UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

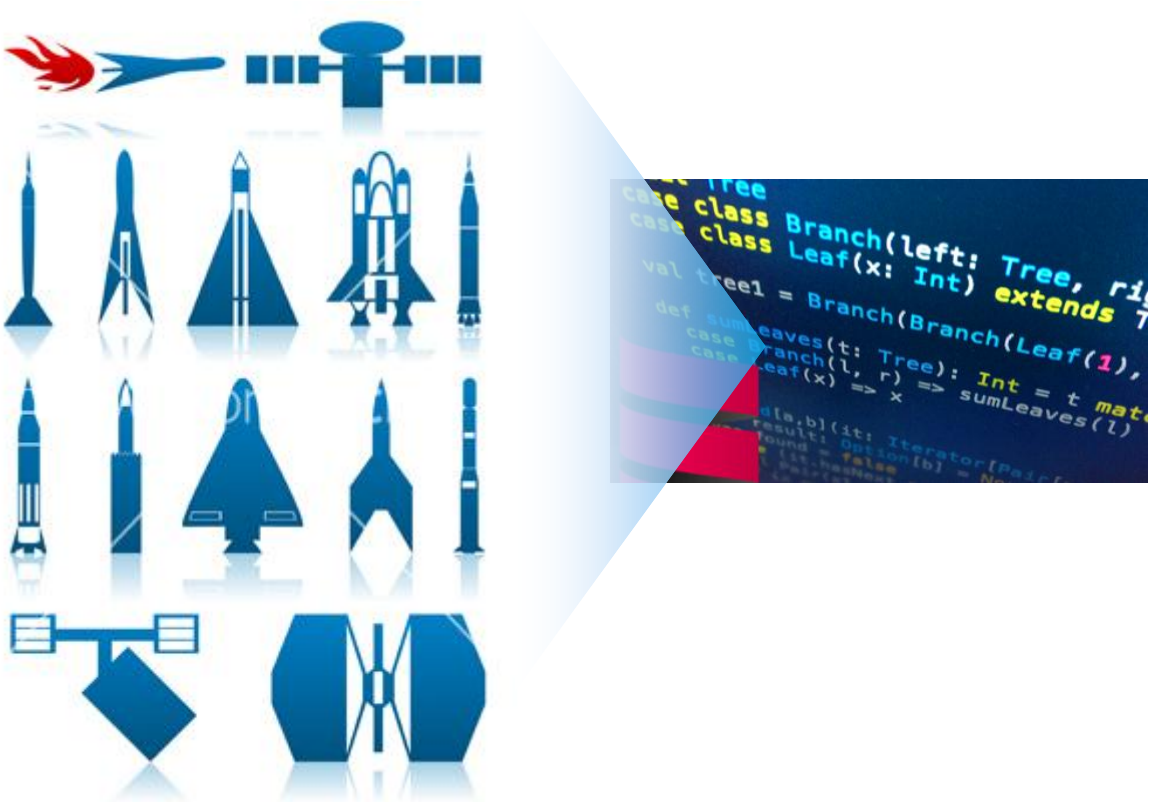
MÁSTER UNIVERSITARIO EN INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

ITINERARIO DE INGENIERÍA DE SOFTWARE

CÓDIGO 31105128

TRABAJO FIN DE MASTER

- Anexo ArchE -



Estudiante: Jose Antonio Miranda Rodera

Profesor: José Félix Estívariz López

Curso: 2014/2015

Convocatoria: Junio - 2015

Índice de Contenido

1 CAPÍTULO 1 – Introducción. Objetivos 25
1.1 Introducción al Trabajo Fin de Máster 25
1.2 Objetivos del Trabajo Fin de Máster 27
1.2.1 Objetivos Generales 27
1.2.2 Objetivos Específicos 27
2 CAPÍTULO 2 - Arquitectura Software: Introducción 28
2.1 Definición de Arquitectura Software 28
2.2 Características Fundamentales de una Arquitectura Software 30
2.2.1 La Arquitectura Define una Estructura 30
2.2.2 La Arquitectura Especifica la Comunicación entre Componentes 31
2.2.3 La Arquitectura Analiza Requisitos No Funcionales 31
2.2.4 La Arquitectura es una Abstracción 31
2.3 Vistas 31
2.4 Patrones y Estilos Arquitectónicos 32
2.4.1 Patrones Software 33
2.4.2 Estilos, Patrones y Dialectos 33
2.4.3 Categorías de los Estilos de Arquitectura 34
2.4.3.1 “From Mud to Structure” 34
2.4.3.1.1 Patrón de Capas 34
2.4.3.1.2 Patrón de Tuberías y Filtros 34
2.4.3.1.3 El Patrón Pizarra o *Blackboard* 35
2.4.3.2 Abstracción de Datos y Organización Orientada a Objetos [8] 35
2.4.3.3 Invocación Implícita Basada en Eventos [8] 36
2.4.3.4 Intérpretes Dirigidos por Tablas [8] 36
2.4.3.5 Sistemas Distribuidos [7] 36
2.4.3.6 Sistemas Interactivos [7] 37
2.4.3.6.1 Patrón Modelo-Vista-Controlador (MVC) 37
2.4.3.6.2 Patrón Presentación-Abstracción-Control (PAC) 37
2.4.3.7 Sistemas Adaptativos [7] 37
2.4.3.7.1 Patrón Microkernel 37
2.4.3.7.2 Patrón de Reflexión 37
3 CAPÍTULO 3 – Atributos de Calidad en Arquitecturas Software 38
3.1 Atributos de Calidad 38

3.1.1 Calidades del Sistema....38
3.1.1.1 Escenarios de Atributos de Calidad....39
3.1.1.2 Descripción General de Atributos de Calidad....39
3.1.1.2.1 Disponibilidad....39
3.1.1.2.2 Modificabilidad....40
3.1.1.2.3 Rendimiento....40
3.1.1.2.4 Seguridad....41
3.1.1.2.5 Testabilidad....41
3.1.1.2.6 Usabilidad....41
3.1.1.2.7 Escalabilidad....41
3.1.1.2.8 Integración....42
3.1.1.2.9 Portabilidad....42
3.1.2 Calidades del Negocio....42
3.1.3 Calidades de la Arquitectura....42
3.2 Alcanzando la Calidad a través de Tácticas....43
3.2.1 Tácticas de Disponibilidad....43
3.2.1.1 Detección de Fallos....44
3.2.1.2 Recuperación de Fallos....44
3.2.1.3 Prevención de Fallos....45
3.2.2 Tácticas de Modificabilidad....45
3.2.2.1 Localizar Cambios/Modificaciones....46
3.2.2.2 Prevenir Efectos de Propagación....46
3.2.2.3 Diferir “binding time”....46
3.2.3 Tácticas de Rendimiento....47
3.2.3.1 Demanda de Recursos....47
3.2.3.2 Gestión de Recursos....48
3.2.3.3 Arbitraje de Recursos....48
3.2.4 Tácticas de Seguridad....48
3.2.4.1 Resistir Ataques....49
3.2.4.2 Detectar Ataques....49
3.2.4.3 Recuperación Después de un Ataque....49
3.2.5 Tácticas de Testabilidad....49
3.2.5.1 Gestionar Entradas/Salidas....50
3.2.5.2 Monitorización Interna....50

3.2.6 Tácticas de Usabilidad 50
3.2.6.1 Tácticas en Tiempo de Ejecución 51
3.2.6.2 Tácticas en Tiempo de Diseño 52
3.3 Diseño Dirigido por Atributos 52
4 CAPÍTULO 4 – Métodos de Análisis de Arquitecturas Software 55
4.1 Introducción a la Metodología de Análisis de Arquitecturas 55
4.2 Clasificación de los Principales Métodos 55
4.3 Perspectiva de los Principales Métodos de Análisis. 55
4.3.1 Método de Análisis de Arquitectura Basado en Escenarios (SAAM) 55
4.3.2 SAAM Fundado en Escenarios Complejos (SAAMCS) 56
4.3.3 Extendiendo SAAM Mediante la Integración en el Dominio (ESAAMI) 57
4.3.4 Método de Análisis de Arquitectura Software para Evolución y Reusabilidad (SAAMER) 57
4.3.5 Método de Análisis de Compensación de la Arquitectura 57
4.3.6 Reingeniería de la Arquitectura Basada en Escenarios (SBAR) 60
4.3.7 Predicción del Mantenimiento de Software a Nivel de Arquitectura (ALPSM) 60
4.3.8 Modelo de Evaluación de Arquitectura Software (SAEM) 61
5 CAPÍTULO 5 – Introducción al Diseño Software Aeronáutico 62
5.1 Introducción 62
5.2 Procesos de Desarrollo Software 62
5.2.1 Proceso de Requisitos de Software. 63
5.2.2 Proceso de Diseño de Software. 64
5.2.3 Proceso de Codificación de Software. 64
5.2.4 Proceso de Integración. 64
5.3 Trazabilidad del Proceso de Desarrollo Software 64
5.4 Proceso de Pruebas Software 65
5.5 Niveles de Diseño DAL 66
5.5.1 Categorías de las Condiciones de Fallo 66
5.5.2 Definición del Nivel de Software 67
5.5.3 Determinación del Nivel de Software 68
5.6 Consideraciones Arquitectónicas 68
5.6.1 Particionamiento 68
5.6.2 Software Disimilar de Versiones Múltiples 69
5.6.3 Monitorización de Seguridad 69

5.7 Estándares de Calidad de Procesos Software 69
5.7.1 Introducción 69
5.7.2 Visión General de SW-CMM 69
5.7.3 Visión General de la DO-178C/ED-12C 71
5.7.4 Comparación entre SW-CMM 74
6 Arquitecturas Software Aeroespaciales y Aeronáuticas 75
6.1 Introducción a las Arquitecturas Software Aeroespaciales y Aeronáuticas 75
6.2 Tipos de Arquitecturas de Sistemas Software 75
6.2.1 Arquitecturas Federadas 75
6.2.2 Arquitecturas IMA 77
6.2.2.1 Componentes del Sistema IMA 80
6.2.2.1.1 Cabinets 80
6.2.2.1.2 Buses de Datos (ARINC 629, ARINC 429) 82
6.2.2.1.3 Dispositivos Compatibles con ARINC 629 82
6.2.2.1.4 Concentradores de Datos Compatibles con ARINC 629 82
6.2.2.2 Ejemplo de Arquitectura IMA 82
6.3 El concepto de Arquitectura Software IMA 84
6.3.1 Introducción a la Arquitectura Software IMA 84
6.3.2 Elementos Principales de la Arquitectura Software 87
6.3.2.1 Funciones Software 87
6.3.2.2 Interfaces Software 87
6.3.2.2.1 Interfaz APEX 87
6.3.2.2.2 Interfaz COEX 88
6.3.2.3 Software de Aplicación 88
6.3.2.4 Sistema Operativo 89
6.3.3 Análisis Detallado de la Arquitectura Software IMA 89
6.3.3.1 Particionamiento IMA y Descomposición Software 90
6.3.3.2 Descripción Detallada de la Arquitectura Software IMA 92
6.3.3.3 Funcionalidad del Sistema 93
6.3.3.3.1 Hardware 93
6.3.3.3.2 Sistema Operativo: Gestión de las Particiones 93
6.3.3.3.2.1 Definición de Atributos de una Partición 94
6.3.3.3.2.2 Modos de la Partición 94
6.3.3.3.2.3 Planificación de Particiones 95

6.3.3.3.3 Gestión de Procesos 95
6.3.3.3.3.1 Control de Procesos 95
6.3.3.3.3.2 Planificación de Procesos 96
6.3.3.3.4 Gestión de Tiempo 96
6.3.3.3.5 Asignación de Memoria 96
6.3.3.3.6 Comunicación entre Particiones 97
6.3.3.3.7 Comunicación dentro de las Particiones 97
6.3.3.3.8 Flujo de Datos entre Particiones y Mecanismos de Protección 97
6.3.3.3.8.1 Copia Directa a través del Kernel 98
6.3.3.3.8.2 Copia Indirecta a través del Kernel 99
6.3.3.3.8.3 Copia Cero Síncrona 100
6.3.3.3.8.4 Copia Cero Asíncrona 100
6.3.3.4 Monitor de Salud 101
6.3.3.5 Consideraciones acerca de la Configuración 101
6.3.3.6 Algunos Ejemplos de Implementaciones IMA 102
6.3.3.6.1 VxWorks 653 Partition Operating System 102
6.3.3.6.2 LynxOS-178 102
6.3.3.6.3 INTEGRITY-178B Operating System 103
6.3.3.6.4 Arquitectura del Sistema AIR 104
7 CAPÍTULO 7 – Análisis de Arquitecturas Software con ArchE 106
7.1 Introducción 106
7.2 Simulador de Vuelo 106
7.2.1 Introducción a los Simuladores de Vuelo 106
7.2.2 Arquitectura y Diseño de un Simulador de Vuelo 108
7.2.2.1 Casos de Uso y Actores principales 108
7.2.2.2 Funcionalidades del Sistema 116
7.2.2.3 Modelo de Arquitectura del Simulador de Vuelo 117
7.2.2.4 Escenarios de Calidad 117
7.2.3 Análisis de la Arquitectura con ArchE 119
7.2.3.1 Definición de Funciones 119
7.2.3.2 Definición de Responsabilidades 119
7.2.3.3 Definición de Relaciones 121
7.2.3.4 Definición de Escenarios 122
7.2.3.5 Mapeo de Escenarios a Responsabilidades 123

7.2.3.6 Análisis de ArchE 124
7.3 Sistema de Aviónica con Software Empotrado 134
7.3.1 Arquitectura del Sistema de Aviónica 134
7.3.1.1 Arquitectura Hardware 134
7.3.1.2 Arquitectura Software 135
7.3.1.3 Escenarios de Calidad 137
7.3.2 Análisis de la Arquitectura con ArchE 139
7.3.2.1 Definición de Funciones y Responsabilidades 139
7.3.2.2 Definición de Relaciones 141
7.3.2.3 Definición de Escenarios 142
7.3.2.4 Mapeo de Escenarios a Responsabilidades 145
7.3.2.5 Análisis de ArchE 146
8 ArchE Como Asistente de Arquitecturas SW IMA 187
8.1 Introducción. Objetivos 187
8.2 Arquitectura Software Real del Sistema de Aviónica 187
8.3 Definición de los Nuevos Marcos de Razonamiento en ArchE 190
8.3.1 Marco de Razonamiento *Space Partitioning*. 190
8.3.2 Marco de Razonamiento *Time Partitioning* 196
8.4 Análisis de la Arquitectura IMA con Arche 200
9 CAPÍTULO 9 – Conclusiones. Trabajos Futuros 222
9.1 Conclusiones 222
9.2 Trabajos Futuros 223
10 Bibliografía 224
11 Glosario 228
12 ANEXO 1 - Introducción a ArchE 230
12.1 Conceptos Básicos de ArchE 230
12.2 Operación de ArchE 232
12.2.1 Conceptos Claves de ArchE 232
12.2.2 Actividades Básicas de ArchE e Interacciones con el Usuario 233
12.2.2.1 Interacciones Básicas 233
12.2.2.2 Adquirir Requisitos 234
12.2.2.3 Refinar Escenarios 234
12.2.2.4 Elegir Marco de Razonamiento 234
12.2.2.5 Construir Los Modelos de Atributos de Calidad. 234

12.2.2.6 Construir el Diseño 234
12.3 Marcos de Razonamiento en ArchE 235
12.3.1 Modificabilidad 236
12.3.1.1 Especificando Requisitos de Modificabilidad 237
12.3.1.2 Elementos Primarios que Afectan a la Modificabilidad 237
12.3.1.3 Tácticas de Modificabilidad 238
12.3.1.4 Análisis de Modificabilidad 238
12.3.1.5 Implementación del Marco de Razonamiento de Modificabilidad en ArchE 239
12.3.2 Rendimiento 240
12.3.2.1 Especificando Requisitos de Rendimiento 241
12.3.2.2 Elementos Primarios que Afectan al Rendimiento 241
12.3.2.3 Tácticas de Rendimiento 242
12.3.2.4 Análisis de Rendimiento 242
12.3.2.5 Implementación del Marco de Razonamiento de Rendimiento en ArchE 244
12.3.2.6 Implementación en ArchE del Marco de Razonamiento a través de Lambda-WBA 245
12.4 Desarrollar ArchE 246
13 ANEXO 2 – Instalación y Configuración de ArchE 253
13.1 Introducción 253
13.2 Instalación del Java runtime 5.0 254
13.3 Instalación de Eclipse 260
13.4 Instalación de MySQL 263
13.5 Instalación de GEF (Graphical Editing Framework) 274
13.6 Instalación de JESS 277
13.7 Instalación de xmlBlaster 282
13.8 Instalación de ArchE 289
13.9 Método Alternativo de Instalación de ArchE 300
14 ANEXO 3 – Guía de Usuario de ArchE 301
14.1 Guía de Usuario de ArchE 301
14.1.1 Arrancar xmlBlaster 301
14.1.2 Iniciar ArchE 301
14.1.3 Creación de un Proyecto en ArchE 303
14.1.4 Introducir Responsabilidades 305
14.1.5 Asignar Relaciones entre Responsabilidades 307

14.1.6 Introducir Escenarios 310
14.1.7 Mapeo Escenarios-Responsabilidades 312
14.1.8 Análisis de Resultados 315
14.2 Notas y Limitaciones de ArchE 325

14 ANEXO 3 – GUÍA DE USUARIO DE ARCHE

14.1 Guía de Usuario de ArchE

En este apartado se dará una guía rápida para aprender a utilizar ArchE.

