la siguiente información general del sistema: ▪ Autor. ▪ Versión. ▪ Año de desarrollo. ▪ Propósito del sistema.

Identificador	CP-006
Propósito	Verificar el correcto funcionamiento de la opción de reporte de la taxonomía base.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Reporte* → *Taxonomía Base*. **2.** Verificar que las categorías y riesgos listados corresponden a los indicados en la taxonomía base (referirse a **Apéndice F: Taxonomía de Riesgos Ajustada**). **3.** Seleccionar el botón *Continuar*.
Resultado Esperado	Se muestra una ventana conteniendo la taxonomía base descripta en **Apéndice F: Taxonomía de Riesgos Ajustada**.

Identificador	CP-007
Propósito	Verificar el correcto funcionamiento de la ventana de inicio de sesión.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Nueva.*
Resultado Esperado	Se muestra en pantalla la ventana de inicio de sesión de identificación de riesgos.

Identificador	CP-008
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-007.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Organizacion*.

Identificador	CP-009
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-007.
Entradas	No Aplica.
Procedimiento	**1.** Completar el campo *Organizacion* con espacios en blanco. **2.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Organizacion*.

Identificador	CP-010
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-007.
Entradas	No Aplica.
Procedimiento	**1.** Completar el campo *Organizacion* con el valor: *Organizacion1*. **2.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Proyecto*.

Identificador	CP-011
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-010.
Entradas	No Aplica.
Procedimiento	**1.** Completar el campo *Proyecto* con espacios en blanco. **2.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Proyecto*.

Identificador	CP-012
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-010.

Entradas	No Aplica.
Procedimiento	**1.** Completar el campo Proyecto con el valor: *Proyecto1*. **2.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Responsable*.

Identificador	CP-013
Propósito	Verificar las validaciones previas al inicio de una sesión de identificación.
Preparación	Ejecución de CP-012.
Entradas	No Aplica.
Procedimiento	**1.** Completar el campo *Responsable* con espacios en blanco. **2.** Seleccionar el botón *Identificar.*
Resultado Esperado	Se muestra un mensaje de error indicando que debe completarse el campo de *Responsable*.

Identificador	CP-014
Propósito	Verificar la correcta ejecución de un proceso de identificación de riesgos.
Preparación	Ejecución de CP-012.
Entradas	- Taxonomía base provista con la instalación del sistema sin modificaciones.
Procedimiento	**1.** Completar el campo *Responsable* con el valor: *Responsable1*. **2.** Seleccionar la solapa de *Caracteristicas*. **3.** Verificar que se encuentre seleccionado el *Tipo* denominado *Software Comercial de Mercado*. **4.** Seleccionar el botón *Identificar.* **5.** Repetir hasta que se presente la ventana de *Identificación de Riesgos*: **a.** Seleccionar la primera alternativa para la consulta realizadas por el sistema. **b.** Presionar el botón *Siguiente*.
Resultado Esperado	Se muestra una seria de diálogos presentando opciones a las siguientes categorías y riesgos: ▪ Experiencia con el proceso. ▪ Espíritu y actitud del equipo. ▪ Experiencia. ▪ Compromiso con el proceso. ▪ Calendario de desarrollo. ▪ Soporte al administrador de proyectos. ▪ Experiencia en la aplicación. ▪ Estabilidad organizacional. ▪ Comunicación interna del equipo. ▪ Soporte de la gerencia. ▪ Enfoque de aseguramiento de la calidad. ▪ Estabilidad del proceso.

	▪ Actitud. Al finalizar el proceso se muestra un diálogo conteniendo las opciones: ▪ *Reporte.* ▪ *Almacenar.* ▪ *Cancelar.*

Identificador	CP-015
Propósito	Verificar la correcta identificación de riesgos.
Preparación	Ejecución de CP-014
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar el botón *Reporte*.
Resultado Esperado	Se muestra una ventana conteniendo la siguiente información respecto a la identificación de riesgos realizada: ▪ **Proyecto:** *Proyecto1*. ▪ **Organización:** *Organizacion1*. ▪ **Responsable:** *Responsable1*. ▪ Riesgo 1 → Bajo. ▪ Riesgo 11 → Bajo. ▪ Riesgo 12 → Medio. ▪ Riesgo 14 → Medio. ▪ Riesgo 15 → Alto. ▪ Riesgo 16 → Medio. ▪ Riesgo 17 → Bajo. ▪ Riesgo 19 → Bajo. ▪ Riesgo 20 → Medio. ▪ Riesgo 21 → Bajo. ▪ Riesgo 22 → Bajo. ▪ Riesgo 23 → Bajo. ▪ Riesgo 30 → Bajo. ▪ Riesgo 31 → Bajo. ▪ Riesgo 32 → Bajo. ▪ Riesgo 36 → Medio. ▪ Riesgo 37 → Bajo. ▪ Riesgo 38 → Medio. ▪ Riesgo 39 → Bajo. ▪ Riesgo 40 → Bajo. ▪ Riesgo 41 → Bajo. ▪ Riesgo 42 → Bajo. ▪ Riesgo 44 → Bajo. ▪ Riesgo 45 → Bajo. ▪ Riesgo 46 → Medio. ▪ Riesgo 48 → Bajo. ▪ Riesgo 49 → Bajo. ▪ Riesgo 51 → Bajo. ▪ Riesgo 66 → Medio. ▪ Riesgo 67 → Bajo.

	▪ Riesgo 68 → Bajo. ▪ Riesgo 70 → Bajo. ▪ Riesgo 71 → Bajo. ▪ Riesgo 72 → Bajo. ▪ Riesgo 73 → Medio. ▪ Riesgo 74 → Medio. ▪ Riesgo 75 → Bajo. ▪ Riesgo 76 → Medio. ▪ Riesgo 78 → Bajo. ▪ Riesgo 79 → Bajo. ▪ Riesgo 80 → Bajo. ▪ Riesgo 82 → Bajo. ▪ Riesgo 83 → Bajo. ▪ Riesgo 84 → Medio. ▪ Riesgo 87 → Bajo. ▪ Riesgo 88 → Bajo. ▪ Riesgo 92 → Bajo. ▪ Riesgo 93 → Bajo. ▪ Riesgo 94 → Bajo. ▪ Riesgo 95 → Bajo. ▪ Riesgo 96 → Bajo. ▪ Riesgo 97 → Bajo. ▪ Riesgo 98 → Bajo. ▪ Riesgo 99 → Bajo. ▪ Riesgo 100 → Bajo. ▪ Riesgo 101 → Bajo. ▪ Riesgo 102 → Bajo. ▪ Riesgo 103 → Bajo. ▪ Riesgo 104 → Medio. ▪ Riesgo 105 → Bajo. ▪ Riesgo 106 → Bajo. ▪ Riesgo 108 → Medio. ▪ Riesgo 109 → Medio. ▪ Riesgo 111 → Bajo. Se muestran además los botones: ▪ *Guardar.* ▪ *Imprimir.* ▪ *Cancelar.*

Identificador	CP-016
Propósito	Verificar la posibilidad de realizar almacenamiento de archivos desde el diálogo de identificación de riesgos.
Preparación	Ejecución de CP-014.
Entradas	No Aplica.
Procedimiento	**1.** Presionar el botón *Almacenar.* **2.** Seleccionar una ubicación (referenciada como **<DIR-016>** a partir de este punto). **3.** Indicar CP-016 como nombre de archivo.

	4. Seleccionar el botón *Save* o *Guardar*.
Resultado Esperado	Se regresa al diálogo que contiene las opciones: ▪ *Reporte.* ▪ *Almacenar.* ▪ *Cancelar.*

Identificador	CP-017
Propósito	Verificar la posibilidad de realizar almacenamiento de archivos desde la ventana principal del sistema.
Preparación	Ejecución de CP-014.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Reporte* → *Guardar*. **2.** Seleccionar una ubicación (referenciada como **<DIR-017>** a partir de este punto). **3.** Indicar CP-017 como nombre de archivo. **4.** Seleccionar el botón *Save* o *Guardar*.
Resultado Esperado	Se regresa al diálogo que contiene las opciones: ▪ *Reporte.* ▪ *Almacenar.* ▪ *Cancelar.*

Identificador	CP-018
Propósito	Verificar la correcta validación del formato de los archivos de identificación de riesgos a ser abiertos por la aplicación.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Reporte* → *Abrir*. **2.** Ubicarse en el directorio de instalación de la aplicación. **3.** Seleccionar *message_format_es_AR.xml* como archivo a abrir. **4.** Seleccionar el botón *Open* o *Abrir*.
Resultado Esperado	Se muestra un diálogo con el siguiente mensaje de error: *Ha seleccionado un archivo con formato no valido*.

Identificador	CP-019
Propósito	Verificar el correcto funcionamiento de la opción de cancelado de una sesión de identificación de riesgos que se encuentra en proceso.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Nueva*. **2.** Completar el campo Organizacion con el valor: *Organizacion1*. **3.** Completar el campo Proyecto con el valor: *Proyecto1*.

	4. Completar el campo Responsable con el valor: *Responsable1*. **5.** Seleccionar el botón *Identificar*. **6.** Presionar el botón *Cancelar*.
Resultado Esperado	Se muestra una ventana solicitando confirmación de cancelación de la sesión.

Identificador	CP-020
Propósito	Verificar el correcto funcionamiento de la opción de cancelado de una sesión de identificación de riesgos que se encuentra finalizada.
Preparación	Ejecución de CP-014.
Entradas	No Aplica.
Procedimiento	**1.** Presionar el botón *Cancelar.*
Resultado Esperado	Se muestra una ventana solicitando confirmación de cancelación de la sesión.

Identificador	CP-021
Propósito	Verificar el correcto almacenamiento de las sesiones de identificación de riesgos.
Preparación	Ejecución de CP-016. Ejecución de CP-020.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Reporte* → *Abrir*. **2.** Ubicarse en el directorio **<DIR-016>**. **3.** Seleccionar el archivo *CP-016.xml*. **4.** Seleccionar el botón *Open* o *Abrir*.
Resultado Esperado	Se muestra una ventana conteniendo la siguiente información respecto a la identificación de riesgos realizada: ▪ **Proyecto:** *Proyecto1*. ▪ **Organización:** *Organizacion1*. ▪ **Responsable:** *Responsable1*. ▪ Riesgo 1 → Bajo. ▪ Riesgo 11 → Bajo. ▪ Riesgo 12 → Medio. ▪ Riesgo 14 → Medio. ▪ Riesgo 15 → Alto. ▪ Riesgo 16 → Medio. ▪ Riesgo 17 → Bajo. ▪ Riesgo 19 → Bajo. ▪ Riesgo 20 → Medio. ▪ Riesgo 21 → Bajo. ▪ Riesgo 22 → Bajo. ▪ Riesgo 23 → Bajo. ▪ Riesgo 30 → Bajo. ▪ Riesgo 31 → Bajo. ▪ Riesgo 32 → Bajo. ▪ Riesgo 36 → Medio.

	▪ Riesgo 37 → Bajo. ▪ Riesgo 38 → Medio. ▪ Riesgo 39 → Bajo. ▪ Riesgo 40 → Bajo. ▪ Riesgo 41 → Bajo. ▪ Riesgo 42 → Bajo. ▪ Riesgo 44 → Bajo. ▪ Riesgo 45 → Bajo. ▪ Riesgo 46 → Medio. ▪ Riesgo 48 → Bajo. ▪ Riesgo 49 → Bajo. ▪ Riesgo 51 → Bajo. ▪ Riesgo 66 → Medio. ▪ Riesgo 67 → Bajo. ▪ Riesgo 68 → Bajo. ▪ Riesgo 70 → Bajo. ▪ Riesgo 71 → Bajo. ▪ Riesgo 72 → Bajo. ▪ Riesgo 73 → Medio. ▪ Riesgo 74 → Medio. ▪ Riesgo 75 → Bajo. ▪ Riesgo 76 → Medio. ▪ Riesgo 78 → Bajo. ▪ Riesgo 79 → Bajo. ▪ Riesgo 80 → Bajo. ▪ Riesgo 82 → Bajo. ▪ Riesgo 83 → Bajo. ▪ Riesgo 84 → Medio. ▪ Riesgo 87 → Bajo. ▪ Riesgo 88 → Bajo. ▪ Riesgo 92 → Bajo. ▪ Riesgo 93 → Bajo. ▪ Riesgo 94 → Bajo. ▪ Riesgo 95 → Bajo. ▪ Riesgo 96 → Bajo. ▪ Riesgo 97 → Bajo. ▪ Riesgo 98 → Bajo. ▪ Riesgo 99 → Bajo. ▪ Riesgo 100 → Bajo. ▪ Riesgo 101 → Bajo. ▪ Riesgo 102 → Bajo. ▪ Riesgo 103 → Bajo. ▪ Riesgo 104 → Medio. ▪ Riesgo 105 → Bajo. ▪ Riesgo 106 → Bajo. ▪ Riesgo 108 → Medio. ▪ Riesgo 109 → Medio. ▪ Riesgo 111 → Bajo. Se muestran además los botones:

	▪ *Guardar.* ▪ *Imprimir.* ▪ *Cancelar.*

Identificador	CP-022
Propósito	Verificar el correcto funcionamiento de la opción de cancelado de una sesión de identificación de riesgos que se encuentra reportada.
Preparación	Ejecución de CP-015.
Entradas	No Aplica.
Procedimiento	**1.** Presionar el botón *Cancelar.*
Resultado Esperado	Se muestra una ventana solicitando confirmación de cancelación de la sesión.

Identificador	CP-023
Propósito	Verificar la posibilidad de realizar almacenamiento de archivos desde la ventana de reporte de una sesión de identificación.
Preparación	Ejecución de CP-015.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar el botón *Guardar*. **2.** Seleccionar una ubicación (referenciada como **<DIR-023>** a partir de este punto). **3.** Indicar CP-023 como nombre de archivo. **4.** Seleccionar el botón *Save* o *Guardar*.
Resultado Esperado	Se regresa al diálogo que contiene el reporte de identificación de riesgos.

Identificador	CP-024
Propósito	Verificar la correcta impresión de los riesgos identificados.
Preparación	Ejecución de CP-015
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar el botón *Imprimir*.
Resultado Esperado	Se muestra una ventana conteniendo la siguiente información respecto a la identificación de riesgos realizada en el formato definido en **4.10.4-Formatos de Impresión**: ▪ **Proyecto:** *Proyecto1*. ▪ **Organización:** *Organizacion1*. ▪ **Responsable:** *Responsable1*. ▪ Riesgo 1 → Bajo. ▪ Riesgo 11 → Bajo. ▪ Riesgo 12 → Medio. ▪ Riesgo 14 → Medio. ▪ Riesgo 15 → Alto. ▪ Riesgo 16 → Medio. ▪ Riesgo 17 → Bajo.

	▪ Riesgo 19 → Bajo. ▪ Riesgo 20 → Medio. ▪ Riesgo 21 → Bajo. ▪ Riesgo 22 → Bajo. ▪ Riesgo 23 → Bajo. ▪ Riesgo 30 → Bajo. ▪ Riesgo 31 → Bajo. ▪ Riesgo 32 → Bajo. ▪ Riesgo 36 → Medio. ▪ Riesgo 37 → Bajo. ▪ Riesgo 38 → Medio. ▪ Riesgo 39 → Bajo. ▪ Riesgo 40 → Bajo. ▪ Riesgo 41 → Bajo. ▪ Riesgo 42 → Bajo. ▪ Riesgo 44 → Bajo. ▪ Riesgo 45 → Bajo. ▪ Riesgo 46 → Medio. ▪ Riesgo 48 → Bajo. ▪ Riesgo 49 → Bajo. ▪ Riesgo 51 → Bajo. ▪ Riesgo 66 → Medio. ▪ Riesgo 67 → Bajo. ▪ Riesgo 68 → Bajo. ▪ Riesgo 70 → Bajo. ▪ Riesgo 71 → Bajo. ▪ Riesgo 72 → Bajo. ▪ Riesgo 73 → Medio. ▪ Riesgo 74 → Medio. ▪ Riesgo 75 → Bajo. ▪ Riesgo 76 → Medio. ▪ Riesgo 78 → Bajo. ▪ Riesgo 79 → Bajo. ▪ Riesgo 80 → Bajo. ▪ Riesgo 82 → Bajo. ▪ Riesgo 83 → Bajo. ▪ Riesgo 84 → Medio. ▪ Riesgo 87 → Bajo. ▪ Riesgo 88 → Bajo. ▪ Riesgo 92 → Bajo. ▪ Riesgo 93 → Bajo. ▪ Riesgo 94 → Bajo. ▪ Riesgo 95 → Bajo. ▪ Riesgo 96 → Bajo. ▪ Riesgo 97 → Bajo. ▪ Riesgo 98 → Bajo. ▪ Riesgo 99 → Bajo. ▪ Riesgo 100 → Bajo. ▪ Riesgo 101 → Bajo. ▪ Riesgo 102 → Bajo.

	▪ Riesgo 103 → Bajo. ▪ Riesgo 104 → Medio. ▪ Riesgo 105 → Bajo. ▪ Riesgo 106 → Bajo. ▪ Riesgo 108 → Medio. ▪ Riesgo 109 → Medio. ▪ Riesgo 111 → Bajo.

Identificador	CP-025
Propósito	Verificar la existencia de ayuda en línea del sistema.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Nueva*. **a.** Mantener el apuntador en la opción por al menos un segundo. **2.** Seleccionar la opción *Sesion* → *Salir*. **a.** Mantener el apuntador en la opción por al menos un segundo. **3.** Seleccionar la opción *Reporte* → *Abrir*. **a.** Mantener el apuntador en la opción por al menos un segundo. **4.** Seleccionar la opción *Reporte* → *Guardar*. **a.** Mantener el apuntador en la opción por al menos un segundo. **5.** Seleccionar la opción *Reporte* → *Taxonomía Base*. **a.** Mantener el apuntador en la opción por al menos un segundo. **6.** Seleccionar la opción *Ayuda* → *Acerca de...*. **a.** Mantener el apuntador en la opción por al menos un segundo.
Resultado Esperado	Para cada uno de los casos, se muestra un mensaje describiendo la utilidad de la opción de menú seleccionada.

Identificador	CP-026
Propósito	Verificar la correcta habilitación de las opciones del menú principal del sistema.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Nueva*. **2.** Seleccionar la opción *Sesion* → *Salir*. **3.** Seleccionar la opción *Reporte* → *Abrir*. **4.** Seleccionar la opción *Reporte* → *Guardar*. **5.** Seleccionar la opción *Reporte* → *Taxonomía Base*. **6.** Seleccionar la opción *Ayuda* → *Acerca de...*.
Resultado Esperado	Para cada uno de los casos (exceptuando *Reporte* → *Guardar*) la opción se encuentra habilitada para su uso.

Identificador	CP-027
Propósito	Verificar la imposibilidad de abrir múltiplese sesiones de identificación de riesgo.
Preparación	Ejecución de CP-001.
Entradas	No Aplica.
Procedimiento	**1.** Seleccionar la opción *Sesion* → *Nueva*. **2.** Seleccionar la opción *Sesion* → *Nueva*.
Resultado Esperado	Se muestra un mensaje de error indicando que ya existe una sesión de identificación de riesgos iniciada.

Identificador	CP-028
Propósito	Verificar la posibilidad de modificar el idioma presentado por el sistema.
Preparación	Ejecución de CP-001.
Entradas	- Archivo message_format_en_US.xml. ▪ Realizar una copia del archivo message_format_es_AR.xml y renombrarla como message_format_en_US.xml. ▪ Editar el archivo message_format_en_US.xml y reemplazar `<message key = "NEW"><value>Nueva</value></message>` por `<message key = "NEW"><value>New</value></message>`. ▪ Editar el archivo seldon.props y reemplazar `propsIdioma = es` por `propsIdioma = en` y `propsRegion = AR` por `propsRegion = US`.
Procedimiento	**1.** Seleccionar la opción *Sesion*.
Resultado Esperado	Se muestran las siguientes opciones: ▪ *New*. ▪ *Salir*.

Identificador	CP-029
Propósito	Verificar que el sistema puede ser ejecutado en múltiples plataformas.
Preparación	Emplear la **Configuración #2**. Instalar el sistema Seldon.
Entradas	- Sistema instalado exitosamente.
Procedimiento	**1.** Ubicarse en el directorio base de la instalación de la aplicación Seldon. **2.** Ejecutar la siguiente instrucción desde línea de comandos: `javaw -classpath .;./lib/commons-beanutils.jar;./lib/commons-beanutils-bean-collections.jar;./lib/commons-beanutils-core.jar;./lib/commons-collections-3.1.jar;./lib/commons-digester-1.7.jar;./lib/commons-logging.jar;./lib/itext-1.3.jar;./lib/jasperreports-`

	`0.6.8.jar;./lib/JEOPS.jar;./lib/xml.jar;./lib/i18n.jar edu.itba.mvc.controller.Main seldon.props`
Resultado Esperado	Se muestra en pantalla la ventana principal de la aplicación Seldon.

5.13. Establecimiento de Requisitos de Implantación

No se han especificado ni se han detectado requisitos especiales relacionados con la documentación que el usuario requiere para operar con el nuevo sistema, ni tampoco relativos a la propia implantación del sistema en el entorno de operación.

La afirmación antes realizada ha sido validada y acordada luego de considerar los siguientes aspectos:

- La condición de prototipo de herramienta del sistema y el hecho de que los usuarios del mismo hayan participado tanto en la revisión de objetivos y de pruebas como en la validación del análisis y del diseño realizado facilita, simplifica y minimiza el entendimiento requerido para llevar a cabo la operación de la aplicación.

- La terminología estándar empleada para nombrar las opciones de los menús del sistema facilita y simplifica el uso de la aplicación a usuarios habituados al manejo de diferentes aplicaciones.

- El soporte gráfico adaptable en cuanto a su modo de visualización a los estándares de distintos sistemas operativos facilita y simplifica el uso de la aplicación a usuarios habituados al manejo de diferentes entornos gráficos.

- La característica de internacionalización facilita y simplifica el uso de la aplicación a usuarios habituados al manejo de diferentes idiomas.

- Los usuarios principales del sistema serán administradores de proyecto que no requieren de un entrenamiento especial para comprender y utilizar las facilidades brindadas por la aplicación y los conceptos en los cuales se basa su operación.

- La implantación de una aplicación del tipo standalone, ejecutable en cualquier sistema operativo con soporte para Java, con una interfaz gráfica de navegación y con conexión a un repositorio local de datos no requiere de una formación, una infraestructura y una instalación especiales que deban ser preparadas y organizadas con antelación.

5.14. Aprobación del Diseño del Sistema

Luego de realizada la presentación del diseño del sistema a la Gerencia se obtuvo la aprobación final del mismo.

Durante la sesión fueron presentados, analizados, discutidos y aprobados los siguientes artefactos:

- Diseño de la Arquitectura del Sistema
- Entorno Tecnológico del Sistema
- Diseño Detallado de los Subsistemas de Soporte
- Modelo Físico de Datos
- Especificaciones de Construcción del Sistema
- Plan de Migración y Carga Inicial de Datos
- Diseño de la Realización de los Casos de Uso
- Modelo de Clases y Comportamiento de Diseño

Capítulo

6

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Construcción del Sistema de Información

6. Construcción del Sistema de Información (CSI)

En el presente capítulo se describe y desarrolla la construcción del sistema según las actividades propuestas por la metodología utilizada.

6.1. Introducción

En la fase de Construcción se incluye la codificación, la integración y las pruebas de los módulos definidos durante la fase de Diseño en un esqueleto progresivamente más completo del sistema final empleando diferentes técnicas de programación.

6.2. La construcción del sistema en MÉTRICA Versión 3

Durante la Construcción del Sistema se genera el código de los componentes del Sistema, se desarrollan todos los procedimientos de operación y seguridad y se elaboran todos los manuales de usuario final y de explotación con el objetivo de asegurar el correcto funcionamiento del Sistema para su posterior implantación.

Para conseguir dicho objetivo, en este proceso se realizan las pruebas unitarias, las pruebas de integración de los subsistemas y componentes y las pruebas del sistema, de acuerdo al plan de pruebas establecido.

Asimismo, se define la formación de usuario final y, si procede, se construyen los procedimientos de migración y carga inicial de datos.

Al ser MÉTRICA Versión 3 una metodología que cubre tanto desarrollos estructurados como orientados a objetos, las actividades de ambas aproximaciones están integradas en una estructura común.

Las actividades que engloba este proceso se recogen en la siguiente figura (véase **Figura 76 - Tareas de la CSI**), en la que se indican las tareas que pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores:

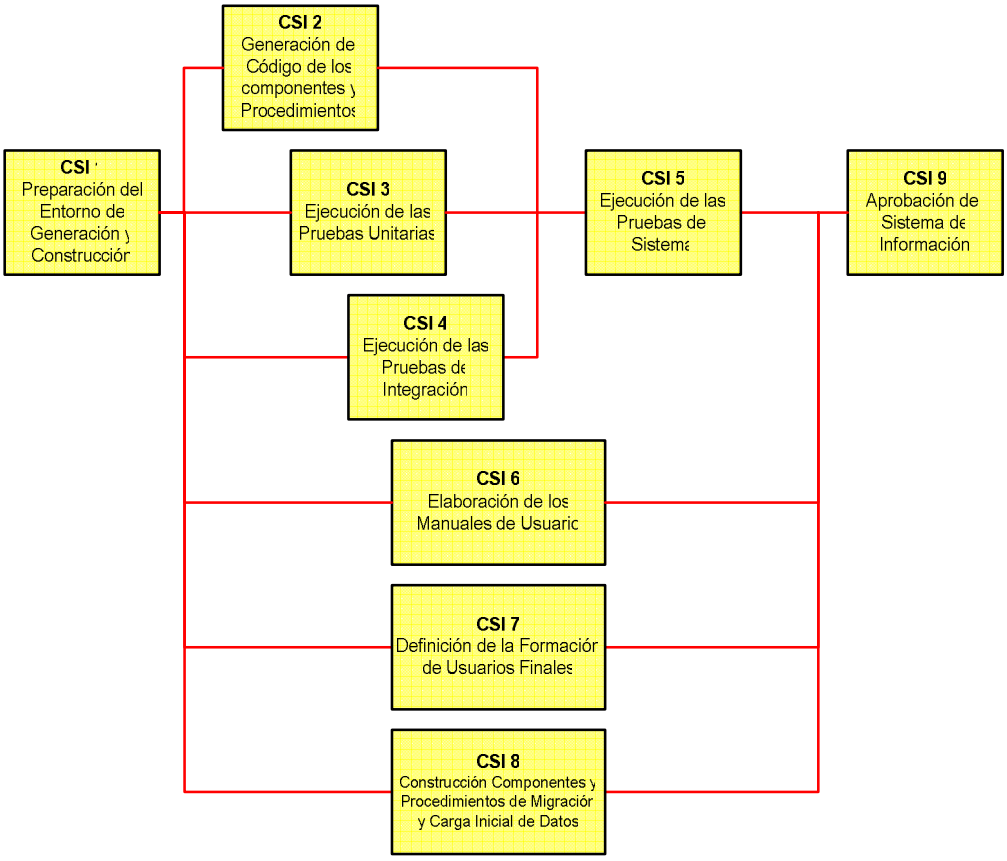


Figura 76 - Tareas de la CSI

Dada las características del sistema a realizar, y considerando que la metodología centra su proceso en desarrollos de tipo Estructurado y Orientado a Objetos, se revisarán como parte del presente proyecto todas las actividades propuestas.

6.3. Preparación del Entorno de Generación y Construcción

Al realizar esta actividad se aseguró la disponibilidad de todos los medios y facilidades para llevar a cabo la construcción del sistema. Entre estos medios, cabe destacar la preparación de los puestos de trabajo, equipos físicos y lógicos, archivos, bibliotecas de programas, herramientas de generación de código, archivos de prueba, entre otros.

6.3.1. Implantación de Archivos

Se produjo la inicialización de los archivos del sistema según lo especificado en las siguientes actividades:

- Archivos de configuración:
  Referirse a la sección **5.5.4**.**i18n** para detalles de implantación.
- Archivos del sistema:
  Referirse a la sección **5.4**.**Diseño Físico de Datos** para detalles de implantación y a la sección **Apéndice F: Taxonomía de Riesgos Ajustada** para detalles de contenidos.

6.3.2. Preparación del Entorno de Construcción

Se produjo la preparación del entorno de construcción según lo especificado en la sección **5.10.1**.**Especificación del Entorno de Construcción**.

Los siguientes pasos constituyeron la preparación del entorno de construcción del sistema:

1. Instalación y pruebas mínimas de funcionamiento de la plataforma J2SE versión 1.4.2. [J2SE, 2004]
2. Instalación, configuración y pruebas mínimas de funcionamiento del entorno integrado de desarrollo Eclipse 3.0.1 [Eclipse, 2005]

6.4. Generación del Código de los Componentes

El objetivo de esta tarea es la codificación de los componentes del sistema a partir de las especificaciones de construcción obtenidas en el proceso diseño.

Una vez preparado el entorno de construcción, se procedió a generar el código fuente de cada uno de los componentes contemplados en el diseño del sistema siguiendo el orden establecido en la sección **5.10.3**.**Plan de Integración del Sistema**.

Para cada una de las clases de los modelos de diseño, se generó el código fuente en lenguaje Java desde la herramienta Racional Rose y, posteriormente, dicho código fue completado utilizando el entorno de desarrollo Eclipse y agregando el comportamiento especificado en los diagramas de colaboración y de robustez.

Para generar el código fuente se consideraron los estándares de nomenclatura, codificación y calidad utilizados por la organización y referidos en los planes de proyecto.

Con el fin de verificar que el código especificaba de forma correcta el componente correspondiente, se realizó su compilación, verificando y corrigiendo los errores sintácticos, y el enlace del código objeto obtenido con las correspondientes bibliotecas.

Finalmente, las clases se integraron entre sí y se efectuaron las pruebas unitarias y de integración.

6.5. Ejecución de las Pruebas Unitarias

En esta actividad se realizaron las pruebas unitarias de cada uno de los componentes del sistema, una vez codificados, con el objeto de comprobar que su estructura es correcta y que se ajustan a la funcionalidad establecida.

6.5.1. Preparación del Entorno de las Pruebas Unitarias

Se aseguró la disponibilidad del entorno y de los datos necesarios para la ejecución de las pruebas unitarias y se prepararon las bibliotecas oportunas para la realización de las mismas, así como los procedimientos manuales o automáticos necesarios, conforme a la especificación del entorno definida en el plan de pruebas.

6.5.2. Realización y Evaluación de las Pruebas Unitarias

Las pruebas unitarias se efectuaron en paralelo con la generación del código de las clases que conforman el sistema lo cual implica que, a medida que se escribía el código, se ejecutaban las pruebas a manera de regresión con el propósito de garantizar que el nuevo código producido no introducía errores sobre el existente (se incluyeron en esta etapa a las pruebas de tipo Alpha).

Para cada uno de los componentes que conforman el sistema se ejecutaron los casos de prueba unitarios especificados.

Todos los defectos detectados fueron corregidos pertinentemente durante la ejecución de las tareas de construcción del sistema.

6.6. Ejecución de las Pruebas de Integración

En esta actividad se realizaron las pruebas de integración de los componentes o subsistemas con el propósito de verificar si los mismos interactuaban correctamente a través de sus interfaces, tanto internas como externas, cubrían la funcionalidad establecida, y se ajustaban a los requisitos especificados en las verificaciones correspondientes.

6.6.1. Preparación del Entorno de las Pruebas de Integración

Se aseguró la disponibilidad del entorno y de los datos necesarios para la ejecución de las pruebas de integración y se prepararon las bibliotecas

oportunas para la realización de las mismas, así como los procedimientos manuales o automáticos necesarios, conforme a la especificación del entorno definida en el plan de pruebas.

6.6.2. Realización y Evaluación de las Pruebas de Integración

Las pruebas de integración se efectuaron en paralelo con la generación del código de las clases que conforman el sistema y una vez que los componentes objeto de las mismas habían sido verificados de manera unitaria.

Para cada uno de los componentes o subsistemas que conforman la aplicación se ejecutaron los casos de prueba de integración especificados.

Todos los defectos detectados fueron corregidos pertinentemente durante la ejecución de las tareas de construcción del sistema.

6.7. Ejecución de las Pruebas del Sistema

En esta actividad se realizaron las pruebas del sistema con el propósito de comprobar la integración del sistema globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas con los que se comunica.

6.7.1. Preparación del Entorno de las Pruebas del Sistema

Se aseguró la disponibilidad del entorno y de los datos necesarios para la ejecución de las pruebas del sistema y se prepararon las bibliotecas oportunas para la realización de las mismas, así como los procedimientos manuales o automáticos necesarios, conforme a la especificación del entorno definida en el plan de pruebas.

6.7.2. Realización y Evaluación de las Pruebas del Sistema

Las pruebas del sistema se efectuaron una vez finalizada la codificación del sistema e incluyeron: pruebas de sistema, pruebas de compatibilidad y pruebas de instalación.

Para cada una de las funcionalidades requeridas para la aplicación se ejecutaron los casos de prueba del sistema especificados.

Todos los defectos detectados fueron corregidos pertinentemente durante la ejecución de las tareas de construcción del sistema.

En la sección **IX.1**.**Resultados de la Ejecución de las Pruebas** se detalla un resumen de los resultados obtenidos durante la ejecución de las pruebas.

6.8. Elaboración de los Manuales de Usuario

6.8.1. Requisitos de Hardware y Software

Se listan a continuación los requisitos de hardware y software necesarios para utilizar la aplicación Seldon.

Hardware:

- Computadora personal Pentium III o superior.
- 64 Mbytes de memoria RAM mínimo.
- Espacio mínimo en disco de 10 MBytes.

Software:

- Plataforma J2SE 1.4.2.
- Sistema operativo con soporte para la plataforma J2SE.

6.8.2. Instalación de la Aplicación Seldon

La instalación de la aplicación Seldon se inicia mediante la escritura de la siguiente instrucción en línea de comandos:

```
java -jar Seldon.jar
```

El proceso de instalación comienza con la selección del lenguaje del instalador (véase **Figura 77 - Selección del Lenguaje del Instalador**).

El usuario debe desplegar la lista de idiomas, posicionarse en aquel que desea utilizar y presionar el botón *OK*.



Figura 77 - Selección del Lenguaje del Instalador

La siguiente pantalla (véase **Figura 78 - Bienvenida al Proceso de Instalación**) corresponde a la bienvenida al proceso de instalación y muestra información general del sistema Seldon.

Para continuar es necesario presionar el botón *Siguiente*.

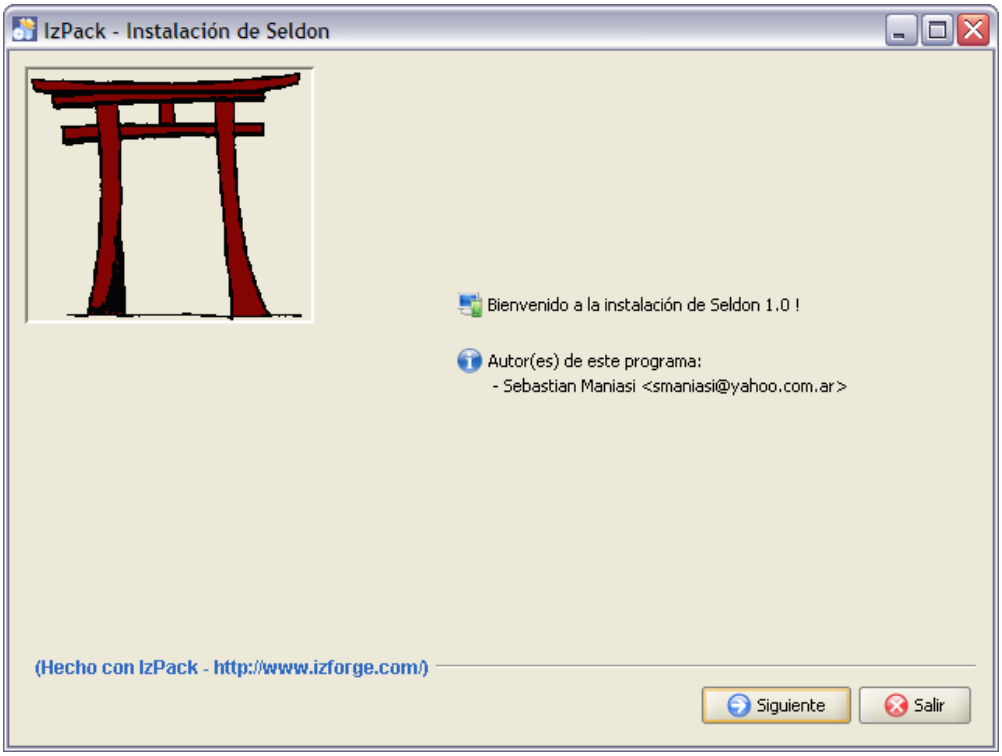


Figura 78 - Bienvenida al Proceso de Instalación

A continuación se presenta una pantalla que posibilita al usuario realizar la selección del directorio donde será instalada la aplicación (véase **Figura 79 - Selección del Directorio de Instalación**). El instalador propone un directorio pero el usuario puede modificarlo presionando el botón *Escoger* y realizando su elección mediante una ventana estándar de apertura de archivos.

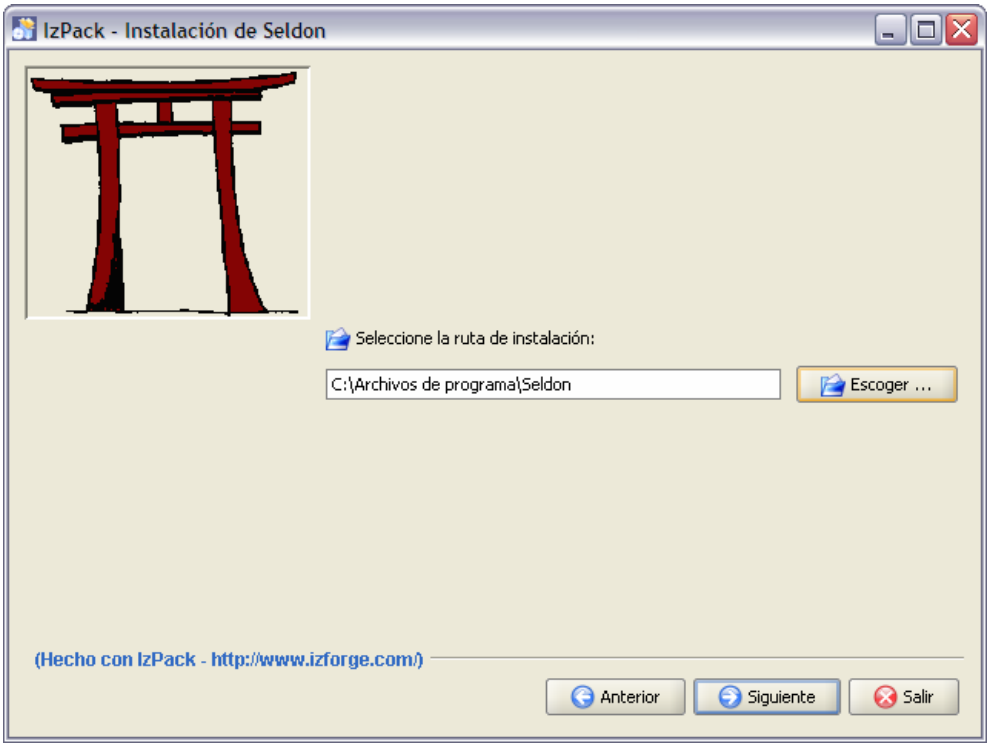


Figura 79 - Selección del Directorio de Instalación

Una vez finalizada la selección el proceso continúa utilizando el botón *Siguiente*.