Los pasos correctos en la creación de un proyecto son los siguientes:

1.- Arrancar XmlBlaster.
2.- Iniciar ArchE.
3.- Crear un nuevo Proyecto
4.- Introducir Funciones y Responsabilidades
5.- Asignar relaciones entre responsabilidades
6.- Introducir escenarios
7.- Mapear escenarios a responsabilidades
8.- Analizar resultados

14.1.1 Arrancar xmlBlaster

xmlBlaster es necesario para que ArchE pueda comunicarse con los marcos de razonamiento; por tanto, lo primero que hay que hacer es iniciarlo, mediante la línea de comandos: (Figura 303)

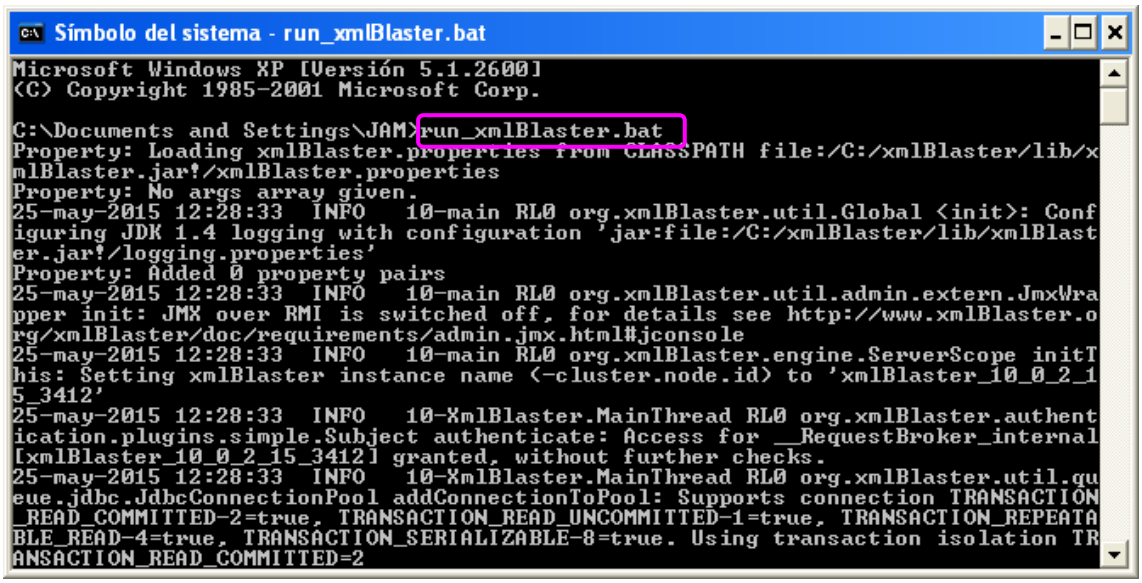


Figura 303. Arrancar XmlBlaster.

Una vez finalice el arranque del fichero run_xmlBlaster. bat, hay que presionar “r”.

14.1.2 Iniciar ArchE

Para iniciar ArchE, se hace doble clic en el icono de ArchE y a continuación el entorno Eclipse se cargará: (Figura 304)

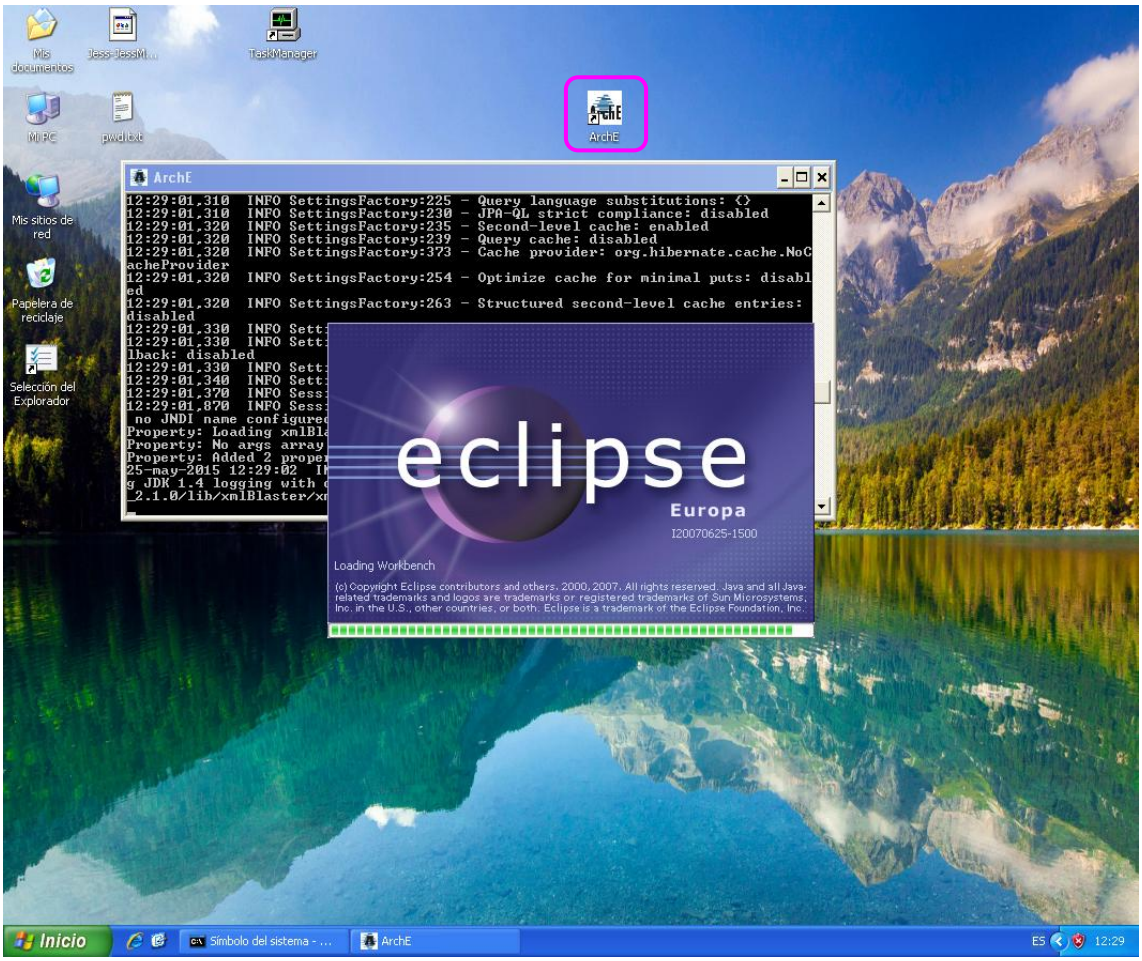


Figura 304. Aranque de ArchE.

El entorno ArchE se carga, como se puede ver en la Figura 305:

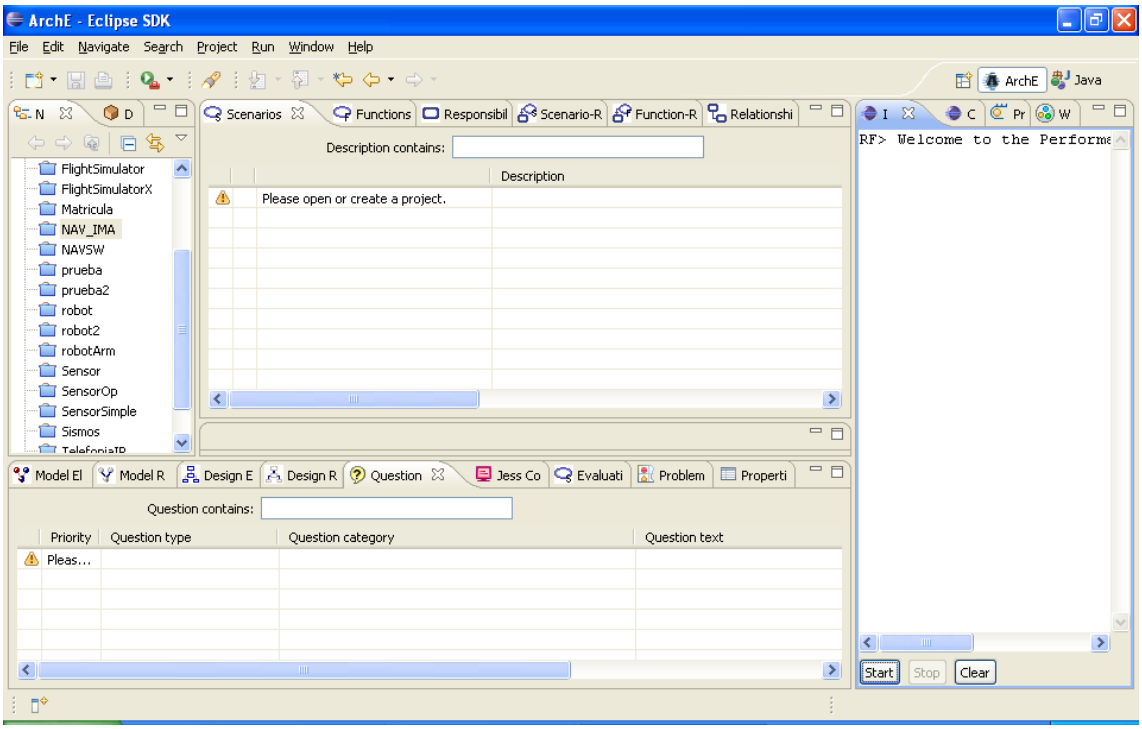


Figura 305. Entorno ArchE.

14.1.3 Creación de un Proyecto en ArchE

Para crear un proyecto, hay que ir a la opción File/New/Arche Project: (Figura 306)

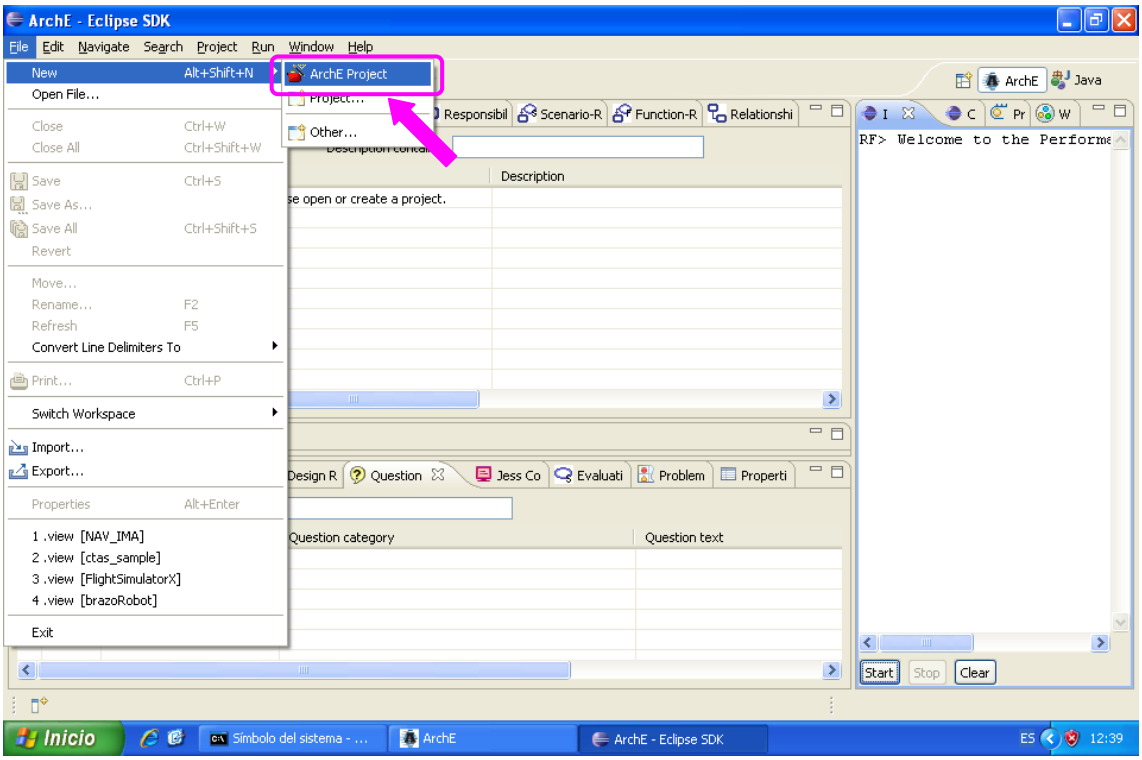


Figura 306. Creación de un Nuevo Proyecto en ArchE.

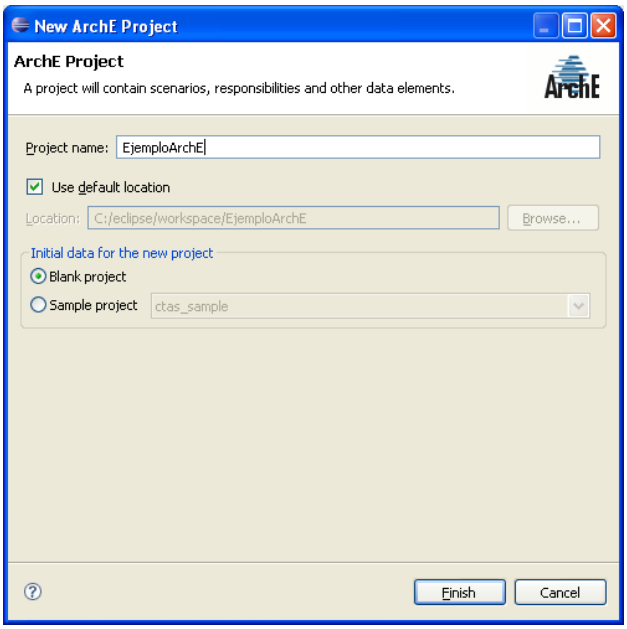


Figura 307. Nuevo Proyecto de ArchE.

Una vez abierto, ya se puede trabajar con el proyecto. Es importantísimo recalcar que todo cambio que se haga en el proyecto no se salvará si no se sigue el siguiente procedimiento: (Figura 308)

Project/Persist fact base

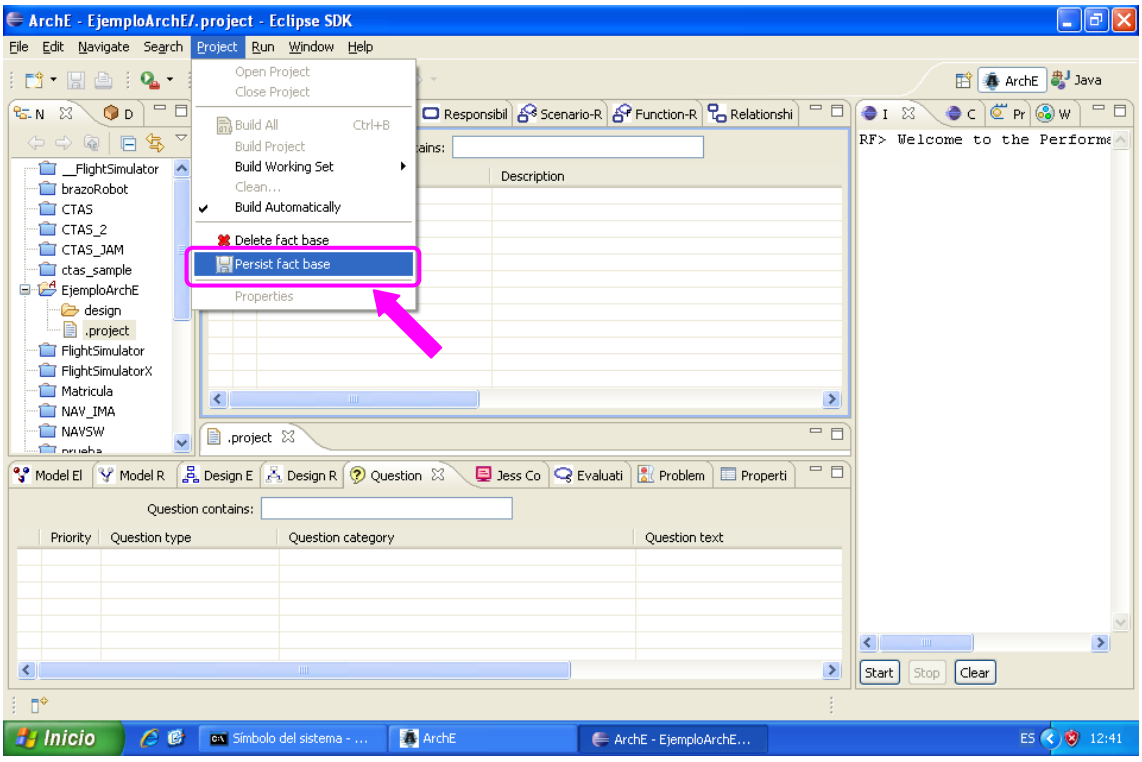


Figura 308. Guardar Cambios en Proyecto ARchE.

14.1.4 Introducir Responsabilidades

Una vez generado el proyecto, a continuación se introducirán sus funciones, haciendo clic en el botón derecho del ratón y seleccionando “New function”: (Figura 309, Figura 310 y Figura 311)

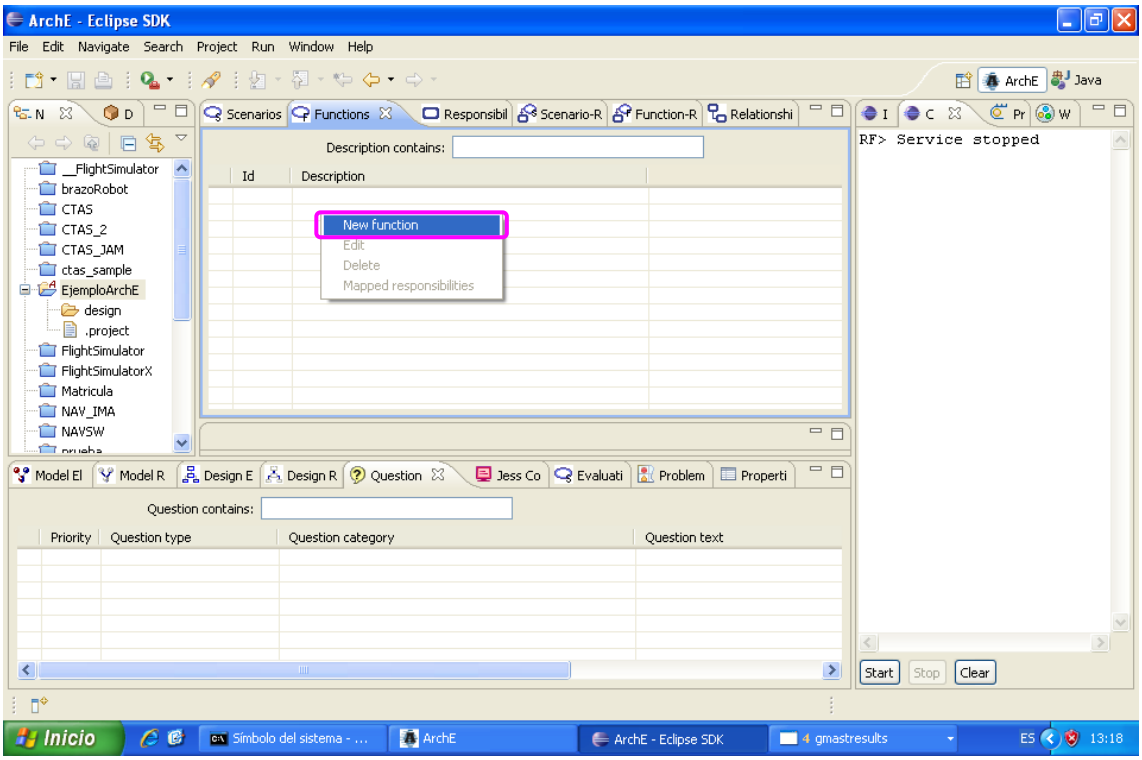


Figura 309. Introducir Funciones en Proyecto ArchE.

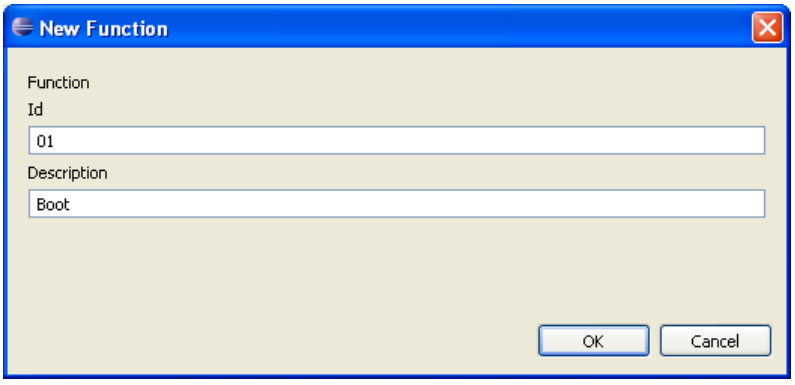


Figura 310. Nueva Función.

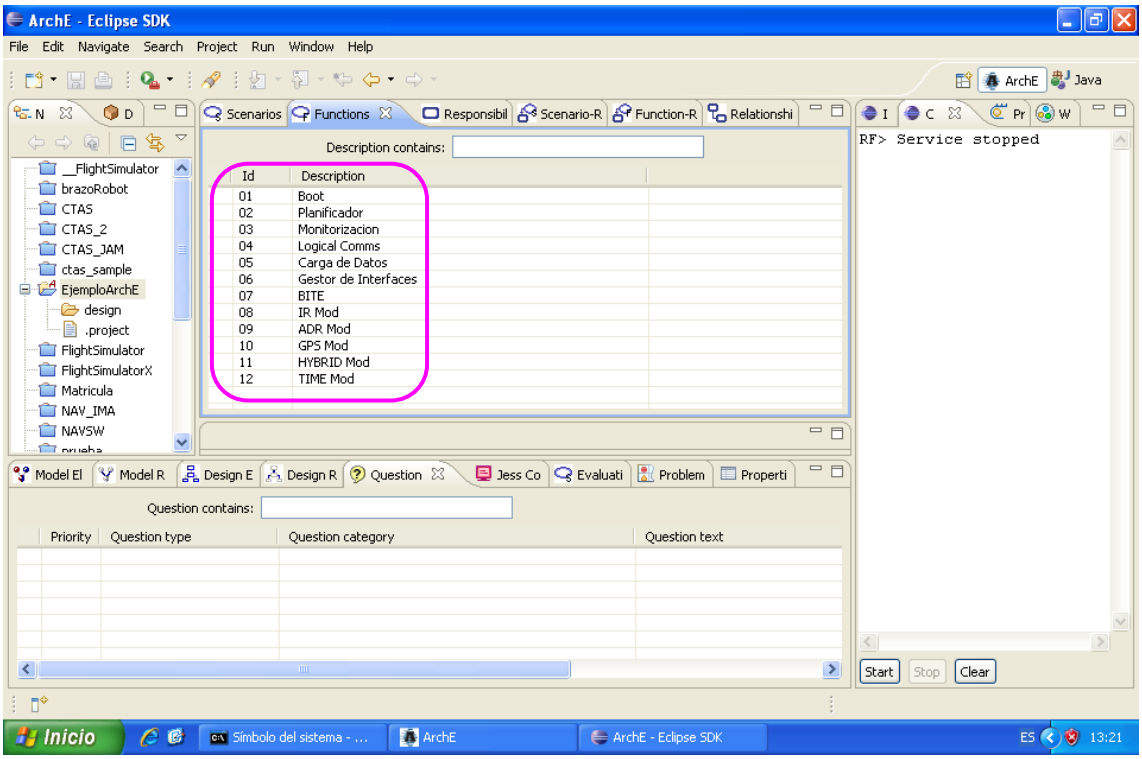


Figura 311. Las Funciones están introducidas en ArchE.

Las responsabilidades son asignadas de manera inmediata por ArchE a su respectiva funcionalidad. El arquitecto puede después modificar o asignar nuevas responsabilidades. (Figura 312 y Figura 313)

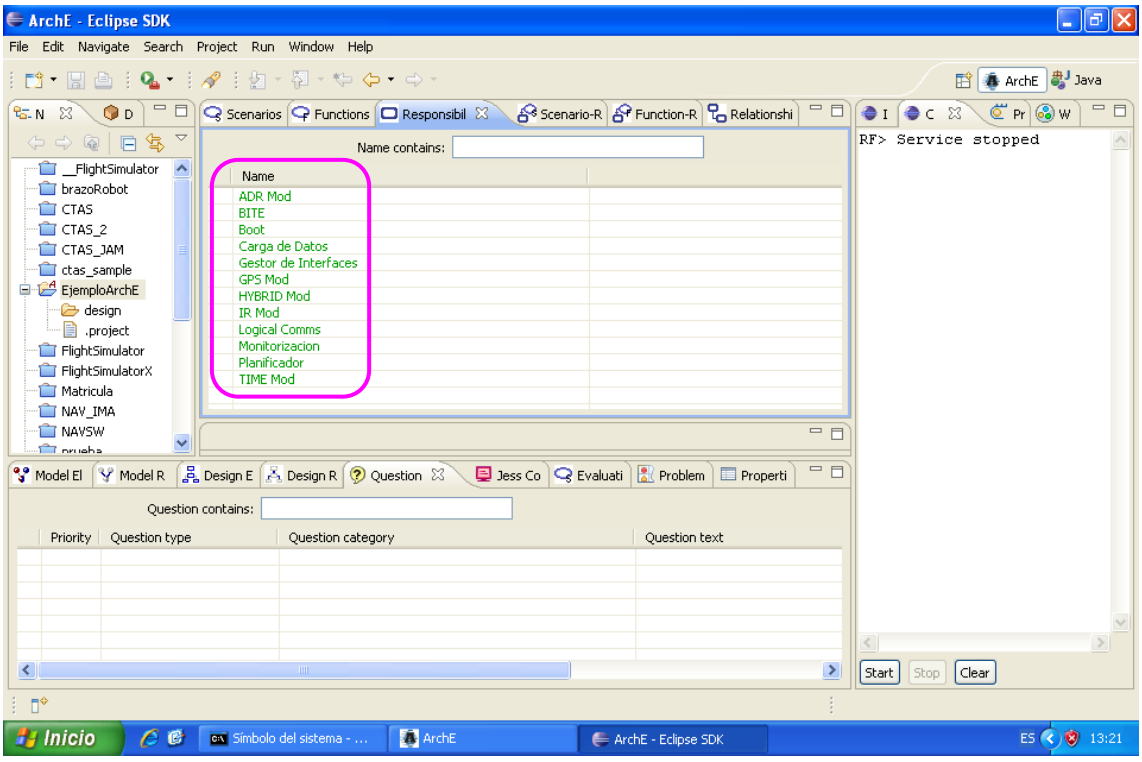


Figura 312. Introducción Automática de Responsabilidades por ArchE.

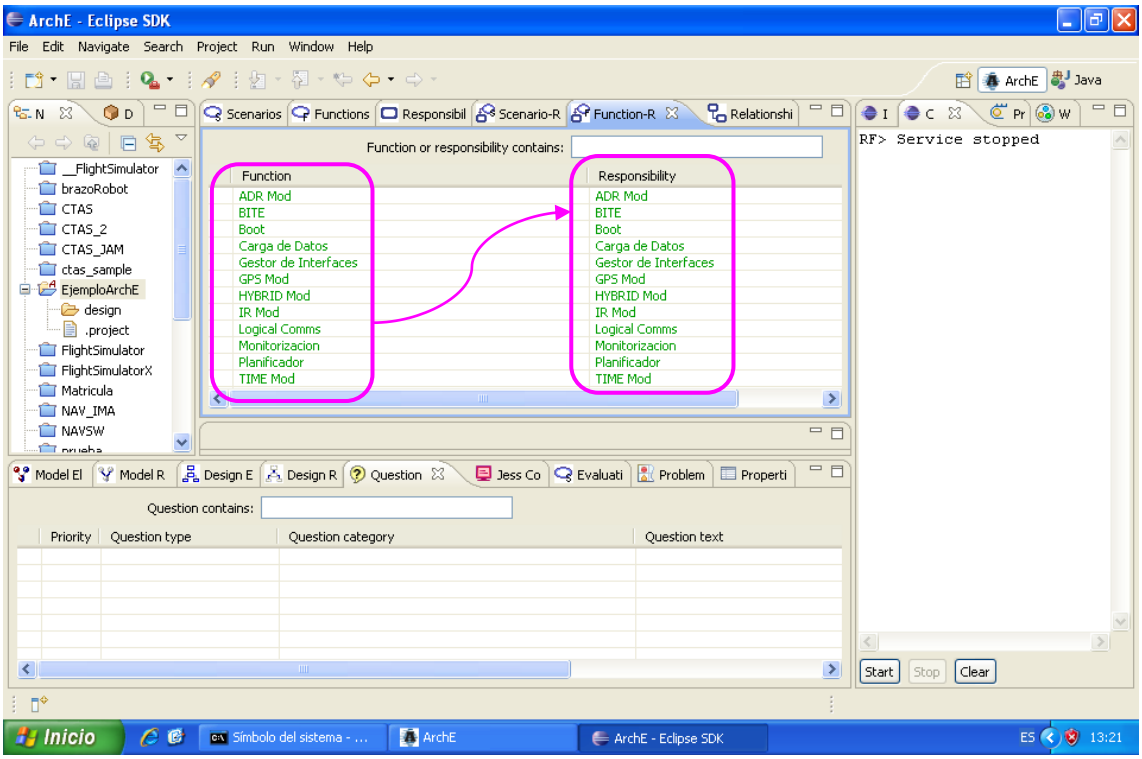


Figura 313. Mapeo Funciones-Responsabilidades en ArchE.