El instalador informará respecto a la creación del directorio de instalación (véase **Figura 80 - Información de Creación del Directorio de Instalación**) y el usuario debe continuar el proceso presionando el botón *Aceptar*.



Figura 80 - Información de Creación del Directorio de Instalación

A continuación, se comenzará con la instalación de la aplicación Seldon y se reportará el progreso del proceso (véase **Figura 81 - Progreso de la Instalación**).

Una vez concluido el mismo y mediante el botón *Siguiente* es posible avanzar a la próxima pantalla.

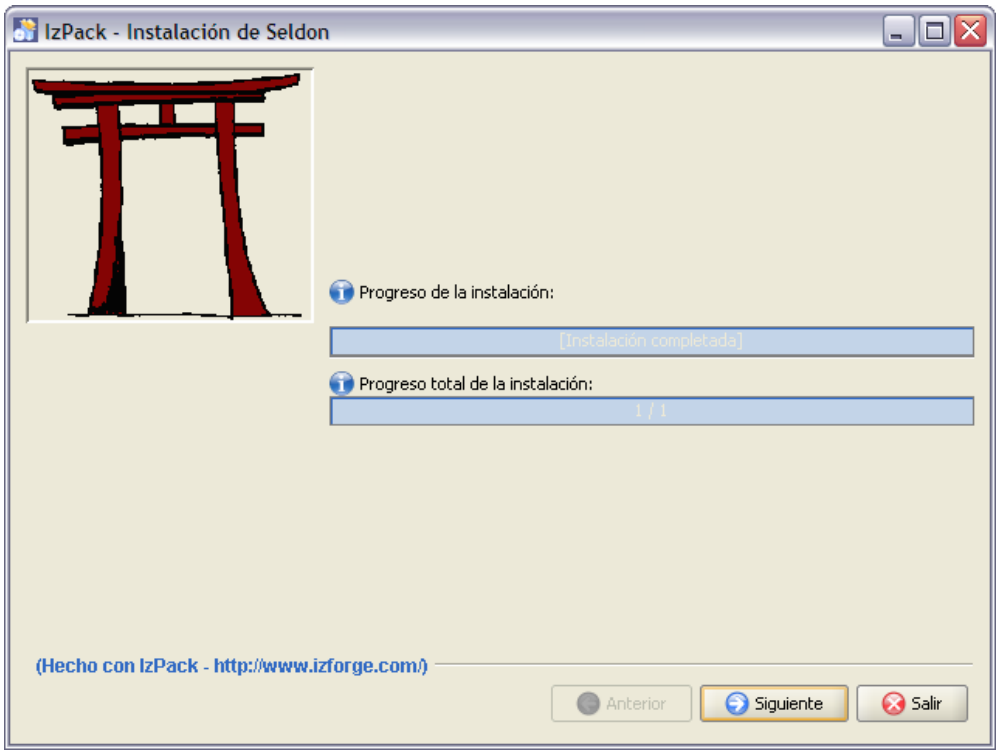


Figura 81 - Progreso de la Instalación

El procedimiento termina con la pantalla de finalización y resumen del proceso que informa respecto al sistema instalado y al directorio creado (véase **Figura**

82 - Finalización del Proceso de Instalación). El botón *Hecho* posibilita concluir el proceso y terminar el instalador.



Figura 82 - Finalización del Proceso de Instalación

Nota: En todas las ventanas de la aplicación instaladora, el botón *Salir* permite al usuario abandonar el proceso.

6.8.3. Configuración de la Aplicación Seldon

Luego de finalizada la instalación de la aplicación Seldon es posible cambiar su configuración mediante la modificación de diferentes archivos del sistema.

Configuración del Idioma

El sistema Seldon obtiene los mensajes a mostrar al usuario a través de la interfaz gráfica de un archivo de nombre message_format_<**ID**>_<**REG**>.xml, donde <**ID**> hace referencia al idioma a emplear y <**REG**> a la regionalización a considerar al detallar las expresiones y formatos de texto, numéros y de fechas.

La adopción de otro idioma para el sistema requiere de la realización de tres acciones:

- Creación de un nuevo archivo de mensajes siguiendo el formato definido en el archivo `message_format_es_AR.xml`.

- Modificación de las propiedades `propsIdioma` y `propsRegion` del archivo `seldon.props` según los nuevos <**ID**> y <**REG**> definidos para el sistema.

- Traducción de los archivos de taxonomía base y tipos de proyecto (`taxonomia.xml` y `tiposProyectos.xml` respectivamente).

Configuración de la Taxonomía Base

La aplicación obtiene la taxonomía base para su operación de un archivo de nombre `taxonomia.xml` que puede ser modificado con el propósito de:

- Cambiar los nombres de categorías y las descripciones de riesgos.

- Modificar la valoración del riesgo por cada tipo de proyecto.

- Agregar y eliminar categorías y riesgos.

Configuración de Tipos de Proyecto

La aplicación obtiene los tipos de proyecto empleados durante su operación de un archivo de nombre `tiposProyectos.xml` que puede ser modificado con el propósito de:

- Cambiar los nombres de los tipos de proyectos.

- Agregar y eliminar tipos de proyectos.

Nota: Con el propósito de mantener consistencia en el sistema, el agregado o eliminación de tipos de proyectos debe ser reflejado en el archivo de taxonomía base del sistema.

Configuración de la Base de Reglas

La aplicación obtiene las reglas a emplear durante su operación de un archivo de nombre `Seudoreglas.rules` (localizado bajo las carpetas `Seldon\Application\edu\itba\mvc\controller`) que puede ser modificado con el propósito de:

- Cambiar las condiciones de las reglas existentes.

- Agregar y eliminar reglas.

Nota: Las modificaciones al archivo de reglas requerirán de la regeneración de la clase de reglas; para detalles relacionados a dicha regeneración referirse a la guía de uso de JEOPS [JEOPS, 2005].

6.8.4. Ejecución de la Aplicación

Inicio de la Aplicación Seldon

La aplicación se inicia mediante la ejecución por línea de comandos de la siguiente instrucción:

```
javaw -classpath .;./lib/commons-beanutils.jar;./lib/commons-
beanutils-bean-collections.jar;./lib/commons-beanutils-
core.jar;./lib/commons-collections-3.1.jar;./lib/commons-digester-
1.7.jar;./lib/commons-logging.jar;./lib/itext-
1.3.jar;./lib/jasperreports-
0.6.8.jar;./lib/JEOPS.jar;./lib/xml.jar;./lib/i18n.jar
edu.itba.mvc.controller.Main seldon.props
```

Nota: En sistemas Windows, la aplicación puede ser inicializada mediante la ejecución del comando `Seldon` en la línea de comandos sobre el directorio `Application` dentro de la carpeta de instalación del sistema.

Una vez ejecutada esta instrucción, se accede a la ventana principal del sistema Seldon (véase **Figura 83 - Ventana Principal de Seldon**).



Figura 83 - Ventana Principal de Seldon

Nota: Con el propósito de mantener funcionando el sistema, debe mantenerse abierta la ventana de comandos utilizada para invocar la aplicación.

Realización de una Sesión de Identificación de Riesgos

Una sesión de identificación de riesgos comienza al seleccionar la opción *Sesion → Nueva* (véase **Figura 84 - Inicio de Sesión de Identificación de Riesgos**) del menú de la ventana principal del sistema.

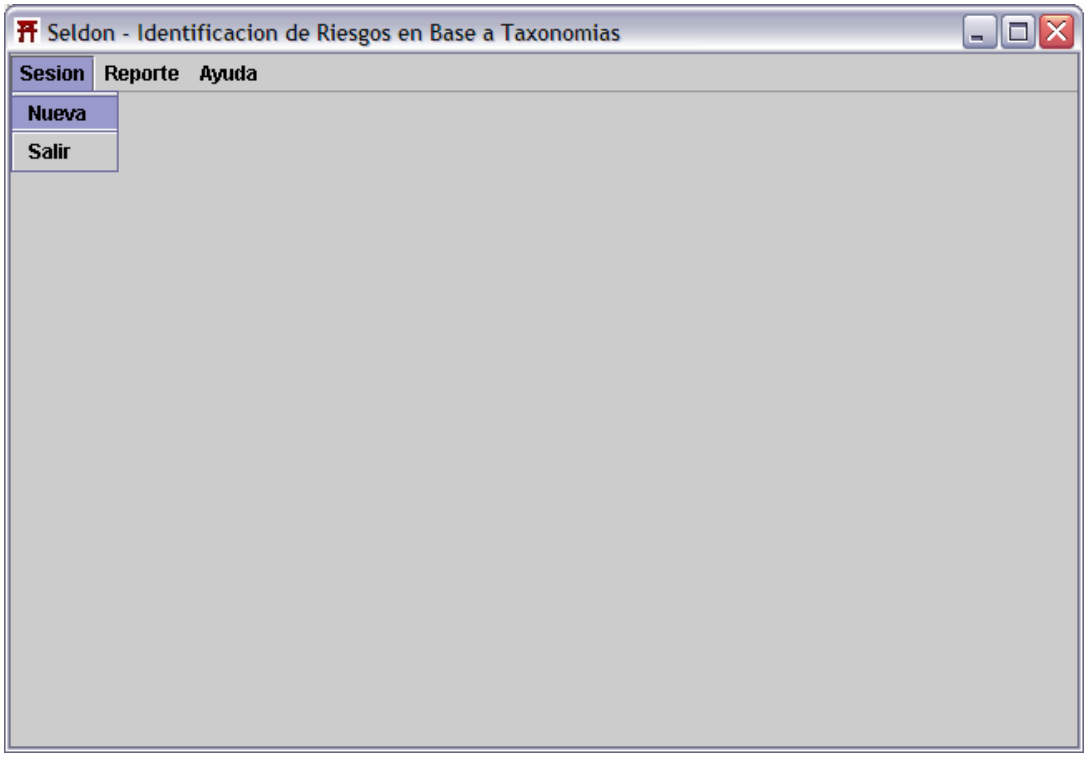


Figura 84 - Inicio de Sesión de Identificación de Riesgos

La primera acción a realizar consiste en la especificación de los datos que caracterizarán a la sesión de identificación de riesgos y que se encuentran en la solapa *Identificación* de la ventana de *Sesión de Identificación de Riesgos* (véase **Figura 85 - Alta de Datos de la Sesión de Identificación**).



Figura 85 - Alta de Datos de la Sesión de Identificación

A continuación se deben detallar las características del proyecto para el cual se realizará la sesión de identificación de riesgos en la solapa *Características* de la ventana de *Sesión de Identificación de Riesgos* (véase **Figura 86 - Alta de Características de la Sesión de Identificación**).

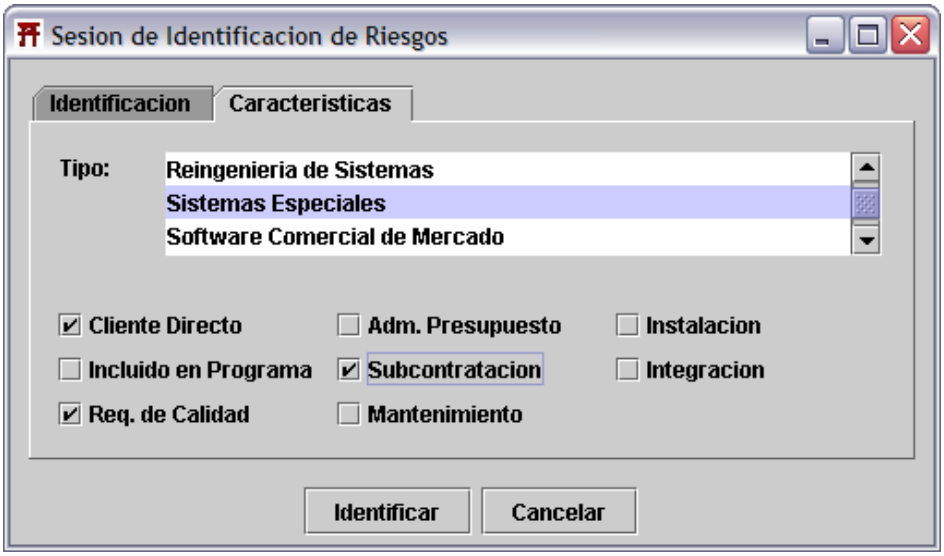


Figura 86 - Alta de Características de la Sesión de Identificación

El botón *Identificar* posibilita dar comienzo efectivo a la sesión de identificación de riesgos.

El sistema consultará al usuario respecto al valor a otorgar a diferentes parámetros del proyecto (véase **Figura 87 - Parámetros del Proyecto**) hasta poder contar con todos los datos necesarios para concluir la identificación de los riesgos asociados al mismo. El botón *Siguiente* posibilita avanzar en la caracterización de los diferentes parámetros del proyecto.

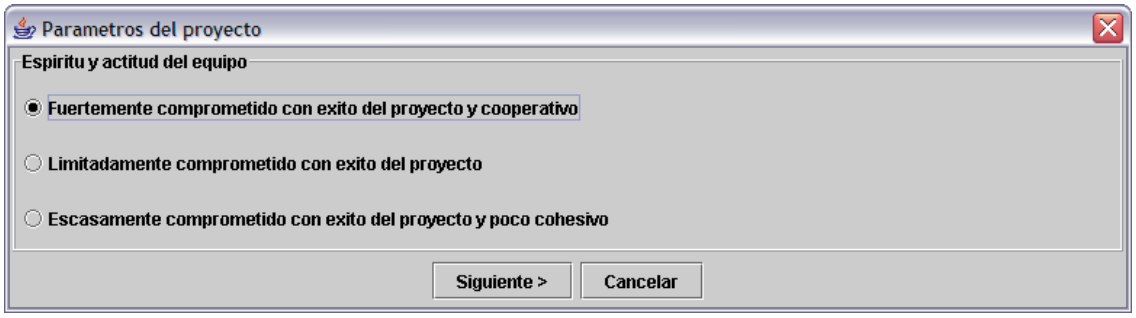


Figura 87 - Parámetros del Proyecto

Una vez que el usuario ha asociado valores a cada uno de los parámetros requeridos por el sistema Seldon, es posible generar el reporte del proceso de identificación de riesgos mediante la presión del botón *Reporte* en la ventana de *Identificación de Riesgos* (véase **Figura 88 - Generación del Reporte de Identificación de Riesgos**).

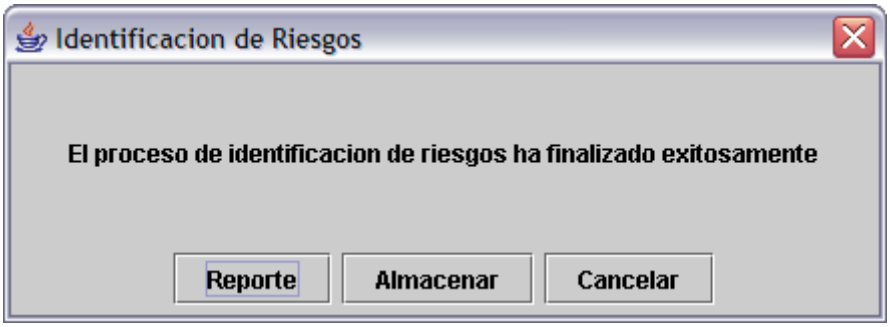


Figura 88 - Generación del Reporte de Identificación de Riesgos

El reporte de identificación de riesgos (véase **Figura 89 - Reporte de Identificación de Riesgos**) posibilita consultar los resultados obtenidos luego de concluida la sesión de identificación para el proyecto.

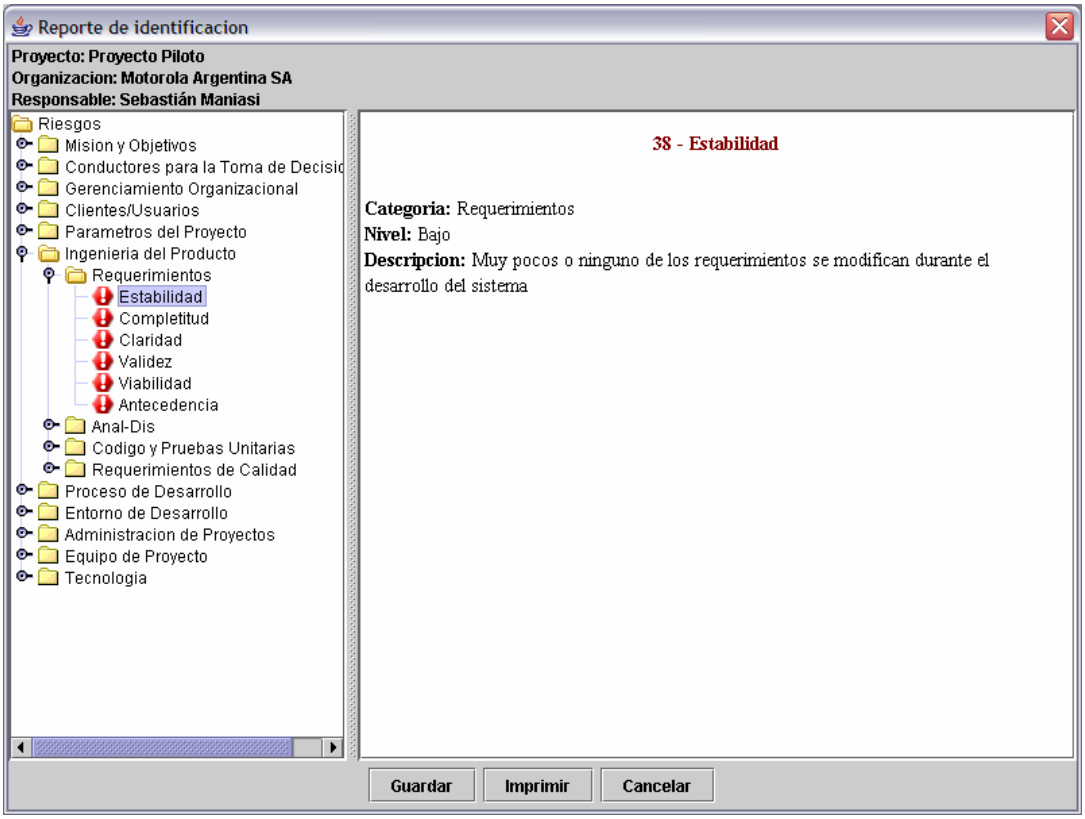


Figura 89 - Reporte de Identificación de Riesgos

Almacenamiento de una Sesión de Identificación de Riesgos

Existen tres accesos diferentes a la opción de almacenamiento en disco de los resultados de una sesión de identificación de riesgos:

- Desde el menú principal de la aplicación (véase **Figura 90 - Almacenamiento de una Sesión - Menú Principal**), opción que se habilita una vez generado un reporte de identificación de riesgos.
- Desde la ventana de *Reporte de Identificación* (véase **Figura 91 - Almacenamiento de una Sesión - Reporte de Identificación**), mientras se consulta el reporte de identificación de riesgos.

- Desde la ventana de *Identificación de Riesgos* (véase **Figura 92 - Almacenamiento de una Sesión - Identificación de Riesgos**), antes de generar el reporte de identificación de riesgos.

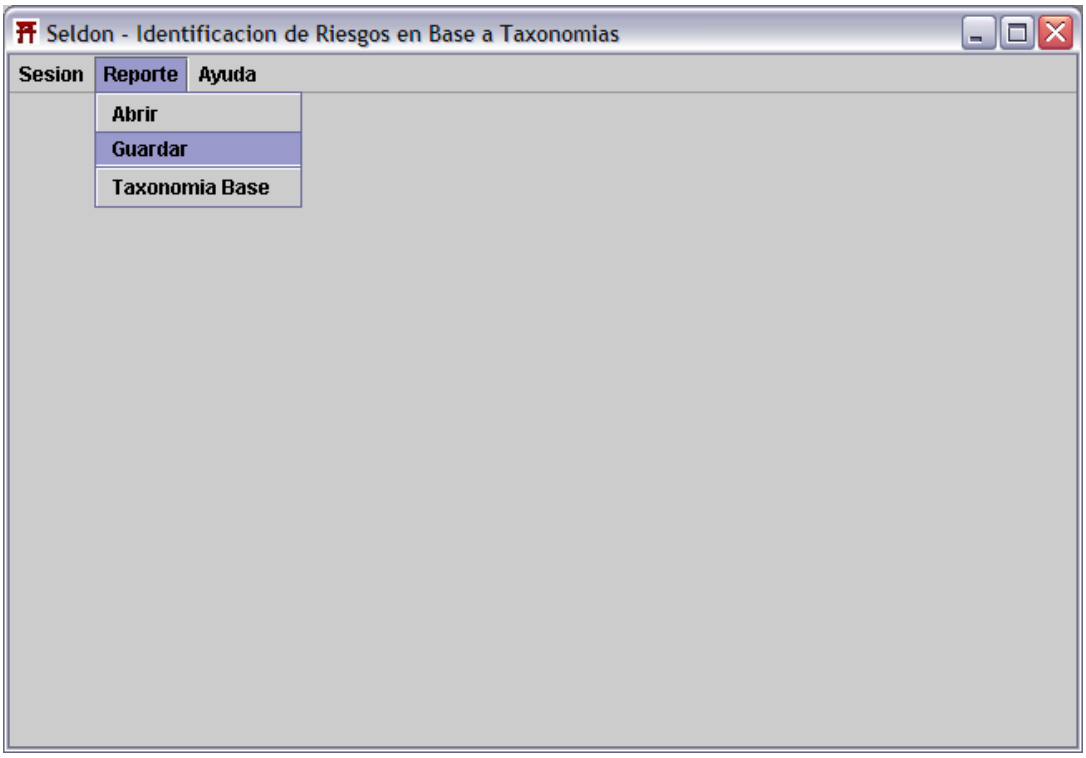


Figura 90 - Almacenamiento de una Sesión - Menú Principal

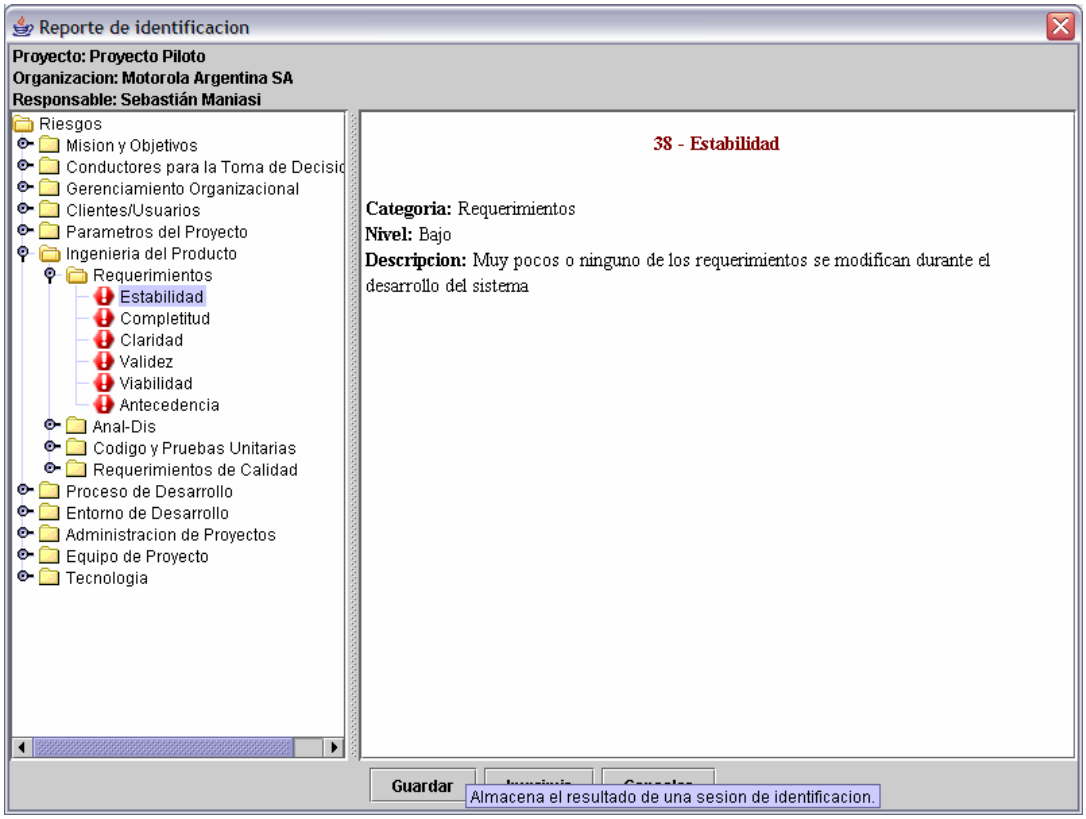


Figura 91 - Almacenamiento de una Sesión - Reporte de Identificación

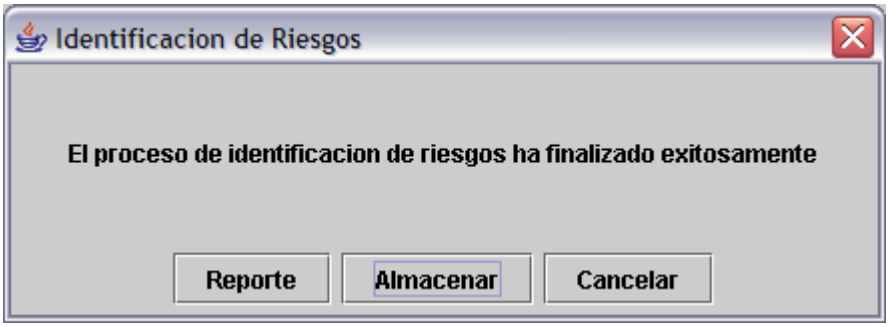


Figura 92 - Almacenamiento de una Sesión - Identificación de Riesgos

Luego de la selección de cualquiera de estas opciones, el sistema mostrará una ventana estándar de almacenamiento de archivos.

Recuperación de una Sesión de Identificación de Riesgos

La recuperación de los resultados de una sesión de identificación de riesgos se lleva a cabo seleccionando la opción *Reporte* → *Abrir* (véase Figura 93 **-** Recuperación de una Sesión de Identificación de Riesgos) del menú de la ventana principal del sistema.

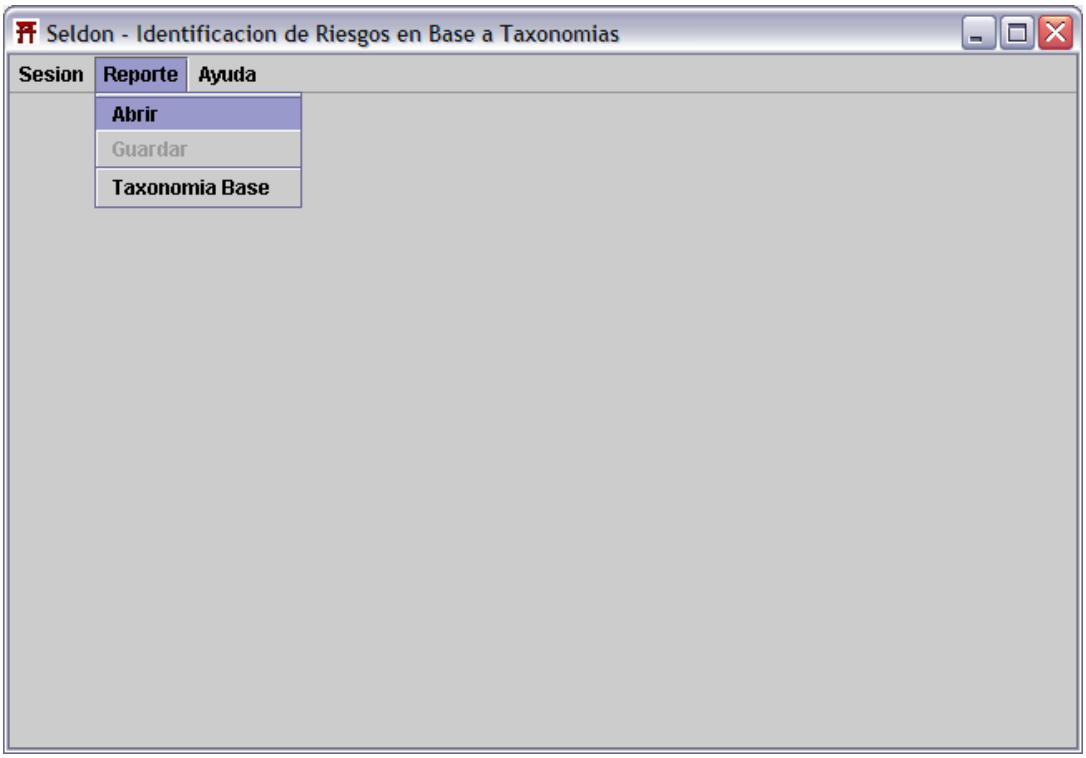


Figura 93 - Recuperación de una Sesión de Identificación de Riesgos

Luego de la selección de esta opción, el sistema mostrará una ventana estándar de apertura de archivos.

Impresión de una Sesión de Identificación de Riesgos

La impresión de los resultados de una sesión de identificación de riesgos se lleva a cabo presionando el botón *Imprimir* en la ventana de *Reporte de Identificación* (véase Figura 94 **-** Impresión de una Sesión de Identificación de Riesgos).

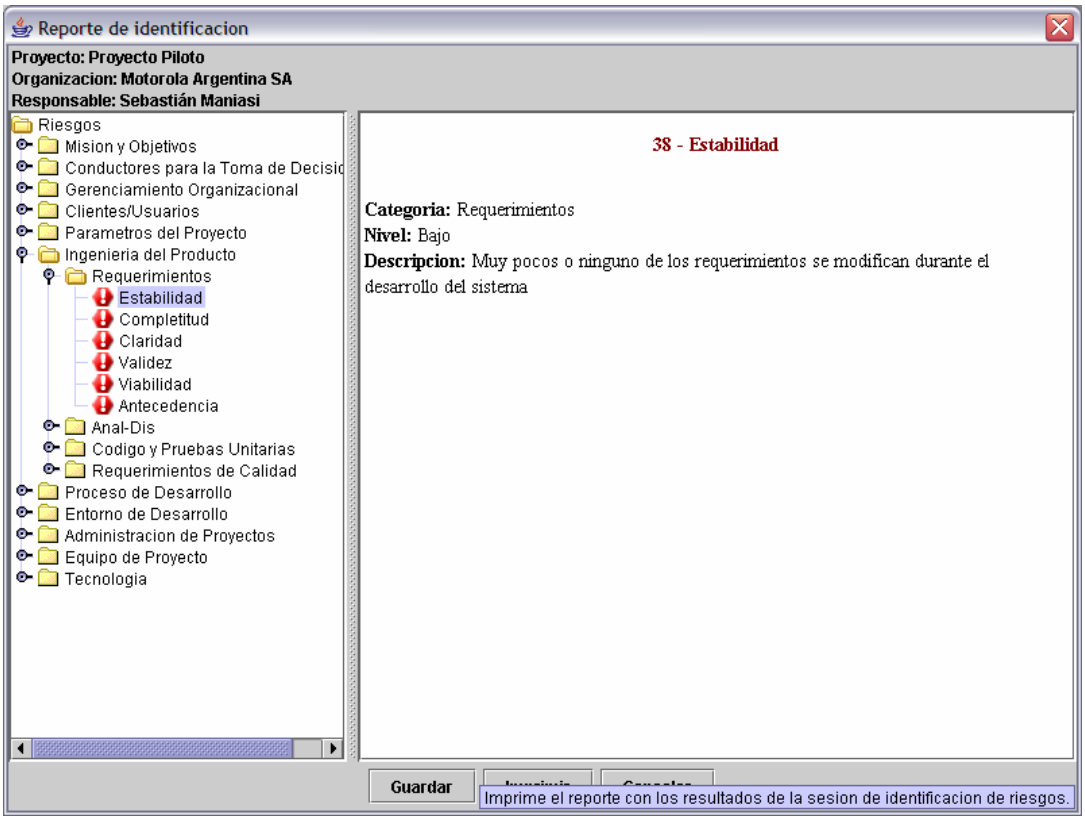


Figura 94 - Impresión de una Sesión de Identificación de Riesgos

Luego de la selección de esta opción, el sistema mostrará una ventana estándar de visualización previa a la impresión de reportes.

Consulta de la Taxonomía Base de Riesgos

La consulta de la taxonomía base de riesgos se produce seleccionando la opción *Reporte* → *Taxonomía Base* (véase **Figura 95 - Consulta del Reporte de Taxonomía Base de Riesgos**) del menú de la ventana principal del sistema.



Figura 95 - Consulta del Reporte de Taxonomía Base de Riesgos

El sistema mostrará a continuación una ventana conteniendo el reporte de la taxonomía base de riesgos empleada por la aplicación (véase **Figura 96 - Reporte de Taxonomía Base de Riesgos**).

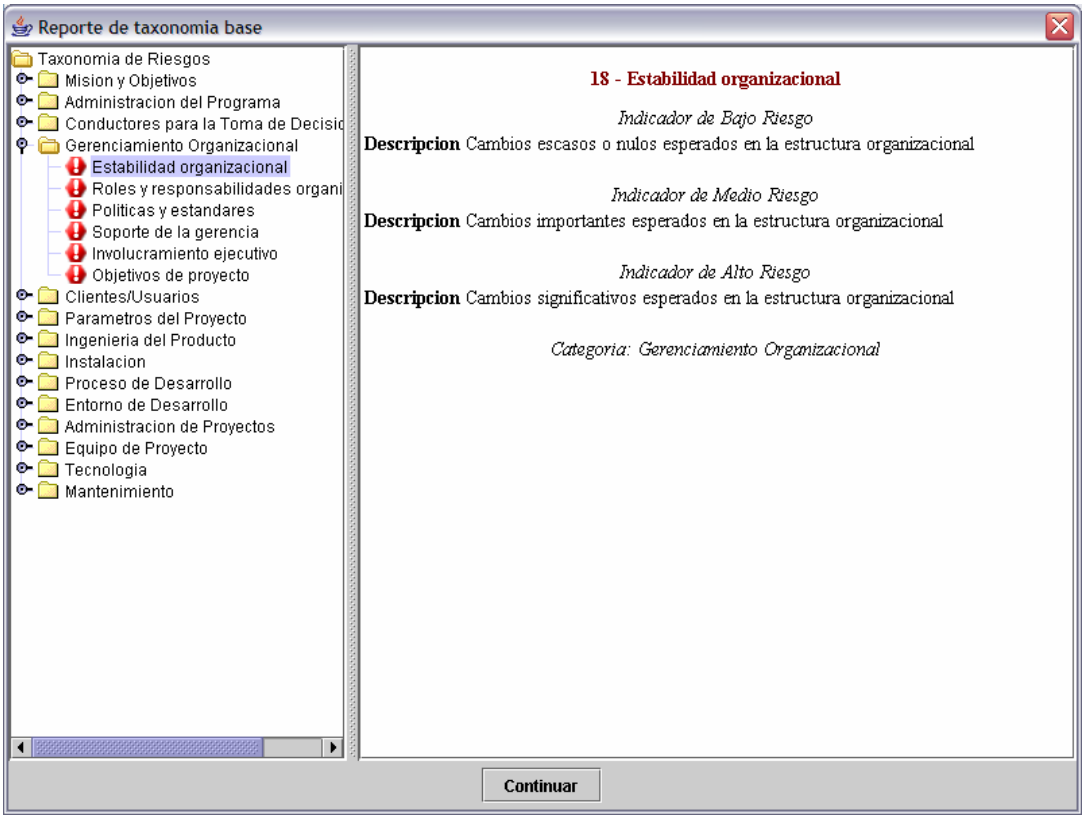


Figura 96 - Reporte de Taxonomía Base de Riesgos

La consulta finaliza al presionar el botón *Continuar*.

Consulta de la Información del Sistema

La consulta de la información del sistema se puede realizar al seleccionar la opción *Ayuda* → *Acerca de…* (véase **Figura 97 - Consulta de la Información del Sistema**) del menú de la ventana principal del sistema.

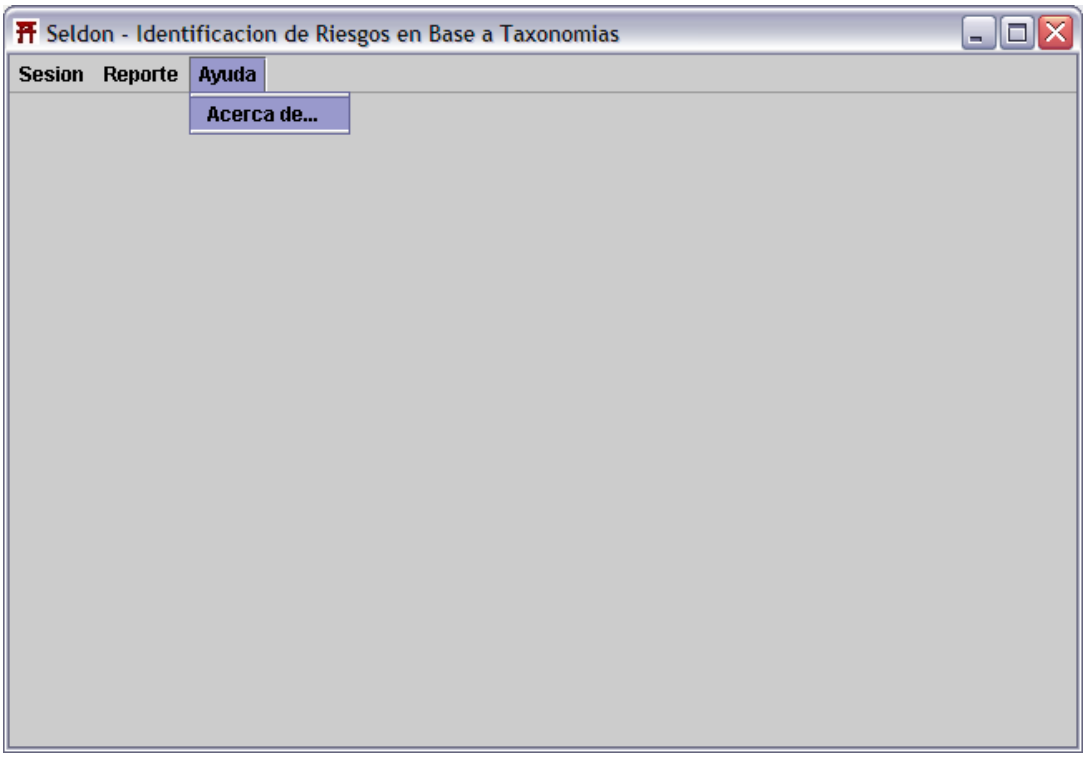


Figura 97 - Consulta de la Información del Sistema

El sistema mostrará a continuación una ventana conteniendo información general del sistema (véase **Figura 98 - Información del Sistema**).



Figura 98 - Información del Sistema

La consulta finaliza al presionar el botón *Aceptar*.

Salida del Sistema

El usuario puede abandonar el sistema mediante la opción *Sesion* → *Salir* (véase **Figura 99 - Salida del Sistema**) del menú de la ventana principal del sistema.