14.1.5 Asignar Relaciones entre Responsabilidades

En este paso, se establecerán las relaciones entre responsabilidades. Es muy importante arrancar ahora los dos marcos de razonamiento, modifiability y performance, puesto que si no, no se podrán establecer dichas relaciones: (Figura 314)

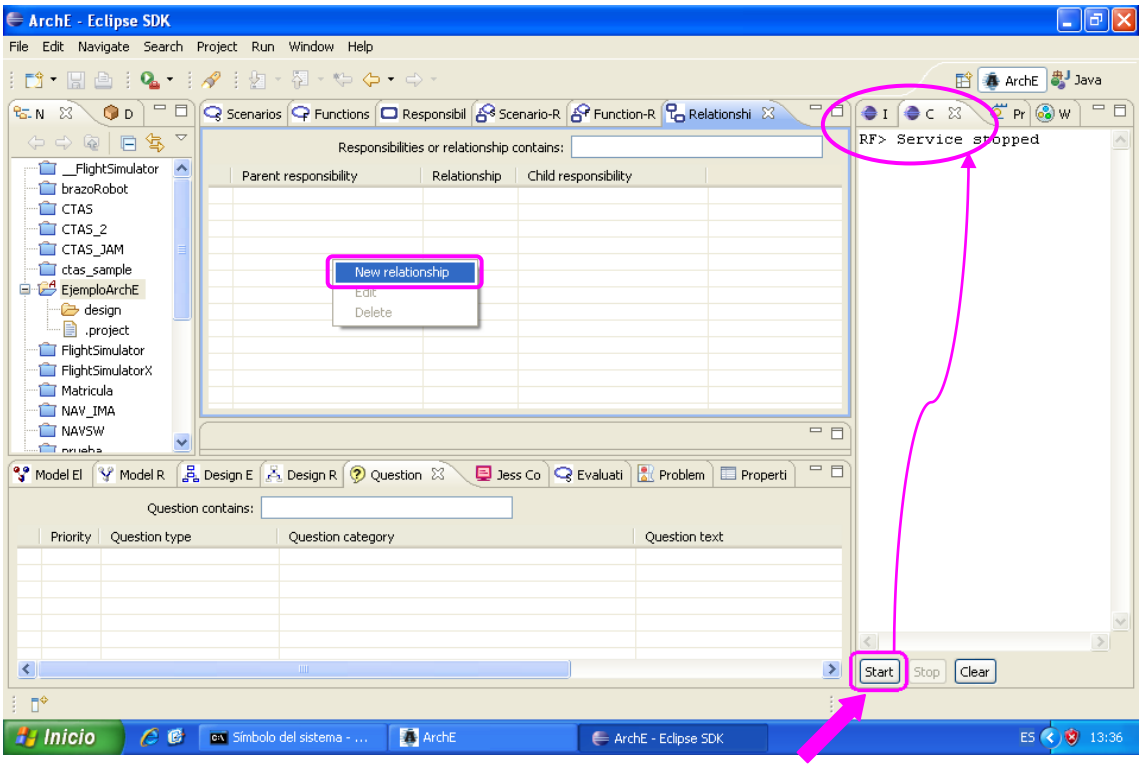


Figura 314. Establecer una nueva Relación entre Responsabilidades en ArchE.

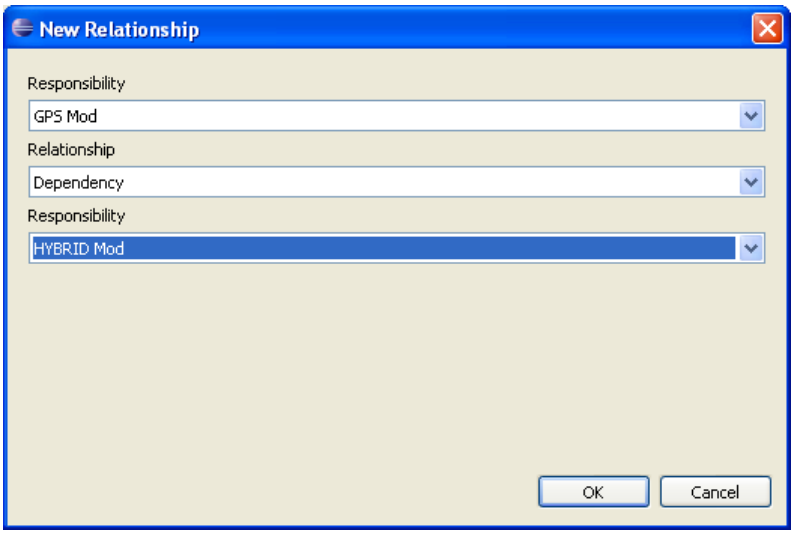


Figura 315. Selección de Responsabilidades y Tipo de Relación entre Ellas.

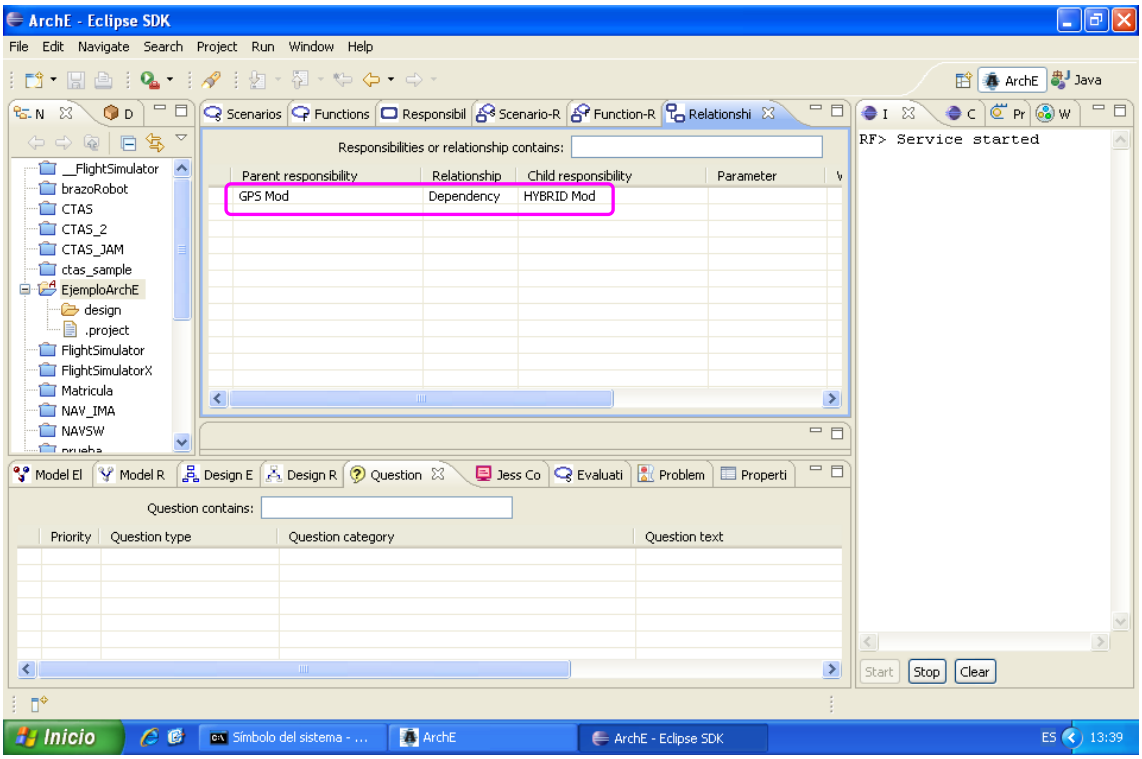


Figura 316. La Relación entre Responsabilidades queda creada.

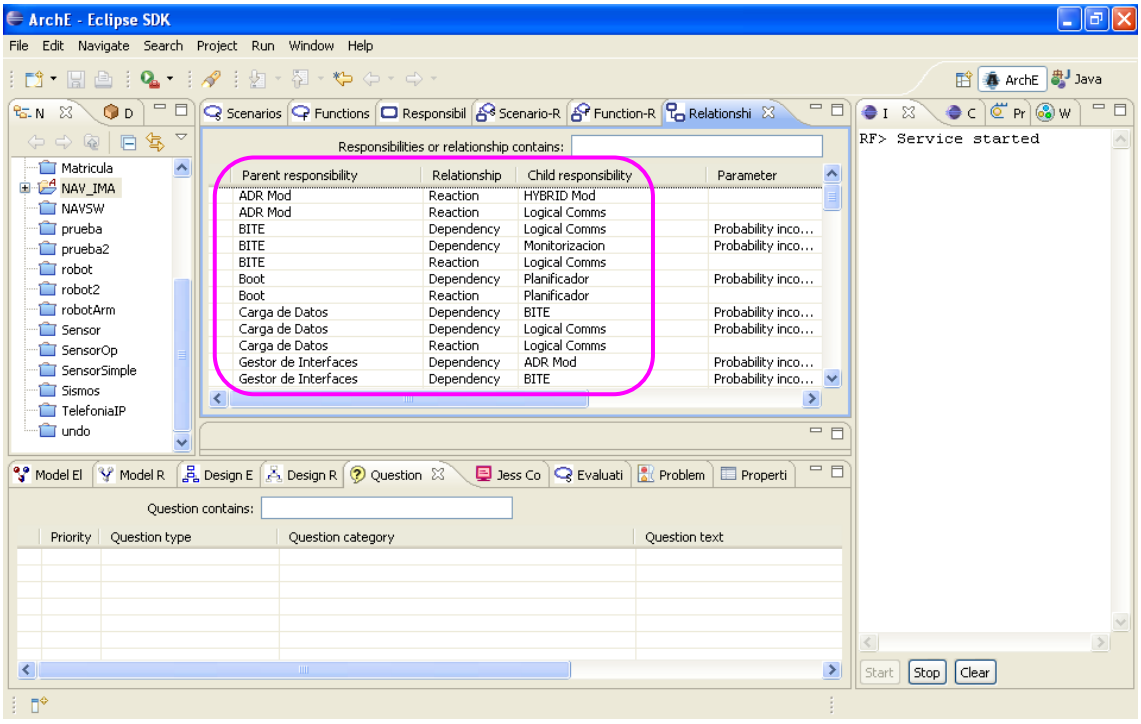


Figura 317. Conjunto de Relaciones en el Proyecto ArchE.

14.1.6 Introducir Escenarios

El paso siguiente es introducir los escenarios de atributos de calidad; para ello, los marcos de razonamiento han de seguir arrancados: (Figura 318)

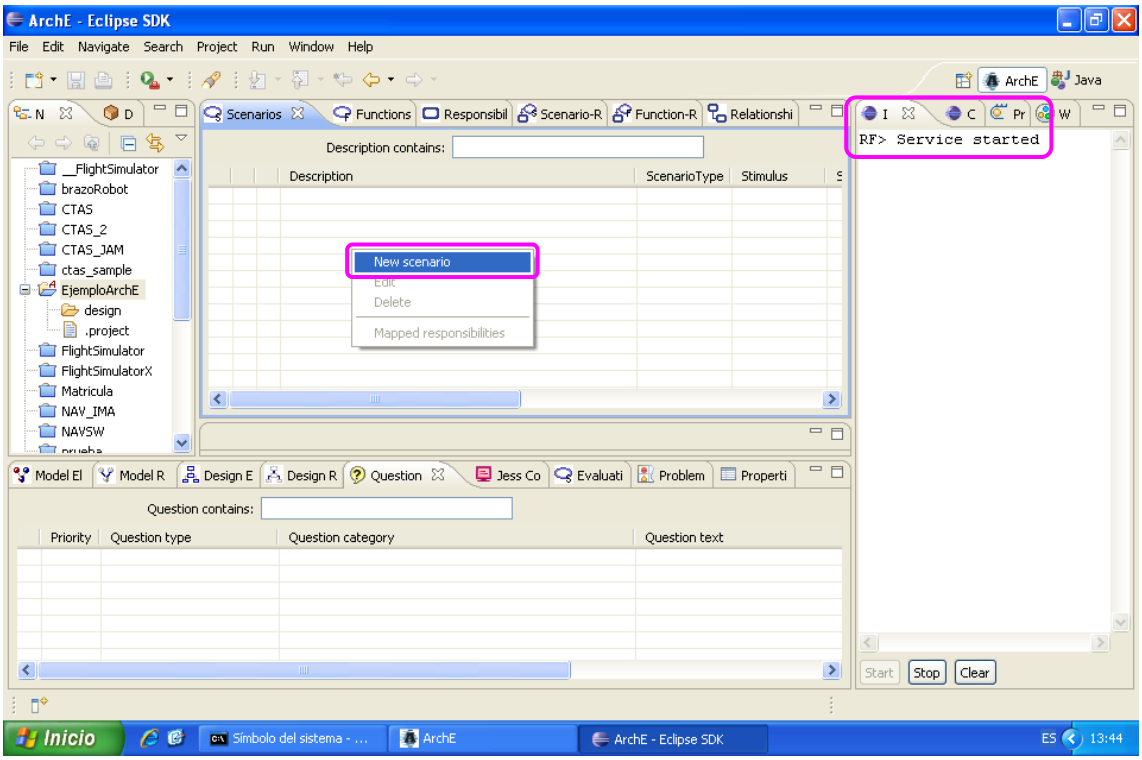


Figura 318. Introducir Nuevo Escenario en ARchE.