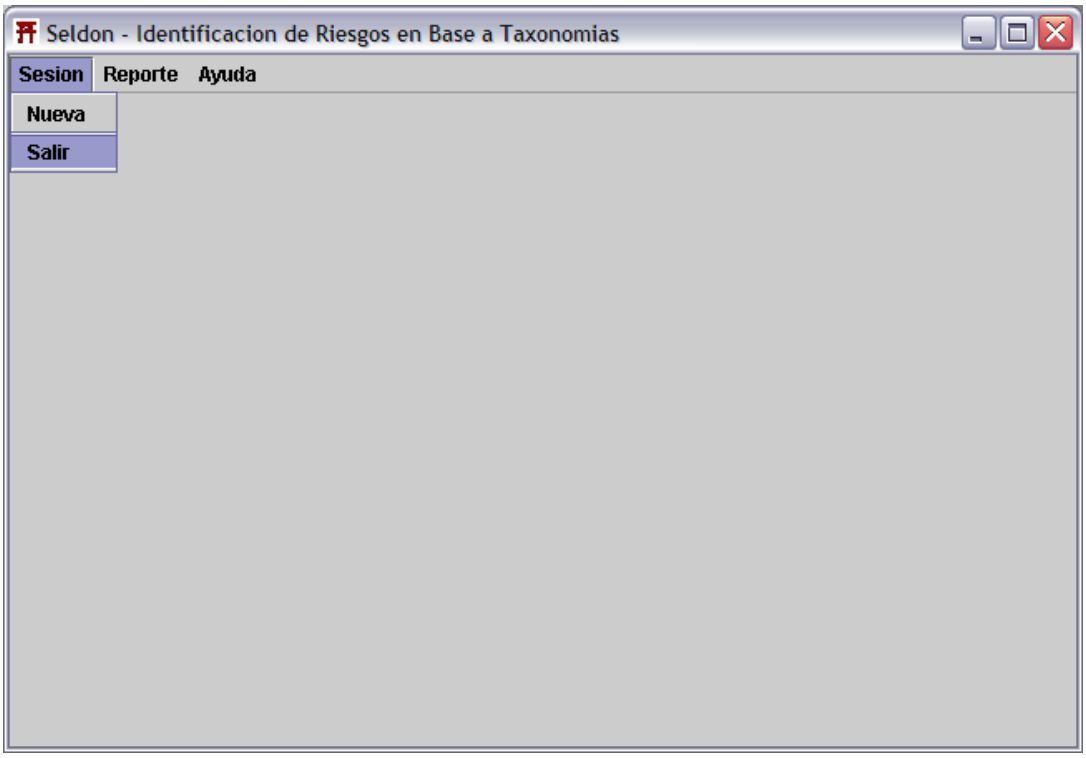


Figura 99 - Salida del Sistema

Luego de la selección de esta opción, el sistema mostrará una ventana estándar de confirmación de cierre de la aplicación.

Nota: En todas las ventanas de la aplicación Seldon, el botón *Cancelar* permite anular la acción que el usuario se encuentra realizando.

6.9. Definición de la Formación de Usuarios Finales

Dado que el problema resuelto por el sistema (identificación de riesgos) es ampliamente conocido en la industria y generalmente aplicado por la audiencia principal de la aplicación (líderes de proyecto de software y desarrolladores de software en general) y considerando que la aplicación ha sido desarrollada con una interfaz gráfica estándar de navegación orientada a facilitar su empleo a los usuarios finales no se requiere la definición de una formación específica para los mismos.

6.10. Construcción de los Componentes y Procedimientos de Migración y Carga Inicial de Datos

Dado que no existen datos previos asociados al sistema y que la matriz de trazabilidad ha sido introducida en la aplicación como uno de los componentes durante la fase de construcción, no es necesario llevar a cabo ningún proceso relacionado con la migración y carga inicial de datos.

6.11. Aprobación del Sistema

Luego de realizada la presentación del sistema y los resultados de los casos de prueba a la Gerencia se obtuvo la aprobación final del mismo.

Capítulo

7

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Implantación y Aceptación del Sistema

7. Implantación y Aceptación del Sistema de Información (IAS)

En el presente capítulo se describe y desarrolla la implantación y aceptación del sistema según las actividades propuestas por la metodología utilizada.

7.1. Introducción

La implantación es el proceso de verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla. Sin importar cuál sea la estrategia utilizada, los encargados de desarrollar el sistema se deben procuran porque el uso inicial del sistema se encuentre libre de problemas.

Debido a que los sistemas deben mantenerse siempre al día, la implantación es un proceso de constante evolución.

7.2. La implantación y aceptación del sistema en MÉTRICA Versión 3

Este proceso tiene como objetivo principal la entrega y aceptación del sistema en su totalidad, y la realización de todas las actividades necesarias para el paso a producción del mismo.

En primer lugar, se revisa la estrategia de implantación: se estudia su alcance y, en función de sus características, se define un plan de implantación y se especifica el equipo que lo va a llevar a cabo.

Las actividades previas al inicio de la producción incluyen la preparación de la infraestructura necesaria para configurar el entorno, la instalación de los componentes, la activación de los procedimientos manuales y automáticos asociados y, cuando proceda, la migración o carga inicial de datos. Para ello se toman como punto de partida los productos software probados, obtenidos en el proceso Construcción del Sistema de Información y su documentación asociada. Luego se realizan las pruebas de implantación y de aceptación del sistema en su totalidad.

Asimismo, se llevan a cabo las tareas necesarias para la preparación del mantenimiento, siempre y cuando se haya decidido que éste va a efectuarse.

Además hay que determinar los servicios (y niveles para cada uno) que requiere el sistema que se va a implantar, y el acuerdo que se adquiere una vez que se inicie la producción. Finalmente, se realizan las acciones necesarias para el inicio de la producción.

Las actividades que engloba este proceso se recogen en la siguiente figura (véase **Figura 100 - Tareas de la IAS**), en la que se indican las tareas que pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores:

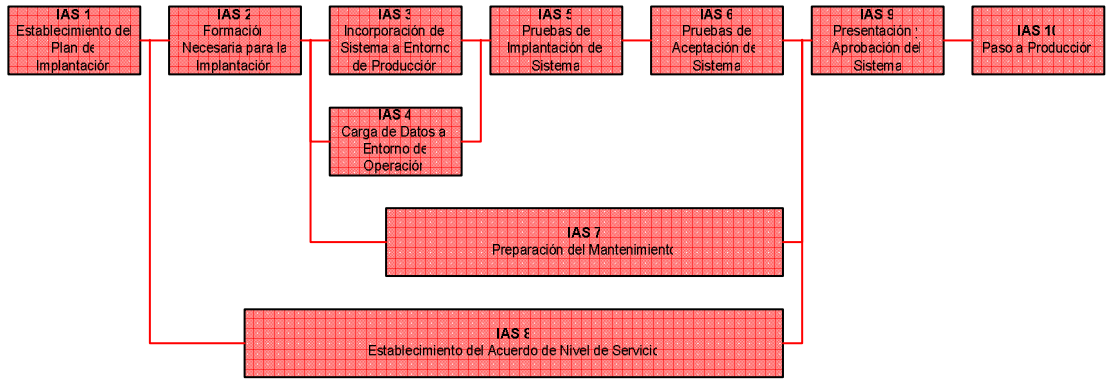


Figura 100 - Tareas de la IAS

7.3. Instalador del Sistema Seldon

La generación de un instalador para un sistema escrito en Java no es una tarea menor debido a que se pretende mantener la independencia de plataforma aún en esta etapa final de desarrollo.

Entre las múltiples aplicaciones existentes en el mercado, se optó por IzPack 3.7.2 [IzPack, 2005] por las siguientes razones:

- Es de uso libre.
- Es independiente de plataforma y genera instaladores independientes de plataforma.
- Posee gran flexibilidad y modularidad.
- Emplea la máxima compresión en la generación de archivos.

- Realiza substitución automática de variables del sistema.
- Esta totalmente integrado con Jakarta Ant.
- Permite la creación de un desinstalador.
- Provee un completo manual de usuario y se brinda un importante soporte al usuario.

7.3.1. Archivo de Configuración

Una de las características fundamentales de IzPack lo constituye el hecho de que puede ser personalizado por el usuario para cada aplicación a través de la escritura de un archivo de configuración en formato XML.

Algunos de los elementos que constituyen este archivo son:

- <info>
  Empleado para especificar información general respecto al instalador en si mismo.
- <guiprefs>
  Posibilita indicar las caracteristicas de interfaz de usuario para el proceso de instalación.
- <locale>
  Es utilizado para especificar los lenguajes que estarán disponibles durante el proceso de instalación.
- <resources>
  Se emplea como repositorio para datos adicionales a emplear por las diferentes utilidades componentes del instalador (imágenes, archivos de licencia, etc.).
- <panels>
  Permite indicar los paneles que serán mostrados durante el proceso de instalación.
- <packs>
  Lista los archivos que se necesita instalar.

Se mustra a continuación el archivo de configuración desarrollado para la aplicación Seldon:

```
<installation version="1.0">
    <info>
        <appname>Seldon</appname>
        <appversion>1.0</appversion>
        <authors>
            <author name="Sebastian Maniasi"
            email="smaniasi@yahoo.com.ar" />
        </authors>
        <javaversion>1.4</javaversion>
    </info>

    <guiprefs height="480" resizable="yes" width="640">
        <laf name="looks">
            <os family="windows" />
            <param name="variant" value="plasticXP" />
        </laf>
```

```
    </guiprefs>

    <locale>
        <langpack iso3="spa" />
        <langpack iso3="eng" />
    </locale>

    <resources>
        <res id="Installer.image" src="seldon_big.gif" />
    </resources>

    <panels>
        <panel classname="HelloPanel" />
        <panel classname="TargetPanel" />
        <panel classname="InstallPanel" />
        <panel classname="SimpleFinishPanel" />
    </panels>

    <packs>
        <pack name="Seldon" required="yes">
            <description>Aplicacion Seldon.</description>
            <fileset dir="" targetdir="$INSTALL_PATH">
                <include name="Application/**/*.*" />
            </fileset>
        </pack>
    </packs>

</installation>
```

7.3.2. Generación del Instalador

El instalador de la aplicación ha sido generado mediante las facilidades brindadas por IzPack y empleando el archivo de configuración antes descripto utilizando la siguiente instrucción por línea de comandos:

```
.\bin\compile Seldon.xml -b . -o Seldon.jar -k Standard
```

7.3.3. Ejecución del Instalador

La instalación de la aplicación se completa mediante la ejecución del instalador en los equipos cliente utilizando la siguiente instrucción por línea de comandos:

```
java -jar Seldon.jar
```

7.4. Tareas de Implantación y Aceptación

Una vez lograda esta aceptación de la gerencia, la herramienta esta siendo utilizada en sesiones de identificación de riesgos de proyectos reales de *Motorola Argentina SA* que son empleados como pilotos que permitirán

completar todas las fases del proceso definido para la inserción de nuevas tecnologías en la organización (Reporte de Evaluación de Herramientas, en inglés *Tool Evaluation Report*).

En caso de que el resultado final del proceso antes mencionado constituya una recomendación de uso del prototipo de la herramienta, se planifica dar al mismo las características de proyecto organizacional y el soporte de recursos tanto materiales como humanos que permitan la generación de una herramienta estándar que pueda ser usada en un entorno de producción a nivel organizacional.

Capítulo

8

Lic. Sebastián D. Maniasi

Conclusiones Finales y Futuras Líneas de Trabajo

8. Conclusiones Finales y Futuras Líneas de Trabajo

En este capítulo se detallan las conclusiones finales obtenidas luego de la realización del trabajo final de Tesis de Magíster y se especifican, además, las probables futuras líneas de trabajo que pueden ser tomadas como guías por quienes pretendan realizar una extensión del presente estudio.

8.1. Conclusiones Finales sobre la Metodología

El trabajo final de Tesis de Magíster fue desarrollando siguiendo los pasos propuestos por la metodología Métrica V3 [Métrica V3, 2000] y tal como se esperaba la misma contribuyó como una guía en la ejecución de un desarrollo ordenado y organizado a través de cada una de las etapas, actividades y tareas definidas para el ciclo de vida de desarrollo del producto de software.

La falta de plantillas para la generación de documentación, informes y modelos representa una de las principales carencias detectadas en la metodología utilizada y por este motivo han sido empleados mayormente durante el desarrollo del trabajo final de Tesis de Magíster los provistos por IEEE.

Considerando que el sistema ha sido desarrollado para su aplicación en una organización que cuenta y que aplica consistentemente una metodología con un alcance similar al de Métrica V3 y que el tesista cuenta con una amplia experiencia en dicho proceso, es destacable el alto grado de similitud conceptual, metodológica y organizativa que ambos presentan.

Desde el punto de vista tecnológico, la metodología UML fue utilizada como base para la aplicación del paradigma de orientación a objetos y pudo

comprobarse nuevamente su amplia adecuación y su alto grado de utilidad en este tipo de sistemas.

Finalmente, y como conclusión, es posible afirmar que una vez mas han sido comprobadas las enormes ventajas que representa el uso de una metodología durante el proceso de desarrollo de un sistema.

8.2. Conclusiones Finales sobre la Tecnología y Herramientas

El entorno tecnológico seleccionado para la construcción de la herramienta resultó apropiado y no fueron detectados inconvenientes de importancia durante el desarrollo del sistema que pudieran afectar o contradecir la elección realizada.

El alto grado de disponibilidad de bibliotecas de propósitos variados y de tipo libre existentes para la tecnología Java constituyen otro de los motivos que justifican y alientan el uso de la misma tanto para desarrollos de tipo académico como comercial.

Lo antes indicado permite reafirmar que la tecnología Java se encuentra en un estado de madurez más que suficiente como para ser considerada en la construcción de herramientas similares a la producida como producto del trabajo final de Tesis de Magíster.

Las herramientas utilizadas durante el desarrollo mostraron su utilidad y efectividad a lo largo de todo el proceso, incrementando enormemente la productividad, organización y seguridad de todas las actividades de proyecto.

Se presentan a continuación algunas conclusiones relacionadas con algunas de las herramientas utilizadas durante el desarrollo de la Tesis de Magíster:

8.2.1. Rational Rose [Rational, 2002]

Fue empleado para la generación y mantenimiento de la documentación técnica del sistema, desde las actividades de especificación de requerimientos hasta las de diseño detallado.

La principal ventaja obtenida lo representó la posibilidad de llevar a cabo y propagar modificaciones fácilmente en todos los modelos y la simplicidad en la generación y obtención de reportes de diferentes tipos que fueron empleados en la presente Tesis de Magíster. Asimismo, la posibilidad de generar automáticamente código Java desde los modelos de clases significó un importante aporte durante la fase de Construcción del Sistema.

Como carencia destacable puede mencionarse la falta de un mecanismo en la herramienta para mantener la trazabilidad entre los diferentes componentes de cada uno de los modelos y entre los modelos entre sí.

Como conclusión final puede mencionarse que se considera a la herramienta como un apropiado soporte de las actividades técnicas de construcción de un sistema de software orientado a objetos y documentado empleando UML.

8.2.2. Eclipse [Eclipse, 2005]

Fue empleado para la generación, compilación, ejecución y verificación del código fuente de la aplicación.

El alto grado de integración entre esta herramienta y diferentes adiciones de tipo libre posibilitaron la construcción de la interfaz de usuario y la realización y ejecución de pruebas unitarias en este mismo entorno. Es destacable además la simplicidad y amigabilidad del producto ya que el mismo facilita por esto el logro de una alta productividad en su uso.

Como crítica principal a esta herramienta se puede mencionar el alto grado de requerimientos de hardware que posee para su instalación y utilización.

Como conclusión final puede mencionarse que se considera a la herramienta como un apropiado soporte de las actividades de programación y pruebas iniciales de un sistema.

8.2.3. Rational ClearCase [Rational ClearCase, 2004]

Fue empleado para llevar a cabo las tareas de gestión de la configuración a lo largo de todo el ciclo de vida del proyecto. La misma constituye la herramienta estándar para este tipo de tareas en Motorola Argentina SA y nuevamente ha demostrado su altísima eficiencia, su facilidad de uso y sus muy amplias capacidades de visualización y gestión de versiones, etiquetas y líneas base.

8.2.4. IzPack [IzPack, 2005]

Fue empleado para llevar a cabo la generación del instalador del sistema. Demostró ser una potente herramienta independiente de plataforma y con grandes posibilidades de personalización y adaptación a las necesidades de diferentes potenciales usuarios. A pesar de ser de uso libre posee un muy buen nivel de soporte y un completo tutorial y manual de usuario que facilitan su rápido aprendizaje. Su principal falencia lo constituye la falta de una interfaz gráfica para la configuración del archivo de instalación lo cual obliga a la escritura manual de un archivo XML siguiendo una estructura predefinida.

Como conclusión final puede mencionarse que se considera a la herramienta como una opción muy apropiada para cumplir con las tareas de generación de instaladores de aplicaciones Java.

8.3. Conclusiones Finales sobre el Trabajo de Tesis de Magíster

El principal aspecto a destacar lo constituye el hecho de haber alcanzado todos los objetivos propuestos tanto a nivel metodológico como académico y técnico dentro de los plazos establecidos y con las restricciones definidas.

Otro aspecto a mencionar es el de haber logrado la efectiva aplicación de una completa gestión de proyectos a pesar de haber desarrollado el producto mediante la ejecución de un proyecto prácticamente unipersonal.

Cabe destacarse además la aplicación de algunas técnicas de Ingeniería del Conocimiento en el análisis, diseño y construcción de la solución propuesta por el sistema.

Finalmente, la continua y progresiva consulta y validación de las actividades realizadas, de las conclusiones obtenidas y los productos generados ha demostrado nuevamente ser un poderoso y efectivo mecanismo de aseguramiento de la calidad del software.

8.4. Futuras Líneas de Trabajo

Inicialmente la herramienta obtenida como resultado de la Tesis de Magíster será utilizada por la empresa Motorola Argentina SA como parte de las actividades de implementación del PA de Gestión de Riesgos del modelo de calidad CMMI, lo cual brindará la valiosa realimentación del uso del sistema en entornos y situaciones reales lo cual dará origen muy probablemente a nuevas propuestas de ampliaciones, modificaciones y hasta a posibles correcciones.

Aún antes de esto, entre las futuras líneas de trabajo ya identificadas para la herramienta producida pueden destacarse las siguientes:

- Ampliación de la Taxonomía Base mediante la incorporación de nuevas categorías de riesgos, nuevos riesgos y nuevos tipos de proyectos.

- Ampliación de la base de reglas del sistema mediante la identificación y la adición de nuevas interrelaciones entre los riesgos.

- Ampliación de las facilidades del sistema con ayudas en línea y tutores brinden explicaciones y permitan conocer y comprender el modelo de gestión de riesgos que sirve de base para el sistema.

- Incorporación de las restantes fases del modelo de gestión de riesgos.

- Definición e incorporación de métricas que posibiliten realizar un seguimiento cuantitativo de los resultados generados por la aplicación del nuevo modelo de identificación de riesgos planteado.

- Agregado de reportes gráficos que destaquen las principales áreas de riesgo dentro de un proyecto y las relaciones detectadas entre los riesgos identificados.

- Incorporación de un sistema de recursos que brinden un soporte de gestión de reglas mas complejo y mas completo (Mandarax [Mandarax, 2005], por ejemplo) con el fin de posibilitar el uso de diferentes mecanismos de control, considerar el manejo de umbrales de confianza, operar bajo incertidumbre, etc.

- Optimización del mecanismo de búsqueda de preguntas a realizar al usuario con el propósito de lograr mejores prestaciones y una mayor eficiencia del sistema (considerando el pequeño número de relgas existente, la estrategia actual se basa en un mecanismo similar al de generar y comprobar: las preguntas se realizan por orden de identificación de los riesgos sin emplear una heurística particular).

- Reemplazo de la escala difusa utilizada actualmente en la Taxonomía Base (riesgos de probabilidad Alta, Media o Baja) por una escala numérica que posibilite indicar la probabilidad de ocurrencia de cada uno de los riesgos y que permita la utilización de distintos umbrales o niveles de confianza para el sistema.

- Adaptación de la herramienta para su ejecución en modo remoto o en un ambiente Internet.

Capítulo

9

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Bibliografía

9. Bibliografía

En este capitulo se detallan la bibliografía y abreviaturas empleadas durante el desarrollo del presente trabajo.

9.1. REFERENCIAS BIBLIOGRAFICAS

[Active Risk Manager, 2004]. *Active Risk Manager*. Disponible en http://www.strategicthought.com/QuickPlace/stlwebsite/PageLibrary80256D6D007EC8CB.nsf/h_Toc/d84e8c26d5c18c1888256deb006ef979/?OpenDocument. Página vigente al 24-Jul-2005.

[ANSI/IEEE 1042, 1987]. *IEEE Guide to Software Configuration Management (ANSI/IEEE STD 1042-1987).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[ASC, 2003]. *Risk Management Process & Implementation*. American Systems Corporation. Disponible en http://www.2asc.com. Página vigente al 24-Jul-2005.

[Booch & Jacobson & Rumbaugh, 2000]. *El Proceso Unificado de Desarrollo de Software*. 438 páginas. Editorial Addison-Wesley. ISBN 8478290362.

[Caper Jones, 1994]. *Assessment and Control of Software Risks*. Editorial Prentice Hall. ISBN 0137414064.

[Charette, 1989]. *Software Engineering Risk Analysis and Management*. McGraw-Hill/Intertext.

[CMMI, 2002]. *Capability Maturity Model Integration*. Software Engineering Institute. Disponible en http://www.sei.cmu.edu/cmmi/cmmi.html. Página vigente al 24-Jul-2005.

[Eclipse, 2005]. *Eclipse Platform SDK 3.0.1*. Disponible en http://www.eclipse.org/. Página vigente al 24-Jul-2005.

[Futrell, Shafer & Shafer, 2002]. *Quality Software Project Management*. Editorial Prentice Hall. ISBN 0130912972.

[Hunt, 2001]. *The Hierarchical MVC.* Disponible en http://www.jaydeetechnology.co.uk/planetjava/tutorials/swing/hmvc.PDF. Página vigente al 24-Jul-2005.

[Hantos, 2000]. *A Practical Approach to Quantifying Risk Evaluation Results*. Disponible en http://www.stsc.hill.af.mil/crosstalk/2000/02/hantos.html. Página vigente al 24-Jul-2005.

[IEEE 1058, 1987]. *IEEE Standard for Software Project Management Plans (IEEE Std 1058.1-1987).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[IEEE 1540, 2001]. *IEEE Standard for Software Life Cycle Processes-Risk Management (IEEE Std 1540-2001).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[IEEE 730, 2002]. *IEEE Standard for Software Quality Assurance Plans (IEEE Std 730-2002).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[IEEE 829,1983]. *IEEE Standard for Software Test Documentation (ANSI/IEEE Std 829-1983).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[IEEE 830,1998]. *IEEE recommended practice for software requirements specifications (IEEE Std 830-1998).* Disponible en http://www.standards.ieee.org/. Página vigente al 24-Jul-2005.

[IzPack, 2005]. *IzPack 3.7.2*. Disponible en http://www.izforge.com/izpack/. Página vigente al 24-Jul-2005.

[J2SE, 2004]. *Java 2 Standard Edition, Sun Microsystems.* Disponible en http://java.sun.com/j2se. Página vigente al 24-Jul-2005.

[JasperReports, 2005]. *JasperReports, versión 0.6.4.* Disponible en http://jasperreports.sourceforge.net/. Página vigente al 24-Jul-2005.

[JEOPS, 2005]. *Jeops, versión 2.2.* Disponible en http://www.jeops.org/. Página vigente al 24-Jul-2005.

[Mandarax, 2005]. *Mandarax, versión 3.3.2.* Disponible en http://mandarax.sourceforge.net/. Página vigente al 24-Jul-2005.

[Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich y Clay F. Walker, 1993]. *Taxonomy-Based Risk Identification*. 90 páginas. Disponible en www.sei.cmu.edu/pub/documents/93.reports/pdf/tr06.93.pdf. Página vigente al 24-Jul-2005.

[Métrica V3, 2000]. *Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información*. Ministerio de Administraciones Públicas Español. Disponible en http://www.csi.map.es/csi/metrica3/index.html. Página vigente al 24-Jul-2005.

[Microsoft, 2002]. *MSF Risk Management Discipline v.1.1*. 54 Páginas. Disponible en http://www.microsoft.com/downloads/details.aspx?FamilyID=6c2f2c7e-ddbd-448c-a218-074d88240942&DisplayLang=en. Página vigente al 24-Jul-2005.

[Motorola LMPS, 1999]. Motorola Land Mobile Products Sector, "Risk Management". Publicado el 20-Abr-1999

[Pazos, 2003]. *Unidad 22, Estudio de Viabilidad*. Facultad de Informática, Universidad Politécnica de Madrid.

[Rational, 2002]. *Rational Rose – Modeler Edition.*

[Rational ClearCase, 2004]. *Rational ClearCase,.*

[RiskTrak, 2004]. *RiskTrak*. Disponible en http://www.iog.com.au/risktrack.html. Página vigente al 24-Jul-2005.

[Rosenberg, L., Hammer, T., Gallo, A., 1999]. *Continuous Risk Management at NASA*. Disponible en http://satc.gsfc.nasa.gov/support/ASM_FEB99/crm_at_nasa.html. Página vigente al 24-Jul-2005.

[SEI, 2004]. *Software Engineering Institute*. Disponible en http://www.sei.cmu.edu/programs/sepm/risk. Página vigente al 24-Jul-2005.

[TBS, 2002]. *Treasury Board Secretariat of Canada: Risk Management.* Disponible en http://www.cio-dpi.gc.ca/emf-cag/solutions/ef-quest-ca/ef-quest-ca_e.asp. Página vigente al 24-Jul-2005.

[TeraQuest, 1998]. *Risk Management Process for Projects*. Material del curso dictado por Liveware IS.

[TRIMS, 2004]. *Technical Risk Identification and Mitigation System*. Disponible en http://www.bmpcoe.org/pmws/download/trims.html. Página vigente al 24-Jul-2005.

[Walker, 1998]. *Software Project Management, A Unified Framework*. Addison Wesley.

[Webster, 2004]. *Merriam-Webster's Collegiate Dictionary.* Disponible en http://www.webster.com. Página vigente al 24-Jul-2005.

[WelcomRisk, 2004]. *WelcomRisk*. Disponible en http://www.welcom.com/content.cfm?page=489. Página vigente al 24-Jul-2005.

[Wideman, 1998]. *Project and Program Risk Management: A Guide to Managing Project Risks and Opportunities (PMBOK Handbooks)*. Editorial PMI. ISBN 1880410060.

[Williams Ray, 2003]. *New directions in Risk Management at SEI*. NASA RMC IV. Disponible en www.sei.cmu.edu/programs/acquisition-support/presentations/williams/new-directions/new-directions.pdf. Página vigente al 24-Jul-2005.

9.2. Abreviaturas

CC	ClearCase
CCC	Comité de Control de Cambios
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
DTD	Document Type Definition
EVS	Estudio de Viabilidad del Sistema
GC	Gestor de Configuración
GCG	Gestor de Configuración Global
GCS	Gestión de Configuración de Software
I+D	Investigación y Desarrollo
IC	Ítems de Configuración
IEEE	Institute of Electrical and Electronics Engineers
IPD	Integrated Product Development
IPPD	Integrated Product and Process Development
MIRT	Modelo para la Identificación de Riesgos en base a Taxonomías
N/A	No Aplica
PA	Process Area
SE	System Engineering
SEI	Software Engineering Institute
SG	Specific Goal
SOW	Statement of Work
SP	Specific Practice
SW	Software

SS	Supplier Sourcing
TBQ	Taxonomy-Based Questionnaire
UML	Unified Modeling Language
VOB	Versioned Object Base
WBS	Work Breakdown Structure
XML	eXtensible Markup Language

Apéndice

A

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Plan de Gestión del Proyecto

I. Apéndice A: Plan de Gestión del Proyecto

I.1. La Gestión de Proyectos en MÉTRICA Versión 3

La **Gestión de Proyectos** tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de una herramienta. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos de manera inmediata.

Se distinguen tres grupos de actividades:

- *Actividades de Inicio del Proyecto (GPI).*
  Al principio del proyecto, al concluir el proceso Estudio de Viabilidad del Sistema, se realizarán las actividades de Estimación de Esfuerzo y Planificación del proyecto.

- *Actividades de Seguimiento y Control (GPS).*
  Comprenden desde la asignación de las tareas hasta su aceptación interna por parte del equipo de proyecto, incluyendo la gestión de incidencias y cambios en los requisitos que puedan presentarse y afectar a la planificación del proyecto. El Seguimiento y Control del proyecto se realizan durante los procesos de Análisis, Diseño, Construcción, Implantación y Aceptación, y Mantenimiento del Sistema, para vigilar el correcto desarrollo de las actividades y tareas establecidas en la planificación.

- *Actividades de Finalización del Proyecto (GPF).*
  Por último, al concluir el proyecto se realizan las tareas propias de Cierre del Proyecto y Registro de la Documentación de Gestión.

En función de las características del proyecto puede ser aconsejable emplear herramientas software de soporte a las técnicas, disponibles en el mercado.

I.2. Introducción

Este plan describe la forma en la cual se llevará a cabo la Gestión del Proyecto para el proyecto de desarrollo de software para la Tesis de Magíster.

El plan de Gestión del Proyecto ha sido desarrollado según [IEEE 1058, 1987].

I.3. Descripción del Proyecto

El presente proyecto tiene como objetivo central el de desarrollar una herramienta que brinde la posibilidad de efectuar tareas de Identificación de Riesgos en base a Taxonomías estándar en organizaciones de software de cualquier tipo.

I.3.a. Entregables

- Tesis de Magíster.
- Prototipo de la herramienta.

I.3.b. Evolución del Plan

El presente plan puede evolucionar como consecuencia de:
- Cambios significativos en el alcance del proyecto.
- Demoras mayores a tres semanas.
- La realización de algunos de los riesgos de alta exposición identificados.

I.3.c. Documentos Referenciados

- Plan de Gestión de la Configuración.
- Plan de Aseguramiento de la Calidad.
- Plan de Gestión de Riesgos.
- Calendario de Proyecto.

I.4. Organización del Proyecto

I.4.a. Modelo de Proceso

Con el propósito de llevar a cabo el desarrollo de la herramienta durante el proyecto de Tesis de Magíster se utilizará la Metodología MÉTRICA Versión 3 [Métrica V3, 2000], se emplearán estándares de documentación basados en los propuestos por la IEEE y se utilizará un ciclo de vida tendiente a la generación de prototipos incrementales.

I.4.b. Modelo de Estimación

Para la realización de las estimaciones del presente proyecto se utilizará un Wide Band Delphi con expertos y datos provenientes de la industria.

I.4.c. Estructura Organizacional

El equipo de proyecto esta compuesto por un grupo permanente conformado por un Desarrollador de Tesis, un Especialista en Calidad y un Gerente de Proyecto. Asimismo, y según la disponibilidad de personal y las necesidades de proyecto, podrán incorporarse desarrolladores que colaborarán en etapas específicas y actividades puntuales de la construcción del producto.

I.4.d. Interfaces de Proyecto

La gerencia de Motorola Argentina SA cumplirá el rol de cliente del proyecto.

La actualización de estado de proyecto será realizada de manera periódica y bimensual por el Desarrollador de Tesis a la gerencia de Motorola Argentina SA.

I.4.e. Responsabilidades de Proyecto

Las responsabilidades de administración de proyecto, administración de la configuración, aseguramiento de la calidad, análisis, diseño, construcción, validación y verificación de la solución propuesta están a cargo del Desarrollador de Tesis quién recibirá soporte en dichas tareas tanto del Especialista en Calidad como del Gerente de Proyecto y de los miembros temporales del equipo de proyecto.

El Especialista en Calidad y el Gerente de Proyecto serán los encargados de garantizar la calidad de los productos de proyecto producidos mediante su participación en actividades de revisión por pares.

I.5. Proceso de Administración

I.5.a. Objetivos y Prioridades de la Administración

El objetivo central es proveer de una herramienta para lograr ordenar y sistematizar la actual dispersión de información relacionada con la identificación de riesgos, compatibilizar los criterios y procedimientos para su tratamiento, y homogeneizar los niveles de desagregación y las posibilidades de análisis.

I.5.b. Asunciones, Dependencias y Restricciones

Para el cumplimiento del presente proyecto se asume que cada una de las áreas de la organización ha comprometido hasta un 2.5% del tiempo de trabajo de sus miembros y hasta un 20% del tiempo de sus miembros que han sido seleccionados como miembros permanentes del equipo de proyecto; sin el cumplimiento de este compromiso el proyecto no podrá llevarse a cabo.

También se requiere el apoyo y compromiso de la Dirección de Motorola Argentina SA en su rol de auspiciante de proyecto y de la gerencia en su rol de cliente del proyecto.

Finalmente es necesaria la predisposición y colaboración de todo el personal de la compañía que puede ser consultado o requerido ante situaciones de necesidad especiales.

I.5.c. Manejo de Riesgos

Referirse al Plan de Gestión de Riesgos.

I.5.d. Plan de Prevención de Riesgos

El plan de prevención de riesgos incluye:

- Capacitación.
- Desarrollo de guías y listas de control.
- Análisis causal de errores.
- Reuniones periódicas de equipo.

I.5.e. Plan de Personal

El personal del proyecto y estructura del equipo de trabajo permanecerán esTabla durante el proceso completo.

I.5.f. Plan de Capacitación

Durante el desarrollo del proyecto el equipo revisará la capacitación recibida en:

- Técnicas de análisis y diseño.
- Técnicas de realización de entrevistas.
- Técnicas de descripción de estructuras de datos y procesos.

I.6. Proceso Técnicos

I.6.a. Métodos, Herramientas y Técnicas

Las metodologías, herramientas y técnicas descriptas por el material correspondiente al Magíster en Ingeniería de Software serán empleadas para realizar las actividades de análisis, diseño y construcción del producto del proyecto.

Todos los artefactos de proyecto utilizarán una revisión por pares como método de aseguramiento de la calidad.

MS Project será usada como herramienta de calendarización.

IzPack será empleado como utilitario para la generación de los instaladores del sistema.

I.6.b. Funciones de Soporte del Proyecto

El proyecto va a requerir el soporte de todas las áreas de la organización.

I.7. Artefactos, Calendario y Presupuesto

I.7.a. Artefactos

- Plan de Gestión del Proyecto.
- Plan de Gestión de la Configuración.
- Plan de Gestión de Riesgos.
- Plan de Aseguramiento de Calidad.
- Documentos auxiliares de proyecto: minutas, acuerdos, etc.
- Tesis de Magíster.

I.7.b. Recursos Requeridos

- 3 Licencias de MS Windows 2000.

- 2 Licencias de MS Project.
- 3 Licencias de MS Word.
- 1 Licencia de Rational Rose – Modeler Edition.
- 3 PCs.
- 1 Impresora.

I.7.c. Calendario

Referirse al Calendario de Proyecto.

I.7.d. Presupuesto

La gestión presupuestaria no se encuentra incluida en el alcance de las actividades de gestión del presente proyecto.

Apéndice

B

Lic. Sebastián D. Maniasi

Estimaciones y Calendario

II. Apéndice B: Estimaciones y Calendario

II.1. Estimaciones

Con el propósito de llevar a cabo el desarrollo de la herramienta durante el proyecto de Tesis de Magíster se utilizará la Metodología MÉTRICA Versión 3 [Métrica V3, 2000].

La actividad de estimación se llevo a cabo siguiendo el Proceso de Estimación definido en Motorola Argentina SA. Entre los aspectos mas relevantes del mismo pueden destacarse los siguientes:

1. Debe realizarse una estimación de tamaño para cada uno de los productos a generar por el proyecto (tanto software como documentación) empleando al menos un método de estimación. En caso de emplearse mas de un método de estimación para determinar el tamaño del proyecto, deberá verificarse que todos los valores obtenidos converjan a un valor (considerando como desvío aceptable ± 15%); si los valores no convergen debe repetirse el proceso. Una vez lograda la convergencia, se selecciona un valor como final para el tamaño del proyecto (el criterio de los estimadores determina qué valor se selecciona, puede tomarse, por ejemplo, el mayor valor obtenido, el menor valor obtenido, un promedio entre ambos o cualquier otro criterio que se considere apropiado).
2. Tomando como base el tamaño del proyecto se realiza una estimación de esfuerzo de desarrollo empleando al menos dos métodos de estimación (las actividades de revisiones y retrabajo no se consideran en esta etapa). Deberá verificarse que todos los valores obtenidos converjan a un valor (considerando como desvío aceptable ± 15%); si los valores no convergen debe repetirse el proceso. Una vez lograda la

convergencia, se selecciona un valor como final para el esfuerzo del proyecto (el criterio de los estimadores determina qué valor se selecciona, puede tomarse, por ejemplo, el mayor valor obtenido, el menor valor obtenido, un promedio entre ambos o cualquier otro criterio que se considere apropiado).

3. Una vez logrado un valor final para el esfuerzo se realiza un ajuste (+ 25%) relacionado con la realización de actividades de revisiones y retrabajo. El esfuerzo obtenido de este ajuste constituirá al esfuerzo finalmente asociado al proyecto.

La determinación de tamaño del proyecto se llevó a cabo empleando dos métodos diferentes según se tratara de determinar la cantidad de documentación a generar o la cantidad de código a producir:

- Documentación:
  Para la determinación del tamaño de la documentación a generar se empleó la consulta a Registros Históricos como método de estimación; con este propósito, se accedió a la base de datos disponible en el ITBA para trabajos de tesis que han seguido la misma metodología que el actual. El valor final obtenido para la documentación es de 450 páginas.
- Código:
  Para la determinación del tamaño del código a generar se empleó Wide-Band Delphi como método de estimación; con este propósito, se informó a los estimadores los objetivos, alcance, restricciones y suposiciones del proyecto. El valor final obtenido para el código es de 4800 líneas de código fuente Java. Es importante destacar que el valor antes mencionado no incluye el código reusado en forma de componentes existentes ni el código que será automáticamente generado tanto por la herramienta de análisis/diseño a utilizar ni por el entorno de desarrollo a emplear.

La determinación de esfuerzo del proyecto se llevó a cabo empleando dos métodos diferentes tanto para determinar el esfuerzo asociado a la producción de la documentación a generar como para determinar el esfuerzo asociado a la producción del código a construir:

- Documentación:
  Para la determinación del esfuerzo de producción de la documentación a generar se empleó la consulta a Registros Históricos y Wide-Band Delphi como métodos de estimación; con este propósito, se accedió a la base de datos disponible en Motorola Argentina para proyectos que han seguido las actividades detalladas por la metodología a utilizar y se informó a los estimadores los objetivos, alcance, restricciones y suposiciones del proyecto.
- Código:
  Para la determinación del tamaño del código a generar se empleó la consulta a Registros Históricos y PERT como métodos de estimación; con este propósito, se accedió a la base de datos disponible en Motorola Argentina para proyectos de similar complejidad que han empleado el mismo lenguaje a ser utilizado en el desarrollo de la Tesis de Magíster y se informó a un grupo de estimadores los objetivos,

alcance, restricciones y suposiciones del proyecto solicitando la determinación de valores de esfuerzo pesimista, optimista y probable.

Tanto para esfuerzo de producción de documentación como de código se obtuvo convergencia entre los valores determinados y se empleó un promedio entre el esfuerzo obtenido por cada uno de los métodos para determinar el valor final que, luego de realizar el ajuste por revisiones y retrabajo, se empleará durante la actividad de calendarización del proyecto.