Se abre una ventana en la cual se introducen los parámetros del escenario, así como el tipo de marco de razonamiento: (Figura 319 y Figura 320)

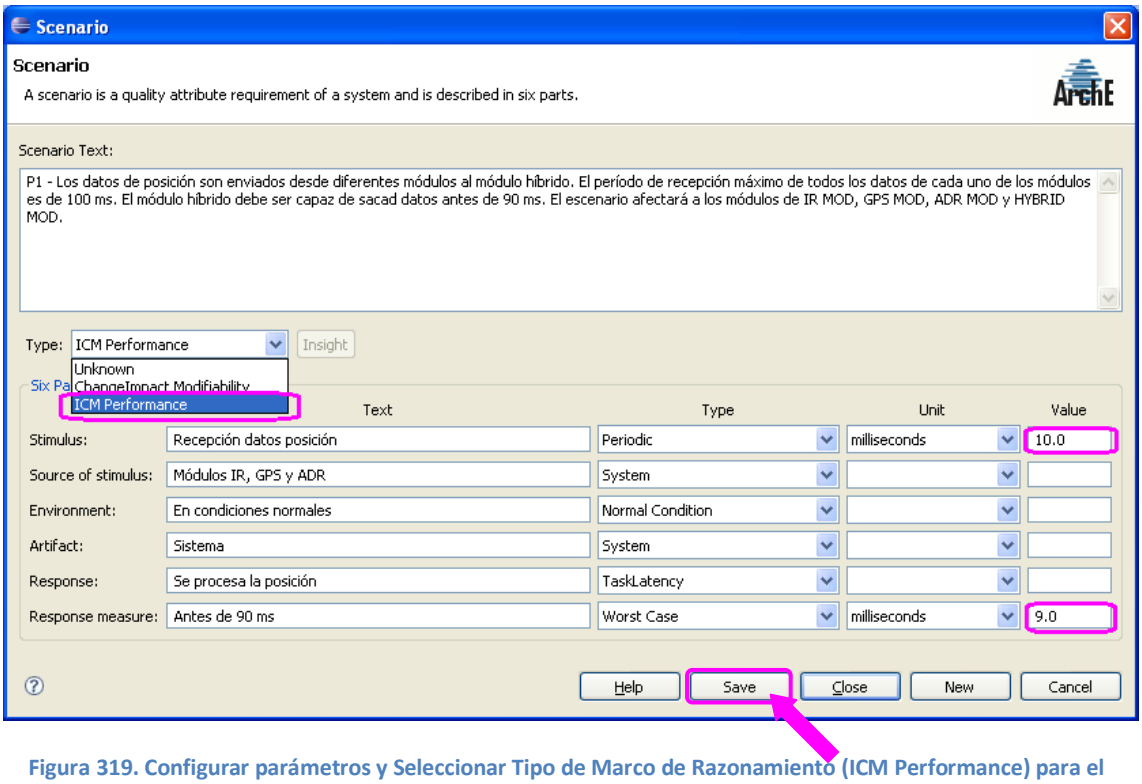


Figura 319. Configurar parámetros y Seleccionar Tipo de Marco de Razonamiento (ICM Performance) para el Escenario.

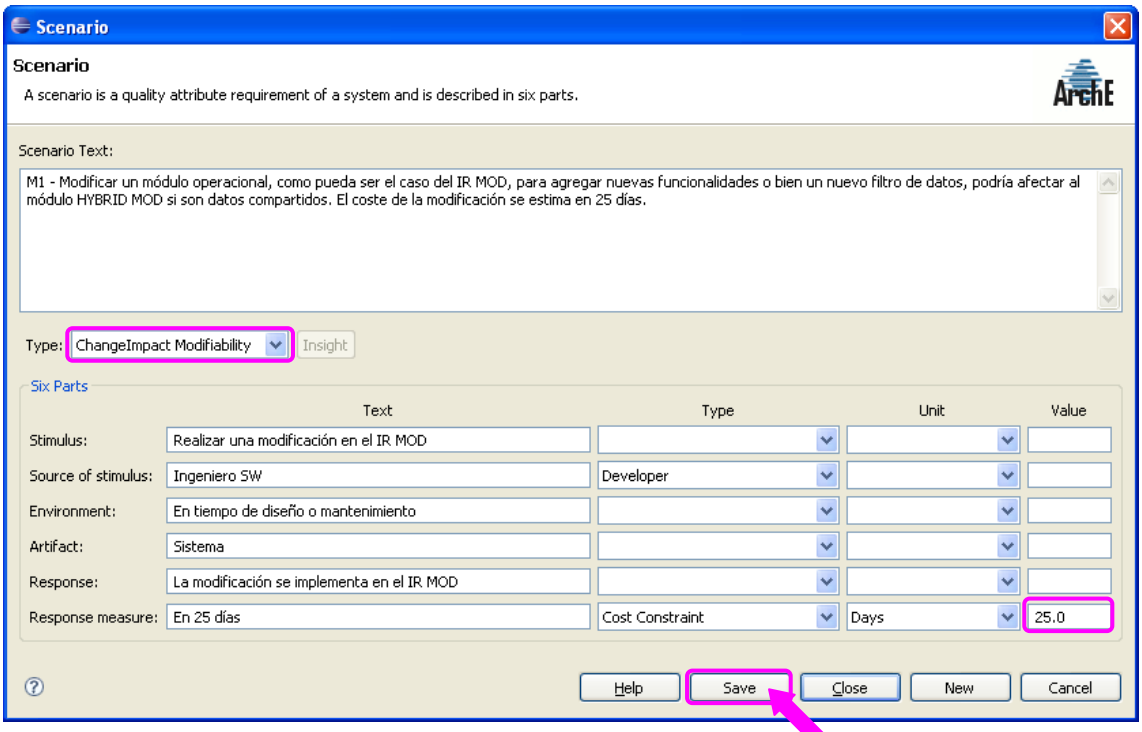


Figura 320. Configurar parámetros y Seleccionar Tipo de Marco de Razonamiento (ChangeImpact Modifiability) para el Escenario.

Una vez introducidos todos los escenarios, la vista de escenarios de ArchE quedaría de la siguiente manera: (Figura 321)

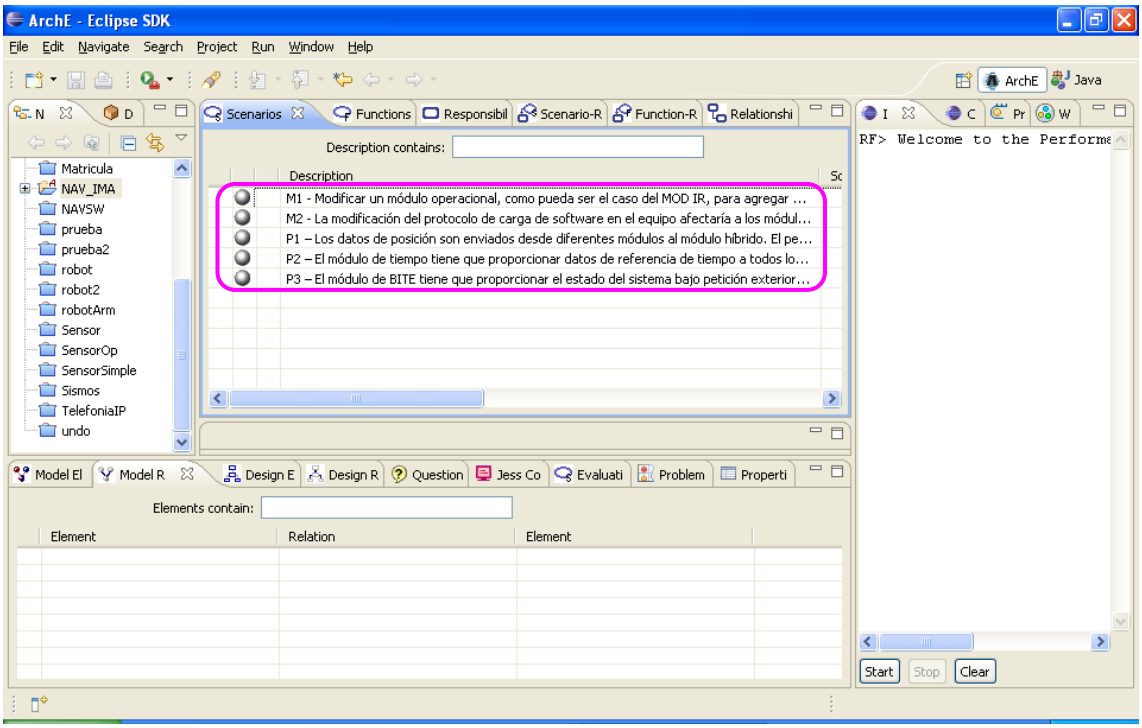


Figura 321. Escenarios de Atributos de Calidad en el Proyecto ArchE.

14.1.7 Mapeo Escenarios-Responsabilidades

Por último, se realiza el mapeo de escenarios a responsabilidades, haciendo como siempre clic derecho del ratón y seleccionando New Mapping en la pestaña de Scenario-Responsibility Mapping. Esto hará que los marcos de razonamiento (que permanecerán arrancados de pasos anteriores) empiecen a ejecutarse, lo que ralentizará el proceso, pues al final de dichas ejecuciones, ArchE presentará los resultados: (Figura 322)

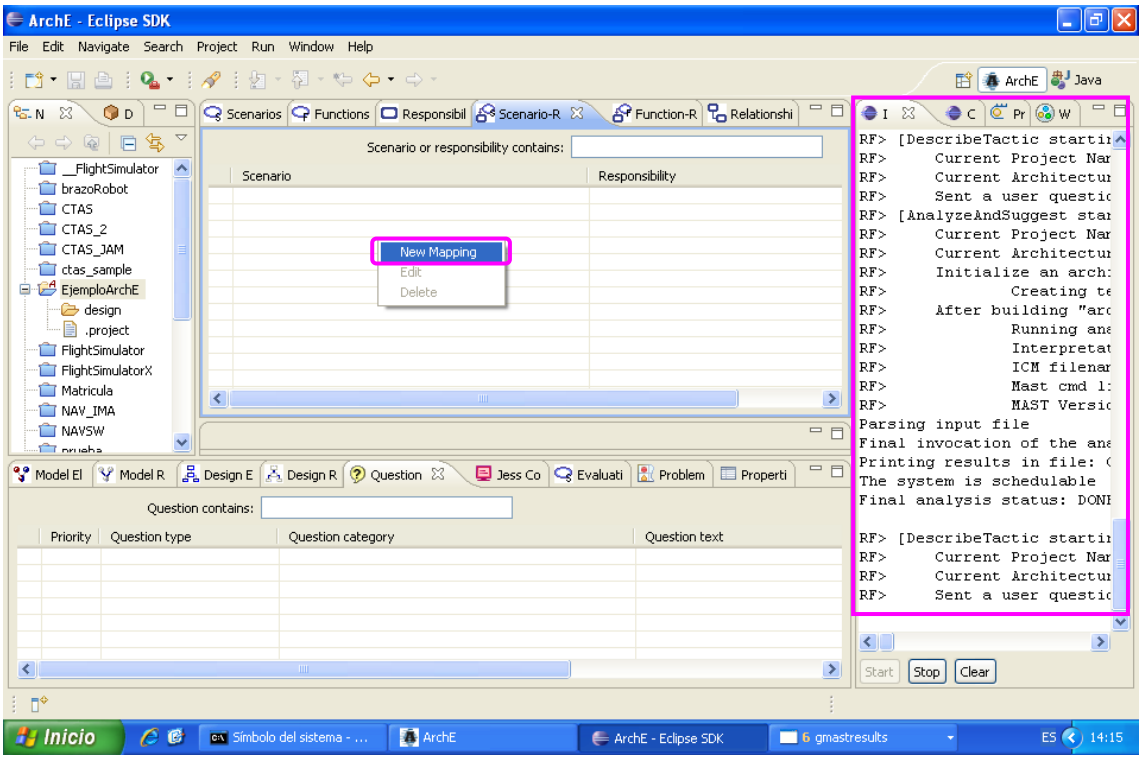


Figura 322. Mapeo Escenario-Responsabilidades.

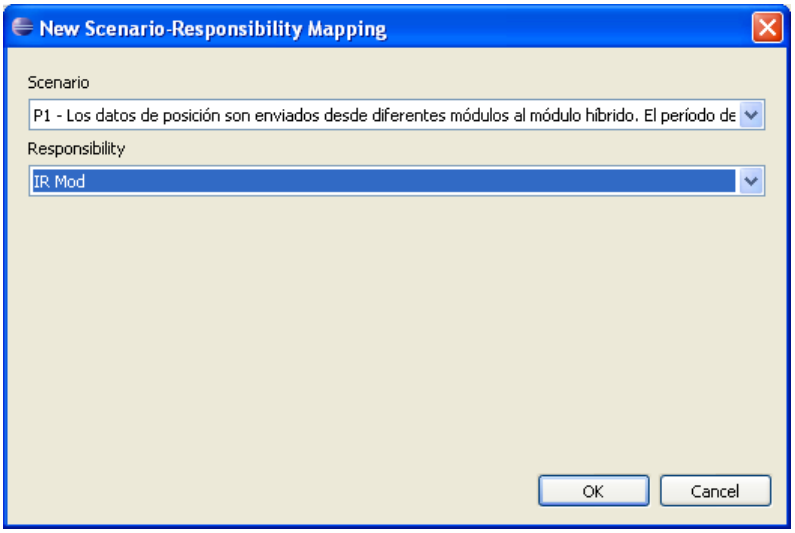


Figura 323. Se abre un Menú de Selección de Escenario y Responsabilidad Asociada.

El mapeo queda registrado en ArchE: (Figura 324)

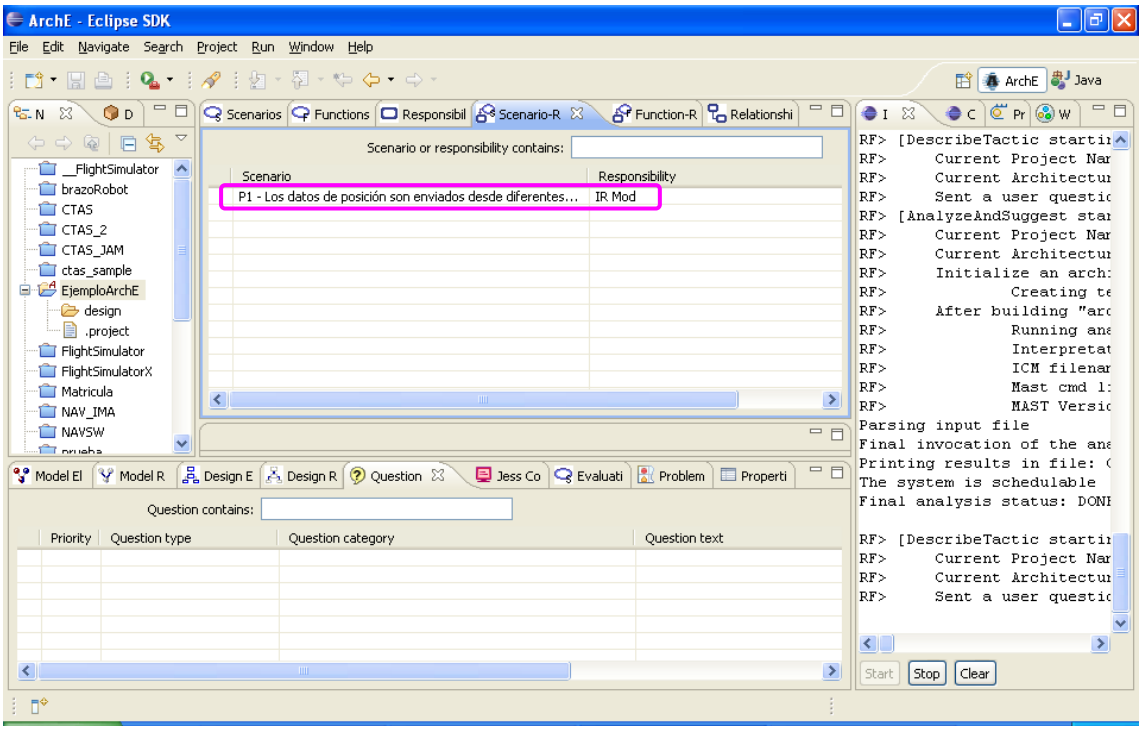


Figura 324. Mapeo Escenario-Responsabilidad ha sido creado.

Al final del proceso, todos los escenarios han de quedar mapeados a sus respectivas responsabilidades: (Figura 325)

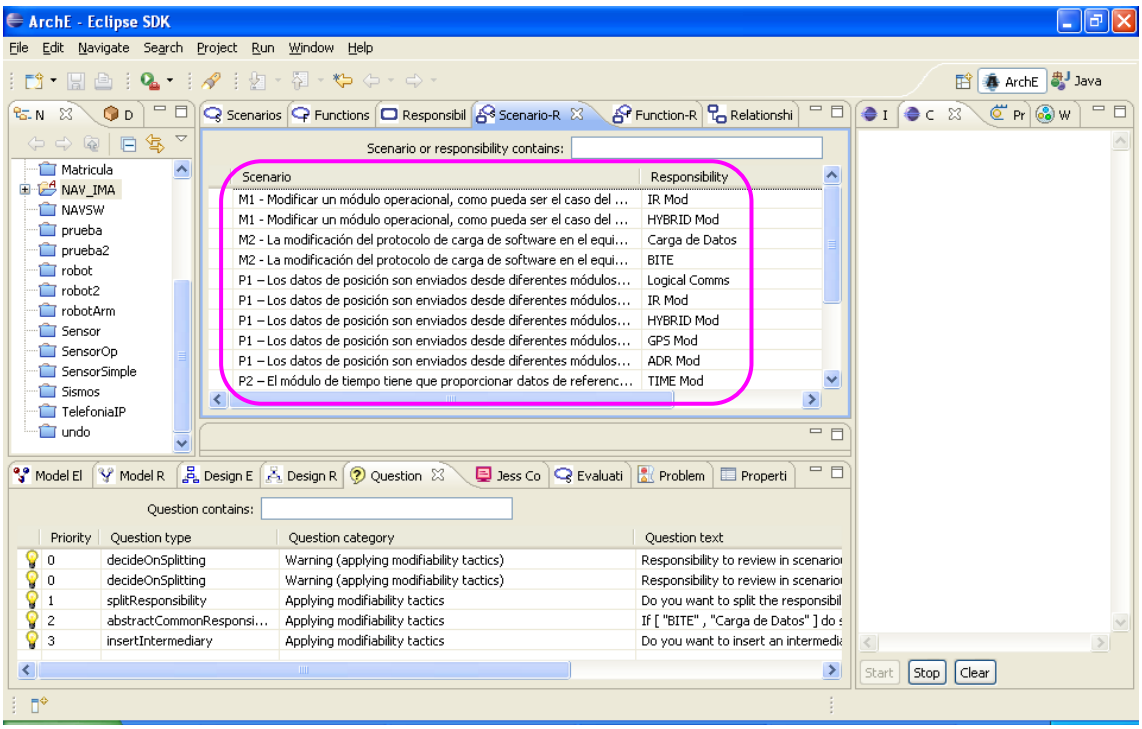


Figura 325. Mapeo Final Escenarios-Responsabilidades.

Para el resto de la guía, se utilizará el proyecto NAV_IMA desarrollado durante el trabajo.

14.1.8 Análisis de Resultados

En la pantalla inicial de escenarios pueden verse los resultados obtenidos en los escenarios, una vez los marcos de razonamiento han sido ejecutados: (Figura 326)

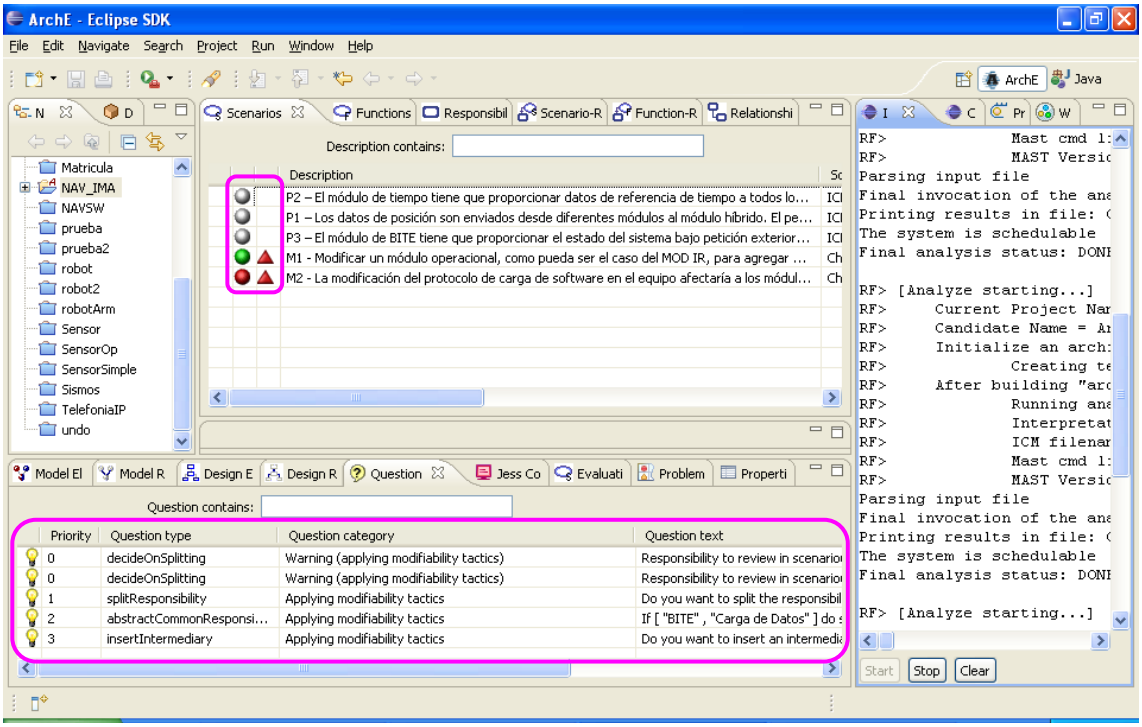


Figura 326. La Vista Escenarios muestra el Estado de Cumplimiento de los Escenarios con los Atributos de Calidad.

Los resultados se interpretan de la siguiente forma:

- Un círculo rojo significa que el escenario no se ha cumplido
- Un círculo verde significa que el escenario se cumple
- Un triángulo verde significa que la última modificación trajo consigo una mejora positiva del escenario.
- Un triángulo ámbar significa que la última modificación fue indiferente y no supuso ni mejora ni empeoramiento del escenario.
- Un triángulo verde significa que la última modificación trajo consigo un empeoramiento del escenario.

Si se han definido escenarios de modificabilidad y además se han asociado a responsabilidades, al pulsar sobre “Design Tree View” se obtiene una vista UML de la arquitectura propuesta por ARchE: (Figura 327)

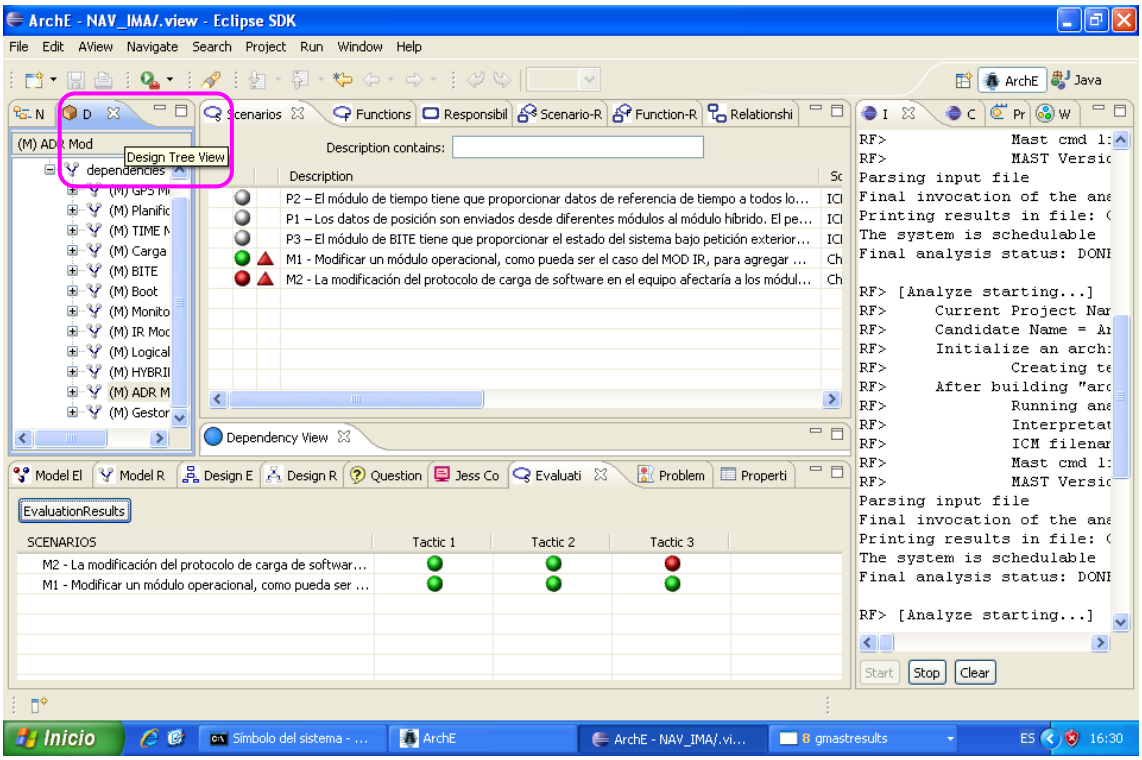


Figura 327. Selección de la Vista Arbol de Diseño en ArchE.

El diagrama UML de la arquitectura propuesto por ArchE e muestra: (Figura 328)

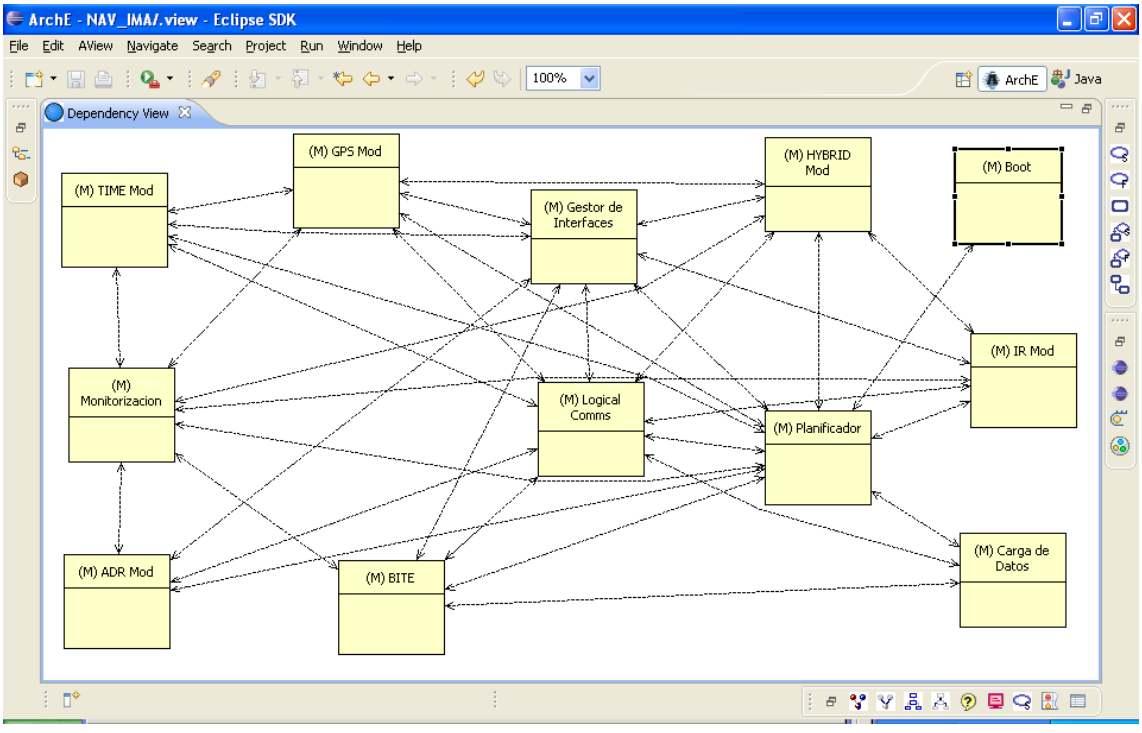


Figura 328. Arquitectura Propuesta por ArchE – Vista Design Tree.

En la pestaña “Responsibilities” pueden modificarse los parámetros de las responsabilidades, en este caso el coste del cambio y el tiempo de ejecución: (Figura 329)

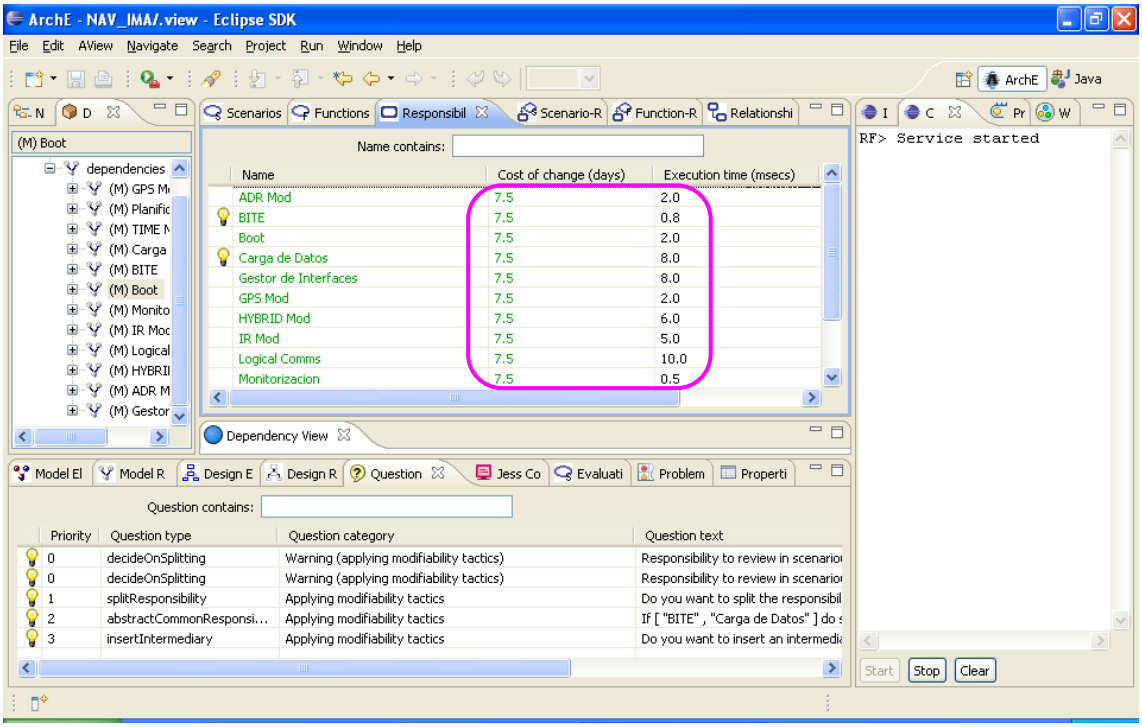


Figura 329. Vista Responsabilidades – Parámetros Coste del Cambio y Tiempo de Ejecución.