Al emplear el método de **Wide-Band Delphi** se dan las especificaciones del proyecto y del producto a generar a un grupo de expertos que discuten sobre su entendimiento respecto a los mismos y luego remiten sus estimaciones individuales al coordinador de la sesión quién establece la media de las estimaciones, así como también la estimación máxima y mínima, entregándolas luego a los expertos. Cada estimador recibe información sobre su estimación y las ajenas de forma anónima previamente a una nueva sesión donde se discuten los resultados obtenidos y se ajustan, revisan y, en caso de ser preciso, corrigen los valores obtenidos. El proceso se repite hasta que la estimación converge de forma razonable. La principal ventaja de este método de estimación es que es relativamente barato y puede ser bastante exacto si los expertos tienen experiencia directa en sistemas similares; la desventaja principal lo constituye el hecho de que los valores obtenidos pueden ser muy imprecisos si no se dispone de los expertos adecuados.

El emplear el método de **Registros Históricos** implica comparar las especificaciones de un proyecto, con las de otros proyectos previos de los cuales se poseen métricas de esfuerzo y tamaño considerando factores tales como tamaño, complejidad, tecnología y personal afectado al proyecto. La principal ventaja de este método de estimación es que puede ser bastante preciso si se disponen de datos exactos de proyectos previos; la desventaja principal lo constituye el hecho de que los valores obtenidos pueden ser muy imprecisos si no se realiza un apropiado ajuste según los factores antes mencionados.

Al emplear el método de **PERT** se determina para cada actividad una estimación más probable (denotada por R), una estimación optimista (denotada por O) y una estimación pesimista (denotada por P). La estimación más probable intenta ser la estimación más realista del tamaño o del tiempo que puede requerir una actividad; en términos estadísticos, es una estimación de la moda de la distribución de probabilidad para el tiempo de la actividad. La estimación optimista procura ser el tiempo o tamaño poco probable pero posible en caso de que todo salga correctamente y es una estimación de la cota inferior de la distribución de la probabilidad. Por ultimo, se intenta que la estimación pesimista sea el tiempo poco probable pero posible si se producen demasiados inconvenientes y se trata de una estimación de la cota superior de la distribución de probabilidad. Una vez determinados estos valores, el valor final de la estimación queda determinado por la siguiente fórmula:

$$\frac{O + 4 * R + P}{6}$$

El siguiente cuadro (véase **Tabla 28 - Estimación de Esfuerzo de Construcción del Sistema**) detalla, por cada uno de los procesos y actividades que componen la metodología, el esfuerzo estimado luego de completar el proceso antes detallado para su completitud:

PROCESOS		
Proceso y Actividades		**Esfuerzo Estimado (Hs.)**
Planificación de Sistemas de Información (PSI)		
Desarrollo de Sistemas de Información – Estudio de Viabilidad del Sistema (EVS)		61.5
	Descripción del Dominio	14
	Establecimiento del Alcance del Sistema	8
	Estudio de la Situación Actual	14
	Definición de Requisitos del Sistema	6
	Estudio de Alternativas de Solución	10
	Valoración de las Alternativas	8
	Selección de la Solución	1.5
Desarrollo de Sistemas de Información – Análisis del Sistema de Información (ASI)		140
	Definición del Sistema	16
	Establecimiento de Requisitos	20
	Identificación de Subsistemas de Análisis	4
	Análisis de los Casos de Uso	20
	Análisis de Clases	22
	Definición de Interfaces de Usuario	22
	Análisis de Consistencia y Especificación de Requisitos	14
	Especificación del Plan de Pruebas	20
	Aprobación del Análisis del Sistema de Información	2
Desarrollo de Sistemas de Información – Diseño del Sistema de Información (DSI)		144
	Definición de la Arquitectura del Sistema	12
	Diseño de la Arquitectura de Soporte	8
	Diseño de Casos de Uso Reales	20
	Diseño de Clases	28
	Diseño Físico de Datos	10
	Verificación y Aceptación de la Arquitectura del Sistema	4
	Generación de Especificaciones de Construcción	12

	Diseño de la Migración y Carga Inicial de Datos	6
	Especificación Técnica del Plan de Pruebas	40
	Establecimiento de Requisitos de Implantación	4
	Aprobación del Diseño del Sistema de Información	4
Desarrollo de Sistemas de Información – Construcción del Sistema de Información (CSI)		146
	Preparación del Entorno de Generación y Construcción	8
	Generación del Código de los Componentes y Procedimientos	66
	Ejecución de las Pruebas Unitarias	8
	Ejecución de las Pruebas de Integración	16
	Ejecución de las Pruebas del Sistema	24
	Definición de la Formación de Usuarios Finales	7
	Construcción de los Componentes y Procedimientos de Migración y Carga Inicial de Datos	13
	Aprobación del Sistema de Información	4
Desarrollo de Sistemas de Información – Implantación y Aceptación del Sistema (IAS)		56
	Establecimiento del Plan de Implantación	12
	Formación Necesaria para la Implantación	7
	Incorporación del Sistema al Entorno de Operación	4
	Carga de Datos al Entorno de Operación	3
	Pruebas de Implantación del Sistema	6
	Pruebas de Aceptación del Sistema	18
	Presentación y Aprobación del Sistema	4
	Paso a Producción	2
Mantenimiento de Sistemas de Información (MSI)		
Subtotal		*547.5*
INTERFACES		
Interfaz		**Esfuerzo Estimado (Hs.)**
Gestión de Proyectos		57
Seguridad		28

Gestión de la Configuración	28
Aseguramiento de la Calidad	142
Subtotal	*255*
TOTAL	***802.5***

Tabla 28 - Estimación de Esfuerzo de Construcción del Sistema

Se estima además una dedicación semanal de 20 horas/hombre empleadas en el desarrollo del sistema, motivo por el cual el tiempo total estimado para la completitud de la tesis es de 40 semanas o 10 meses.

Finalmente, es preciso indicar que en la estimación realizada no han sido incluidos los procesos PSI y MSI debido a que:

- El sistema propuesto es independiente de un ámbito organizativo determinado motivo por el cual no es necesario llevar a cabo el desarrollo de un Plan de Desarrollo de Sistemas específico, sino de una herramienta que pueda ser empleada en organizaciones de software de cualquier tipo.
- El alcance del proyecto esta fijado hasta la obtención de la primera versión del asistente por lo que no se contemplan actividades relacionadas al mantenimiento del mismo.

II.2. Calendario

El proyecto estará guiado por el siguiente calendario (véase **Figura 101 - Calendario del Proyecto**):

Id	Nombre de tarea	Comienzo	Fin
1	**Identificación de riesgos de proyectos de software en base a taxonomías**	**lu 01/11/04**	**ju 11/08/05**
2	**Desarrollo de Sistemas de Información**	**lu 01/11/04**	**ma 09/08/05**
3	**Estudio de Viabilidad del Sistema (EVS)**	**lu 01/11/04**	**ma 07/12/04**
11	**Análisis del Sistema de Información (ASI)**	**ma 07/12/04**	**ma 15/02/05**
21	**Diseño del Sistema de Información (DSI)**	**ma 15/02/05**	**sá 30/04/05**
33	**Construcción del Sistema de Información (CSI)**	**sá 30/04/05**	**ma 12/07/05**
42	**Implantación y Aceptación del Sistema (IAS)**	**ma 12/07/05**	**ma 09/08/05**
51	**Interfaces**	**lu 01/11/04**	**ju 11/08/05**
52	Gestión de Proyectos	lu 01/11/04	ju 11/08/05
53	Seguridad	lu 01/11/04	ju 11/08/05
54	Gestión de la Configuración	lu 01/11/04	ju 11/08/05
55	Aseguramiento de la Calidad	lu 01/11/04	ju 11/08/05

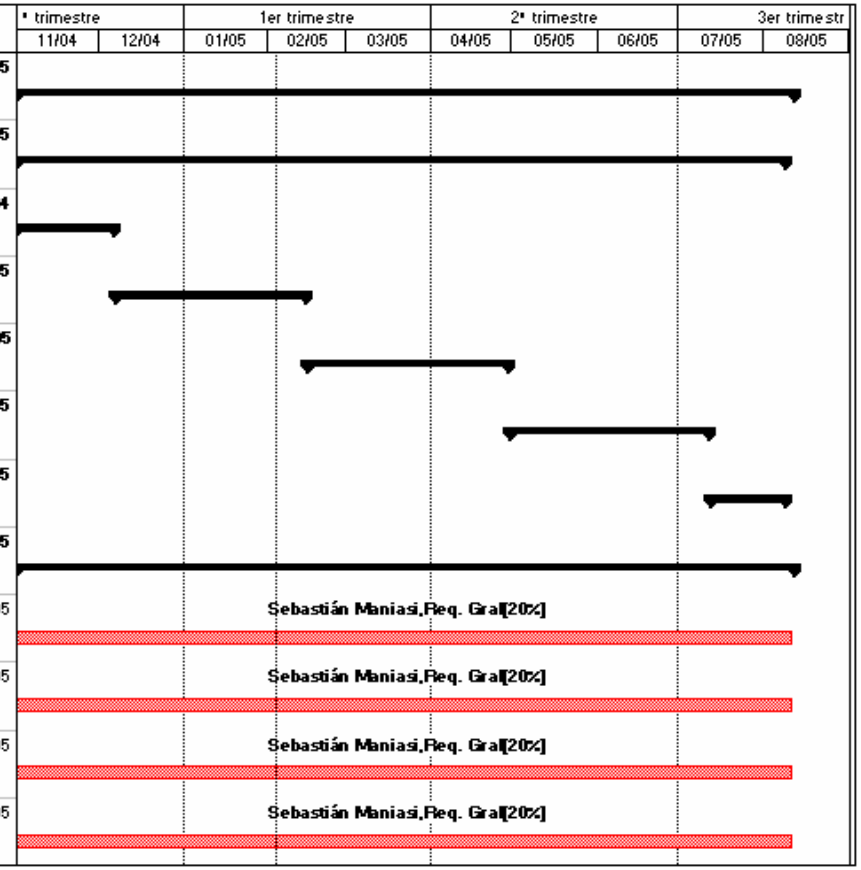


Figura 101 - Calendario del Proyecto

El proceso de Estudio de Viabilidad del Sistema (EVS) estará guiado por el siguiente calendario (véase **Figura 102 - Calendario EVS**):

Id	Nombre de tarea	Comienzo	Fin
1	Identificación de riesgos de proyectos de software en base a taxonomías	lu 01/11/04	ju 11/08/05
2	Desarrollo de Sistemas de Información	lu 01/11/04	ma 09/08/05
3	Estudio de Viabilidad del Sistema (EVS)	lu 01/11/04	ma 07/12/04
4	Descripción del Dominio	lu 01/11/04	lu 08/11/04
5	Establecimiento del Alcance del Sistema	lu 08/11/04	vi 12/11/04
6	Estudio de la Situación Actual	vi 12/11/04	vi 19/11/04
7	Definición de Requisitos del Sistema	vi 19/11/04	lu 22/11/04
8	Estudio de Alternativas de Solución	lu 22/11/04	sá 27/11/04
9	Valoración de las Alternativas	sá 27/11/04	mi 01/12/04
10	Selección de la Solución	mi 01/12/04	ma 07/12/04
11	Análisis del Sistema de Información (ASI)	ma 07/12/04	ma 15/02/05

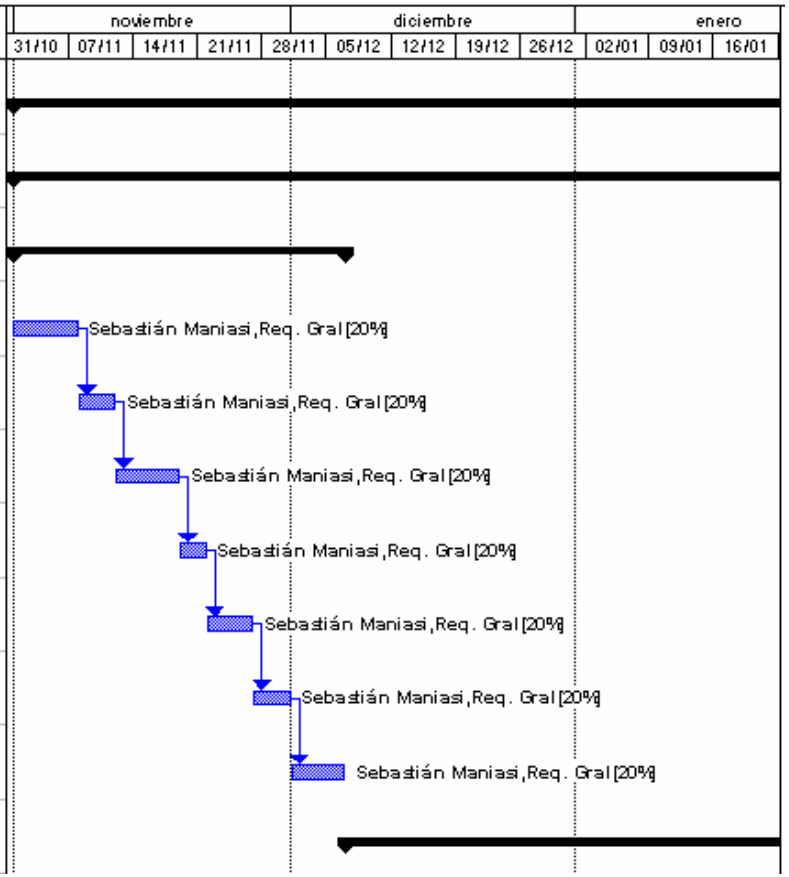


Figura 102 - Calendario EVS

El proceso de Análisis del Sistema de Información (ASI) estará guiado por el siguiente calendario (véase **Figura 103 - Calendario ASI**):

Id	Nombre de tarea	Comienzo	Fin
1	**Identificación de riesgos de proyectos de software en base a taxonomías**	**ma 07/12/04**	**ma 15/02/05**
2	**Desarrollo de Sistemas de Información**	**ma 07/12/04**	**ma 15/02/05**
3	**Análisis del Sistema de Información (ASI)**	**ma 07/12/04**	**ma 15/02/05**
4	Definición del Sistema	ma 07/12/04	mi 15/12/04
5	Establecimiento de Requisitos	mi 15/12/04	sá 25/12/04
6	Identificación de Subsistemas de Análisis	sá 25/12/04	lu 27/12/04
7	Análisis de los Casos de Uso	lu 27/12/04	ju 06/01/05
8	Análisis de Clases	ju 06/01/05	lu 17/01/05
9	Definición de Interfaces de Usuario	lu 17/01/05	vi 28/01/05
10	Análisis de Consistencia y Especificación de Requisitos	vi 28/01/05	vi 04/02/05
11	Especificación del Plan de Pruebas	vi 04/02/05	lu 14/02/05
12	Aprobación del Análisis del Sistema de Información	lu 14/02/05	ma 15/02/05

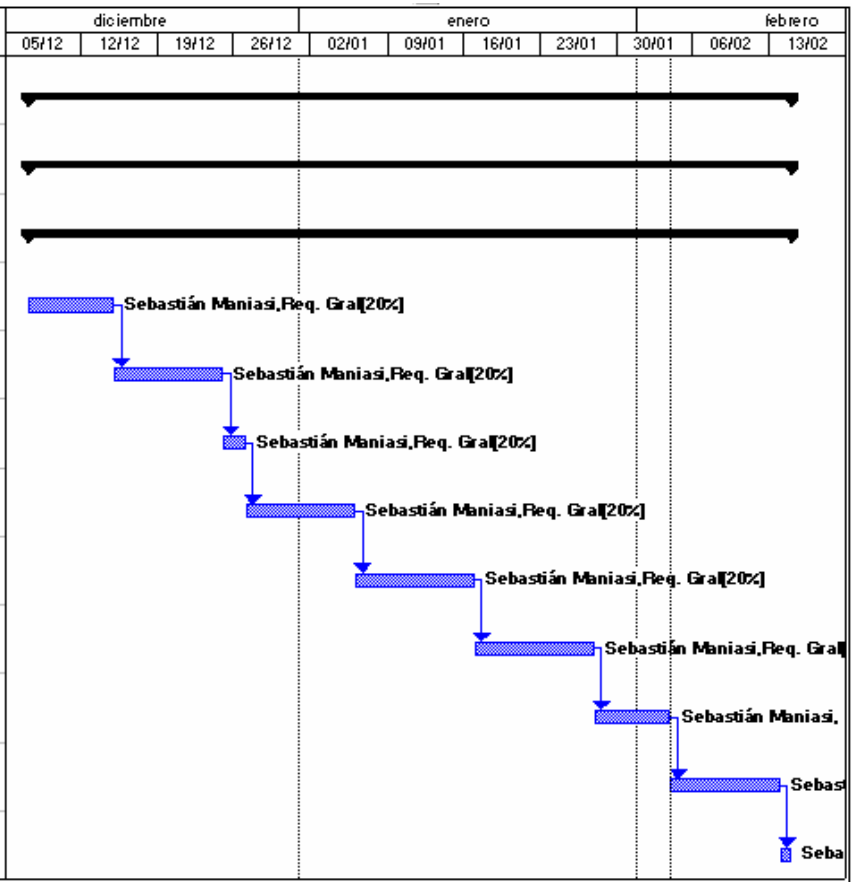


Figura 103 - Calendario ASI

El proceso de Diseño del Sistema de Información (DSI) estará guiado por el siguiente calendario (véase **Figura 104 - Calendario DSI**):

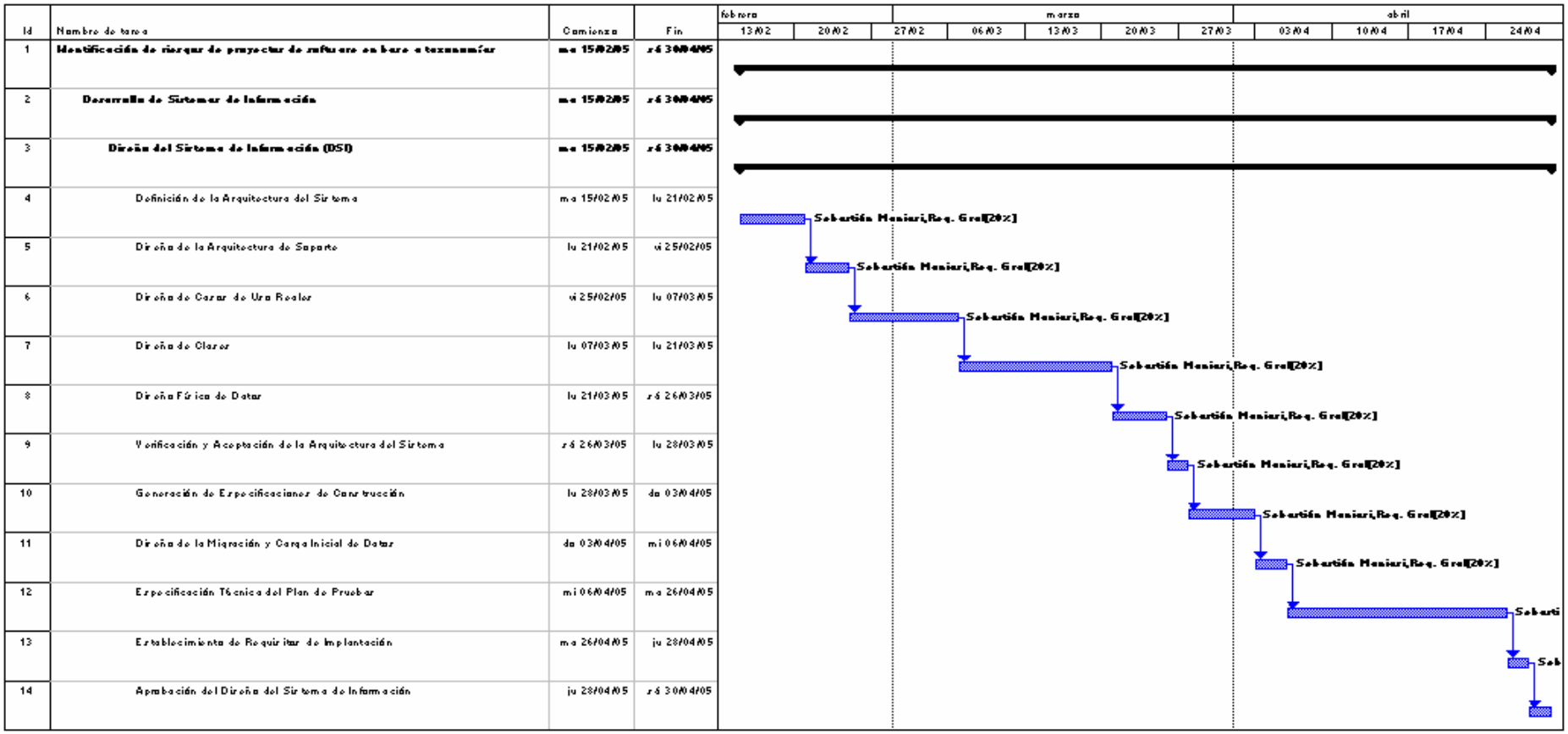


Figura 104 - Calendario DSI

El proceso de Construcción del Sistema de Información (CSI) estará guiado por el siguiente calendario (véase **Figura 105 - Calendario CSI**):

Id	Nombre de tarea	Comienzo	Fin
1	Identificación de riesgos de proyectos de software en base a taxonomías	sá 30/04/05	ma 12/07/05
2	Desarrollo de Sistemas de Información	sá 30/04/05	ma 12/07/05
3	Construcción del Sistema de Información (CSI)	sá 30/04/05	ma 12/07/05
4	Preparación del Entorno de Generación y Construcción	sá 30/04/05	mi 04/05/05
5	Generación del Código de los Componentes y Procedimientos	mi 04/05/05	lu 06/06/05
6	Ejecución de las Pruebas Unitarias	lu 06/06/05	vi 10/06/05
7	Ejecución de las Pruebas de Integración	vi 10/06/05	sá 18/06/05
8	Ejecución de las Pruebas del Sistema	sá 18/06/05	ju 30/06/05
9	Definición de la Formación de Usuarios Finales	ju 30/06/05	lu 04/07/05
10	Construcción de los Componentes y Procedimientos de Migración y Carga	lu 04/07/05	do 10/07/05
11	Aprobación del Sistema de Información	do 10/07/05	ma 12/07/05

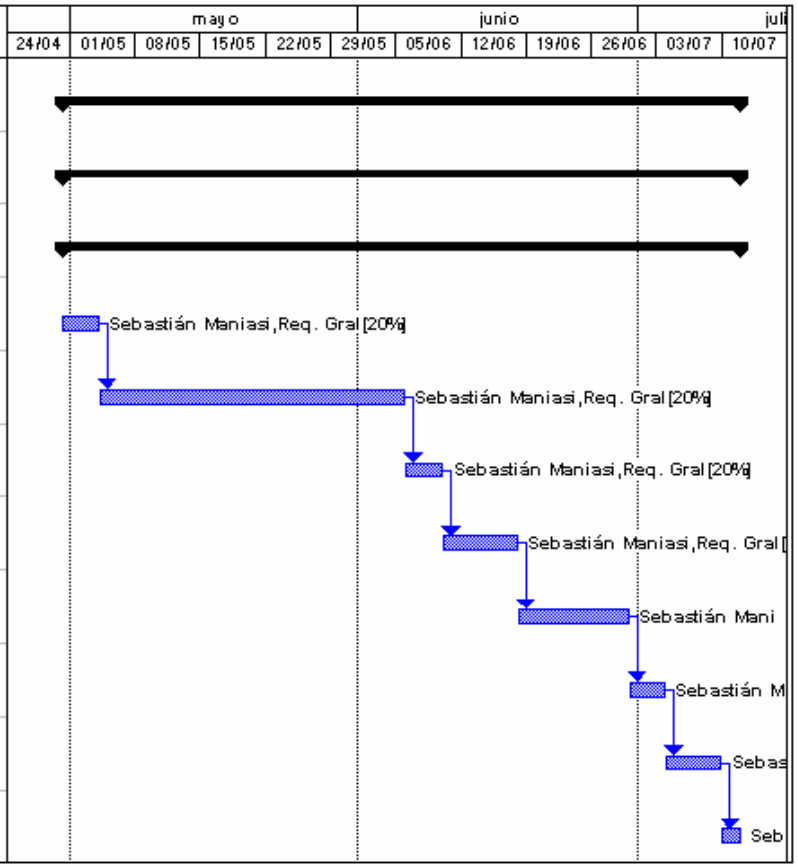


Figura 105 - Calendario CSI

El proceso de Implantación y Aceptación del Sistema (IAS) estará guiado por el siguiente calendario (véase **Figura 106 - Calendario IAS**):

Id	Nombre de tarea	Comienzo	Fin
1	Identificación de riesgos de proyectos de software en base a taxonomías	ma 12/07/05	ma 09/08/05
2	Desarrollo de Sistemas de Información	ma 12/07/05	ma 09/08/05
3	Implantación y Aceptación del Sistema (IAS)	ma 12/07/05	ma 09/08/05
4	Establecimiento del Plan de Implantación	ma 12/07/05	lu 18/07/05
5	Formación Necesaria para la Implantación	lu 18/07/05	ju 21/07/05
6	Incorporación del Sistema al Entorno de Operación	vi 22/07/05	sá 23/07/05
7	Carga de Datos al Entorno de Operación	do 24/07/05	lu 25/07/05
8	Pruebas de Implantación del Sistema	lu 25/07/05	ju 28/07/05
9	Pruebas de Aceptación del Sistema	ju 28/07/05	sá 06/08/05
10	Presentación y Aprobación del Sistema	sá 06/08/05	lu 08/08/05
11	Paso a Producción	lu 08/08/05	ma 09/08/05

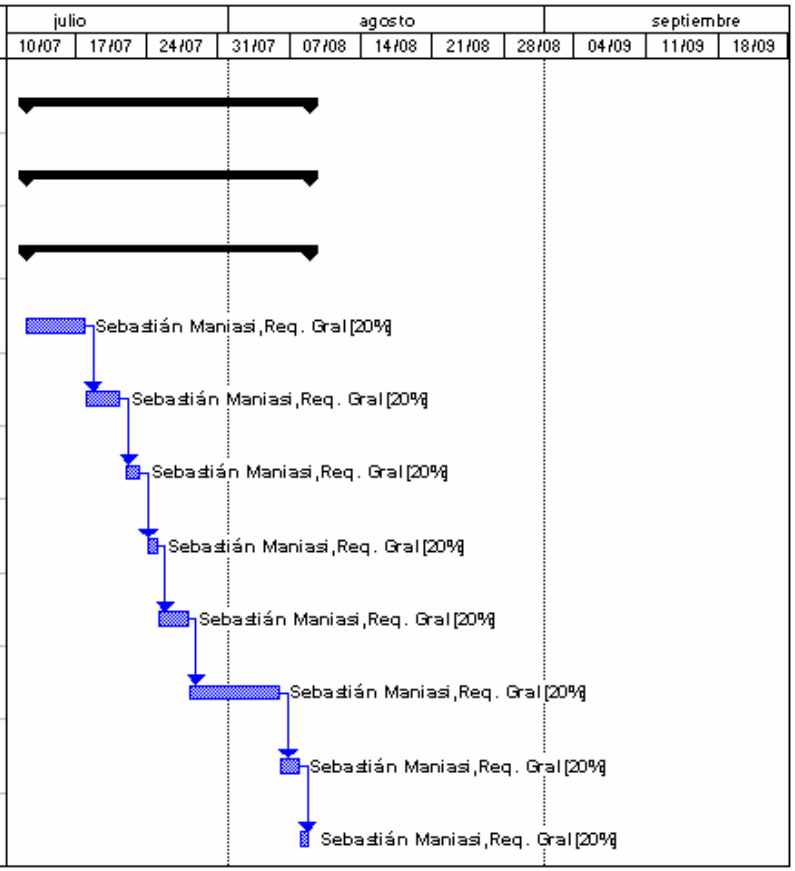


Figura 106 - Calendario IAS

Por simplicidad se han omitido de manera explícita actividades de estimaciones, aseguramiento de calidad, auditorías, retrabajo y aquellas no relacionadas directamente al desarrollo del producto del proyecto como así también se ha incluido como un único recurso (**Req. Gral**) a los participantes de las diferentes áreas de la empresa. Adicionalmente, y con el mismo propósito, se ha omitido en este calendario el detalle de las tareas componentes de cada una de las actividades detallados en el calendario.

Asimismo, en la construcción del calendario, no se ha considerado la posibilidad de contar con recursos humanos extra que colaboren con el desarrollo del proyecto.

Apéndice

C

Lic. Sebastián D. Maniasi

Plan de Gestión de la Configuración

III.Apéndice C: Plan de Gestión de la Configuración

III.1. La Gestión de la Configuración en MÉTRICA Versión 3

En el desarrollo de software los cambios, debidos principalmente a modificaciones de requisitos y fallos, son inevitables. Normalmente se trabaja en equipo por lo que es preciso llevar un control y registro de los cambios con el fin de reducir errores, aumentar la calidad y la productividad y evitar los problemas que puede acarrear una incorrecta sincronización en dichos cambios, al afectar a otros elementos del sistema o a las tareas realizadas por otros miembros del equipo de proyecto.

El objetivo de la **Gestión de la Configuración** es mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas, 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. Así, entre los elementos de configuración software, se encuentran no únicamente ejecutables y código fuente, sino también los modelos de datos, modelos de procesos, especificaciones de requisitos, pruebas, etc.

La gestión de configuración se realiza durante todas las actividades asociadas al desarrollo del sistema, y continua registrando los cambios hasta que éste deja de utilizarse. Además, facilita el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como correctivo. Asimismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que

se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.

La interfaz de gestión de configuración de MÉTRICA Versión 3 permite definir las necesidades de gestión de configuración para cada sistema, recogiéndolas en un plan de gestión de configuración, en el que se especifican las actividades de identificación y registro de productos en el sistema de gestión de configuración durante el desarrollo y posterior mantenimiento de la herramienta.

Si en la organización ya existe un sistema de gestión de configuración estándar, para el sistema en concreto deberán analizarse las necesidades de configuración específicas respecto a dicho sistema estándar y determinar las diferencias, si las hubiera, así como aquellas necesidades concretas que no se encuentren recogidas, estableciendo así el plan de gestión de configuración de la herramienta.

Los productos registrados en el sistema de gestión de la configuración se encuentran identificados y localizados unívocamente, de manera que la información relativa a los productos es de fácil acceso.

La información que puede solicitarse al sistema de gestión de la configuración es variada:

- Información relacionada con Análisis, Diseño, Construcción, Implantación y Aceptación del Sistema, como productos globales que integran todos los productos que lo componen.
- Información de un producto en concreto, su versión, estado, traza de su evolución y cualquier dato que el plan de gestión de la configuración determine de interés (por ejemplo, participantes en la elaboración o modificación del producto).

III.2. Introducción

Este plan describe la forma en la cual se llevará a cabo la Gestión de Configuración para el proyecto de desarrollo de software para la Tesis de Magíster.

El plan de Gestión de la Configuración ha sido desarrollado según [ANSI/IEEE 1042, 1987].

III.2.a. Propósito

Este documento permitirá a los participantes del proyecto entender cómo se llevarán a cabo las tareas de Gestión de Configuración en el mismo.

Este documento indicará fundamentalmente:

- Cómo realizar la Gestión de Configuración en un proyecto.

- Roles y responsabilidades.
- Actividades a realizar.

III.2.b. Alcance

El alcance de este plan se aplica al proyecto de desarrollo para la Tesis de Magíster e incluye ítems de configuración que corresponden tanto a código fuente como a documentación del proyecto extendiéndose a repositorios tanto electrónicos cómo físicos.

III.2.c. Definición de alto nivel del proceso de GCS

La Gestión de Configuración de Software es el proceso de administración que implica identificar y definir Ítems de Configuración en el sistema, controlar los cambios que se produzcan en los mismos a lo largo del ciclo de vida del proyecto, almacenar y reportar el estado de dichos ítems y requerimientos de cambio y verificar la completitud y correctitud de los mismos. Los Ítems de Configuración son salidas del proceso de producción de software.

Las actividades de Gestión de Configuración están presentes durante todas las etapas del ciclo de vida del proyecto y es un componente central del enfoque general de desarrollo.

El Plan de Gestión de Configuración deberá ser revisado al comienzo de cada una de las fases principales de proyecto, modificado en caso de ser necesario y distribuido al equipo de proyecto.

Todas las modificaciones al Plan de Gestión de Configuración deberán ser realizadas por el Gestor de Configuración del Proyecto y una revisión formal se realizará con el propósito de establecer una línea base para el mismo.

III.3. Especificaciones de Gestión

En esta sección se identifican las tareas de coordinación y gestión que serán necesarias para llevar a cabo las actividades de GCS en el proyecto de desarrollo de Tesis de Magíster.

III.3.a. Organización

Las responsabilidades de GCS estarán a cargo de GC asignados para cada proyecto y las actividades de GCS serán llevadas a cabo a través de todo el ciclo de vida del proyecto. El GC será considerado un miembro del equipo de proyecto y deberá cumplir con las responsabilidades definidas para los mismos.

Las actividades de GCS serán auditadas de manera periódica o predefinida por un área independiente de la compañía (GCG). Dicho organismo brindará además soporte al proyecto en las actividades de GCS.

Se formará un CCC para el proyecto que será responsable y tendrá la autoridad para realizar cambios sobre la configuración del proyecto.

El nivel superior de administración de la compañía será responsable por promover y controlar el cumplimiento de las actividades de GCS en el proyecto.

III.3.b. Responsabilidades

El siguiente es un listado de las responsabilidades sobre las tareas de GCS en el proyecto de desarrollo para la Tesis de Magíster:

- GCG:
  - Crear y mantener un Plan de Gestión de Configuración para la organización.
  - Asegurar el cumplimiento a nivel organizacional del estándar de gestión de configuración.
  - Proveer guías organizacionales para todas las actividades de Gestión de Configuración.
  - Auditar los proyectos.
- GC:
  - Coordinar las actividades GCS dentro del proyecto.
  - Desarrollar, documentar y distribuir los artefactos de GCS relacionados con el proyecto.
  - Asegurar que los cambios sobre la línea base se registran con suficiente detalle.
  - Realizar las actividades de GCS según las políticas y procedimientos generales.
  - Asegurar que no se realicen cambios no autorizados sobre la línea base.
  - Operación de la herramienta de GCS.
- Equipo de Proyecto:
  - Cada miembro del equipo debe cumplir con los procesos y actividades de GCS que se encuentran especificadas en el presente documento.
- CCC:
  - Aprobar, monitorear y controlar requerimientos de cambios.
  - Establecer líneas base para los IC.

III.3.c. Implementación del plan de Gestión de Configuración

Todos los IC de un proyecto (documentos y código fuente) que se encuentren bajo control de configuración serán almacenados en un repositorio electrónico de proyecto cuyo acceso será limitado a aquellos participantes autorizados del

proyecto. En caso de requerirse copias físicas de algún documento, dicha copia será numerada de manera unívoca; cualquier cambio sobre el documento requiere de la obtención de la versión controlada anterior y la sustitución de la misma por una nueva.

El CCC debe considerarse como el mecanismo de control central necesario para asegurar que todos los cambios sean considerados, coordinados y documentados de manera apropiada. El CCC incluirá miembros del equipo de desarrollo, aseguramiento de calidad, gestión de configuración y administración del proyecto.

El CCC será responsable por aprobar, rechazar y controlar la correcta implementación de los cambios que se produzcan sobre aquellos documentos que formen parte de la línea base del proyecto. El CCC será responsable también para resolver situaciones relacionadas con el cambio de alcance del proyecto.

Debe utilizarse un procedimiento de cambio una vez que un IC ha pasado satisfactoriamente sus revisiones y el mismo forma parte de la línea base o ante cambios que afecten el calendario, el costo o el tamaño del proyecto.

El siguiente cuadro (véase **Tabla 29 - Formación del CCC**) muestra la formación del CCC para el proyecto de desarrollo de Tesis de Magíster:

NOMBRE	ROL	RESPONSABILIDAD
Sebastián Maniasi	Responsable de CCC (*) Líder de Proyecto (*)	▪ Obtener aprobaciones de la gerencia ante cambios aprobados por el CCC. ▪ Realizar reasignaciones de trabajo. ▪ Agendar, controlar y reportar los cambios. ▪ Fijar la agenda y calendario de las reuniones. ▪ Analizar, asignar y controlar la resolución de problemas y cambios. ▪ Realizar las minutas de reunión.
Patricio M.	Especialista en Calidad (*)	
Ramiro P.B.	Gerente de Proyecto	

Tabla 29 - Formación del CCC

Los roles marcados con (*) deberán tener asistencia obligatoria a las reuniones del CCC. Los demás participantes serán invitados según necesidades de proyecto.

Las reuniones tendrán lugar según lo requiera el Líder de Proyecto, el GC o el Especialista en Calidad.

Una línea base se utiliza como una marca en un punto del desarrollo de un proyecto y estará conformada por un conjunto de IC que han superado exitosamente sus revisiones.

El procedimiento para incorporar un documento a la línea base del proyecto comprende la realización de los siguientes pasos:

1. Cuando un documento se encuentra en condiciones de ser revisados, el autor del mismo debe aplicar al mismo una etiqueta con el siguiente formato: “Versión.Mayor.Menor_Borrador_Identificador”; tal es el caso, por ejemplo de: “2.1.0_Borrador_A”.
2. Llevar a cabo la reunión de revisión.
3. Luego de realizar la revisión el documento es corregido y se le asigna una nueva etiqueta de línea base según el siguiente formato: “Documento_Versión.Mayor.Menor”; por ejemplo: “PlanProyecto_2.1.0”.

Las auditorías de configuración se llevarán a cabo en el proyecto a solicitud del Especialista en Calidad con el propósito de garantizar el cumplimiento de la metodología de GCS.

ClearCase será la herramienta de Control de Configuraciones que se empleará para llevar a cabo el presente proyecto, el siguiente cuadro (véase **Tabla 30 - Información de CC**) lista información relacionada con el VOB de ClearCase y el servidor del mismo:

ADMINISTRADOR	Jerónimo R.
VERSIÓN	ClearCase 4.1
VERSIÓN MULTISITE	N/A
VERSIÓN DE SISTEMA OPERATIVO	Solaris 8
NOMBRE DEL SERVIDOR DEL VOB	xxx01
IP DEL SERVIDOR DEL VOB	xxx.yyy.www.zzz
NOMBRE DEL SERVIDOR DE VISTAS	xxx01
IP DEL SERVIDOR DEL VISTAS	xxx.yyy.www.zzz
GRUPO DE RED	xxx.yyy.www.zzz
TAGNAME DEL VOB DE PROYECTO	\vobs\TCM_Tax_Riesgos
PATH DE ALMACENAMIENTO DEL VOB DE PROYECTO	\\xxx01\vob\TCM_Tax_Riesgos\riesgos.vbs

Tabla 30 - Información de CC

Los siguientes cuadros muestran los permisos a aplicar sobre el VOB (véase **Tabla 31 - Permisos en el VOB**) y la estructura de grupos prevista (véase **Tabla 32 - Estructura de Grupos en el VOB**):

DUEÑO	GRUPO	OTROS
Lectura	Lectura	Lectura
Escritura	Escritura	-
Ejecución	Ejecución	Ejecución

Tabla 31 - Permisos en el VOB

DUEÑO DEL VOB	Motorola Argentina SA
GRUPO DEL VOB	Equipo de Proyecto
GRUPOS ADICIONALES	N/A
OTROS	Personal de Motorola Argentina SA

Tabla 32 - Estructura de Grupos en el VOB

Toda la documentación física del proyecto será almacenada en la Biblioteca de Proyecto y al finalizar el proyecto será agregada a la Biblioteca Organizacional.

III.3.d. Políticas, Directivas y Procedimientos Aplicables

A nivel organizacional, existe un conjunto de directivas y procedimientos que serán considerados para la realización de las actividades de GCS, se destacan aquellas empleadas: para la escritura de documentación, para la gestión de la herramienta de configuración y para las actividades de GCS.

III.4. Actividades de Gestión de Configuración

En esta sección se planifica cada una de las actividades de GCS que se realizarán en el proyecto de desarrollo de la Tesis de Magíster.

III.4.a. Identificación de la Configuración

La identificación de la configuración permitirá definir e identificar todos los IC que formarán parte del proyecto.

Todos los IC serán almacenados en el VOB del proyecto. La identificación de los IC de documentación, misceláneas, código fuente, especiales y fases del desarrollo será realizada según lo especificado en el presente documento.

Se adjuntarán etiquetas a los IC para facilitar su identificación.

El nombre de los documentos deberá seguir la siguiente convención: `<ID_Documento>.<Extensión>`

La siguiente tabla (véase **Tabla 33 - Líneas Base y sus Etiquetas**) enumera las líneas base definidas para el proyecto y su etiqueta correspondiente:

FASE	ETIQUETA DE LÍNEA BASE
Desarrollo de Sistemas de Información – Estudio de Viabilidad del Sistema (EVS)	Fin_EVS
Desarrollo de Sistemas de Información – Análisis del Sistema de Información (ASI)	Fin_ASI
Desarrollo de Sistemas de Información – Diseño del Sistema de Información (DSI)	Fin_DSI
Desarrollo de Sistemas de Información – Construcción del Sistema de Información (CSI)	Fin_CSI
Desarrollo de Sistemas de Información – Implantación y Aceptación del Sistema (IAS)	Fin_IAS

Tabla 33 - Líneas Base y sus Etiquetas

El siguiente cuadro (véase **Figura 107 - Estructura del Repositorio**) muestra la estructura del repositorio:

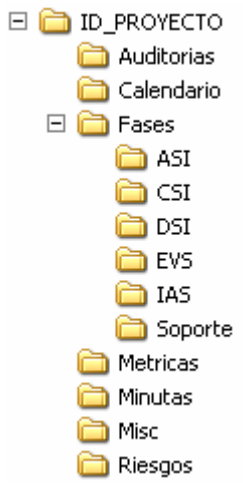


Figura 107 - Estructura del Repositorio

La siguiente tabla (véase **Tabla 34 - IC del Proyecto**) enumera los IC que serán administrados durante el proyecto:

IC (PATH RELATIVO RESPECTO AL VOB TAG)	ID DEL DOCUMENTO
Estudio de Viabilidad del Sistema \Fases\EVS (*Pueden crearse subcarpetas*)	EVS

IC (PATH RELATIVO RESPECTO AL VOB TAG)	ID DEL DOCUMENTO
Análisis del Sistema de Información \Fases\ASI (*Pueden crearse subcarpetas*)	ASI
Diseño del Sistema de Información \Fases\DSI (*Pueden crearse subcarpetas*)	DSI
Construcción del Sistema de Información \Fases\CSI (*Pueden crearse subcarpetas*)	CSI
Implantación y Aceptación del Sistema \Fases\IAS (*Pueden crearse subcarpetas*)	IAS
Documentos de Soporte y Anexos \Fases\Soporte (*Pueden crearse subcarpetas*)	**Sin Convención**
Documento Integrador de Tesis de Magister \Fases	IS 362 - Sebastián Maniasi - Tesis Magíster
Estimaciones de Proyecto \Calendario	IS 362 - Estimaciones
Calendario de Proyecto \Calendario	IS 362 - Calendario
Lista de Especificación y Seguimiento de Riesgos \Riesgos	IS 362 - Riesgos
Libro de Métricas del Proyecto \Métricas	IS 362 - MBOOK
Documentos Auxiliares \Misc	**Sin Convención**
Minutas \Minutas	Minuta_*aaaammdd_nn* Donde: *aaaammdd* año, mes y día. *nn* número consecutivo.
Auditorías \Auditorías	Auditoria_*aaaammdd_nn* Donde: *aaaammdd* año, mes y día. *nn* número consecutivo.

Tabla 34 - IC del Proyecto

Todas las etiquetas usadas en el proyecto seguirán las siguientes convenciones (véase **Tabla 35 - Convención para Etiquetas**):

ETIQUETA	SIGNIFICADO
Ma.Mi_Borrado_Identificador	Versión lista para revisión.

ETIQUETA	SIGNIFICADO
	Versión.Mayor.Menor_Borrador_Identificador Ejemplo: “2.1.0_Borrador_A”.
<ID_Documento>.Ma.Mi	Línea base para el documento luego de su revisión. Ejemplo: “EVS_2.1.0”.
Despliegue_aaaammdd_nn	Identificador de un despliegue realizado. Ejemplo: “Despliegue_20030614_01”.
Fin_EVS Fin_ASI Fin_DSI Fin_CSI Fin_IAS	Etiqueta de fin de fase para marcar una línea base.

Tabla 35 - Convención para Etiquetas

Para conocer el procedimiento para incorporar un documento a la línea base del proyecto referirse a la sección **III.3.c**.**Implementación del plan de Gestión de Configuración**.

El fin de cada fase deberá marcarse en el VOB con la etiqueta correspondiente y dicha actividad permitirá recuperar completamente el contenido del repositorio en un momento dado al final de cada fase, por lo cual dicha etiqueta deberá aplicarse al VOB completo (todos los IC serán incluidos). Antes de la aplicación de esta etiqueta el GC debe realizar las siguientes acciones:

- Verificar que todas las vistas privadas estén vacías.
- Verificar que todos los IC estén checked in.

La realización de un despliegue deberá marcarse en el VOB con la etiqueta correspondiente y dicha actividad permitirá recuperar completamente los elementos empleados para realizar un despliegue, por lo cual dicha etiqueta deberá aplicarse a los elementos del código fuente e implementación del VOB. Antes de la aplicación de esta etiqueta el GC debe realizar las siguientes acciones:

- Verificar que todas las vistas privadas estén vacías.
- Verificar que todos los IC estén checked in.

La rama principal será usada para almacenar las versiones de línea base de todos los artefactos de proyecto. No se realizará ninguna integración sobre esta rama principal. Se crearán diferentes ramas para cada desarrollador; las ramas de cada uno de los desarrolladores será nombrada según la siguiente convención: `<ID_Desarrollador>`. El merge necesario para colocar disponible el código generado por cada uno de los desarrolladores se realizará desde la rama del desarrollador hacia la rama principal empleando la herramienta de merge automática provista por ClearCase. Debido al hecho de que cada implementación para cada desarrollador será independiente no es necesario definir un proceso de merge entre las ramas.

Cada miembro del equipo será responsable de crear y mantener su propia vista privada. El tag de la vista deberá tener el siguiente formato: `<ID_Desarrollador>`. Las vistas creadas deberán usar cualquiera de los "config spec" creados por el GC.

Todos los IC del proyecto deberán tener asignados los permisos correspondientes. Los comandos `rmelem` y `rmver` deberán ser empleados para eliminar CI y serán de uso exclusivo del GC y/o Líder de Proyecto; los demás integrantes del equipo de proyecto podrán emplear el comando `rmname` para eliminar un CI obsoleto.

Todos los checkins deberán ser acompañados por una descripción corta que refiera el propósito del mismo. En caso de que el cambio se encuentre asociado a un cambio el mismo debe ser referido.

Todos los checkouts deben restringirse a artefactos disponibles según la vista del miembro del equipo. Todos los checkouts deben realizarse bajo el esquema de reservas y con un checkin mínimo y obligatorio de frecuencia semanal.

Los siguientes triggers (véase **Tabla 36 - Triggers del VOB**) serán aplicados al VOB del proyecto:

ID DEL TRIGGER	OPERACIÓN	DESCRIPCIÓN
RestricEmptyComment	Checkin	Restringe el checkin sin comentarios.
RestrictRemoveElement	Rmelem	Restringe la eliminación de artefactos.
RestrictCOFromMain	checkout	Restringe el checkout sin comentarios.

Tabla 36 - Triggers del VOB

Adicionalmente, no existen requerimientos para la aplicación de un esquema multisite en el presente proyecto.

Finalmente, no existen requerimientos para la aplicación de un esquema de control de interfaces en el presente proyecto.

III.4.b. Control de la Configuración

El control de configuración del proyecto será regido por lo indicado en el proceso organizacional sin realizarse ningún tipo de desviación de dicho procedimiento, además, las actividades de Control de Configuración se llevarán a cabo según lo especificado en la guía organizacional.

III.4.c. Contabilidad del estado de la configuración

El “Configuration Status Accounting Report” será producido de manera regular por el GC y será generado de manera automática a través de la herramienta CSAR de ClearCase.

El “Configuration Status Accounting Report” deberá realizarse antes de la realización de auditorías y al finalizar cada fase de proyecto.

El siguiente cuadro (véase **Tabla 37 - Actividades de Reporte**) lista las actividades mínimas de reporte requeridas:

TAREA	FRECUENCIA
Configuration Status Accounting Report	Según lo indicado en la presente sección
Reporte de Implementación	Después de cada implementación
AFI	Al final del proyecto
AFU	Antes de cada implementación
Notas de Implementación	Antes de cada implementación
Control de Implementación	Antes de cada implementación

Tabla 37 - Actividades de Reporte

III.4.d. Auditoría de la Configuración

Referirse a la sección **III.3.c**.**Implementación del plan de Gestión de Configuración** para detalles de tipo y características.

Referirse a la sección **III.4.a**.**Identificación de la Configuración** para detalles sobre el almacenamiento de las mismas.

III.5. Control de suministradores

No aplicable debido a que en el proyecto no se realiza subcontratación.

III.6. Recogida y retención de registros

El backup del VOB será responsabilidad del área de “Redes y Tecnología” de la compañía y deberá ser realizado siguiendo lo indicado el plan organizacional.

Una cinta conteniendo un backup completo de los artefactos electrónicos del proyecto deberá almacenarse en el repositorio físico una vez finalizado el proyecto. El período de retención de archivo será de cinco años.

Apéndice

D

Lic. Sebastián D. Maniasi

Plan de Aseguramiento de la Calidad

IV. Apéndice D: Plan de Aseguramiento de la Calidad

IV.1. El Aseguramiento de la Calidad en MÉTRICA Versión 3

El objetivo de la interfaz de **Aseguramiento de la Calidad** de MÉTRICA Versión 3 es proporcionar un marco común de referencia para la definición y puesta en marcha de planes específicos de aseguramiento de calidad aplicables a proyectos concretos. Si en la organización ya existe un sistema de calidad, dichos planes deberán ser coherentes con el mismo, completándolo en los aspectos no contemplados relativos a normas particulares del cliente, usuario o sistema concreto.

La calidad se define como el grado en que un conjunto de características inherentes cumple con unos requisitos. El Aseguramiento de la Calidad pretende dar confianza en que el producto reúne las características necesarias para satisfacer todos los requisitos de la herramienta.

Por tanto, para asegurar la calidad de los productos resultantes el equipo de calidad deberá realizar un conjunto de actividades que servirán para:

- Reducir, eliminar y lo más importante, prevenir las deficiencias de calidad de los productos a obtener.
- Alcanzar una razonable confianza en que las prestaciones y servicios esperados por el cliente o el usuario queden satisfechas.

Para conseguir estos objetivos, es necesario desarrollar un plan de aseguramiento de calidad específico que se aplicará durante la planificación del proyecto de acuerdo a la estrategia de desarrollo adoptada en la gestión del proyecto. En el plan de aseguramiento de calidad se reflejan las

actividades de calidad a realizar (normales o extraordinarias), los estándares a aplicar, los productos a revisar, los procedimientos a seguir en la obtención de los distintos productos durante el desarrollo en MÉTRICA Versión 3 y la normativa para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

El grupo de aseguramiento de calidad participa en la revisión de los productos seleccionados para determinar si son conformes o no a los procedimientos, normas o criterios especificados, siendo totalmente independiente del equipo de desarrollo. Las actividades a realizar por el grupo de aseguramiento de calidad vienen gobernadas por el plan. Sus funciones están dirigidas a:

- Identificar las posibles desviaciones en los estándares aplicados, así como en los requisitos y procedimientos especificados.
- Comprobar que se han llevado a cabo las medidas preventivas o correctoras necesarias.

Las revisiones son una de las actividades más importantes del aseguramiento de la calidad, debido a que permiten eliminar defectos lo más pronto posible, cuando son menos costosos de corregir. Además existen procedimientos extraordinarios, como las auditorías, aplicables en desarrollos singulares y en el transcurso de las cuales se revisarán tanto las actividades de desarrollo como las propias de aseguramiento de calidad. La detección anticipada de errores evita el que se propaguen a los restantes procesos de desarrollo, reduciendo substancialmente el esfuerzo invertido en los mismos. En este sentido es importante destacar que el establecimiento del plan de aseguramiento de calidad comienza en el Estudio de Viabilidad del Sistema y se aplica a lo largo de todo el desarrollo, en los procesos de Análisis, Diseño, Construcción, Implantación y Aceptación del Sistema y en su posterior Mantenimiento.

IV.2. Introducción

Este plan describe la forma en la cual se llevará a cabo el Aseguramiento de la Calidad para el proyecto de desarrollo de software para la Tesis de Magíster.

El plan de Aseguramiento de la Calidad ha sido desarrollado según [IEEE 730, 2002].

IV.2.a. Objetivos de Calidad

Los siguientes objetivos de calidad han sido seleccionados para el proyecto:

1. Ningún defecto de severidad 3 (funcionalidad menor deshabilitada) o severidad 4 (problemas cosméticos) detectado durante las pruebas de aceptación final. Este objetivo será soportado por la recolección de métricas relacionadas con los resultados obtenidos en las pruebas de aceptación final.
2. Resultado global de la encuesta de satisfacción de clientes superior o igual a 8.5. Este objetivo será soportado por la recolección de métricas

relacionadas con los resultados obtenidos en el cuestionario de satisfacción de clientes.

IV.2.b. Métricas de Proyecto

El Desarrollador de Tesis será responsable por asegurar que existen los medios adecuados para recolectar y documentar métricas como así también será el responsable por la recolección efectiva de las mismas.

El Especialista en Calidad será responsable por procesar, analizar y proveer de información de soporte respecto a las métricas colectadas tanto al Desarrollador de Tesis como al Gerente de Proyecto de forma tal de posibilitar el actuar en consecuencia respecto a los resultados obtenidos.

Métricas a ser Colectadas y Reportadas

El siguiente cuadro muestra el listado de las métricas a ser colectadas y reportadas por el proyecto (véase **Tabla 38 - Métricas de Proyecto**):

MÉTRICA	COLECTADA?	UBICACIÓN DE RECOLECCIÓN	COMENTARIOS
Satisfacción del cliente	Si	VOB de Calidad y Procesos	Resultados de la encuesta realizada luego de la entrega del producto.
Datos de calidad	Si	Libro de Métricas	
Datos de calidad de pruebas	Si	Libro de Métricas	
Defectos posteriores a la entrega (Delta)	No	Libro de Métricas	Los defectos producidos durante el desarrollo serán controlados durante la fase de mantenimiento que será administrada como un proyecto independiente.
Defectos posteriores a la entrega (Total)	Si	Libro de Métricas	
Defectos detectados por el cliente (Delta)	No	Libro de Métricas	Los defectos producidos durante el desarrollo serán controlados durante la fase de mantenimiento que será administrada como un proyecto independiente.
Defectos detectados por el cliente (Total)	Si	Libro de Métricas	

MÉTRICA	COLECTADA?	UBICACIÓN DE RECOLECCIÓN	COMENTARIOS
Total de errores y defectos conocidos en el código al momento de entregar el producto	No	Libro de Métricas	El producto debe ser entregado libre de errores conocidos.
Tiempo calendario y productividad para mejoras del producto	No	Libro de Métricas	El proyecto implica el desarrollo de un producto nuevo.
Tiempo calendario y productividad para un producto nuevo	Si	Libro de Métricas	
Adecuación de la estimación de esfuerzo	Si	Libro de Métricas	
Adecuación de la estimación de calendario	Si	Libro de Métricas	
Adecuación de la estimación de tamaño del producto	No	Libro de Métricas	
Tamaño del producto del proyecto (Total)	Si	Libro de Métricas	
Tamaño del producto del proyecto (Delta)	No	Libro de Métricas	
Productividad	Si	Libro de Métricas	
Cantidad de código reutilizado	No	Libro de Métricas	
Cantidad de requerimientos de cambio	No	Libro de Métricas	No se implementarán requerimientos de cambio para el prototipo.
Cantidad de personas integrantes del equipo de proyecto	Si	Libro de Métricas	
Inspecciones y datos de revisiones	Si	Libro de Métricas	
Costos	No	Libro de Métricas	Los costos son controlados por el área de finanzas.
Gestión de riesgos	Si	Plan de Riesgos	
Cantidad de replanificaciones	Si	Libro de Métricas	
Recursos críticos de computación	Si	Libro de Métricas	

Tabla 38 - Métricas de Proyecto

IV.2.c. Auditorías

El departamento de Calidad y Procesos conducirá evaluaciones independientes de los productos de software y los procesos realizados de forma tal de determinar y asegurar el nivel de efectividad y cumplimiento respecto a los planes y procesos. El Gerente de Calidad y Procesos deberá designar un auditor independiente responsable por las auditorías al proyecto.

Se desarrollarán los siguientes tipos de auditorías:

- Auditorías Funcionales.
- Auditorías de Configuración.

El calendario de auditorías es mantenido de forma separada e independiente por el Gerente de Calidad y Procesos.

El Desarrollador de Tesis será responsable por asegurar que se aplican respuestas de manera adecuada y a tiempo a las acciones correctivas surgidas durante las auditorías.

Apéndice

E

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Plan de Riesgos

V. Apéndice E: Plan de Riesgos

V.1. La Seguridad en MÉTRICA Versión 3

El objetivo de la interfaz de **Seguridad** de MÉTRICA Versión 3 es incorporar en los sistemas diferentes mecanismos de seguridad que son adicionales a los que se proponen en la propia metodología, asegurando el desarrollo de cualquier tipo de sistema a lo largo de los procesos que se realicen para su obtención.

La seguridad de la aplicación ya se considera en MÉTRICA Versión 3 como requisito funcional, es decir previamente al desarrollo del mismo. La interfaz de Seguridad hace posible incorporar durante la fase de desarrollo las funciones y mecanismos que refuerzan la seguridad del nuevo sistema y del propio proceso de desarrollo, asegurando su consistencia y seguridad.

El *Análisis de los Riesgos* constituye una pieza fundamental en el diseño y desarrollo de sistemas seguros. Si bien los riesgos que afectan a una aplicación son de distinta índole: naturales (inundaciones, incendios, etc.) o lógicos (fallos propios, ataques externos, virus, etc.) son estos últimos los contemplados en la interfaz de Seguridad de MÉTRICA Versión 3.

De lo anterior se desprende que existen dentro de la interfaz dos tipos de actividades diferenciadas:

- Actividades relacionadas con la seguridad intrínseca de la herramienta. Estas actividades no se consideran como parte del alcance del presente Apéndice.

- Actividades que velan por la seguridad del propio proceso de desarrollo de la herramienta. Estas actividades se consideran como parte del alcance del presente Apéndice.

Si en la organización ya existe un plan de seguridad o una metodología de análisis y gestión de riesgos para cada sistema deberán analizarse las necesidades de seguridad del sistema respecto al método vigente, y determinar las diferencias si las hubiera, así como aquellas necesidades concretas que no se encuentren recogidas, estableciendo así el plan de seguridad de la aplicación. Si no existe un plan de seguridad en la organización habrá que desarrollarlo desde el principio. El plan recogerá además las medidas de seguridad activas o preventivas y reactivas, en respuesta a situaciones en que se produce un fallo reduciendo su efecto, relacionadas con la seguridad de la herramienta de software y del proceso de desarrollo.

Las valoraciones sobre la seguridad deben ser realizadas en función de las características del sistema: complejidad, tamaño, incertidumbre, participantes, etc., por los responsables de la seguridad del sistema, quienes se apoyarán para sus decisiones en su conocimiento y experiencia en la materia sin perder de vista además que, al ser finitos los recursos con los que se cuentan, no pueden asegurarse todos los aspectos del desarrollo de los sistemas, por lo que habrá que aceptar un determinado nivel de riesgo concentrándose en los aspectos más comprometidos o amenazados, que serán diferentes según las circunstancias.

V.2. Introducción

Este plan describe la forma en la cual se llevará a cabo la Gestión de Riesgos para el proyecto de desarrollo de software para la Tesis de Magíster.

El plan de Gestión de Riesgos ha sido desarrollado según [IEEE 1540, 2001].

V.2.a. Propósito y Alcance

Este plan detalla el proceso a ser seguido para identificar, valorizar, controlar y monitorizar eventos de riesgo a lo largo de todo el ciclo de vida del proyecto.

V.3. Roles y Responsabilidades

La siguiente tabla lista los roles y responsabilidades asociados a la Gestión de Riesgos en el proyecto (véase **Tabla 39 - Roles y Responsabilidades en la Gestión de Riesgos**):

ROL	RESPONSABILIDAD
Gerente de Proyecto	Control de la ejecución del plan de riesgos y revisión de estado durante las reuniones periódicas de control de avance.
Oficial de Riesgos	Posee las siguientes responsabilidades y autoridad: coordinación de las actividades de identificación y análisis de riesgos, mantenimiento de la lista de riesgos, notificación a la gerencia de los nuevos riesgos identificados y reporte del estado de resolución de un riesgo a la gerencia. El Desarrollador de Tesis será el Oficial de Riesgos en el presente proyecto.
Miembro del equipo de proyecto responsable por un riesgo	El Oficial de Riesgos asignará cada nuevo riesgo identificado a un miembro del equipo de proyecto, quién valorizará la exposición del riesgo y reportará los resultados de su análisis al Oficial de Riesgos. Cada miembro del equipo de proyecto con un riesgo asignado será responsable además por la implementación de los planes de mitigación y el reporte de sus resultados.

Tabla 39 - Roles y Responsabilidades en la Gestión de Riesgos

V.4. Documentación de Riesgos

La siguiente tabla lista la documentación asociada a la Gestión de Riesgos en el proyecto (véase **Tabla 40 - Documentación de Riesgos**):

ELEMENTO	COMENTARIO
Lista de Riesgos	Contiene los factores de riesgos identificados y manejados para el proyecto.
Elementos de Datos de Riesgos	Se almacenará la siguiente información para cada uno de los riesgos del proyecto: identificador, fecha de identificación, descripción, impacto, probabilidad (0-1), impacto (1-100), exposición, plan de mitigación y contingencia, responsable, fecha de cierre planificado y estado.

Tabla 40 - Documentación de Riesgos

V.5. Actividades de Gestión de Riesgos

La siguiente tabla lista las actividades asociadas a la Gestión de Riesgos en el proyecto como así también los participantes de cada una de ellas (véase **Tabla 41 - Actividades de Gestión de Riesgos**):

ACTIVIDAD	PARTICIPANTE (S)
Identificación de Riesgos	Miembros del equipo de proyecto
Análisis de Riesgos	Miembro del equipo de proyecto responsable por un riesgo Oficial de Riesgos
Planificación de Riesgos	Miembros del equipo de proyecto Oficial de Riesgos
Seguimiento de Riesgos	Gerente de Proyecto Oficial de Riesgos
Control de Riesgos	Miembro del equipo de proyecto responsable por un riesgo Oficial de Riesgos

Tabla 41 - Actividades de Gestión de Riesgos

V.6. Herramientas de Gestión de Riesgos

Las siguientes herramientas serán empleadas para realizar la gestión de riesgos:

- Lista de Riesgos.
- Taxonomías de Riesgos.

V.7. Criterio de Cierre

Un elemento de riesgo puede ser considerado como "**cerrado**" cuando el evento iniciador del mismo no tiene posibilidad de ocurrencia o cuando su impacto ha dejado de ser representativo.

V.8. Calendario para las Actividades de Gestión de Riesgos

La siguiente tabla lista el calendario de las actividades asociadas a la Gestión de Riesgos en el proyecto (véase **Tabla 42 - Actividades de Gestión de Riesgos**):

ACTIVIDAD	EVENTO
Identificación de Riesgos	Final de la fase de Estudio de Viabilidad del Sistema
Análisis de Riesgos	
Planificación de Riesgos	
Seguimiento de Riesgos	Todas las fases del proyecto
Control de Riesgos	

Tabla 42 - Actividades de Gestión de Riesgos

Apéndice

F

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Taxonomía de Riesgos Ajustada

VI. Apéndice F: Taxonomía de Riesgos Ajustada

Se presenta a continuación la Taxonomía de Riesgos Ajustada desarrollada para la Tesis de Magíster.

VI.1. Tipos de Proyectos

La siguiente tabla (véase **Tabla 43 - Tipos de Proyecto**) resume los diferentes tipos de proyecto identificados para la Taxonomía de Riesgos Ajustada de la Tesis de Magíster.

Tipos de Proyectos de Desarrollo de Software		
ID	***Nombre***	***Ejemplos***
SAI	Sistemas de Administración de Información	Sistemas Contables, Sistemas de Seguimiento de Trámites
SSB	Software de Sistemas Base	Sistemas Operativos, Sistema de Control de Dispositivos
SCM	Software Comercial de Mercado	Planillas de Cálculo, Procesadores de Texto, Sistemas de Administración Enpaquetados
SE	Sistemas Especiales	Sistemas Militares, Sistemas Científicos
RS	Reingeniería de Sistemas	

Tabla 43 - Tipos de Proyecto

VI.2. Taxonomía de Riesgos Ajustada

La siguiente tabla (véase **Tabla 44 - Taxonomía de Riesgos Ajustada**) constituye la Taxonomía de Riesgos Ajustada de la Tesis de Magíster.

Taxonomía de Riegos para Proyectos de Desarrollo de Software

ID	***Factor***	***Indicadores de Bajo Riesgo***	***Indicadores de Mediano Riesgo***	***Indicadores de Alto Riesgo***	***Criterio para su Aplicabilidad***	***Probabilidad por Tipo de Proyecto***				
						SAI	*SSB*	*SCM*	*SE*	*RS*

Mision y Objetivos										
1	El proyecto se adecua a la organización proveedora	El proyecto soporta directamente la misión y objetivos de la organización proveedora	El proyecto impacta indirectamente en uno o mas de los objetivos definidos por la organización proveedora	El proyecto no se relaciona con la misión y objetivos de la organización proveedora		Media	Baja	Baja	Baja	Alta
2	Flujo de trabajo	Se preven cambios pequeños o nulos en el flujo de trabajo	Se preven cambios importantes en el flujo de trabajo	Se preven cambios significativos en el flujo de trabajo		Media	Baja	Media	Baja	Media
3	Percepción del cliente	El cliente esperaba que esta organización sea la proveedora del producto del proyecto	La organización se encuentra trabajando en el proyecto en un área no esperada por el cliente	El proyecto no coincide con productos o servicios previos de esta organización	**Riesgos válidos sólo cuando el proyecto cuenta con un "cliente/usuario" directo. Tipo de pregunta de verificación: (Si/No)**	Media	Baja	Media	Baja	Media
4	El proyecto se adecua a la organización cliente	El proyecto soporta directamente la misión y objetivos de la organización cliente	El proyecto impacta indirectamente en uno o mas de los objetivos definidos por la organización cliente	El proyecto no se relaciona con la misión y objetivos de la organización cliente		Baja	Baja	Media	Baja	Baja

Administración del Programa											
5	Objetivos en conflicto	Los objetivos de los proyectos del programa se soportan y complementan mutuamente	Los objetivos de los proyectos del programa no entran en conflico y proveen un pequeño soporte directo	Los objetivos de los proyectos del programa entran, directa o indirectamente, en conflicto	**Riesgos válidos sólo cuando el proyecto forma parte de un programa. Tipo de pregunta de verificación: (Si/No)**	Media	Baja	Baja	Baja	Media
6	Recursos en conflicto	Los proyectos del programa comparten recursos sin ningún conflicto	Los proyectos del programa calendarizan el uso de los recursos de forma tal de evitar conflictos	Los proyectos del programa frecuentemente necesitan los mismos recursos al mismo momento		Media	Media	Media	Alta	Baja
7	Clientes en conflicto	Los múltiples clientes del programa tienen necesidades en común	Los múltiples clientes del programa tienen necesidades diferentes pero que no entran en conflicto	Los múltiples clientes del programa tienen necesidades diferentes que entran en conflicto		Media	Baja	Media	Baja	Media
8	Liderazgo	El programa cuenta con un administrador de programas activo que coordina los proyectos	El programa cuenta con un responsable pero el mismo no puede dedicar el tiempo suficiente para realizar un liderazgo efectivo	El programa no cuenta con un lider		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.

9	Experiencia del administrador del programa	El administrador del programa tiene una amplia experiencia en este tipo de actividades	El administrador del programa tiene cierta experiencia que puede ser soportada por expertos en la tarea	El administrador del programa no cuenta con experiencia en este tipo de actividades		No Deter.	Baja	No Deter.	Baja	No Deter.
10	Definición del programa	El programa se encuentra bien definido y con un alcance que puede ser manejado por la organización	El programa se encuentra bien definido pero su alcance no puede ser manejado completamente por la organización	El programa no se encuentra bien definido o se ha definido incurriendo en conflictos de intereses respecto al alcance		Media	Baja	Baja	Baja	Media
Conductores para la Toma de Decisiones										
11	Influencias políticas	No se han tomado para el proyecto decisiones motivadas por influencias políticas	Se han tomado para el proyecto algunas decisiones motivadas por influencias políticas	Se han tomado para el proyecto una gran cantidad de decisiones motivadas por influencias políticas		Media	Media	Baja	Media	Baja
12	Estimaciones	Las estimaciones de tamaño, esfuerzo y costos se han realizado y validado en base a un proceso definido y apropiado que involucra el uso de diferentes técnicas	Las estimaciones de tamaño, esfuerzo y costos se han realizado en base a un proceso definido pero no han sido validadas	Las estimaciones de tamaño, esfuerzo y costos han sido realizadas sin seguir un proceso formal y sin una validación final		Alta	Baja	Media	Baja	Alta

13	Datos históricos	Los datos históricos tanto a nivel organizacional como industrial que se emplean para realizar estimaciones y determinar niveles esperados de productividad y calidad son confiables y se encuentran disponibles	Los datos históricos tanto a nivel organizacional como industrial que se emplean para realizar estimaciones y determinar niveles esperados de productividad y calidad no son absolutamente confiables o no se encuentran totalmente disponibles	No se emplean datos históricos para realizar estimaciones y determinar niveles esperados de productividad y calidad	Alta	Media	Baja	Media	Alta
14	Bala de plata	El concepto de "bala de plata" es mayormente rechazado tanto a nivel de proyecto como a nivel organizacional y en escasas ocasiones se apela a este como recurso	El concepto de "bala de plata" es considerado como aceptable tanto a nivel de proyecto como a nivel organizacional y suele apelarse al mismo como recurso	El concepto de "bala de plata" es mayormente aceptado tanto a nivel de proyecto como a nivel organizacional y en repetidas ocasiones se apela a este como recurso	Media	Alta	Media	Baja	Media

15	Fecha conveniente	La fecha de entrega ha sido fijada en base a un proceso adecuado y apropiado para el proyecto	La fecha de entrega ha sido fijada parcialmente en base a la necesidad de alcanzar algún objetivo de negocio no necesariamente relacionado con las estimaciones técnicas	La fecha de entrega ha sido fijada en base a la necesidad de alcanzar algún objetivo de negocio sin considerar las estimaciones técnicas	Alta	Media	Alta	Media	Media
16	Tecnología atractiva	La tecnología seleccionada para el proyecto se encuentra en uso desde hace algún tiempo	Uno de los objetivos esperados del proyecto es el dar soporte a la compra o desarrollo de una nueva tecnología	El proyecto se desarrolla como una herramienta para mostrar una nueva tecnología o como excusa para incorporar una nueva tecnología a la organización	Baja	Alta	Media	Media	Baja
17	Solución a corto plazo	El proyecto satisface las necesidades a corto plazo sin comprometer las posibilidades a largo plazo	El proyecto se enfoca en satisfacer necesidades a corto plazo con algún entendimiento de los requerimientos a largo plazo	El proyecto se enfoca en satisfacer necesidades a corto plazo ignorando completamente los requerimientos a largo plazo	Media	Baja	Baja	Media	Baja

Gerenciamiento Organizacional										
18	Estabilidad organizacional	Cambios pequeños o nulos esperados en la estructura organizacional	Cambios importantes esperados en la estructura organizacional	Cambios significativos esperados en la estructura organizacional		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
19	Roles y responsabilidades organizacionales	Los individuos en la organizacion comprenden sus roles y responsabilidades y la de los demás	Los individuos en la organizacion comprenden sus roles y responsabilidades pero no conocen claramente los roles y responsabilidades fuera de su grupo de trabajo inmediato	Los individuos en la organizacion no comprenden sus roles y responsabilidades ni la de los demás		Media	Baja	Baja	Baja	Media
20	Políticas y estándares	Las políticas y estándares se encuentran cuidadosamente definidos y son seguidos de forma conveniente	Las políticas y estándares se encuentran definidos pero no son seguidos de forma cuidadosa	Las políticas y estándares no se encuentran definidos o no son seguidos		Alta	Baja	Media	Baja	Alta
21	Soporte de la gerencia	La gerencia se encuentra fuertemente comprometida con el éxito del proyecto	La gerencia se encuentra comprometida con el éxito del proyecto pero no totalmente	La gerencia no se encuentra comprometida con el éxito del proyecto		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.