En la ventana “Relationships” pueden modificarse las probabilidades de propagación del cambio, tanto de entrada como de salida, de la responsabilidad correspondiente: (Figura 330)

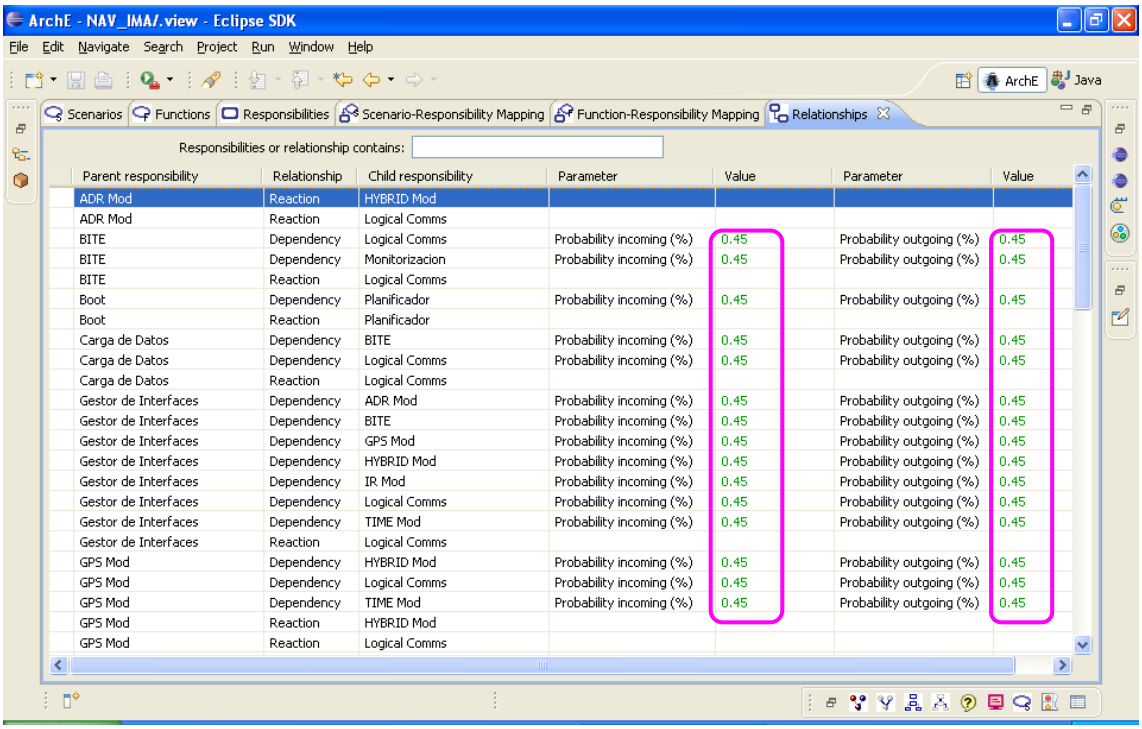


Figura 330. Vista Relaciones – la Probabilidad de Propagación del Cambio puede modificarse.

La ventana “Evaluation” propone las tácticas que pueden ser mejoras a la arquitectura; por ejemplo, en la Figura 331 las tácticas 1 y 2 mejorarían todos los escenarios, mientras que la táctica 3 mejoraría el escenario 1 y empeoraría el 2:

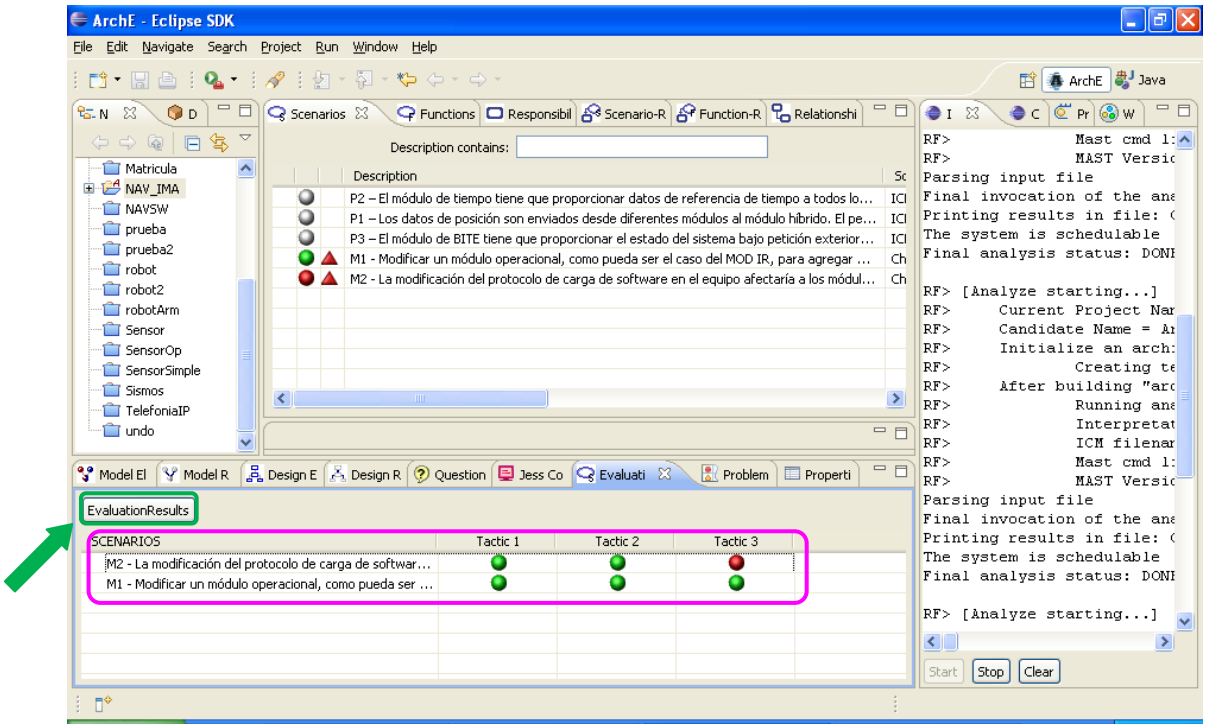


Figura 331. Vista Evaluations con las Posibles Tácticas a Aplicar y su Posible Resultado sobre el Escenario..

En general, el código de colores de los círculos tiene el siguiente significado:

- Un círculo verde ● significa que, aplicando la táctica correspondiente, el escenario se va a cumplir, debido a que la táctica mejorará las condiciones de dicho escenario.
- Un círculo ámbar ● significa que la aplicación de dicha táctica no afectará al cumplimiento del escenario (ni mejora ni degrada las condiciones del escenario).
- Un círculo rojo ● significa que, aplicando la táctica correspondiente, el escenario no se va a cumplir, debido a que la táctica degradará las condiciones de dicho escenario.

Pulsando el botón “EvaluationResults”, ArchE nos muestra otro formato de evaluación de la lista de escenarios, esta vez con triángulos cuyo código de colores tiene el siguiente significado:

- Un triángulo verde ▲ significa que la aplicación de la táctica produjo una mejora en el escenario particular, acercándolo a su umbral de cumplimiento.
- Un triángulo ámbar ▲ significa que la aplicación de la táctica no afectó al escenario particular.
- Un triángulo rojo ▲ significa que la aplicación de la táctica produjo una degradación en el escenario particular, alejándolo de su umbral de cumplimiento.

Si se pulsa en el botón EvaluationResults, se obtiene lo siguiente: (Figura 332)

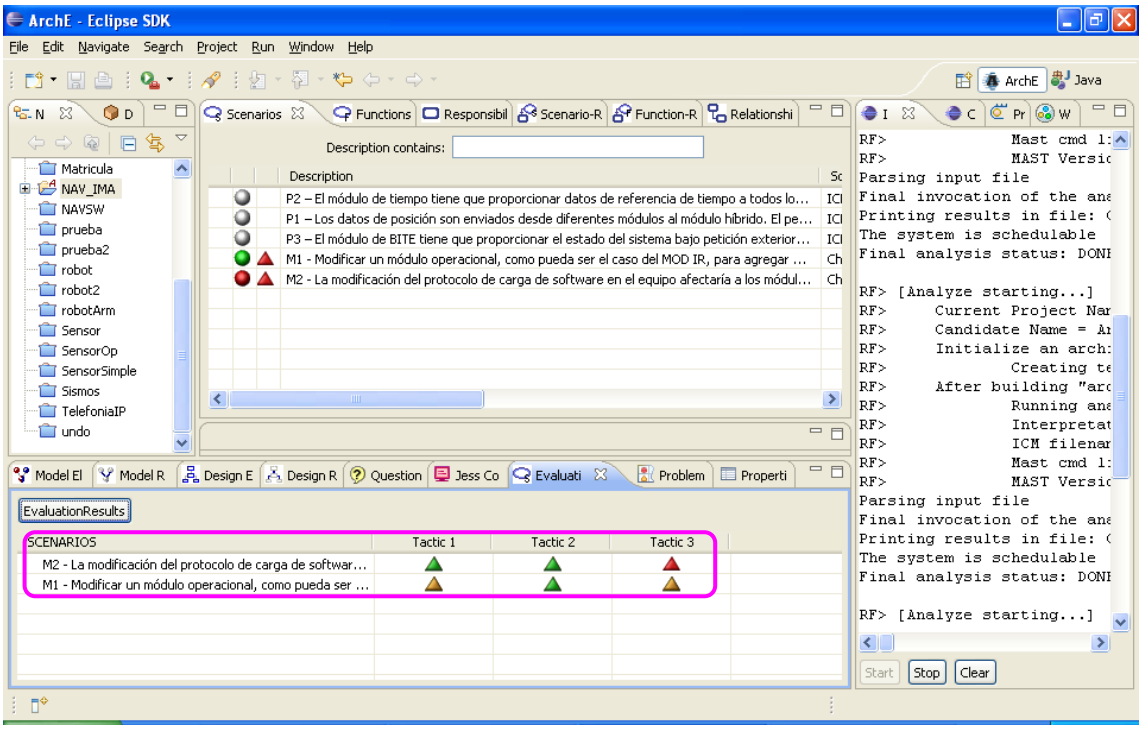


Figura 332. Vista Evaluations con las Posibles Tácticas a Aplicar y su Posible Mejora sobre el Escenario.

Se puede observar el resultado previsto de la aplicación de las tácticas sobre los escenarios, con el criterio descrito anteriormente.

En la ventana “Questions” se pueden ver las tácticas que propone ArchE. (Figura 333)

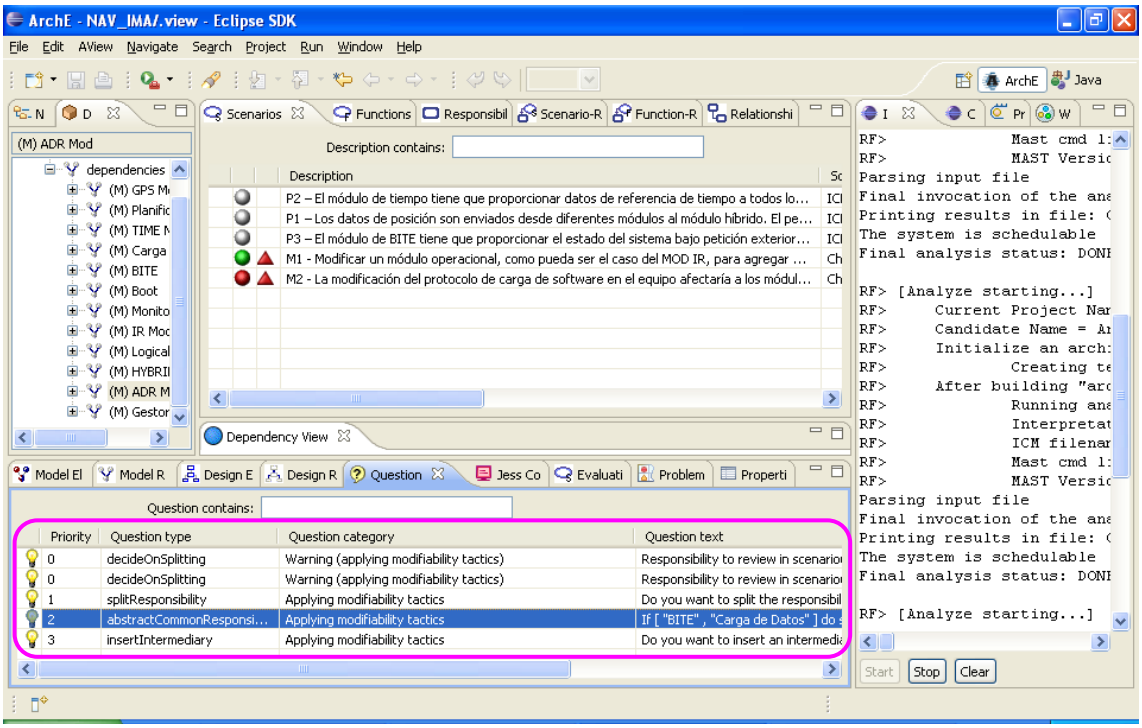


Figura 333. Ventana Questions con las Tácticas Sugeridas por ArchE.

Si se abre una de ellas, por ejemplo la 2: (Figura 334)

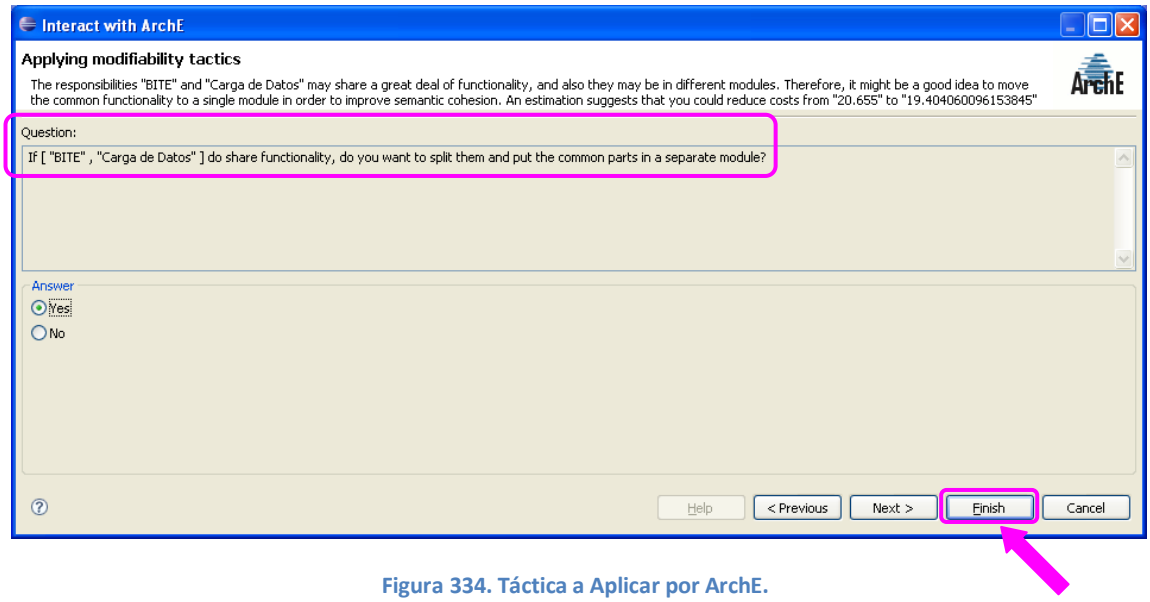


Figura 334. Táctica a Aplicar por ArchE.