22	Involucramiento ejecutivo	El soporte ejecutivo es visible y fuerte	El soporte ejecutivo es ocasional y ante situaciones especiales	No existe soporte ejecutivo aún en situaciones especiales		Media	Media	Baja	Baja	Media
23	Objetivos de proyecto	Los objetivos de proyecto son verificables para requerimientos razonables	Los objetivos de proyecto han sido definidos pero son dificilmente medibles	Los objetivos de proyecto no han sido definidos o no son medibles		Media	Baja	Baja	Baja	Baja
	Clientes/Usuarios									
24	Involucramiento del usuario	Los usuarios se encuentran altamente involucrados con el equipo de proyecto y proveen un importante soporte	Los usuarios se encuentran desarrollando roles poco significativos con un bajo impacto	Los usuarios tienen un involucramiento nulo o su soporte es inadecuado	**Riesgos válidos sólo cuando el proyecto cuenta con un "cliente/usuario" directo. Tipo de pregunta de verificación: (Si/No)**	Media	No Aplica	No Aplica	Baja	Media
25	Experiencia de los usuarios	Los usuarios cuentan con una amplia experiencia en proyectos similares y aportan ideas respecto a como podría resolverse el problema	Los usuarios cuentan con alguna experiencia en proyectos similares y conocen algunas de sus necesidades	Los usuarios no cuentan con experiencia en proyectos similares y no conocen completamente sus necesidades		Media	No Aplica	No Aplica	Media	Alta

26	Aceptación del usuario	Los usuarios aceptan los conceptos y detalles del sistema y se ha definido un proceso de aprobación por parte de los mismos	Los usuarios aceptan muchos de los conceptos y detalles del sistema y se ha definido un proceso de aprobación por parte de los mismos	Los usuarios no aceptan los conceptos y detalles del sistema o no se ha definido un proceso de aprobación por parte de los mismos		Media	Baja	No Aplica	Baja	Alta
27	Necesidades de entrenamiento de los usuarios	Las necesidades de entrenamiento de los usuarios han sido consideradas y planificadas o realizadas	Las necesidades de entrenamiento de los usuarios han sido consideradas pero no han sido planificadas o realizadas	Las necesidades de entrenamiento de los usuarios no han sido consideradas		Media	Baja	Baja	Baja	Media
28	Justificación a los usuarios	Se ha realizado una completa y adecuada justificación del sistema a los usuarios	Se ha realizado una justificación del sistema a los usuarios pero aún existen dudas respecto a la aplicabilidad del mismo	No se ha realizado una justificación del sistema a los usuarios o la misma resulta inadecuada		Media	Baja	Baja	Baja	Alta
Parametros del Proyecto										
29	Tamaño del proyecto	Pequeño, no complejo o fácilmente divisible	De mediano tamaño, moderada complejidad y divisible	Grande, altamente complejo o no divisible		Baja	Alta	Media	Alta	Media

30	Restricciones de hardware	Escasas o nulas restricciones de hardware o plataforma única	Algunas restricciones de hardware o varias plataformas	Importantes restricciones de hardware o múltiples plataformas		Baja	Alta	Baja	Media	Baja
31	Componentes reusables	Existen componentes disponibles y compatibles con el enfoque a seguir	Existen componentes disponibles pero necesitan cierta revisión	Se han identificado algunos componentes que necesitan importantes modificaciones para su uso		Baja	Alta	Baja	Media	Media
32	Componentes provistos	Existen componentes disponibles y usables directamente	Existen componentes que trabajan en la mayor parte de las circunstancias	Existen componentes que fallan en ciertos casos y pueden ser incompatibles con parte del enfoque seleccionado		Baja	Alta	Baja	Media	Baja
33	Presupuesto	Suficiente presupuesto asignado	Moderado presupuesto asignado	Escaso presupuesto asignado	**Riesgos válidos sólo cuando el proyecto realiza administración de presupuesto.** **Tipo de pregunta de verificación: (Si/No)**	Alta	Baja	Media	Baja	Alta
34	Restricciones presupuestarias	Fondos asignados sin restricciones	Algunas dudas respecto a la disponibilidad de fondos	Asignaciones presupuestarias en duda o sujetas a cambiar sin notificación previa		Alta	Media	Media	Media	Alta
35	Controles de costo	Existe un sistema establecido y disponible	Existe un sistema establecido pero con algunas carencias	No existe un sistema establecido		Media	Baja	Baja	Baja	Media

36	Compromiso de entrega	Fechas comprometidas de entrega estables	Existen algunas fechas comprometidas de entrega inciertas	Fechas de compromiso inestables y fluctuantes	Alta	Baja	Media	Baja	Media
37	Calendario de desarrollo	El equipo acuerda que el calendario es aceptable y puede ser cumplido	El equipo encuentra el calendario de alguna de las fases del plan demasiado agresivo	El equipo encuentra el calendario de algunas de las fases del plan dificilmente alcanzable	No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
Ingeniería del Producto										
Requerimientos										
38	Estabilidad	Muy pocos o ninguno de los requerimientos se modifican durante el desarrollo del sistema	Algunos de los requerimientos se modifican durante el desarrollo del sistema	Muchos de los requerimientos se modifican durante el desarrollo del sistema	Alta	Baja	Media	Baja	Baja
39	Completitud	La mayor parte de los requerimientos del cliente han sido detectados y se encuentran completamente documentados; existen pocas faltas de definiciones	Muchos de los requerimientos del cliente han sido detectados y se encuentran documentados aunque de forma incompleta; existen algunas faltas de definiciones	Muchos de los requerimientos del cliente no han sido detectados o no se encuentran documentados apropiadamente; existen muchas faltas de definiciones	Alta	Baja	Baja	Media	Baja

40	Claridad	Los requerimientos se encuentran claramente especificados y no existen problemas de ambigüedades o entendimientos con el cliente	Los requerimientos se encuentran especificados y existen problemas de ambigüedades y entendimiento que se espera puedan resolverse facilmente	Los requerimientos se encuentran especificados pero existen grandes problemas de ambigüedades y entendimiento con el cliente	Media	Baja	Baja	Baja	Media
41	Validez	Los requerimientos expresan las necesidades de los clientes y han sido validados empleando diferentes técnicas (ejemplo: prototipado)	Los requerimientos expresan la mayor parte de las necesidades de los clientes aunque no han sido validados empleando diferentes técnicas	Los requerimientos expresan algunas de las necesidades de los clientes y no han sido validados empleando diferentes técnicas	Alta	Baja	Baja	Baja	Media
42	Viabilidad	La mayor parte de los requerimientos son técnicamente implementables tanto individual como grupalmente pero no se ha realizado un estudio formal al respecto	La mayor parte de los requerimientos son técnicamente implementables individualmente pero se esperan algunos problemas al realizar la implementación conjunta	La mayor parte de los requerimientos no son técnicamente implementables tanto individual como grupalmente	Baja	Media	Baja	Media	Baja

43	Antecedencia	La mayor parte de los requerimientos se relacionan con actividades y tecnologías antes implementadas en la organización	Muchos los requerimientos se relacionan con actividades y tecnologías antes implementadas en industria	Solo algunos de requerimientos se relacionan con actividades y tecnologías antes implementadas en industria, muchos de ellos no tienen precedentes		Baja	Alta	Baja	Alta	Baja
44	Escalabilidad	El tamaño, complejidad y esfuerzo requerido por los requerimientos es comparable al de experiencias previas en la organización	El tamaño, complejidad o esfuerzo requerido por los requerimientos es comparable al de experiencias previas en la industria	El tamaño, complejidad y esfuerzo requerido por los requerimientos no es comparable al de experiencias previas en la industria		Baja	Alta	Baja	Alta	Baja
Diseño										
45	Funcionalidad	Los algoritmos y modelos seleccionados satisfacen los requerimientos funcionales pero no se ha realizado un estudio formal al respecto	Los algoritmos y modelos seleccionados satisfacen muchos de los requerimientos funcionales	Los algoritmos y modelos seleccionados no satisfacen muchos de los requerimientos funcionales		Media	Baja	Baja	Baja	Media

46	Dificultad	Los algoritmos y modelos seleccionados presentan una moderada complejidad y todos los requerimientos tienen una solución asociada	Los algoritmos y modelos seleccionados presentan cierta complejidad y todos los requerimientos tienen una solución asociada	Muchos de los algoritmos y modelos presentan una alta complejidad y no todos los requerimientos tienen una solución asociada	Baja	Alta	Media	Alta	Media	
47	Interfaces	Las interfaces internas (hardware y software) han sido correctamente definidas y se encuentran controladas	Algunas de las interfaces internas (hardware y software) han sido definidas y se encuentran controladas	Las interfaces internas (hardware y software) no han sido definidas o no se encuentran controladas	Media	Media	Baja	Baja	Media	
48	Posibilidad de desarrollar pruebas	El producto admite facilmente la realización de pruebas y el personal que las desarrollará ha sido involucrado en el proceso de diseño	El producto admite la realización de pruebas y el personal que las desarrollará no ha sido involucrado en el proceso de diseño	Las características del producto dificultan la realización de pruebas y el personal que las desarrollará no ha sido involucrado en el proceso de diseño	Baja	Alta	Baja	Alta	Media	

Código y Pruebas Unitarias										
49	Viabilidad	La mayor parte del diseño es técnicamente implementable tanto individual como grupalmente pero no se ha realizado un estudio formal al respecto	La mayor parte del diseño es técnicamente implementable individualmente pero se esperan algunos problemas al realizar la implementación conjunta	La mayor parte del diseño no es técnicamente implementable tanto individual como grupalmente		Baja	Media	Baja	Media	Baja
50	Pruebas unitarias	Las pruebas unitarias han sido estimadas y planificadas pero no se ha definido el nivel de detalle para las mismas	Las pruebas unitarias han sido estimadas y planificadas pero no serán documentadas	Las pruebas unitarias no han sido estimadas ni planificadas		Media	Baja	Media	Baja	Media
51	Codificación e implementación	Existen pocas restricciones a nivel de programación, el entorno de desarrollo coincide con el de despliegue, se desarrollará en un único lenguaje y las especificaciones de diseño cuentan con un preciso nivel de detalle	Existen algunas restricciones a nivel de programación, el entorno de desarrollo es similar al de despliegue, se desarrollará en mas de un lenguaje y las especificaciones de diseño cuentan con un buen nivel de detalle	Existen varias restricciones a nivel de programación, el entorno de desarrollo difiere del de despliegue, se desarrollará en múltiples lenguajes y las especificaciones de diseño tienen un insuficiente nivel de detalle		Baja	Media	Baja	Media	Media

Integración y Pruebas										
52	Entorno	Se cuenta con hardware y software suficientes para realizar la integración, los casos de prueba reflejan escenarios operacionales realistas, el conjunto de datos elegido es apropiado y la actividad ha sido planificada y acordada	Se cuenta con hardware y software para realizar la integración, los casos de prueba reflejan escenarios operacionales, el conjunto de datos elegido es básico y la actividad ha sido acordada pero no planificada	No se cuenta con hardware y software suficientes para realizar la integración o los casos de prueba no reflejan escenarios operacionales realistas o el conjunto de datos elegido no es apropiado o la actividad no ha sido planificada ni acordada	**Riesgos válidos sólo cuando el producto del proyecto será integrado con HW/SW.** **Tipo de pregunta de verificación: (Si/No)**	Media	Alta	Baja	Alta	Media
53	Producto	Las interfaces de hardware y software internas han sido definidas, documentadas, acordadas e implementadas pero no se ha especificado un criterio de aceptación	Las interfaces de hardware y software internas han sido definidas e implementadas pero no acordadas ni documentadas	Las interfaces de hardware y software internas han sido definidas pero no documentadas, acordadas o implementadas		Alta	Baja	Baja	Baja	Media

54	Sistema	Las interfaces de hardware y software externas han sido definidas, documentadas, acordadas e implementadas pero no se ha especificado un criterio de aceptación	Las interfaces de hardware y software externas han sido definidas e implementadas pero no acordadas ni documentadas	Las interfaces de hardware y software externas han sido definidas pero no documentadas, acordadas o implementadas		Media	Baja	Media	Baja	Baja	
Requerimientos de Calidad											
55	Rendimiento	No existen requerimientos de rendimiento o se ha realizado un análisis de rendimiento con un alto nivel de confianza	Existen algunos requerimientos relacionados con el rendimiento hacia usuarios finales	Existen gran cantidad de requerimientos relacionados con el rendimiento y la ejecución en tiempo real	**Riesgos válidos sólo cuando el producto del proyecto tiene requerimientos de calidad. Tipo de pregunta de verificación: (Si/No)**	Media	Alta	Baja	Alta	Media	
56	Confiabilidad	Los requerimientos de confiabilidad y disponibilidad se encuentran expresados en términos absolutos (hardware + software) y el sistema cuenta con interfaces externas confiables	Los requisitos de confiabilidad y disponibilidad se encuentran expresados en términos absolutos (hardware + software) pero la confiabilidad de las interfaces externas no puede ser garantizada	Los requerimientos de confiabilidad y disponibilidad se encuentran expresados en términos relativos de hardware o software y el sistema no cuenta con interfaces externas confiables		Media	Media	Baja	Media	Media	

57	Protección	Los requerimientos relacionados con la protección pueden ser verificados antes de que el sistema sea puesto en operación	Algunos de los requerimientos relacionados con la protección pueden ser verificados antes de que el sistema sea puesto en operación	Los requerimientos relacionados con la protección no pueden ser verificados antes de que el sistema sea puesto en operación		Baja	Media	Baja	Media	Baja
58	Seguridad	Los requerimientos de seguridad se encuentran enmarcados dentro del alcance de las prácticas actuales y no dependen de acreditaciones externas tanto para el hardware como para el software	Algunos de los requerimientos de seguridad no se encuentran enmarcados dentro del alcance de las prácticas actuales y no dependen de acreditaciones externas tanto para el hardware como para el software	Los requerimientos de seguridad no se encuentran enmarcados dentro del alcance de las prácticas actuales o dependen de acreditaciones externas tanto para el hardware como para el software		Media	Baja	Baja	Baja	Media
59	Factores humanos	Los requerimientos de interfaz han sido acordados y documentados mediante el uso de prototipos con los usuarios	Los requerimientos de interfaz han sido acordados y documentados pero no se ha hecho uso de prototipos para asegurar el proceso	Los requerimientos de interfaz no han sido acordados o documentados		Media	Baja	Baja	Baja	Baja

60	Calidad	Los requerimientos de calidad se encuentran claramente documentados y son compatibles con valores alcanzables historicamente en la industria o en la organización	Los requerimientos de calidad se encuentran documentados y son parcialmente compatibles con valores alcanzables historicamente en la industria o en la organización	Los requerimientos de calidad se encuentran documentados pero son dificilmente alcanzables comparados con valores históricos de la industria o la organización		Media	Baja	Baja	Baja	Media
				Instalación						
61	Recursos de hardware	Maduros, con capacidad de crecimiento y flexibles	Disponibles con alguna capacidad de crecimiento	Sin capacidad de crecimiento e inflexibles	**Riesgos válidos sólo cuando el proyecto realiza instalación del producto. Tipo de pregunta de verificación: (Si/No)**	Media	Alta	Baja	Alta	Media
62	Impacto en el servicio al cliente	La instalación del sistema implica cambios menores en el servicio al cliente	La instalación del sistema implica cambios importantes en el servicio al cliente	La instalación del sistema implica cambios mayores en el servicio al cliente		Media	Baja	Baja	Media	Alta
63	Necesidad de migración de datos	Pocos datos a migrar	Importante cantidad de datos a migrar con un buen soporte de documentación	Importante cantidad de datos a migrar con diferentes bases de datos y escasa o nula documentación		Media	Baja	Baja	Baja	Alta

64	Prueba piloto	Participantes del piloto disponibles, capacitados e interesados en participar de las pruebas	Participantes del piloto disponibles pero con importantes necesidades de entrenamiento	Participantes del piloto no disponibles o no interesados en participar de las pruebas		Media	Baja	Baja	Media	Media
65	Interfaces externas de software y hardware	Interfaces escasas o nulas	Alguna integración entre interfaces es requerida	Gran cantidad de interfaces requeridas		Baja	Alta	Baja	Media	Media
Proceso de Desarrollo										
66	Uso de un proceso de ingeniería definido	Efectivo proceso de desarrollo establecido y respetado	Proceso de desarrollo establecido pero no empleado efectivamente	Proceso de desarrollo no establecido		Alta	Media	Media	Baja	Alta
67	Estabilidad del proceso	Proceso de desarrollo definido, probado y con alto nivel de madurez	Proceso de desarrollo definido, probado pero con importantes mejoras requeridas	Nuevo proceso de desarrollo sin experiencias previas de aplicación o proceso no definido		No Deter.	No Deter.	No Deter.	Baja	Alta

68	El proceso de desarrollo se adecua al proyecto	El proceso de desarrollo se adecua a las necesidades de proyecto y esta soportado por los métodos y herramientas seleccionados	El proceso de desarrollo se adecua a las necesidades de proyecto pero requiere de ciertas adaptaciones y esta soportado parcialmente por los métodos y herramientas seleccionados	El proceso de desarrollo no se adecua a las necesidades de proyecto o no esta soportado por los métodos y herramientas seleccionados	Media	Baja	Media	Baja	Alta
69	Análisis de alternativas	Análisis de alternativas completo con todas las posibilidades consideradas y suposiciones verificables	Análisis de alternativas completo con algunas posibilidades no consideradas y algunas suposiciones cuestionables	Análisis de alternativas no completo, posibilidades no consideradas y suposiciones cuestionables	Media	Baja	Baja	Baja	Media
70	Compromiso con el proceso	Los cambios a los compromisos en cuanto a alcance, contenidos y calendario son revisados y aprobados por todos los involucrados	Los cambios a los compromisos en cuanto a alcance, contenidos y calendario son comunicados a todos los involucrados	Los cambios a los compromisos en cuanto a alcance, contenidos y calendario son realizados sin participar a los involucrados	No Deter.	No Deter.	No Deter.	Baja	No Deter.

71	Enfoque de aseguramiento de la calidad	Efectivo sistema de aseguramiento de la calidad establecido y respetado	Procesos para el aseguramiento de la calidad establecidos pero no empleados efectivamente	Sistema de aseguramiento de la calidad o procesos no establecidos	No Deter.	No Deter.	No Deter.	Baja	No Deter.
72	Documentación de Desarrollo	Correcta y disponible	Disponible con algunas deficiencias	Inexistente	Media	Baja	Baja	Baja	Alta
73	Identificación temprana de defectos	Incorporación permanente de un proceso de revisión de pares	Incorporación espóradica y selectiva de un proceso de revisión de pares	Proceso de revisión de pares no incorporado	Media	Baja	Media	Baja	Media
74	Seguimiento de defectos	Se ha definido un efectivo y consistente proceso de seguimiento de defectos	Se ha definido un proceso de seguimiento de defectos pero es usado de forma inconsistente	No se ha definido un proceso para realizar el seguimiento de defectos	Alta	Media	Media	Baja	Alta
75	Proceso de control de cambios	Se ha definido un efectivo y consistente proceso de control de cambios	Se ha definido un proceso de control de cambios pero es usado de forma inconsistente	No se ha definido un proceso de control de cambios	Media	Baja	Baja	Baja	Media

76	Métricas	Las métricas a colectar se encuentran claramente definidas y son facilmente colectables en base a los procesos y herramientas con las que se cuenta	Las métricas a colectar se definidas y en su mayoría son colectables en base a los procesos y herramientas con las que se cuenta	Las métricas a colectar no se encuentran correctamente definidas o no son colectables en base a los procesos y herramientas con las que se cuenta		Alta	Media	Media	Baja	Alta
77	Interdependencia de procesos	No se han identificado interdependencias respecto a los procesos de la organización cliente	Se han identificado algunas interdependencias respecto a los procesos de la organización cliente	Se han identificado importantes interdependencias respecto a los procesos de la organización cliente	**Riesgos válidos sólo cuando el proyecto cuenta con un "cliente/usuario" directo.** **Tipo de pregunta de verificación: (Si/No)**	Media	Media	No Aplica	Alta	Baja
				Entorno de Desarrollo						
78	Adecuación del entorno de desarrollo	El entorno de desarrollo soporta los modelos, procesos, métodos, procedimientos, actividades y fases definidas para el proyecto	El entorno de desarrollo soporta parcialmente los modelos, procesos, métodos, procedimientos, actividades y fases definidas para el proyecto	El entorno de desarrollo no soporta los modelos, procesos, métodos, procedimientos, actividades y fases definidas para el proyecto		Baja	Alta	Baja	Media	Media

79	Instalaciones físicas	Escasas o nulas modificaciones requeridas	Algunas modificaciones requeridas	Amplias modificaciones requeridas o instalaciones inexistentes	Baja	Alta	Baja	Media	Baja
80	Plataforma de hardware	Estable, sin cambios esperados y con capacidad suficiente	Con algunos cambios en proceso pero controlados	Plataforma en desarrollo paralelamente con el software	Baja	Alta	Baja	Media	Baja
81	Disponibilidad de herramientas	Disponibles, documentadas y validadas	Disponibles y validadas pero con algunas adaptaciones requeridas	No validadas, propietarias, sin documentación o con grandes adaptaciones requeridas	Baja	Media	Baja	Media	Media
82	Usabilidad	Los elementos componentes del entorno de desarrollo son faciles de usar, se encuentran accesibles y documentados	Los elementos componentes del entorno de desarrollo se encuentran accesibles y son relativamente faciles de usar aunque no están documentados	Los elementos componentes del entorno de desarrollo no son faciles de usar o no se encuentran accesibles o no están documentados	Baja	Alta	Baja	Media	Baja
83	Soporte de los proveedores	Completo soporte, a precio razonable y en el marco de tiempo requerido	Soporte adecuado al precio contratado con un tiempo de respuesta razonable	Escaso o nulo soporte, con alto costo y pobre tiempo de respuesta	Media	Media	Baja	Media	Baja

84	Recuperación de desastres	Todas las áreas siguen las guías de seguridad y los datos se encuentran respaldados	Se han tomado algunas medidas de seguridad, los datos se encuentran respaldados, las guías y procesos están incompletas o no son respetadas	No se han tomado medidas de seguridad, hay falta de respaldo de datos y las medidas de recuperación de desastres no son consideradas	Media	Baja	Media	Baja	Media
85	Tipo de contrato	Las condiciones del contrato son favorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo)	Las condiciones del contrato son algo desfavorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo)	Las condiciones del contrato son desfavorables en cuanto a formas de pago y adecuación a documentos de acuerdos previos entre las partes (SOW, por ejemplo)	Baja	No Deter.	No Aplica	Media	No Deter.
86	Restricciones contractuales	El contrato no posee restricciones	El contrato posee ciertas restricciones que pueden limitar o impedir el desarrollo del proyecto por forzar el uso de determinadas tecnologías o equipos	El contrato posee muchas restricciones que pueden limitar o impedir el desarrollo del proyecto por forzar el uso de determinadas tecnologías o equipos	Media	Alta	No Aplica	Alta	Alta

87	Dependencias contractuales	El proyecto tiene escasas o nulas dependencias respecto a proveedores externos, equipos del cliente o cualquier otro producto o servicio externo	El proyecto tiene alguanas dependencias respecto a proveedores externos, equipos del cliente o cualquier otro producto o servicio externo	El proyecto tiene significativas dependencias respecto a proveedores externos, equipos del cliente o cualquier otro producto o servicio externo		Media	Alta	Baja	Alta	Media
88	Propiedad intelecutal	Existen un acuerdo previo respecto a los derechos de propiedad intelectual y la metodología para garantizar su cumplimiento.	Existen un acuerdo previo respecto a los derechos de propiedad intelectual pero no se ha establecido una metodología para garantizar su cumplimiento.	No existen un acuerdo previo respecto a los derechos de propiedad intelectual.		Media	Media	Baja	Baja	Baja
89	Subcontratos	Definición adecuada de actividades, mecanisimos de gerenciamiento de subcontrataciones implementados y funcionando correctamente	Definición parcial de actividades y mecanisimos de gerenciamiento de subcontrataciones implementados pero sin funcionar correctamente	Falta de definición de actividades y mecanismos de gerenciamiento de subcontrataciones no implementados	**Riesgo válidos sólo cuando el proyecto realiza subcontratacion. Tipo de pregunta de verificación: (Si/No)**	Media	Media	Baja	Baja	Media
90	Conocimiento de los subcontratados	Experiencias anteriores en contratos de similares características	Experiencias anteriores en contratos de diferentes características	Sin experiencias anteriores		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.

Administración de Proyectos										
91	Enfoque de administración de proyectos	Planificación y monitorización del producto y proyecto existentes	La planificación y monitorización del producto y proyecto necesitan de un mejoramiento	Planificación y monitorización del producto y proyecto inexistentes		Media	Media	Media	Baja	Media
92	Comunicación	Claramente se comunican los objetivos y el estado entre el equipo y el resto de la organización	Se comunica parte de la información solo en algunas situaciones especiales	Esporadicamente se comunican los objetivos y el estado entre el equipo y el resto de la organización		Media	Baja	Baja	Baja	Media
93	Experiencia	El administrador de proyectos posee una amplia experiencia en proyectos similares	El administrador de proyectos posee cierta experiencia o posee amplia experiencia en otro tipo de proyectos	El administrador de proyectos no posee experiencia		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
94	Actitud	El administrador de proyectos esta fuertemente comprometido con éxito del proyecto	El administrador de proyectos esta limitadamente comprometido con éxito del proyecto	El administrador de proyectos esta escasamente comprometido con éxito del proyecto		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.

95	Autoridad	El administrador de proyectos posee formalmente la autoridad para ejercer un liderazgo efectivo	El administrador de proyectos es capaz de ejercer ciertas influencias en la organización en base a sus relaciones personales	El administrador de proyectos no posee una autoridad formal ni cuenta con relaciones peronales con los responsables por la toma de decisiones que le permitan ejercer sus funciones		Media	Baja	Baja	Baja	Media
96	Soporte al administrador de proyectos	Completo soporte del equipo de proyecto	Soporte parcial del equipo de proyecto (con ciertas reservas)	Sin soporte del equipo de proyecto		No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
97	Administración de la configuración	Adecuado sistema de administración de la configuración implementado y utilizado, según los requerimientos del proyecto y con un buen soporte de herramientas	Sistema de administración de la configuración implementado y utilizado sin un buen soporte de herramientas	Sistema de administración de la configuración no implementado		Media	Baja	Baja	Baja	Media
				Equipo de Proyecto						
98	Disponibilidad de los miembros del equipo	Disponibles, con pocas posibilidades de pérdidas y cambios	Disponibles, con altas posibilidades de pérdidas y cambios	Pérdidas y cambios frecuentes y poca posibilidad de retención		Baja	Alta	Baja	Media	Baja
99	Combinación de habilidades del equipo	Buena combinación de disciplinas	Algunas disciplinas inadecuadamente representadas	Algunas disciplinas no se encuentran representadas		Media	Media	Baja	Baja	Media

100	Comunicación interna del equipo	Clara y fluida	Adecuada pero escasa en ciertas situaciones	Confusa o inexistente	No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
101	Experiencia en la aplicación	Alto grado de experiencia del equipo en este tipo de proyectos	Alguna experiencia previa en proyectos similares	Escasa o ninguna experiencia en proyectos similares	No Deter.	Baja	No Deter.	Baja	No Deter.
102	Experiencia con el hardware y software	Alto grado de experiencia con el hardware y el software	Alguna experiencia con el hardware y el software	Escasa o ninguna experiencia con el hardware y el software	Baja	Media	Baja	Alta	Baja
103	Experiencia con el proceso	Alto grado de experiencia con el proceso	Alguna experiencia con el proceso	Escasa o ninguna experiencia con el proceso	No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
104	Entrenamiento del equipo	Plan de entrenamiento existente o entrenamientos iniciados	El entrenamiento para ciertos aspectos no esta disponible o esta planificado a futuro	No existen plan de entrenamiento o el entrenamiento requerido no esta disponible	Alta	Baja	Media	Baja	Alta
105	Espíritu y actitud del equipo	Fuertemente comprometido con éxito del proyecto y cooperativo	Limitadamente comprometido con éxito del proyecto	Escasamente comprometido con éxito del proyecto y poco cohesivo	No Deter.	No Deter.	No Deter.	No Deter.	No Deter.
106	Productividad del equipo	Alta productividad, los hitos son alcanzados y las entregas se realizan a tiempo	Aceptable productividad, los hitos son alcanzados y las entregas se realizan en algunos casos con demoras	Baja productividad, los hitos no son alcanzados y las entregas se realizan con demoras	Media	Media	Baja	Baja	Media

107	Experiencia con el dominio de la aplicación	Alto grado de conocimiento en el dominio de la aplicación	Alguna experiencia en el dominio de la aplicación o capacidad para contactar a los expertos	Nula experiencia en el dominio de la aplicación o sin capacidad para contactar a los expertos		Media	Baja	Baja	Alta	Media
Tecnología										
108	Uso de un proceso de adquisición definido	Efectivo proceso de adquisición establecido y respetado	Proceso de adquisición establecido pero no empleado efectivamente	Proceso de adquisición no establecido		Alta	Baja	Media	Baja	Alta
109	Correspondencia entre la tecnología y el proyecto	La tecnología planificada para el proyecto tiene una alta correspondencia con el problema y las necesidades de los clientes	Alguna de las tecnologías planificadas para el proyecto no se corresponde con el problema o el cliente	La tecnología planificada para el proyecto tiene una baja correspondencia con el problema y las necesidades de los clientes		Media	Baja	Media	Media	Baja
110	Disponibilidad de expertos en la tecnología	Alta disponibilidad de los expertos en la tecnología en el equipo de proyecto	Disponibilidad de expertos en la tecnología en la organización	Escasez de expertos en la tecnología y necesidad de contactarlos fuera de la organización		Media	Alta	Baja	Alta	Media
111	Madurez de la tecnología	La tecnología ha estado en uso en la industria por bastante tiempo	La tecnología es bien entendida en la industria	El proyecto empleará tecnología sin demasiado uso en la industria		Baja	Alta	Baja	Alta	Baja

Mantenimiento										
112	Complejidad de diseño	Mantenibilidad estructural	Algunos aspectos son de dificil mantenimiento	Extremadamente dificil de mantener	**Riesgos válidos sólo cuando el proyecto realiza mantenimiento. Tipo de pregunta de verificación: (Si/No)**	Media	Baja	Baja	Baja	Media
113	Soporte del personal	Disponible, experimentado y en un número adecuado	Los representantes de algunas de las áreas no estarán disponibles	La mayor parte del personal no se encontrará disponible		Media	Alta	Baja	Alta	Baja
114	Soporte a futuro de los proveedores	Completo soporte, a precio razonable y en el marco de tiempo requerido	Soporte adecuado al precio contratado con un tiempo de respuesta razonable	Escaso o nulo soporte, con alto costo y pobre tiempo de respuesta		Baja	Media	Baja	Media	Baja

Tabla 44 - Taxonomía de Riesgos Ajustada

Apéndice

G

Lic. Sebastián D. Maniasi

Matrices de Trazabilidad

VII. Apéndice G: Matrices de Trazabilidad

Se presentan a continuación un conjunto de matrices que permiten realizar la trazabilidad de los artefactos de cada una de las fases completadas.

VII.1. Trazabilidad Requerimientos-Casos de Uso

La siguiente tabla (véase **Tabla 45 - Trazabilidad Requerimientos-Casos de Uso**) permite realizar la trazabilidad entre los requerimientos definidos y los casos de uso identificados.

REQUERIMIENTO	CASO DE USO
RF-001	CU-001
RF-002	CU-002
RF-003	CU-003
RF-004	CU-004
RF-005	CU-005
RF-006	CU-006
RF-007	CU-007
RNF-001	No Aplicable
RNF-002	CU-006 CU-007
RNF-003	No Aplicable
RNF-004	No Aplicable
RNF-005	No Aplicable
RNF-006	No Aplicable

Tabla 45 - Trazabilidad Requerimientos-Casos de Uso

VII.2. Trazabilidad Casos de Uso-Requerimientos

La siguiente tabla (véase **Tabla 46 - Trazabilidad Casos de Uso-Requerimientos**) permite realizar la trazabilidad entre los casos de uso identificados y los requerimientos definidos.

CASO DE USO	REQUERIMIENTO
CU-001	RF-001
CU-002	RF-002
CU-003	RF-003
CU-004	RF-004
CU-005	RF-005
CU-006	RF-006 RNF-002
CU-007	RF-007 RNF-002

Tabla 46 - Trazabilidad Casos de Uso-Requerimientos

VII.3. Trazabilidad Casos de Uso-Interfaces

La siguiente tabla (véase **Tabla 47 - Trazabilidad Casos de Uso-Interfaces**) permite realizar la trazabilidad entre los casos de uso identificados y las interfaces definidas.

CASO DE USO	INTERFAZ
CU-001	INT-001 INT-002 INT-003 INT-004 INT-008
CU-002	INT-005 INT-006 INT-007 INT-012
CU-003	INT-012 INT-013 INT-016
CU-004	INT-006 INT-007 INT-008 INT-009
CU-005	INT-010 INT-011
CU-006	INT-014

CASO DE USO	INTERFAZ
	INT-015
CU-007	INT-015

Tabla 47 - Trazabilidad Casos de Uso-Interfaces

VII.4. Trazabilidad Interfaces-Casos de Uso

La siguiente tabla (véase **Tabla 48 - Trazabilidad Interfaces-Casos de Uso**) permite realizar la trazabilidad entre las interfaces definidas y los casos de uso identificados.

INTERFAZ	CASO DE USO
INT-001	CU-001
INT-002	CU-001
INT-003	CU-001
INT-004	CU-001
INT-005	CU-001 CU-002
INT-006	CU-002 CU-004
INT-007	CU-002 CU-004
INT-008	CU-001 CU-004
INT-009	CU-004
INT-010	CU-005
INT-011	CU-005
INT-012	CU-002 CU-003
INT-013	CU-003
INT-014	CU-006
INT-015	CU-006 CU-007
INT-016	CU-003

Tabla 48 - Trazabilidad Interfaces-Casos de Uso

VII.5. Trazabilidad Casos de Uso-Diagramas de Robustez

La siguiente tabla (véase **Tabla 49 - Trazabilidad Casos de Uso-Diagramas de Robustez**) permite realizar la trazabilidad entre los casos de uso identificados y los diagramas de robustez definidos.

CASO DE USO	DIAGRAMA DE ROBUSTEZ
CU-001	DR-001
CU-002	DR-002
CU-003	DR-003
CU-004	DR-004
CU-005	DR-005
CU-006	DR-006
CU-007	DR-007

Tabla 49 - Trazabilidad Casos de Uso-Diagramas de Robustez

VII.6. Trazabilidad Diagramas de Robustez-Casos de Uso

La siguiente tabla (véase **Tabla 50 - Trazabilidad Diagramas de Robustez-Casos de Uso**) permite realizar la trazabilidad entre los diagramas de robustez definidos y los casos de uso identificados.

DIAGRAMA DE ROBUSTEZ	CASO DE USO
DR-001	CU-001
DR-002	CU-002
DR-003	CU-003
DR-004	CU-004
DR-005	CU-005
DR-006	CU-006
DR-007	CU-007

Tabla 50 - Trazabilidad Diagramas de Robustez-Casos de Uso

VII.7. Trazabilidad Diagramas de Robustez-Clases

La siguiente tabla (véase **Tabla 51 - Trazabilidad Diagramas de Robustez-Clases**) permite realizar la trazabilidad entre los diagramas de robustez definidos y las clases identificadas.

DIAGRAMA DE ROBUSTEZ	CLASES
DR-001	Proyecto Sesion Indicador Identificación Riesgo Categoria
DR-002	Proyecto Sesion Indicador Identificación

DIAGRAMA DE ROBUSTEZ	CLASES
	Riesgo
DR-003	Proyecto Sesion Indicador Identificación Riesgo
DR-004	Proyecto Sesion Indicador Identificación
DR-005	Proyecto Sesion Indicador Identificación
DR-006	Riesgo Categoría
DR-007	Riesgo Categoría

Tabla 51 - Trazabilidad Diagramas de Robustez-Clases

VII.8. Trazabilidad Clases-Diagramas de Robustez

La siguiente tabla (véase **Tabla 52 - Trazabilidad Clases-Diagramas de Robustez**) permite realizar la trazabilidad entre las clases identificadas y los diagramas de robustez definidos.

CLASE	DIAGRAMAS DE ROBUSTEZ
Proyecto	DR-001 DR-002 DR-003 DR-004
Sesion	DR-001 DR-002 DR-003 DR-004
Identificación	DR-001 DR-002 DR-003 DR-004
Categoría	DR-001 DR-006 DR-007
Riesgo	DR-001 DR-002 DR-003

CLASE	DIAGRAMAS DE ROBUSTEZ
	DR-006 DR-007
Indicador	DR-001 DR-002 DR-003 DR-004

Tabla 52 - Trazabilidad Clases-Diagramas de Robustez

VII.9. Trazabilidad Requerimientos-Casos de Prueba

La siguiente tabla (véase **Tabla 53 - Trazabilidad Requerimientos-Casos de Prueba**) permite realizar la trazabilidad entre los requerimientos definidos y los casos de prueba especificados.

REQUERIMIENTO	CASO DE PRUEBA
RF-001	CP-007 CP-008 CP-009 CP-010 CP-011 CP-012 CP-013 CP-014 CP-027
RF-002	CP-015
RF-003	CP-024
RF-004	CP-016 CP-017 CP-023
RF-005	CP-018 CP-021
RF-006	CP-006
RF-007	CP-006
RNF-001	CP-001 CP-002 CP-003 CP-004 CP-005 CP-019 CP-020 CP-022 CP-023 CP-026 CP-029
RNF-002	CP-005

REQUERIMIENTO	CASO DE PRUEBA
	CP-025
RNF-003	CP-029
RNF-004	CP-028
RNF-005	No Aplica
RNF-006	No Aplica

Tabla 53 - Trazabilidad Requerimientos-Casos de Prueba

VII.10. Trazabilidad Casos de Prueba-Requerimientos

La siguiente tabla (véase **Tabla 54 - Trazabilidad Casos de Prueba-Requerimientos**) permite realizar la trazabilidad entre los casos de prueba especificados y los requerimientos definidos.

CASO DE PRUEBA	REQUERIMIENTO
CP-001	RNF-001
CP-002	RNF-001
CP-003	RNF-001
CP-004	RNF-001
CP-005	RNF-001 RNF-002
CP-006	RF-006 RF-007
CP-007	RF-001
CP-008	RF-001
CP-009	RF-001
CP-010	RF-001
CP-011	RF-001
CP-012	RF-001
CP-013	RF-001
CP-014	RF-001
CP-015	RF-002
CP-016	RF-004
CP-017	RF-004
CP-018	RF-005
CP-019	RNF-001
CP-020	RNF-001
CP-021	RF-005
CP-022	RNF-001
CP-023	RF-004 RNF-001
CP-024	RF-003
CP-025	RNF-002
CP-026	RNF-001
CP-027	RF-001

CASO DE PRUEBA	REQUERIMIENTO
CP-028	RNF-004
CP-029	RNF-001 RNF-003

Tabla 54 - Trazabilidad Casos de Prueba-Requerimientos

Apéndice

H

IDENTIFICACIÓN DE RIESGOS DE PROYECTOS DE SOFTWARE EN BASE A TAXONOMÍAS

Lic. Sebastián D. Maniasi

Documentación de Clases

VIII. Apéndice H: Documentación de Clases

Se presentan a continuación un conjunto de cuadros y textos que detallan los componentes de cada uno de los paquetes y clases empleados en el sistema.

VIII.1. Paquetes Principales

Se detallan a continuación los componentes principales de la aplicación a nivel de paquetes.

Paquetes	
controller	Contiene un conjunto de clases e interfaces de control definidas según el patrón de diseño MVC.
model	Contiene un conjunto de clases e interfaces del modelo definidas según el patrón de diseño MVC.
mvc	Contiene el conjunto de clases e interfaces que dan soporte al patrón de diseño MVC (que será efectivamente implementado por las clases de los paquetes model, view y controller).
persistence	Contiene el conjunto de clases e interfaces que permiten lograr el almacenamiento y la recuperación de elementos del sistema empleando el sistema de archivos.
view	Contiene un conjunto de clases e interfaces de vista definidas según el patrón de diseño MVC.

VIII.2. Paquete controller

Se detallan a continuación los componentes del paquete controller.

Resumen de Clases	
DataSource	Clase de control empleada para gestionar los datos requeridos por la utilidad de generación de reportes.
FiltroRiesgos	Clase que realiza el filtrado de riesgos para un proyecto y realiza el análisis con seudoreglas.
IdentificacionFinalizadaController	Clase de control empleada para gestionar las tareas disponibles al finalizar una identificación de riesgos.
Main	Clase que implementa el método main para iniciar la aplicación.
NuevaSesionController	Clase de control empleada para gestionar las tareas disponibles al iniciar una identificación de riesgos.
ParametrosProyectoController	Clase de control empleada para gestionar las tareas disponibles al finalizar la definición de parámetros del proyecto.
PrincipalController	Clase de control empleada para gestionar las tareas disponibles como parte del menú principal del sistema.
ReporteRiesgoController	Clase de control empleada para gestionar las tareas disponibles al generar un reporte luego de una sesión de identificación de riesgos.
Seudoreglas	Clase generada por JEOPS conteniendo la representación en clases Java de las reglas del sistema.

controller

Clase DataSource

```
java.lang.Object
  └controller.DataSource
```

Todas las Interfaces Implementadas:

net.sf.jasperreports.engine.JRDataSource

```
public class DataSource
extends java.lang.Object
implements net.sf.jasperreports.engine.JRDataSource
```

Resumen de Constructores

DataSource()
DataSource(Identificacion[] iden)

Resumen de Métodos

java.lang.Object	**getFieldValue**(JRField field)
boolean	**next**()

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

DataSource

```
public DataSource()
```

DataSource

```
public DataSource(Identificacion[] iden)
```

Detalle de Métodos

next

```
public boolean next()
             throws net.sf.jasperreports.engine.JRException
```

Especificado por:

next en net.sf.jasperreports.engine.JRDataSource

Dispara:

net.sf.jasperreports.engine.JRException

---

getFieldValue

```
public java.lang.Object
getFieldValue(net.sf.jasperreports.engine.JRField field)
throws net.sf.jasperreports.engine.JRException
```

Especificado por:

getFieldValue en net.sf.jasperreports.engine.JRDataSource

Dispara:

net.sf.jasperreports.engine.JRException

---

controller

Clase FiltroRiesgos

```
java.lang.Object
  └controller.FiltroRiesgos
```

```
public class FiltroRiesgos
extends java.lang.Object
```

Resumen de Constructores

FiltroRiesgos(Sesion sesion, Categoria taxonomia)	

Resumen de Métodos

void	**actualizarRiesgo**(Riesgo riesgo, int nivel) Actualiza el nivel de un riesgo.
Identificacion	**getIdentificacion**(int id) Devuelve la identificación que posee el riesgo con ese identificador.
boolean	**lanzarEvento**() Método que lanza un evento con un riesgo no determinado.

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

FiltroRiesgos

```
public FiltroRiesgos(Sesion sesion,
                     Categoria taxonomia)
```

Constructor de la clase.

Parámetros:

`proyecto` - Proyecto en análisis.

`taxonomia` - Taxonomía de riesgos.

Detalle de Métodos

actualizarRiesgo

```
public void actualizarRiesgo(Riesgo riesgo,
                             int nivel)
```

Actualiza el nivel de un riesgo.

Parámetros:

`riesgo` - Riesgo a cambiar el nivel.

`nivel` - Nuevo nivel del riesgo.

---

lanzarEvento

`public boolean` **`lanzarEvento`**`()`

Método que lanza un evento con un riesgo no determinado.

Retorna:

true - Lanzó un evento false - No hay riesgos no determinados.

---

getIdentificacion

`public Identificacion` **`getIdentificacion`**`(int id)`

Devuelve la identificación que posee el riesgo con ese identificador.

Parámetros:

`id` - Identificador del riesgo buscado.

Retorna:

Identificación del riesgo buscado null - Si no la encuentra.

---

controller

Clase IdentificacionFinalizadaController

```
java.lang.Object
  └mvc.AbstractController
      └controller.IdentificacionFinalizadaController
```

Todas las Interfaces Implementadas:

Controller

```
public class IdentificacionFinalizadaController
extends AbstractController
```

Resumen de Constructores

`IdentificacionFinalizadaController``()`

Resumen de Métodos

`void`	**`almacenarSesion`**`(java.lang.String file)` Método que invoca al almacenado de la sesión por parte del modelo que la soporta.
`void`	**`mostrarReporte`**`()` Método que permite mostrar el reporte de la identificación de riesgos.
`void`	**`showView`**`()` Permite mostrar la vista asociada al controlador.

Métodos heredados de clasemvc.AbstractController

`getModel, getView, setModel, setView`

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

IdentificacionFinalizadaController

```
public IdentificacionFinalizadaController()
```

Detalle de Métodos

showView

```
public void showView()
```

Descripción copiada de class: `AbstractController`

Permite mostrar la vista asociada al controlador.

Especificado por:

`showView` en `AbstractController`

---

almacenarSesion

```
public void almacenarSesion(java.lang.String file)
                    throws PersistenceException
```

Método que invoca al almacenado de la sesión por parte del modelo que la soporta.

Parámetros:

`file` - El nombre del archivo contenedor de la sesión.

Dispara:

`PersistenceException`

---

mostrarReporte

```
public void mostrarReporte()
```

Método que permite mostrar el reporte de la identificación de riesgos.

---

controller

Clase Main

```
java.lang.Object
  └controller.Main
```

```
public class Main
extends java.lang.Object
```

Resumen de Campos

`static MessageFormatManager`	**`messageManager`**

Resumen de Constructores

`Main``()`

Resumen de Métodos

static void	**cargarProps**(java.io.InputStream propsStream) Carga los valores del archivo en forma de properties.
static MessageFormatManager	**getMessageManager**() Devuelve el MessageFormatManager para la extraccion de mensajes.
static void	**main**(java.lang.String[] args) Método principal de la aplicación.

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Campos

messageManager

```
public static MessageFormatManager messageManager
```

Detalle de Constructores

Main

```
public Main()
```

Detalle de Métodos

getMessageManager

```
public static MessageFormatManager getMessageManager()
```

Devuelve el MessageFormatManager para la extraccion de mensajes.

Retorna:

Retorna messageManager.

---

main

```
public static void main(java.lang.String[] args)
```

Método principal de la aplicación.

---

cargarProps

```
public static void cargarProps(java.io.InputStream propsStream)
```

Carga los valores del archivo en forma de properties.

Parámetros:

`propsStream` - Acceso al archivo de propiedades.

---

controller

Clase NuevaSesionController

```
java.lang.Object
  └mvc.AbstractController
      └controller.NuevaSesionController
```

Todas las Interfaces Implementadas:

Controller

```
public class NuevaSesionController
extends AbstractController
```

Resumen de Constructores

NuevaSesionController()

Resumen de Métodos

void	**crearSesion**(java.lang.String org, java.lang.String pro, java.lang.String res, TipoProyecto tipoPro, boolean clienteDirecto, boolean admPresupuesto, boolean instalacion, boolean inclPrograma, boolean subContratacion, boolean integracion, boolean reqCalidad, boolean mantenimiento) Método de creación de nuevas sesiones.
void	**identificarProyecto**(Sesion sesion)
Static boolean	**isSingle**() Método para identificar si existe una única sesión activa.
static void	**setSingle**(boolean single) Método para indicar si existe una única sesión activa.
void	**showView**() Permite mostrar la vista asociada al controlador.

Métodos heredados de clasemvc.AbstractController

getModel, getView, setModel, setView

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

NuevaSesionController

```
public NuevaSesionController()
```

Detalle de Métodos

showView

```
public void showView()
```

Descripción copiada de class: `AbstractController`

Permite mostrar la vista asociada al controlador.

Especificado por:

`showView` en `AbstractController`

---

identificarProyecto

```
public void identificarProyecto(Sesion sesion)
```

---

crearSesion

```
public void crearSesion(java.lang.String org,
                        java.lang.String pro,
                        java.lang.String res,
                        TipoProyecto tipoPro,
                        boolean clienteDirecto,
                        boolean admPresupuesto,
                        boolean instalacion,
                        boolean inclPrograma,
                        boolean subContratacion,
                        boolean integracion,
                        boolean reqCalidad,
                        boolean mantenimiento)
```

Método de creación de nuevas sesiones.

---

isSingle

```
public static boolean isSingle()
```

Método para identificar si existe una única sesión activa.

Retorna:

Retorna el indicador de sesión única.

---

setSingle

```
public static void setSingle(boolean single)
```

Método para indicar si existe una única sesión activa.

Parámetros:

`single` - El indicador de sesión única.

---

controller

Clase ParametrosProyectoController

```
java.lang.Object
  └mvc.AbstractController
      └controller.ParametrosProyectoController
```

Todas las Interfaces Implementadas:

Controller

```
public class ParametrosProyectoController
extends AbstractController
```

Resumen de Constructores

ParametrosProyectoController()

Resumen de Métodos

void	**actualizarRiesgo**(Riesgo riesgo, int nivel) Permite actualizar el nivel de los riesgos en base a los parámetros seleccionados.
void	**filtrarRiesgos**() Permite filtrar riesgos aplicables en base a los parámetros seleccionados.
void	**showView**() Permite mostrar la vista asociada al controlador.

Métodos heredados de clasemvc.AbstractController

getModel, getView, setModel, setView

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

ParametrosProyectoController

public **ParametrosProyectoController**()

Detalle de Métodos

showView

public void **showView**()

Descripción copiada de class: AbstractController

Permite mostrar la vista asociada al controlador.

Especificado por:

showView en AbstractController

filtrarRiesgos

public void **filtrarRiesgos**()

Permite filtrar riesgos aplicables en base a los parámetros seleccionados.

actualizarRiesgo

```
public void actualizarRiesgo(Riesgo riesgo,
                             int nivel)
```

Permite actualizar el nivel de los riesgos en base a los parámetros seleccionados.

Parámetros:

`riesgo` - El riesgo a actualizar.

`nivel` - El nivel a asignar al riesgo.

---

controller

Clase PrincipalController

```
java.lang.Object
  └mvc.AbstractController
      └controller.PrincipalController
```

Todas las Interfaces Implementadas:

Controller

```
public class PrincipalController
extends AbstractController
```

Resumen de Constructores

PrincipalController()	

Resumen de Métodos

void	**almacenarSesion**(java.lang.String file) **Permite almacenar una sesión de identificación.**
static boolean	**getGuardar**() Permite conocer el estado de la opción de guardado de sesiones.
void	**mostrarTaxonomiaBase**() Permite mostrar la taxonomía base.
void	**nuevaSesion**() Permite crear nuevas sesiones de identificación.
void	**recuperarSesion**(java.io.File file) Permite recuperar sesiones de identificación.
static void	**setGuardar**(boolean guardar) Permite habilitar la opción de guardado de sesiones.
void	**showView**() Permite mostrar la vista asociada al controlador.

Métodos heredados de clasemvc.AbstractController

`getModel, getView, setModel, setView`

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

PrincipalController

```
public PrincipalController()
```

Detalle de Métodos

showView

```
public void showView()
```

Descripción copiada de class: `AbstractController`

Permite mostrar la vista asociada al controlador.

Especificado por:

`showView` en `AbstractController`

---

nuevaSesion

```
public void nuevaSesion()
```

Permite crear nuevas sesiones de identificación.

---

recuperarSesion

```
public void recuperarSesion(java.io.File file)
                    throws PersistenceException
```

Permite recuperar sesiones de identificación.

Parámetros:

`file` - El archivo que contiene la sesión a recuperar.

Dispara:

`PersistenceException`

---

mostrarTaxonomiaBase

```
public void mostrarTaxonomiaBase()
```

Permite mostrar la taxonomía base.

---

setGuardar

```
public static void setGuardar(boolean guardar)
```

Permite habilitar la opción de guardado de sesiones.

Parámetros:

`guardar` - El estado de habilitación de la opción.

---

getGuardar

```
public static boolean getGuardar()
```

Permite conocer el estado de la opción de guardado de sesiones.

almacenarSesion

```
public void almacenarSesion(java.lang.String file)
                    throws PersistenceException
```

Permite almacenar una sesión de identificación.

Parámetros:

`file` - El nombre del archivo contenedor de la sesión.

Dispara:

`PersistenceException`

---

controller

Clase ReporteRiesgoController

```
java.lang.Object
  └mvc.AbstractController
      └controller.ReporteRiesgoController
```

Todas las Interfaces Implementadas:

Controller

```
public class ReporteRiesgoController
extends AbstractController
```

Resumen de Constructores

ReporteRiesgoController()

Resumen de Métodos

void	**almacenarSesion**(java.lang.String file) Método que invoca al almacenado de la sesión por parte del modelo que la soporta.
void	**imprimirReporte**() Método para invocar la impresión del reporte.
static boolean	**isSingle**() Método para identificar si existe un único reporte de riesgos activo.
boolean	**isTBase**() Método para identificar si se realiza un reporte de taxonomía base.
static void	**setSingle**(boolean single) Método para indicar si existe un único reporte de riesgos activo.
void	**setTBase**(boolean base) Método para indicar si se realiza un reporte de taxonomía base.
void	**showView**() Permite mostrar la vista asociada al controlador.

Métodos heredados de clasemvc.AbstractController
getModel, getView, setModel, setView

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

ReporteRiesgoController

```
public ReporteRiesgoController()
```

Detalle de Métodos

isTBase

```
public boolean isTBase()
```

Método para identificar si se realiza un reporte de taxonomía base.

Retorna:

Retorna el indicador de taxonomía base.

---

setTBase

```
public void setTBase(boolean base)
```

Método para indicar si se realiza un reporte de taxonomía base.

Parámetros:

`base` - El indicador de taxonomía base.

---

showView

```
public void showView()
```

Descripción copiada de class: `AbstractController`

Permite mostrar la vista asociada al controlador.

Especificado por:

`showView` en `AbstractController`

---

almacenarSesion

```
public void almacenarSesion(java.lang.String file)
                    throws PersistenceException
```

Método que invoca al almacenado de la sesión por parte del modelo que la soporta.

Parámetros:

`file` - El nombre del archivo contenedor de la sesión.

Dispara:

`PersistenceException`

---

imprimirReporte

`public void` **`imprimirReporte`**`()`

Método para invocar la impresión del reporte.

---

isSingle

`public static boolean` **`isSingle`**`()`

Método para identificar si existe un único reporte de riesgos activo.

Retorna:

Retorna el indicador de sesión única.

---

setSingle

`public static void` **`setSingle`**`(boolean single)`

Método para indicar si existe un único reporte de riesgos activo.

Parámetros:

`single` - El indicador de reporte único.

---

controller

Clase Seudoreglas

```
java.lang.Object
  └jeops.AbstractKnowledgeBase
      └controller.Seudoreglas
```

Todas las Interfaces Implementadas:

jeops.conflict.InternalConflictSetListener

`public class` **`Seudoreglas`**
`extends jeops.AbstractKnowledgeBase`

Resumen de Constructores

`Seudoreglas``()`
`Seudoreglas``(jeops.conflict.ConflictSet conflictSet)`

Resumen de Métodos

Métodos heredados de clasejeops.AbstractKnowledgeBase

`addConflictSetListener, addRuleFireListener, flush, getRuleBase, insert, internalElementAdded, internalElementRemoved, newFirableRule, objects, removeConflictSetListener, removeRuleFireListener, retract, run, setConflictSet`

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

Seudoreglas

```
public Seudoreglas(jeops.conflict.ConflictSet conflictSet)
```

Seudoreglas

```
public Seudoreglas()
```

VIII.3. Paquete model

Se detallan a continuación los componentes del paquete model.

Resumen de Interfaz	
DefinicionesComunes	Define constantes de uso frecuente en el sistema

Resumen de Clases	
Categoria	Clase que representa una categoría de la Taxonomía Base para la identificación de riesgos.
Identificacion	Clase que representa cada una de las identificaciones de riesgos realizadas por el sistema.
Indicador	Clase que representa los diferentes indicadores para el nivel de un riesgo.
Probabilidad	Clase que representa la probabilidad asociada a un riesgo.
ProjectElement	Clase adaptada de los ejemplos de SUN para el manejo de elementos de un árbol. Posibilita el mostrar elementos de tipo proyecto.
Proyecto	Clase que representa un proyecto.
Riesgo	Clase que representa un riesgo.
RiskElement	Clase adaptada de los ejemplos de SUN para el manejo de elementos de un árbol. Posibilita el mostrar elementos de tipo riesgo.
Sesion	Clase que representa una sesión de identificación de riesgos.
ShowElement	Clase adaptada de los ejemplos de SUN para el manejo de elementos de un árbol. Posibilita el mostrar elementos de tipo descriptivos de un riesgo.
TaxoElement	Clase adaptada de los ejemplos de SUN para el manejo de elementos de un árbol. Posibilita el mostrar elementos de tipo taxonomía.
TipoProyecto	Clase que representa los diferentes tipos de proyectos

	existentes en el sistema.
TitledElement	Clase adaptada de los ejemplos de SUN para el manejo de elementos de un árbol. Posibilita el mostrar elementos de tipo categorías de riesgos.

model

Clase Categoria

```
java.lang.Object
  └model.Categoria
```

```
public class Categoria
extends java.lang.Object
```

Resumen de Constructores

Categoria()
Categoria(java.lang.String nombre)

Resumen de Métodos

void	**agregarCategoria**(Categoria categoria) Agrega una subcategoría a la categoría.
void	**agregarRiesgo**(Riesgo riesgo) Agrega un riesgo a la categoría.
static Categoria	**cargarTaxonomia**(String archivoProps, java.lang.String archivoTaxonomias) Crea la taxonomía base con la que trabaja el sistema.
boolean	**equals**(java.lang.Object categoria) Permite la comparación de categorías.
Static javax.swing.tree.TreeNode	**getArbol**() Devuelve la raíz para visualizar el árbol en un JTree.
java.util.Set	**getCategorias**() Devuelve el conjunto de categorías.
java.lang.String	**getNombre**() Devuelve el nombre de la categoría.
java.util.Set	**getRiesgos**() Devuelve el conjunto de riesgos.
Categoria[]	**obtenerCategorias**() Devuelve las subcategorías dependientes de esta categoría.

Riesgo	**obtenerRiesgo**(int identificador) Devuelve un riesgo de acuerdo a su identificador.
Riesgo[]	**obtenerRiesgos**() Devuelve todos los riesgos pertenecientes a la categoría.
static Categoria	**obtenerTaxonomia**() Devuelve la taxonomía con la que fue construida la categoría.
void	**setCategorias**(java.util.Set categorias) Almacena el conjunto de categorías.
void	**setNombre**(java.lang.String nombre) Almacena el nombre de la categoría.
void	**setRiesgos**(java.util.Set riesgos) Alamcena el conjunto de riesgos.

Métodos heredados de clasejava.lang.Object
getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Categoria

```
public Categoria()
```

Categoria

```
public Categoria(java.lang.String nombre)
```

Detalle de Métodos

cargarTaxonomia

```
public static Categoria cargarTaxonomia(java.lang.String archivoProps,
java.lang.String archivoTaxonomias)
```

Crea la taxonomía base con la que trabaja el sistema.

Parámetros:

`archivoProps` - Archivo de porpiedades

`archivoTaxonomias` - Archivo XML que posee la taxonomía base.

Retorna:

Categoría que es la raíz de la taxonomía.

---

obtenerTaxonomia

```
public static Categoria obtenerTaxonomia()
```

Devuelve la taxonomía con la que fue construida la categoría.

Retorna:

Categoría que es la raíz de la taxonomía.

---

getArbol

```
public static javax.swing.tree.TreeNode getArbol()
```

Devuelve la raíz para visualizar el árbol en un JTree.

Retorna:

Retorna la raíz del árbol.

---

getNombre

```
public java.lang.String getNombre()
```

Devuelve el nombre de la categoría.

Retorna:

Retorna el nombre.

---

obtenerRiesgos

```
public Riesgo[] obtenerRiesgos()
```

Devuelve todos los riesgos pertenecientes a la categoría.

Retorna:

Vector de riesgos.

---

obtenerRiesgo

```
public Riesgo obtenerRiesgo(int identificador)
```

Devuelve un riesgo de acuerdo a su identificador.

Parámetros:

`identificador` - Identificador del riesgo buscado.

Retorna:

Riesgo que posea el identificador pasado por parámetro.

---

agregarRiesgo

```
public void agregarRiesgo(Riesgo riesgo)
```

Agrega un riesgo a la categoría.

Parámetros:

`riesgo` - Riesgo a agregar.

---

obtenerCategorias

```
public Categoria[] obtenerCategorias()
```

Devuelve las subcategorías dependientes de esta categoría.

Retorna:

Vector de categorías.

---

agregarCategoria

```
public void agregarCategoria(Categoria categoria)
```

Agrega una subcategoría a la categoría.

Parámetros:

`categoria` - Categoría a agregar.

---

getCategorias

`public java.util.Set` **`getCategorias`**`()`

Devuelve el conjunto de categorías.

Retorna:

Retorna el conjunto de categorías.

---

getRiesgos

`public java.util.Set` **`getRiesgos`**`()`

Devuelve el conjunto de riesgos.

Retorna:

Retorna el conjunto de riesgos.

---

setCategorias

`public void` **`setCategorias`**`(java.util.Set categorias)`

Almacena el conjunto de categorías.

Parámetros:

`categorias` - El conjunto a almacenar.

---

setNombre

`public void` **`setNombre`**`(java.lang.String nombre)`

Almacena el nombre de la categoría.

Parámetros:

`nombre` - Nombre de la categoría.

---

setRiesgos

`public void` **`setRiesgos`**`(java.util.Set riesgos)`

Alamcena el conjunto de riesgos.

Parámetros:

`riesgos` - El conjunto a almacenar.

---

equals

`public boolean` **`equals`**`(java.lang.Object categoria)`

Permite la comparación de categorías.

Overrides:

`equals` en `java.lang.Object`

---

model

Clase Identificacion

```
java.lang.Object
   └ model.Identificacion
```

`public class` **`Identificacion`**
`extends java.lang.Object`

Resumen de Constructores

Identificacion()
Identificacion(int nivel, Riesgo riesgo)

Resumen de Métodos

int	**getNivel**() Devuelve el nivel de riesgo identificado.
Riesgo	**getRiesgo**() Devuelve el riesgo identificado.
void	**setNivel**(int nivel) Indica el nivel de riesgo.
void	**setRiesgo**(Riesgo riesgo) Alamcena el riesgo de la identificación.

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Identificacion

```
public Identificacion()
```

Identificacion

```
public Identificacion(int nivel,
                      Riesgo riesgo)
```

Detalle de Métodos

getNivel

```
public int getNivel()
```

Devuelve el nivel de riesgo identificado.

Retorna:

Retorna nivel.

---

getRiesgo

```
public Riesgo getRiesgo()
```

Devuelve el riesgo identificado.

Retorna:

Retorna riesgo.

---

setNivel

`public void` **`setNivel`**`(int nivel)`

Indica el nivel de riesgo.

Parámetros:

`nivel` - Nivel de riesgo.

---

setRiesgo

`public void` **`setRiesgo`**`(Riesgo riesgo)`

Alamcena el riesgo de la identificación.

Parámetros:

`riesgo` - El riesgo a almacenar.

---

model

Clase Indicador

`java.lang.Object`
`└` **`model.Indicador`**

`public class` **`Indicador`**
`extends java.lang.Object`

Resumen de Constructores

`Indicador``()`
`Indicador``(int nivel, java.lang.String descripcion)`

Resumen de Métodos

`java.lang.String`	**`getDescripcion`**`()` Devuelve el valor del atributo descripcion.
`int`	**`getNivel`**`()` Devuelve el valor del atributo nivel.
`void`	**`setDescripcion`**`(java.lang.String descripcion)` Almacena la descripción del indicador.
`void`	**`setNivel`**`(int nivel)` Almacena el nivel del indicador.

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

Indicador

`public` **`Indicador`**`()`

Indicador

```
public Indicador(int nivel,
                 java.lang.String descripcion)
```

Detalle de Métodos

getDescripcion

```
public java.lang.String getDescripcion()
```

Devuelve el valor del atributo descripcion.

Retorna:

Retorna descripcion.

---

getNivel

```
public int getNivel()
```

Devuelve el valor del atributo nivel.

Retorna:

Retorna nivel.

---

setDescripcion

```
public void setDescripcion(java.lang.String descripcion)
```

Almacena la descripcion del indicador.

Parámetros:

`descripcion` - La descripcion a almacenar.

---

setNivel

```
public void setNivel(int nivel)
```

Almacena el nivel del indicador.

Parámetros:

`nivel` - El nivel a almacenar.

---

model

Clase Probabilidad

```
java.lang.Object
   └model.Probabilidad
```

```
public class Probabilidad
extends java.lang.Object
```

Resumen de Constructores

Probabilidad()
Probabilidad(int nivel, TipoProyecto tipoProyecto)

Resumen de Métodos

int	**getNivel**() Devuelve el nivel de riesgo.
TipoProyecto	**getTipoProyecto**() Devuelve el tipo de proyecto asociado.

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Probabilidad

```
public Probabilidad()
```

Probabilidad

```
public Probabilidad(int nivel,
                    TipoProyecto tipoProyecto)
```

Detalle de Métodos

getNivel

```
public int getNivel()
```

Devuelve el nivel de riesgo.

Retorna:

Retorna el nivel de riesgo.

---

getTipoProyecto

```
public TipoProyecto getTipoProyecto()
```

Devuelve el tipo de proyecto asociado.

Retorna:

Retorna el tipo de proyecto.

---

model

Clase Proyecto

```
java.lang.Object
   └model.Proyecto
```

```
public class Proyecto
extends java.lang.Object
```

Resumen de Constructores

Proyecto()
Proyecto(java.lang.String nombre, java.lang.String responsable, java.lang.String organizacion)

Resumen de Métodos

boolean	**getAdministraPresupuesto**() Informa si el proyecto administra presupuestos.
boolean	**getDespliegue**() Informa si el proyecto realiza despliegues.
boolean	**getIntegracionHWSW**() Informa si el proyecto realiza integración de hardware/software.
boolean	**getMantenimiento**() Informa si el proyecto realiza mantenimiento.
java.lang.String	**getNombre**() Informa el nombre del proyecto.
java.lang.String	**getOrganizacion**() Informa la organización del proyecto.
boolean	**getPertenecePrograma**() Informa si el proyecto pertenece a un programa.
boolean	**getRequerimientosCalidad**() Informa si el proyecto tiene requerimientos de calidad.
java.lang.String	**getResponsable**() Informa el responsable del proyecto.
boolean	**getSubcontratacion**() Informa si el proyecto maneja subcontratación.
boolean	**getTieneCliente**() Informa el tipo de cliente del proyecto.
TipoProyecto	**getTipoProyecto**() Informa el tipo del proyecto.
void	**setAdministraPresupuesto**(boolean administraPresupuesto) Indica si el proyecto administra presupuestos.
void	**setDespliegue**(boolean despliegue) Indica si el proyecto realiza despliegues.
void	**setIntegracionHWSW**(boolean integracioHWSW) Indica si el proyecto realiza integración de hardware/software.
void	**setMantenimiento**(boolean mantenimiento) Indica si el proyecto realiza mantenimiento.

void	**setNombre**(java.lang.String nombre) Indica el nombre del proyecto.
void	**setOrganizacion**(java.lang.String organizacion) Indica la organización del proyecto.
void	**setPertenecePrograma**(boolean pertenecePrograma) Indica si el proyecto pertenece a un programa.
void	**setRequerimientosCalidad** (boolean requerimientosCalidad) Indica si el proyecto tiene requerimientos de calidad.
void	**setResponsable**(java.lang.String responsable) Indica el responsable del proyecto.
void	**setSubcontratacion**(boolean subcontratacion) Indica si el proyecto maneja subcontratación.
void	**setTieneCliente**(boolean tieneCliente) Indica el tipo de cliente del proyecto.
void	**setTipoProyecto**(TipoProyecto tipoProyecto) Indica el tipo del proyecto.

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Proyecto

```
public Proyecto()
```

Proyecto

```
public Proyecto(java.lang.String nombre,
                java.lang.String responsable,
                java.lang.String organizacion)
```

Detalle de Métodos

getAdministraPresupuesto

```
public boolean getAdministraPresupuesto()
```

Informa si el proyecto administra presupuestos.

Retorna:

Retorna administraPresupuesto.

---

getDespliegue

```
public boolean getDespliegue()
```

Informa si el proyecto realiza despliegues.

Retorna:

Retorna despliegue.

---

getIntegracionHWSW

`public boolean **getIntegracionHWSW**()`

Informa si el proyecto realiza integración de hardware/software.

Retorna:

Retorna integracionHWSW.

---

getMantenimiento

`public boolean **getMantenimiento**()`

Informa si el proyecto realiza mantenimiento.

Retorna:

Retorna mantenimiento.

---

getNombre

`public java.lang.String **getNombre**()`

Informa el nombre del proyecto.

Retorna:

Retorna nombre.

---

getOrganizacion

`public java.lang.String **getOrganizacion**()`

Informa la organización del proyecto.

Retorna:

Retorna organizacion.

---

getPertenecePrograma

`public boolean **getPertenecePrograma**()`

Informa si el proyecto pertenece a un programa.

Retorna:

Retorna pertenecePrograma.

---

getRequerimientosCalidad

`public boolean **getRequerimientosCalidad**()`

Informa si el proyecto tiene requerimientos de calidad.

Retorna:

Retorna the requerimientosCalidad.

---

getResponsable

`public java.lang.String **getResponsable**()`

Informa el responsable del proyecto.

Retorna:

Retorna responsable.

---

getSubcontratacion

`public boolean **getSubcontratacion**()`

Informa si el proyecto maneja subcontratación.

Retorna:
Retorna subcontratacion.

---

getTieneCliente

`public boolean` **`getTieneCliente`**`()`

Informa el tipo de cliente del proyecto.
Retorna:
Retorna tieneCliente.

---

getTipoProyecto

`public TipoProyecto` **`getTipoProyecto`**`()`

Informa el tipo del proyecto.
Retorna:
Retorna tipoProyecto.

---

setAdministraPresupuesto

`public void` **`setAdministraPresupuesto`**`(boolean administraPresupuesto)`

Indica si el proyecto administra presupuestos.
Parámetros:
`administraPresupuesto` - Valor de administraPresupuesto.

---

setDespliegue

`public void` **`setDespliegue`**`(boolean despliegue)`

Indica si el proyecto realiza despliegues.
Parámetros:
`despliegue` - Valor de despliegue.

---

setIntegracionHWSW

`public void` **`setIntegracionHWSW`**`(boolean integracionHWSW)`

Indica si el proyecto realiza integración de hardware/software.
Parámetros:
`integracionHWSW` - Valor de integracionHWSW.

---

setMantenimiento

`public void` **`setMantenimiento`**`(boolean mantenimiento)`

Indica si el proyecto realiza mantenimiento.
Parámetros:
`mantenimiento` - Valor de mantenimiento.

---

setNombre

`public void` **`setNombre`**`(java.lang.String nombre)`

Indica el nombre del proyecto.
Parámetros:
`nombre` - Valor de nombre.

---

setOrganizacion

`public void` **`setOrganizacion`**`(java.lang.String organizacion)`

Indica la organización del proyecto.

Parámetros:

`organizacion` - Valor de organizacion.

---

setPerteneceProgrma

`public void` **`setPerteneceProgrma`**`(boolean perteneceProgrma)`

Indica si el proyecto pertenece a un programa.

Parámetros:

`perteneceProgrma` - Valor de perteneceProgrma.

---

setRequerimientosCalidad

`public void` **`setRequerimientosCalidad`**`(boolean requerimientosCalidad)`

Indica si el proyecto tiene requerimientos de calidad.

Parámetros:

`requerimientosCalidad` - Valor de requerimientosCalidad.

---

setResponsable

`public void` **`setResponsable`**`(java.lang.String responsable)`

Indica el responsable del proyecto.

Parámetros:

`responsable` - Valor de responsable.

---

setSubcontratacion

`public void` **`setSubcontratacion`**`(boolean subcontratacion)`

Indica si el proyecto maneja subcontratación.

Parámetros:

`subcontratacion` - Valor de subcontratacion.

---

setTieneCliente

`public void` **`setTieneCliente`**`(boolean tieneCliente)`

Indica el tipo de cliente del proyecto.

Parámetros:

`tieneCliente` - Valor de tieneCliente.

---

setTipoProyecto

`public void` **`setTipoProyecto`**`(TipoProyecto tipoProyecto)`

Indica el tipo del proyecto.

Parámetros:

`tipoProyecto` - Valor de tipoProyecto.

---

model

Clase Riesgo

```
java.lang.Object
   └model.Riesgo
```

```
public class Riesgo
extends java.lang.Object
```

Resumen de Constructores

Riesgo()
Riesgo(int identificador, java.lang.String factor, Categoria categoria)

Resumen de Métodos

void	**agregarIndicador**(Indicador indicador) Agrega un indicador de nivel de riesgo.
void	**agregarProbabilidad**(Probabilidad probabilidad) Agrega una probabilidad de riesgo según el tipo de proyecto.
void	**cargarIndicadores**(com.sun.xml.tree.ElementNode node) Carga los indicadores para este riesgo en base a la información del nodo.
int	**getAplicabilidad**() Devuelve la aplicabilidad del riesgo.
Categoria	**getCategoria**() Devuelve la categoría a la que pertenece el riesgo.
java.lang.String	**getFactor**() Devuelve el factor de riesgo.
int	**getIdentificador**() Devuelve el identificador del riesgo.
java.util.Set	**getIndicadores**() Devuelve el conjunto de indicadores.
java.util.Set	**getProbabilidades**() Devuelve el conjunto de probabilidades.
java.lang.String	**obtenerDescripcion**(int nivel) Devuelve la descripción de un nivel de riesgo.
Indicador[]	**obtenerIndicadores**() Devuelve los distintos indicadores para el riesgo.
Probabilidad	**obtenerProbabilidad**(TipoProyecto tipoProyecto) Devuelve la probabilidad para el tipo de proyecto.
Probabilidad[]	**obtenerProbabilidades**() Devuelve las distintas probabilidades para el riesgo.

void	**setAplicabilidad**(int aplicabilidad) Almacena el atributo aplicabilidad.
void	**setCategoria**(Categoria categoria) Almacena el atributo categoría.
void	**setFactor**(java.lang.String factor) Almacena el atributo factor.
void	**setIdentificador**(int identificador) Almacena el atributo identificador.
void	**setIndicadores**(java.util.Set indicadores) Almacena el conjunto de indicadores.
void	**setProbabilidades**(java.util.Set probabilidades) Almacena el conjunto de probabilidades.

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Riesgo

```
public Riesgo()
```

Riesgo

```
public Riesgo(int identificador,
              java.lang.String factor,
              Categoria categoria)
```

Detalle de Métodos

getFactor

```
public java.lang.String getFactor()
```

Devuelve el factor de riesgo.

Retorna:

Retorna factor.

---

getIdentificador

```
public int getIdentificador()
```

Devuelve el identificador del riesgo.

Retorna:

Retorna identificador.

---

obtenerIndicadores

```
public Indicador[] obtenerIndicadores()
```

Devuelve los distintos indicadores para el riesgo.

Retorna:

Retorna un vector de indicadores.

---

getCategoria

`public Categoria` **`getCategoria`**`()`

Devuelve la categoría a la que pertenece el riesgo.

Retorna:

Retorna la categoria.

---

obtenerDescripcion

`public java.lang.String` **`obtenerDescripcion`**`(int nivel)`

Devuelve la descripción de un nivel de riesgo.

Parámetros:

`nivel` - Nivel del que se quiere obtener la descripción.

Retorna:

La descripción del nivel pasado por parámetro.

---

agregarIndicador

`public void` **`agregarIndicador`**`(Indicador indicador)`

Agrega un indicador de nivel de riesgo.

Parámetros:

`indicador` - Indicador a agregar al riesgo.

---

agregarProbabilidad

`public void` **`agregarProbabilidad`**`(Probabilidad probabilidad)`

Agrega una probabilidad de riesgo según el tipo de proyecto.

Parámetros:

`indicador` - Indicador a agregar al riesgo.

---

obtenerProbabilidades

`public Probabilidad[]` **`obtenerProbabilidades`**`()`

Devuelve las distintas probabilidades para el riesgo.

Retorna:

Retorna un vector de probabilidades.

---

obtenerProbabilidad

`public Probabilidad` **`obtenerProbabilidad`**`(TipoProyecto tipoProyecto)`

Devuelve la probabilidad para el tipo de proyecto.

Retorna:

Retorna una probabilidad.

---

cargarIndicadores

`public void` **`cargarIndicadores`**`(com.sun.xml.tree.ElementNode node)`

Carga los indicadores para este riesgo en base a la información del nodo.

Parámetros:

`node` - Nodo del documento XML que representa un riesgo.

---

getAplicabilidad

`public int` **`getAplicabilidad`**`()`

Devuelve la aplicabilidad del riesgo.

Retorna:

Retorna aplicabilidad.

---

getIndicadores

`public java.util.Set` **`getIndicadores`**`()`

Devuelve el conjunto de indicadores.

Retorna:

Retorna los indicadores.

---

getProbabilidades

`public java.util.Set` **`getProbabilidades`**`()`

Devuelve el conjunto de probabilidades.

Retorna:

Retorna las probabilidades.

---

setAplicabilidad

`public void` **`setAplicabilidad`**`(int aplicabilidad)`

Almacena el atributo aplicabilidad.

Parámetros:

`aplicabilidad` - La aplicabilidad a almacenar.

---

setCategoria

`public void` **`setCategoria`**`(Categoria categoria)`

Almacena el atributo categoría

Parámetros:

`categoria` - La categoría a almacenar.

---

setFactor

`public void` **`setFactor`**`(java.lang.String factor)`

Almacena el atributo factor.

Parámetros:

`factor` - El factor a almacenar.

---

setIdentificador

`public void` **`setIdentificador`**`(int identificador)`

Almacena el atributo identificador.

Parámetros:

`identificador` - El identificador a almacenar.

---

setIndicadores

```
public void setIndicadores(java.util.Set indicadores)
```

Almacena el conjunto de indicadores.

Parámetros:

`indicadores` - El conjunto a almacenar.

---

setProbabilidades

```
public void setProbabilidades(java.util.Set probabilidades)
```

Almacena el conjunto de probabilidades.

Parámetros:

`probabilidades` - El conjunto a almacenar.

---

model

Clase Sesion

```
java.lang.Object
   └mvc.AbstractModel
        └model.Sesion
```

Todas las Interfaces Implementadas:

Model

```
public class Sesion
extends AbstractModel
```

Resumen de Constructores

Sesion()
Sesion(Proyecto proyecto)

Resumen de Métodos

void	**agregarIdentificacion**(Identificacion identificacion) Agrega una nueva identificación de un riesgo.
void	**almacenarSesion**(java.lang.String nombreArchivo) Almacena la sesión de identificación en un archivo pasado por parámetro.
java.util.Set	**getIdentificaciones**() Devuelve el conjunto de identificaciones.
Proyecto	**getProyecto**() Devuelve el proyecto de la sesión.
Identificacion[]	**obtenerIdentificaciones**() Devuelve los riesgos identificados en la sesión.
static Sesion	**recuperarSesion**(java.lang.String nombreArchivo) Recupera la sesión de identificación de un archivo pasado por parámetro.
void	**setIdentificaciones**(java.util.Set identificaciones)

	Almacena el conjunto de identificaciones.
void	**setProyecto**(Proyecto proyecto) Almacena el atributo proyecto.

Métodos heredados de clasemvc.AbstractModel
addListener, notifyChanged, removeListener

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

Sesion

```
public Sesion()
```

Sesion

```
public Sesion(Proyecto proyecto)
```

Detalle de Métodos

getProyecto

```
public Proyecto getProyecto()
```

Devuelve el proyecto de la sesión.

Retorna:

Retorna el proyecto.

---

obtenerIdentificaciones

```
public Identificacion[] obtenerIdentificaciones()
```

Devuelve los riesgos identificados en la sesión.

Retorna:

Vector de identificaciones.

---

agregarIdentificacion

```
public void agregarIdentificacion(Identificacion identificacion)
```

Agrega una nueva identificación de un riesgo.

Parámetros:

`identificacion` - Identificación a agregar.

---

almacenarSesion

```
public void almacenarSesion(java.lang.String nombreArchivo)
                    throws PersistenceException
```

Almacena la sesión de identificación en un archivo pasado por parámetro.

Parámetros:
`nombreArchivo` - Ruta donde se desea guardar la sesión.
Dispara:
`PersistenceException` - Excepción lanzada ante algún error al guardar la sesion.

---

recuperarSesion

```
public static Sesion recuperarSesion(java.lang.String nombreArchivo)
                             throws PersistenceException
```

Recupera la sesión de identificación de un archivo pasado por parámetro.
Parámetros:
`nombreArchivo` - Ruta donde se encuentra la sesión.
Retorna:
La sesión leída del archivo.
Dispara:
`PersistenceException` - Excepción lanzada ante algún error al leer la sesión.

---

getIdentificaciones

```
public java.util.Set getIdentificaciones()
```

Devuelve el conjunto de identificaciones.
Retorna:
Retorna el conjunto de identificaciones.

---

setIdentificaciones

```
public void setIdentificaciones(java.util.Set identificaciones)
```

Almacena el conjunto de identificaciones.
Parámetros:
`identificaciones` - El conjunto a almacenar.

---

setProyecto

```
public void setProyecto(Proyecto proyecto)
```

Almacena el atributo proyecto.
Parámetros:
`proyecto` - El proyecto a almacenar.

---

model

Clase TipoProyecto

```
java.lang.Object
   └model.TipoProyecto
```

```
public class TipoProyecto
extends java.lang.Object
```

Resumen de Constructores

TipoProyecto()
TipoProyecto(java.lang.String id, java.lang.String denominacion)

Resumen de Métodos

Static TipoProyecto[]	**cargarTiposProyecto**(java.lang.String archivoProps, java.lang.String archivoTiposProyectos) Devuelve el vector con los distintos tipos de proyecto.
java.lang.String	**getDenominacion**() Devuelve la denominación del tipo de proyecto.
java.lang.String	**getId**() Devuelve el identificador del tipo de proyecto.
static TipoProyecto	**obtenerTipoProyecto**(java.lang.String id) Devuelve el tipo asociado a un identificador.
Static TipoProyecto[]	**obtenerTiposProyectos**() Devuelve los tipos de proyectos.
void	**setDenominacion**(java.lang.String denominacion) Almacena el valor de la denominación.
void	**setId**(java.lang.String id) Almacena el identificador del tipo de proyecto.
java.lang.String	**toString**() Reimplementación del método toString.

Métodos heredados de clasejava.lang.Object

Equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

TipoProyecto

```
public TipoProyecto()
```

TipoProyecto

```
public TipoProyecto(java.lang.String id,
                    java.lang.String denominacion)
```

Detalle de Métodos

cargarTiposProyecto

```
public static TipoProyecto[]
cargarTiposProyecto(java.lang.String archivoProps,
java.lang.String archivoTiposProyectos)
```

Devuelve el vector con los distintos tipos de proyecto.

Parámetros:
`archivoProps` - Archivo de propiedades.
`archivoTiposProyectos` - Archivo XML que posee los tipos de proyectos.
Retorna:
Vector con los tipos de proyectos

---

obtenerTiposProyectos

```
public static TipoProyecto[] obtenerTiposProyectos()
```

Devuelve los tipos de proyectos.
Retorna:
Retorna los tipos de proyecto.

---

obtenerTipoProyecto

```
public static TipoProyecto obtenerTipoProyecto(java.lang.String id)
```

Devuelve el tipo asociado a un identificador.
Retorna:
Retorna el tipo asociado al identificador.

---

getDenominacion

```
public java.lang.String getDenominacion()
```

Devuelve la denominación del tipo de proyecto.
Retorna:
Retorna denominacion.

---

getId

```
public java.lang.String getId()
```

Devuelve el identificador del tipo de proyecto.
Retorna:
Retorna el identificador.

---

toString

```
public java.lang.String toString()
```

Reimplementación del método toString.
Overrides:
`toString` en `java.lang.Object`

---

setDenominacion

```
public void setDenominacion(java.lang.String denominacion)
```

Almacena el valor de la denominación.
Parámetros:
`denominacion` - La denominacion a almacenar.

---

setId

`public void` **`setId`**`(java.lang.String id)`

Almacena el identificador del tipo de proyecto.

Parámetros:

`id` - El identificador a almacenar.

Notas:

- La interfaz `DefinicionesComunes` no ha sido detallada por contener solamente declaraciones de constantes.
- Las clases `ProjectElement`, `RiskElement`, `ShowElement`, `TaxoElement` y `TitledElement` no han sido detalladas por estar basadas en ejemplos provistos por SUN.

VIII.4. Paquete mvc

Se detallan a continuación los componentes del paquete mvc.

Resumen de Interfaz	
Controller	Interfaz que define el comportamiento común esperado para todos los controladores en el patrón MVC.
Model	Interfaz que define el comportamiento común esperado para todos los controladores en el patrón MVC.
ModelListener	Interfaz que define el comportamiento común esperado para todos los objetos que deseen suscribirse como interesados en los cambios de un modelo.
View	Interfaz que define el comportamiento común esperado para todas las vistas en el patrón MVC.

Resumen de Clases	
AbstractController	Implementa los métodos definidos en la interfaz Controller para asociar una vista y modelo al controlador y obtenerlos.
AbstractModel	Implementa los métodos definidos en la interfaz Model para registrar y eliminar suscriptores del modelo y para notificar los ambios producidos en el mismo.
JDialogView	Implementa los métodos definidos en la interfaz View para asociar un modelo y un controlador a la vista y obtenerlos.
JFrameView	Implementa los métodos definidos en la interfaz View para asociar un modelo y un controlador a la vista y obtenerlos.
ModelEvent	Representa los eventos de notificacion mediante los que

	los objetos suscriptos son notificados sobre los cambios en los modelos.

mvc

Interfaz Controller

Todas las Clases que la Implementan:

AbstractController, IdentificacionFinalizadaController, NuevaSesionController, ParametrosProyectoController, PrincipalController, ReporteRiesgoController

```
public interface Controller
```

Resumen de Métodos

Model	**getModel**() Obtiene el modelo asociado al controlador.
View	**getView**() Obtiene la vista asociada al controlador.
void	**setModel**(Model model) Asocia un modelo al controlador.
void	**setView**(View view) Asocia una vista al controlador.

Detalle de Métodos

setModel

```
void setModel(Model model)
```

Asocia un modelo al controlador.

Parámetros:

`model` - Modelo ha ser asociado.

getModel

```
Model getModel()
```

Obtiene el modelo asociado al controlador.

Retorna:

Modelo asociado al controlador.

setView

```
void setView(View view)
```

Asocia una vista al controlador.

Parámetros:

`view` - Vista ha ser asociada.

getView

`View` **`getView`**`()`

Obtiene la vista asociada al controlador.

Retorna:

Vista asociada al controlador.

---

mvc

Interfaz Model

Todas las Clases que la Implementan:

AbstractModel, Sesion

`public interface` **`Model`**

Resumen de Métodos

`void`	**`addListener`**`(ModelListener modelListener)` Permite agregar un objeto a la lista de suscriptos al modelo.
`void`	**`notifyChanged`**`(ModelEvent modelEvent)` Permite notificar a los objetos suscriptos los cambios producidos en el modelo.
`void`	**`removeListener`**`(ModelListener modelListener)` Permite eliminar un objeto de la lista de suscriptos al modelo.

Detalle de Métodos

notifyChanged

`void` **`notifyChanged`**`(ModelEvent modelEvent)`

Permite notificar a los objetos suscriptos los cambios producidos en el modelo.

Parámetros:

`modelEvent` - Evento que indica el cambio producido en el modelo.

---

addListener

`void` **`addListener`**`(ModelListener modelListener)`

Permite agregar un objeto a la lista de suscriptos al modelo.

Parámetros:

`modelListener` - Escucha del modelo.

---

removeListener

`void` **`removeListener`**`(ModelListener modelListener)`

Permite eliminar un objeto de la lista de suscriptos al modelo.

Parámetros:

`modelListener` - Escucha del modelo.

---

mvc

Interfaz ModelListener

Todas las Clases que la Implementan:

AcercaDeDialog, IdentificacionFinalizadaDialog, JDialogView, JFrameView, NuevaSesionFrame, ParametrosProyectoDialog, PrincipalView, ReporteRiesgoDialog

---

```
public interface ModelListener
```

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificacion de los cambios en el modelo.

Detalle de Métodos

modelChanged

```
void modelChanged(ModelEvent modelEvent)
```

Permite realizar la notificacion de los cambios en el modelo.

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

mvc

Interfaz View

Todas las Clases que la Implementan:

AcercaDeDialog, IdentificacionFinalizadaDialog, JDialogView, JFrameView, NuevaSesionFrame, ParametrosProyectoDialog, PrincipalView, ReporteRiesgoDialog

---

```
public interface View
```

Resumen de Métodos

Controller	**getController**() Obtiene el controlador asociado a la vista.
Model	**getModel**() Obtiene el modelo asociado a la vista.
void	**setController**(Controller controller) Asocia un controlador a la vista.
void	**setModel**(Model model) Asocia un modelo a la vista.

Detalle de Métodos

setController

```
void setController(Controller controller)
```

Asocia un controlador a la vista.

Parámetros:

`controller` - Controlador de la vista.

getController

```
Controller getController()
```

Obtiene el controlador asociado a la vista.

Retorna:

Controlador de la vista.

setModel

```
void setModel(Model model)
```

Asocia un modelo a la vista.

Parámetros:

`model` - Modelo de la vista.

getModel

```
Model getModel()
```

Obtiene el modelo asociado a la vista.

Retorna:

Modelo asociado a la vista.

mvc

Clase AbstractController

```
java.lang.Object
   └mvc.AbstractController
```

Todas las Interfaces Implementadas:

Controller

Subclases Directas:

IdentificacionFinalizadaController, NuevaSesionController, ParametrosProyectoController, PrincipalController, ReporteRiesgoController

```
public abstract class AbstractController
extends java.lang.Object
implements Controller
```

Resumen de Constructores

AbstractController()

Resumen de Métodos

Model	**getModel**() Obtiene el modelo asociado al controlador.
View	**getView**() Obtiene la vista asociada al controlador.

void	**setModel**(Model model) Asocia un modelo al controlador.
void	**setView**(View view) Asocia una vista al controlador.
abstract void	**showView**() Permite mostrar la vista asociada al controlador.

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Constructores

AbstractController

```
public AbstractController()
```

Detalle de Métodos

showView

```
public abstract void showView()
```

Permite mostrar la vista asociada al controlador.

---

getModel

```
public Model getModel()
```

Descripción copiada de interface: `Controller`

Obtiene el modelo asociado al controlador.

Especificado por:

`getModel` en `Controller`

Retorna:

Modelo asociado al controlador.

---

getView

```
public View getView()
```

Descripción copiada de interface: `Controller`

Obtiene la vista asociada al controlador.

Especificado por:

`getView` en `Controller`

Retorna:

Vista asociada al controlador.

---

setModel

```
public void setModel(Model model)
```

Descripción copiada de interface: `Controller`

Asocia un modelo al controlador.

Especificado por:

`setModel` en `Controller`

Parámetros:
`model` - Modelo ha ser asociado.

---

setView

```
public void setView(View view)
```

Descripción copiada de interface: `Controller`
Asocia una vista al controlador.
Especificado por:
`setView` en `Controller`
Parámetros:
`view` - Vista ha ser asociada.

---

mvc

Clase AbstractModel

```
java.lang.Object
   └mvc.AbstractModel
```

Todas las Interfaces Implementadas:
Model
Subclases Directas:
Sesion

```
public abstract class AbstractModel
extends java.lang.Object
implements Model
```

Resumen de Constructores

`AbstractModel``()`

Resumen de Métodos

`void`	**`addListener`**`(ModelListener modelListener)` Permite agregar un objeto a la lista de suscriptos al modelo.
`void`	**`notifyChanged`**`(ModelEvent modelEvent)` Permite notificar a los objetos suscriptos los cambios producidos en el modelo.
`void`	**`removeListener`**`(ModelListener modelListener)` Permite eliminar un objeto de la lista de suscriptos al modelo.

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

AbstractModel

```
public AbstractModel()
```

Detalle de Métodos

addListener

```
public void addListener(ModelListener modelListener)
```

Descripción copiada de interface: `Model`

Permite agregar un objeto a la lista de suscriptos al modelo.

Especificado por:

`addListener` en `Model`

Parámetros:

`modelListener` - Escucha del modelo.

---

notifyChanged

```
public void notifyChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `Model`

Permite notificar a los objetos suscriptos los cambios producidos en el modelo.

Especificado por:

`notifyChanged` en `Model`

Parámetros:

`modelEvent` - Evento que indica el cambio producido en el modelo.

---

removeListener

```
public void removeListener(ModelListener modelListener)
```

Descripción copiada de interface: `Model`

Permite eliminar un objeto de la lista de suscriptos al modelo.

Especificado por:

`removeListener` en `Model`

Parámetros:

`modelListener` - Escucha del modelo.

---

mvc

Clase JDialogView

```
java.lang.Object
   └java.awt.Component
        └java.awt.Container
             └java.awt.Window
                  └java.awt.Dialog
                       └javax.swing.JDialog
                            └mvc.JDialogView
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

Subclases Directas:

AcercaDeDialog, IdentificacionFinalizadaDialog, ParametrosProyectoDialog, ReporteRiesgoDialog

---

```
public abstract class JDialogView
extends javax.swing.JDialog
implements ModelListener, View
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

JDialogView()	
JDialogView(Model model, Controller controller)	

Resumen de Métodos

Controller	**getController**() Obtiene el controlador asociado a la vista.
Model	**getModel**() Obtiene el modelo asociado a la vista.
abstract void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificacion de los cambios en el modelo.
void	**setController**(Controller controller) Asocia un controlador a la vista.
void	**setModel**(Model model) Asocia un modelo a la vista.

Métodos heredados de clasejavax.swing.JDialog

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane,

isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Dialog

addNotify, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground,

getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

JDialogView

```
public JDialogView()
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException` - Lanza una excepcion cuando se detecta un dispositivo no soportado.

JDialogView

```
public JDialogView(Model model,
                   Controller controller)
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException` - Lanza una excepcion cuando se detecta un dispositivo no soportado.

Detalle de Métodos

modelChanged

```
public abstract void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificacion de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

setController

```
public void setController(Controller controller)
```

Descripción copiada de interface: `View`

Asocia un controlador a la vista.

Especificado por:

`setController` en `View`

Parámetros:

`controller` - Controlador de la vista.

---

getController

```
public Controller getController()
```

Descripción copiada de interface: `View`

Obtiene el controlador asociado a la vista.

Especificado por:

`getController` en `View`

Retorna:

Controlador de la vista.

---

setModel

```
public void setModel(Model model)
```

Descripción copiada de interface: `View`

Asocia un modelo a la vista.

Especificado por:

`setModel` en `View`

Parámetros:

`model` - Modelo de la vista.

---

getModel

```
public Model getModel()
```

Descripción copiada de interface: `View`

Obtiene el modelo asociado a la vista.

Especificado por:

`getModel` en `View`

Retorna:

Modelo asociado a la vista.

---

mvc

Clase JFrameView

```
java.lang.Object
   └java.awt.Component
        └java.awt.Container
             └java.awt.Window
                  └java.awt.Frame
                       └javax.swing.JFrame
                            └mvc.JFrameView
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

Subclases Directas:

NuevaSesionFrame, PrincipalView

```
public abstract class JFrameView
extends javax.swing.JFrame
implements ModelListener, View
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejavax.swing.JFrame

EXIT_ON_CLOSE

Campos heredados de clasejava.awt.Frame

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

JFrameView()
JFrameView(Model model, Controller controller)

Resumen de Métodos

Controller	**getController**() Obtiene el controlador asociado a la vista.
Model	**getModel**() Obtiene el modelo asociado a la vista.
abstract void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificacion de los cambios en el modelo.
void	**setController**(Controller controller) Asocia un controlador a la vista.
void	**setModel**(Model model) Asocia un modelo a la vista.

Métodos heredados de clasejavax.swing.JFrame

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Frame

addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor,

setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, show, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint,

setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Métodos heredados de interfaz java.awt.MenuContainer
getFont, postEvent

Detalle de Constructores

JFrameView

```
public JFrameView()
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException` - Lanza una excepcion cuando se detecta un dispositivo no soportado.

JFrameView

```
public JFrameView(Model model,
                  Controller controller)
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException` - Lanza una excepcion cuando se detecta un dispositivo no soportado.

Detalle de Métodos

modelChanged

```
public abstract void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificacion de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

setController

```
public void setController(Controller controller)
```

Descripción copiada de interface: `View`

Asocia un controlador a la vista.

Especificado por:

`setController` en `View`

Parámetros:

`controller` - Controlador de la vista.

---

getController

`public Controller` **`getController`**`()`

Descripción copiada de interface: **`View`**
Obtiene el controlador asociado a la vista.
Especificado por:
`getController` en `View`
Retorna:
Controlador de la vista.

---

setModel

`public void` **`setModel`**`(Model model)`

Descripción copiada de interface: **`View`**
Asocia un modelo a la vista.
Especificado por:
`setModel` en `View`
Parámetros:
`model` - Modelo de la vista.

---

getModel

`public Model` **`getModel`**`()`

Descripción copiada de interface: **`View`**
Obtiene el modelo asociado a la vista.
Especificado por:
`getModel` en `View`
Retorna:
Modelo asociado a la vista.

---

mvc

Clase ModelEvent

```
java.lang.Object
   └mvc.ModelEvent
```

```
public class ModelEvent
extends java.lang.Object
```

Resumen de Constructores

`ModelEvent``(java.lang.String type)`	

Resumen de Métodos

`void`	**`addData`**`(java.lang.String id, java.lang.Object data)` Permite agregar un dato.
`boolean`	**`equals`**`(ModelEvent modelEvent)` Reimplementacion del metodo equals de Java para

	comparar dos eventos.
java.lang.Object	**getData**(java.lang.String id) Permite obtener el valor asociado a un dato almacenado en el mapa.
java.lang.String	**getType**() Permite retornar el tipo de evento.
void	**setType**(java.lang.String type) Permite indicar el tipod e evento.
java.lang.String	**toString**() Reimplementacion de metodo toString de Java para mostrar los contenidos del evento.

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

ModelEvent

```
public ModelEvent(java.lang.String type)
```

Constructor de la clase.

Parámetros:

`type` - Indicador de tipo de evento.

Detalle de Métodos

addData

```
public void addData(java.lang.String id,
                    java.lang.Object data)
```

Permite agregar un dato.

Parámetros:

`id` - Identificador del dato.

`data` - Valor del dato.

---

getData

```
public java.lang.Object getData(java.lang.String id)
```

Permite obtener el valor asociado a un dato almacenado en el mapa.

Parámetros:

`id` - Identificador del dato.

Retorna:

Retorna el valor del dato para el identificador dado.

---

getType

```
public java.lang.String getType()
```

Permite retornar el tipo de evento.

Retorna:

Tipo de evento.

setType

`public void` **`setType`**`(java.lang.String type)`

Permite indicar el tipod e evento.

Parámetros:

`type` - Tipo de evento.

toString

`public java.lang.String` **`toString`**`()`

Reimplementacion de metodo toString de Java para mostrar los contenidos del evento.

Overrides:

`toString` en `java.lang.Object`

Retorna:

Contenido del evento.

equals

`public boolean` **`equals`**`(ModelEvent modelEvent)`

Reimplementacion del metodo equals de Java para comparar dos eventos.

Parámetros:

`modelEvent` - Evento de cambio en el modelo.

Retorna:

Retorna true en caso de que sean iguales.

VIII.5. Paquete persistence

Se detallan a continuación los componentes del paquete persistence.

Resumen de Clases	
PersistenceManager	Permite manejar el almacenamiento y recuperacion de objetos.

Exception Summary	
PersistenceException	Representa una excepcion arrojada por el subsistema cuando no es posible leer o escribir los objetos en los archivos.

persistence

Clase PersistenceManager

```
java.lang.Object
   └ persistence.PersistenceManager
```

Todas las Interfaces Implementadas:

java.beans.ExceptionListener

```
public final class PersistenceManager
extends java.lang.Object
implements java.beans.ExceptionListener
```

Resumen de Métodos

void	**exceptionThrown**(java.lang.Exception exception) Este método es invocado cada vez que se produce una excepción sobre un objeto sobre el que se esta escuchando.
static PersistenceManager	**getInstance**() Retorna la instancia de PersistenceManager.
java.lang.Object	**read**(java.lang.String file) Permite leer un objeto desde un archivo.
void	**write**(java.lang.Object data, java.lang.String file) Permite escribir un objeto en un archivo.

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalle de Métodos

getInstance

```
public static PersistenceManager getInstance()
```

Retorna la instancia de PersistenceManager.

Retorna:

Instancia de PersistenceManager.

---

read

```
public java.lang.Object read(java.lang.String file)
                     throws PersistenceException
```

Permite leer un objeto desde un archivo.

Parámetros:

`file` - Archivo fuente.

Retorna:

El objeto leído.

Dispara:

`PersistenceException` - Lanza una excepción cuando se produjo algún error al leer el archivo.

---

write

```
public void write(java.lang.Object data,
                  java.lang.String file)
           throws PersistenceException
```

Permite escribir un objeto en un archivo.

Parámetros:

`data` - Objeto por almacenar.

`file` - Archivo destino.

Dispara:

`PersistenceException` - Lanza una excepción cuando se produjo algún error al escribir el archivo.

---

exceptionThrown

```
public void exceptionThrown(java.lang.Exception exception)
```

Este método es invocado cada vez que se produce una excepción sobre un objeto sobre el que se esta escuchando.

Especificado por:

`exceptionThrown` en `java.beans.ExceptionListener`

Parámetros:

`exception` - Excepción producida por el objeto.

---

persistence

Clase PersistenceException

```
java.lang.Object
   └java.lang.Throwable
        └java.lang.Exception
              └persistence.PersistenceException
```

Todas las Interfaces Implementadas:

java.io.Serializable

```
public class PersistenceException
extends java.lang.Exception
```

Ver también:

Serialized Form

Resumen de Constructores

`PersistenceException``()` Constructor de la clase.
`PersistenceException``(java.lang.String text)` Constructor de la clase.
`PersistenceException``(java.lang.String text, java.lang.Throwable problem)` Constructor de la clase.

Resumen de Métodos

Métodos heredados de clasejava.lang.Throwable

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Detalle de Constructores

PersistenceException

```
public PersistenceException()
```

Constructor de la clase.

PersistenceException

```
public PersistenceException(java.lang.String text)
```

Constructor de la clase.

Parámetros:

`text` - Descripción de la excepción.

PersistenceException

```
public PersistenceException(java.lang.String text,
                            java.lang.Throwable problem)
```

Constructor de la clase.

Parámetros:

`text` - Descripción de la excepción.

`problem` - Problema de la excepción.

VIII.6. Paquete view

Se detallan a continuación los componentes del paquete view.

Resumen de Clases

AcercaDeDialog	Clase que representa la ventana que muestra la información general del sistema.
DataTreeTools	Utilidad que permite convertir una Sesion de riesgos en un arbol para ser mostrado.
FiltroUniversal	Clase que representa la ventana de selección de archivos.
IdentificacionFinalizadaDialog	Clase que representa la ventana que informa sobre la finalización de una sesión de

	identificación de riesgos.
Nodo	Reimplementacion de DefaultMutableTreeNode para soportar un atributo con HTML.
NuevaSesionFrame	Clase que representa la ventana que se emplea para comenzar una sesión de identificación de riesgos.
ParametrosProyectoDialog	Clase que representa la ventana que permite la especificación de parámetros del proyecto.
PrincipalView	Clase que representa la ventana principal del sistema.
ReporteRiesgoDialog	Clase que representa la ventana que muestra el reporte de riesgos luego de una sesión de identificación de riesgos.
Util	Clase que contiene utilidades generales para el manejo de ventanas.

view

Clase AcercaDeDialog

```
java.lang.Object
   └java.awt.Component
        └java.awt.Container
             └java.awt.Window
                  └java.awt.Dialog
                       └javax.swing.JDialog
                            └mvc.JDialogView
                                 └view.AcercaDeDialog
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

```
public class AcercaDeDialog
extends JDialogView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejava.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

AcercaDeDialog()
AcercaDeDialog(Model model, Controller controller)

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificacion de los cambios en el modelo.

Métodos heredados de clasemvc.JDialogView
getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JDialog
getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Dialog
addNotify, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Métodos heredados de clasejava.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

AcercaDeDialog

```
public AcercaDeDialog()
               throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

AcercaDeDialog

```
public AcercaDeDialog(Model model,
                      Controller controller)
               throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificacion de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Especificado por:

`modelChanged` en `JDialogView`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

view

Clase DataTreeTools

```
java.lang.Object
   └view.DataTreeTools
```

```
public class DataTreeTools
extends java.lang.Object
```

Resumen de Constructores

DataTreeTools()	

Resumen de Métodos

static DefaultMutableTreeNode	**getRootNode**(Sesion sesion) Devuelve el nodo raíz del árbol generado.

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

DataTreeTools

```
public DataTreeTools()
```

Detalle de Métodos

getRootNode

```
public static javax.swing.tree.DefaultMutableTreeNode
getRootNode(Sesion sesion)
```

Devuelve el nodo raíz del árbol generado.

Parámetros:

`sesion` - Sesión a convertir en árbol.

Retorna:

Nodo raíz del árbol.

---

view

Clase IdentificacionFinalizadaDialog

```
java.lang.Object
   └java.awt.Component
        └java.awt.Container
             └java.awt.Window
                  └java.awt.Dialog
                       └javax.swing.JDialog
                            └mvc.JDialogView
                                 └view.IdentificacionFinalizadaDialog
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

---

```
public class IdentificacionFinalizadaDialog
extends JDialogView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejava.awt.Component

`BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT`

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

IdentificacionFinalizadaDialog()
IdentificacionFinalizadaDialog(Model model, Controller controller)

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificación de los cambios en el modelo.

Métodos heredados de clasemvc.JDialogView

getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JDialog

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Dialog

addNotify, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Detalle de Constructores

IdentificacionFinalizadaDialog

```
public IdentificacionFinalizadaDialog()
                              throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

IdentificacionFinalizadaDialog

```
public IdentificacionFinalizadaDialog(Model model,
                                      Controller controller)
                              throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificación de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Especificado por:

`modelChanged` en `JDialogView`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

view

Clase Nodo

```
java.lang.Object
   └javax.swing.tree.DefaultMutableTreeNode
      └view.Nodo
```

Todas las Interfaces Implementadas:

java.io.Serializable, java.lang.Cloneable, javax.swing.tree.MutableTreeNode, javax.swing.tree.TreeNode

---

```
public class Nodo
extends javax.swing.tree.DefaultMutableTreeNode
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejavax.swing.tree.DefaultMutableTreeNode

EMPTY_ENUMERATION

Resumen de Constructores

Nodo(java.lang.Object obj)
Nodo(java.lang.Object obj, java.lang.String HTML)

Resumen de Métodos

java.lang.String	**getHTML**() Devuelve la información en formato HTML.
void	**setHTML**(java.lang.String html) Ingresa una nueva información en formato HTML.

Métodos heredados de clasejavax.swing.tree.DefaultMutableTreeNode

add, breadthFirstEnumeration, children, clone, depthFirstEnumeration, getAllowsChildren, getChildAfter, getChildAt, getChildBefore, getChildCount, getDepth, getFirstChild, getFirstLeaf, getIndex, getLastChild, getLastLeaf, getLeafCount, getLevel, getNextLeaf, getNextNode, getNextSibling, getParent, getPath, getPreviousLeaf, getPreviousNode, getPreviousSibling, getRoot, getSharedAncestor, getSiblingCount, getUserObject, getUserObjectPath, insert, isLeaf, isNodeAncestor, isNodeChild, isNodeDescendant, isNodeRelated, isNodeSibling, isRoot, pathFromAncestorEnumeration, postorderEnumeration, preorderEnumeration, remove, remove, removeAllChildren, removeFromParent, setAllowsChildren, setParent, setUserObject, toString

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

Nodo

```
public Nodo(java.lang.Object obj)
```

Nodo

```
public Nodo(java.lang.Object obj,
            java.lang.String HTML)
```

Detalle de Métodos

getHTML

```
public java.lang.String getHTML()
```

Devuelve la información en formato HTML.

Retorna:

Retorna el HTML generado.

---

setHTML

```
public void setHTML(java.lang.String html)
```

Ingresa una nueva información en formato HTML.

Parámetros:

`html` - El HTML a almacenar.

---

view

Clase NuevaSesionFrame

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └mvc.JFrameView
                          └view.NuevaSesionFrame
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

---

```
public class NuevaSesionFrame
extends JFrameView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejavax.swing.JFrame

EXIT_ON_CLOSE

Campos heredados de clasejava.awt.Frame

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

NuevaSesionFrame()
NuevaSesionFrame(Model model, Controller controller)

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificación de los cambios en el modelo.

Métodos heredados de clasemvc.JFrameView

getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JFrame

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Frame

addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext,

getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, show, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener,

removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Métodos heredados de interfaz java.awt.MenuContainer

getFont, postEvent

Detalle de Constructores

NuevaSesionFrame

```
public NuevaSesionFrame()
                throws java.awt.HeadlessException
```

Dispara:

java.awt.HeadlessException

NuevaSesionFrame

```
public NuevaSesionFrame(Model model,
                        Controller controller)
                throws java.awt.HeadlessException
```

Dispara:

java.awt.HeadlessException

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificación de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Especificado por:

`modelChanged` en `JFrameView`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

view

Clase ParametrosProyectoDialog

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └mvc.JDialogView
                          └view.ParametrosProyectoDialog
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

---

```
public class ParametrosProyectoDialog
extends JDialogView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

	ParametrosProyectoDialog(Model model, Controller controller)

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificación de los cambios en el modelo.

Métodos heredados de clasemvc.JDialogView

getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JDialog

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Dialog

addNotify, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,

firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

ParametrosProyectoDialog

```
public ParametrosProyectoDialog(Model model, Controller controller)
                         throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificación de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Especificado por:

`modelChanged` en `JDialogView`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

view

Clase PrincipalView

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └mvc.JFrameView
                          └view.PrincipalView
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

```
public class PrincipalView
extends JFrameView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejavax.swing.JFrame

EXIT_ON_CLOSE

Campos heredados de clasejava.awt.Frame

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

PrincipalView()
PrincipalView(Model model, Controller controller)

Resumen de Métodos

void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificación de los cambios en el modelo.

Métodos heredados de clasemvc.JFrameView

getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JFrame

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Frame

addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, show, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents,

deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Métodos heredados de interfaz java.awt.MenuContainer

`getFont, postEvent`

Detalle de Constructores

PrincipalView

```
public PrincipalView()
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

PrincipalView

```
public PrincipalView(Model model,
                     Controller controller)
            throws java.awt.HeadlessException
```

Dispara:

`java.awt.HeadlessException`

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificación de los cambios en el modelo.

Especificado por:

`modelChanged` en `ModelListener`

Especificado por:

`modelChanged` en `JFrameView`

Parámetros:

`modelEvent` - Evento que informa el cambio en el modelo.

---

view

Clase ReporteRiesgoDialog

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └mvc.JDialogView
                          └view.ReporteRiesgoDialog
```

Todas las Interfaces Implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants, ModelListener, View

---

```
public class ReporteRiesgoDialog
extends JDialogView
```

Ver también:

Serialized Form

Resumen de Campos

Campos heredados de clasejava.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Campos heredados de interfaz javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Campos heredados de interfaz java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Resumen de Constructores

ReporteRiesgoDialog()
ReporteRiesgoDialog(Model model, Controller controller)

Resumen de Métodos

void	**almacenarSesion**() Método que invoca al almacenado de la sesión actual de la vista.
boolean	**isTbase**() Método para identificar si se realiza un reporte de taxonomía base.
void	**modelChanged**(ModelEvent modelEvent) Permite realizar la notificacion de los cambios en el modelo.
void	**setTbase**(boolean tbase) Método para indicar si se realiza un reporte de taxonomía base.
void	**setVisible**(boolean arg) Método que permite indicar el mostrar u ocultar un reporte de riesgos.

Métodos heredados de clasemvc.JDialogView

getController, getModel, setController, setModel

Métodos heredados de clasejavax.swing.JDialog

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, update

Métodos heredados de clasejava.awt.Dialog

addNotify, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Métodos heredados de clasejava.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isAlwaysOnTop, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setAlwaysOnTop, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationByPlatform, setLocationRelativeTo, toBack, toFront

Métodos heredados de clasejava.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate

Métodos heredados de clasejava.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,

firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, show, size, toString, transferFocus, transferFocusUpCycle

Métodos heredados de clasejava.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Detalle de Constructores

ReporteRiesgoDialog

```
public ReporteRiesgoDialog()
                    throws java.awt.HeadlessException
```

Dispara:

java.awt.HeadlessException

ReporteRiesgoDialog

```
public ReporteRiesgoDialog(Model model,
                           Controller controller)
                    throws java.awt.HeadlessException
```

Dispara:

java.awt.HeadlessException

Detalle de Métodos

modelChanged

```
public void modelChanged(ModelEvent modelEvent)
```

Descripción copiada de interface: `ModelListener`

Permite realizar la notificación de los cambios en el modelo.
Especificado por:
`modelChanged` en `ModelListener`
Especificado por:
`modelChanged` en `JDialogView`
Parámetros:
`modelEvent` - Evento que informa el cambio en el modelo.

---

isTbase

`public boolean` **`isTbase`**`()`

Método para identificar si se realiza un reporte de taxonomía base.
Retorna:
Retorna el indicador de taxonomía base.

---

setTbase

`public void` **`setTbase`**`(boolean tbase)`

Método para indicar si se realiza un reporte de taxonomía base.
Parámetros:
`base` - El indicador de taxonomía base.

---

setVisible

`public void` **`setVisible`**`(boolean arg)`

Overrides:
`setVisible` en `java.awt.Component`
Método que permite indicar el mostrar u ocultar un reporte de riesgos.
Parámetros:
`arg` - El indicador de mostrar/ocultar el reporte.

---

almacenarSesion

`public void` **`almacenarSesion`**`()`

Método que invoca al almacenado de la sesión actual de la vista.

---

view

Clase Util

`java.lang.Object`
`  └ `**`view.Util`**

`public class` **`Util`**
`extends java.lang.Object`

Resumen de Constructores

`Util``()`

Resumen de Métodos	
`static javax.swing.JDialog`	**`centrarDialog`**`(javax.swing.JDialog dialog)` Método que realiza los cálculos para el centrado de un dialogo en pantalla.
`static javax.swing.JFrame`	**`centrarFrame`**`(javax.swing.JFrame frame)` Método que realiza los cálculos para el centrado de un frame en pantalla.

Métodos heredados de clasejava.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Detalle de Constructores

Util

```
public Util()
```

Detalle de Métodos

centrarFrame

```
public static javax.swing.JFrame
centrarFrame(javax.swing.JFrame frame)
```

Método que realiza los cálculos para el centrado de un frame en pantalla.

Parámetros:

`frame` - El frame a centrar.

---

centrarDialog

```
public static javax.swing.JDialog
centrarDialog(javax.swing.JDialog dialog)
```

Método que realiza los cálculos para el centrado de un dialogo en pantalla.

Parámetros:

`dialog` - El dialog a centrar.

Nota:

- La clase `FiltroUniversal` no ha sido detallada por estar basada en un ejemplo provisto por SUN.

Apéndice

I

Lic. Sebastián D. Maniasi

Métricas Finales de Desarrollo

IX. Apéndice I: Métricas Finales de Desarrollo

Se presentan a continuación un conjunto métricas de desarrollo que resumen los resultados obtenidos en diferentes aspectos y actividades realizadas durante la construcción al sistema.

IX.1. Resultados de la Ejecución de las Pruebas

Se detallan a continuación los resultados obtenidos durante la ejecución de las pruebas del sistema.

IX.1.a. Primer Ciclo

La siguiente tabla (véase **Tabla 55 - Resúmen del Primer Ciclo de Pruebas**) contiene el resúmen de la ejecución del primer ciclo de pruebas.

Ciclo	Fecha	Ejecutor	ID del Caso de Prueba	Criterio	Comentarios	1	2	3	4	5	Paso	Fallo
1	10-Jul-05	S. Maniasi	CP-001	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-002	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-003	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-004	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-005	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-006	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-007	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-008	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-009	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-010	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-011	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-012	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-013	Fallo			1				0	1
1	10-Jul-05	S. Maniasi	CP-014	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-015	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-016	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-017	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-018	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-019	Fallo						1	0	1
1	10-Jul-05	S. Maniasi	CP-020	Fallo						1	0	1
1	10-Jul-05	S. Maniasi	CP-021	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-022	Fallo						1	0	1
1	10-Jul-05	S. Maniasi	CP-023	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-024	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-025	Fallo	Falta ayuda para la opción de Reporte -> Guardar.				1		0	1
1	10-Jul-05	S. Maniasi	CP-026	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-027	Fallo	El sistema permite iniciar muchas sesiones al mismo tiempo.						0	1
1	10-Jul-05	S. Maniasi	CP-028	Paso							1	0
1	10-Jul-05	S. Maniasi	CP-029	Paso							1	0
						0	6	0	1	3		
											18	11

Tabla 55 - Resúmen del Primer Ciclo de Pruebas

La siguiente figura (véase **Figura 108 - Reporte del Primer Ciclo de Pruebas**) contiene el reporte de la ejecución del primer ciclo de pruebas.

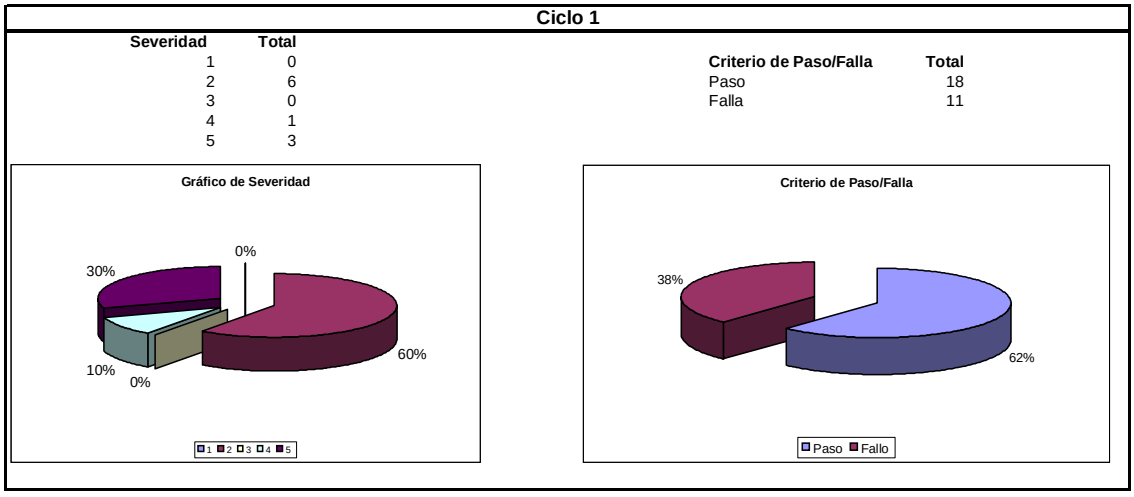


Figura 108 - Reporte del Primer Ciclo de Pruebas

IX.1.b. Segundo Ciclo

La siguiente tabla (véase **Tabla 56 - Resúmen del Segundo Ciclo de Pruebas**) contiene el resúmen de la ejecución del segundo ciclo de pruebas.

Ciclo	Fecha	Ejecutor	ID del Caso de Prueba	Criterio	Comentarios	1	2	3	4	5	Paso	Fallo
2	15-Jul-05	S. Maniasi	CP-001	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-002	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-003	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-004	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-005	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-006	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-007	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-008	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-009	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-010	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-011	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-012	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-013	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-014	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-015	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-016	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-017	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-018	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-019	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-020	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-021	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-022	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-023	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-024	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-025	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-026	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-027	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-028	Paso							1	0
2	15-Jul-05	S. Maniasi	CP-029	Paso							1	0
						0	**0**	**0**	**0**	**0**		
											29	**0**

Tabla 56 - Resúmen del Segundo Ciclo de Pruebas

La siguiente figura (véase **Figura 109 - Reporte del Segundo Ciclo de Pruebas**) contiene el reporte de la ejecución del segundo ciclo de pruebas.

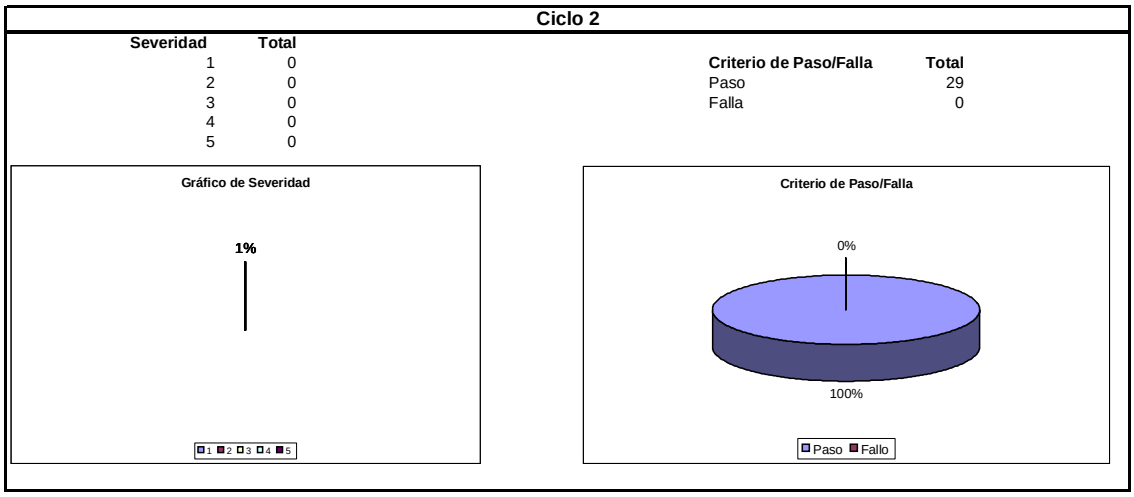


Figura 109 - Reporte del Segundo Ciclo de Pruebas

IX.2. Esfuerzo, Calendario, Líneas de Código y Documentación

La siguiente tabla (véase **Tabla 57 - Comparación de Estimados y Relaes**) compara los valores estimados y reales para la construcción del presente trabajo de tesis en los siguientes aspectos:

- Esfuerzo.
- Calendario.
- Líneas de Código.
- Documentación

Comparaciones Estimados vs. Relaes			
	Estimado	Real	%
Esfuerzo (Hs.)	802.5	895	90%
Calendario (Días)	283	265	107%
Líneas de Código	4800	4747	101%
Documentación	450	460	98%

Tabla 57 - Comparación de Estimados y Relaes

De las mediciones antes detalladas, puede concluirse que las estimaciones realizadas han tenido un alto grado de acierto debido a que para todas las métricas seleccionadas los porcentajes se encuentran contenidos en un ±10% lo cual constituye un porcentual de alta confiabilidad.

Nota: Durante las actividades de codificación del sistema se contó con recursos humanos extra cuyo esfuerzo ha sido considerado para las métricas de la aplicación.