Se pulsa sobre Finish y ArchE aplica cambios: (Figura 335)

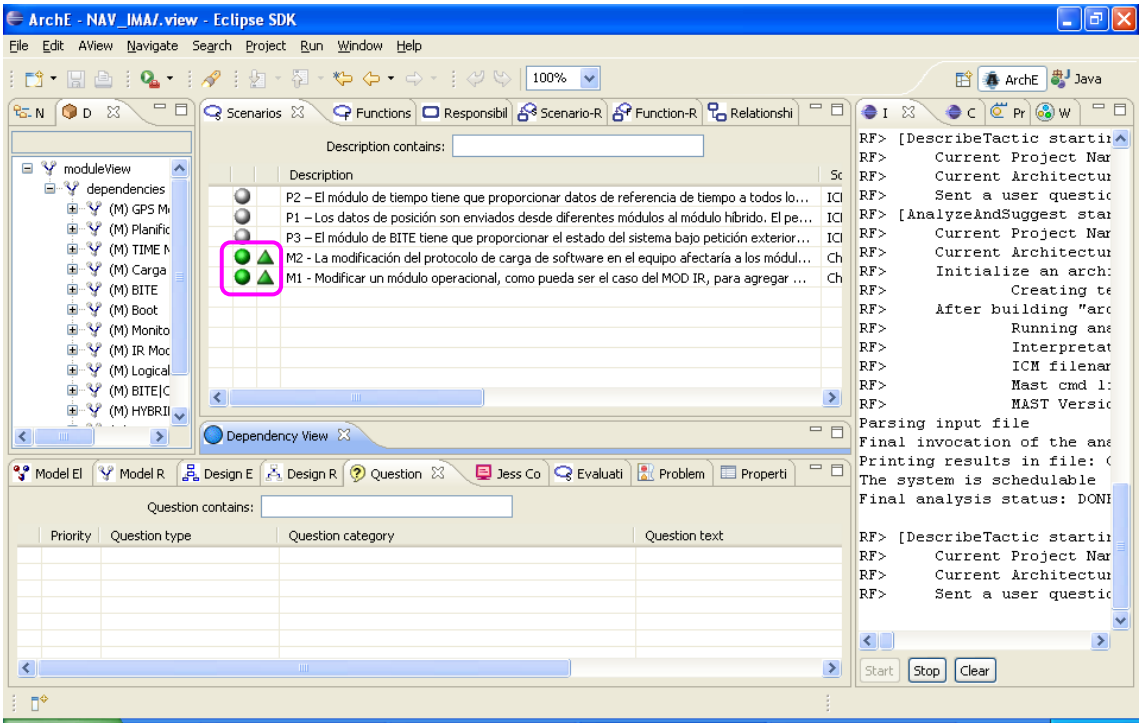


Figura 335. Resultados de Aplicar la Táctica de Arche sobre los Escenarios.

Se puede ver que ahora se cumplen los escenarios y por tanto han desaparecido las tácticas, puesto que ya no son necesarios.

La modificación introducida en el escenario de modificabilidad ha producido el siguiente cambio en la arquitectura: (Figura 336)

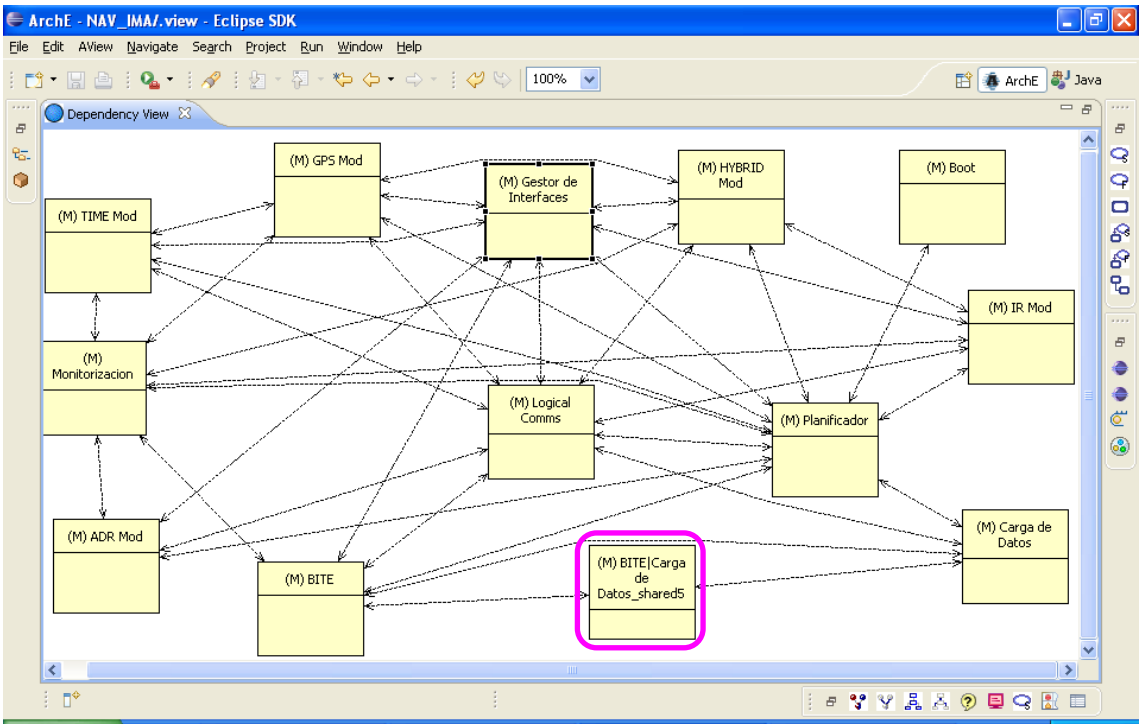


Figura 336. Nueva Vista de Árboles de Diseño con una nueva Responsabilidad creada por la Táctica.

Se observa como una nueva responsabilidad ha sido añadida, para eliminar desacoplamiento entre las antiguas responsabilidades.

En cuanto a rendimiento, se observa la ventana de MAST que indica que el sistema es planificable: (Figura 337)

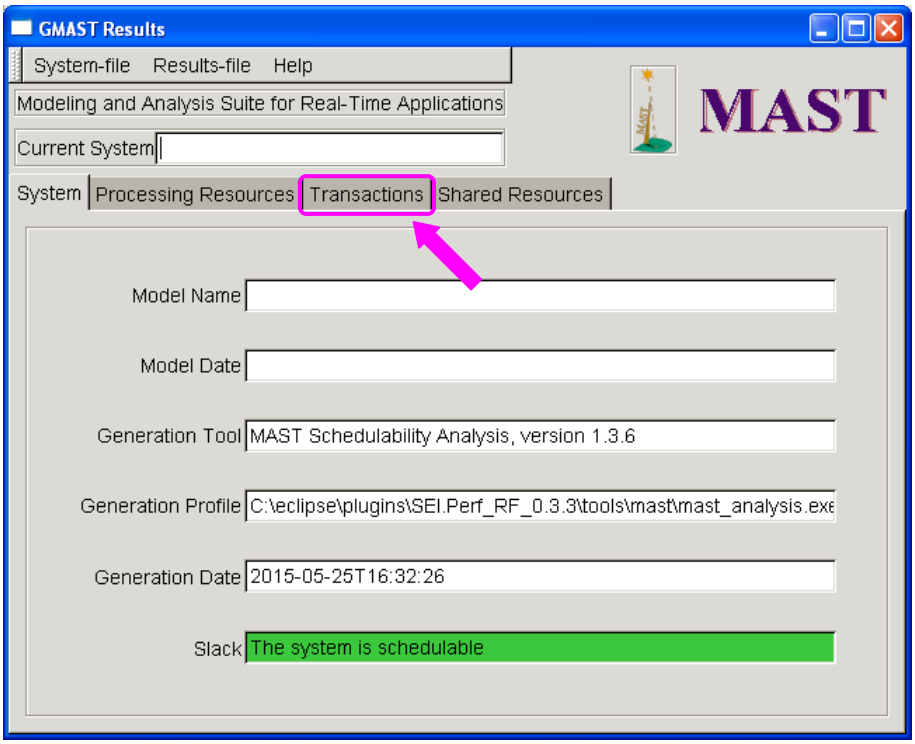


Figura 337. Ventana MAST con el Resultado de Evaluación del Rendimiento del Sistema

Pulsando sobre Transactions: (Figura 338)

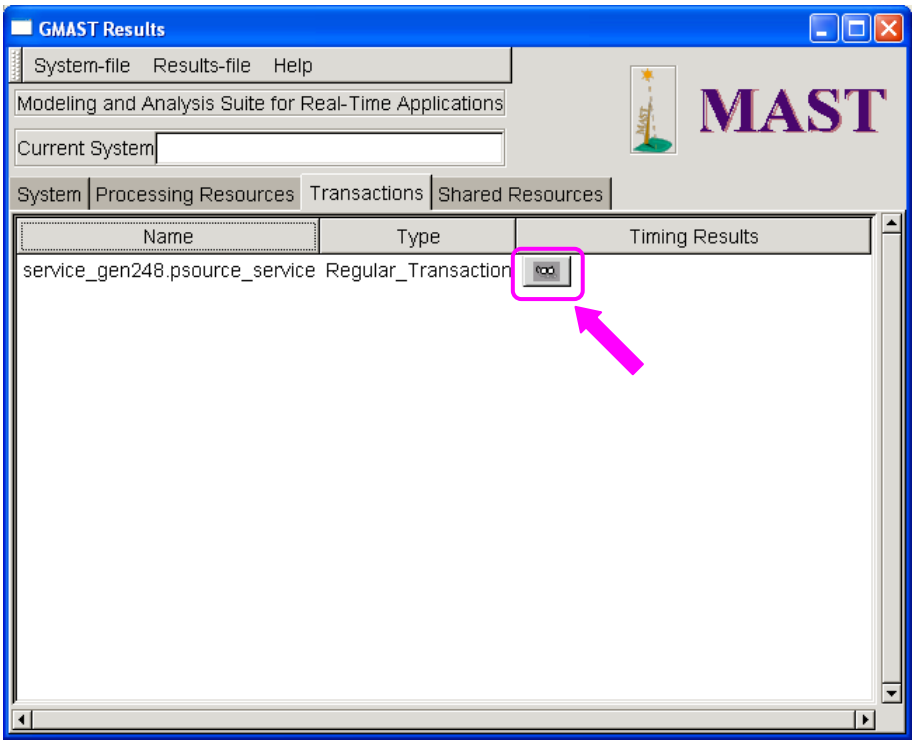


Figura 338. En la Pestaña Transactions se ofrece más Información.

Se pulsa sobre el icono de las gafas y a continuación pulsando sobre "View All" se observan todas las transacciones y que se cumplen las deadlines: (Figura 339)

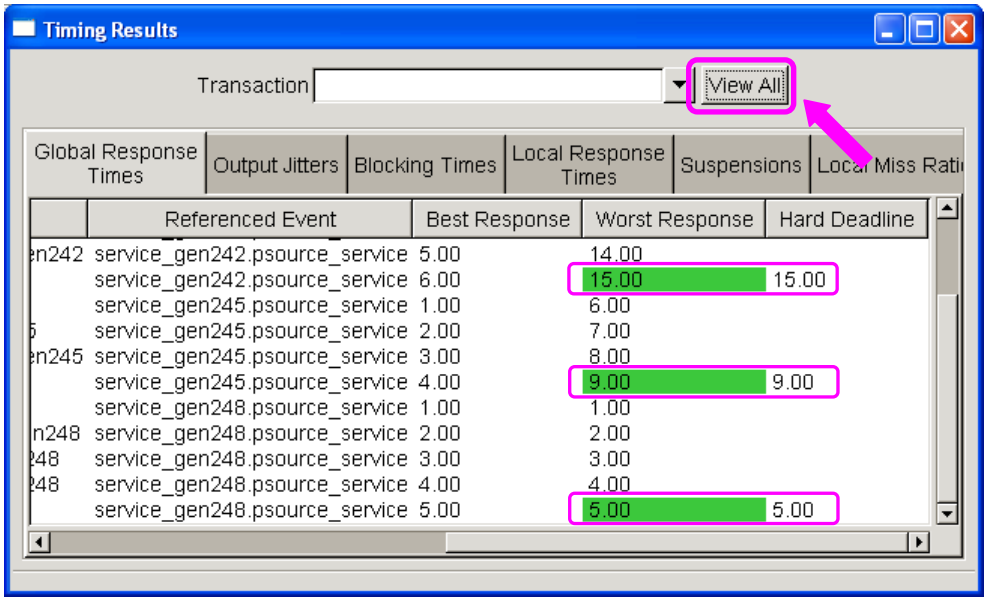


Figura 339. Vista MAST de los Escenarrios Cumpliendo sus Deadlines.

Si el sistema en su conjunto no fuera planificable, se obtendría el siguiente aviso: (Figura 340)

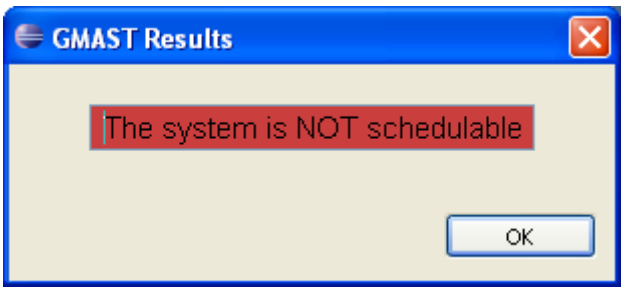


Figura 340. Aviso de Sistema No Planificable.

Si parte del sistema no fuera planificable en alguno de sus escenarios, se obtendría la siguiente ventana MAST: (Figura 341)

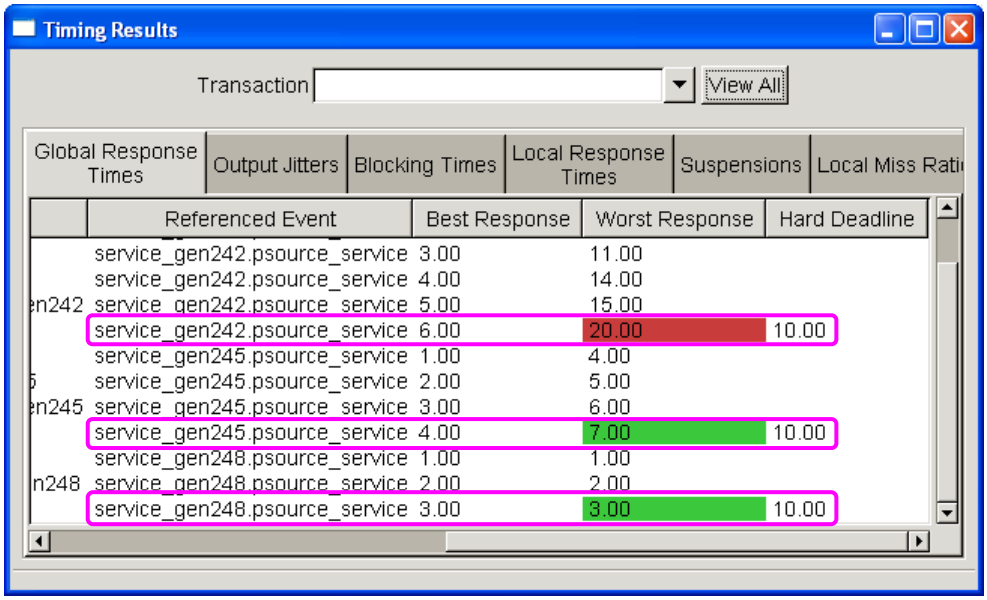


Figura 341. Ventana MAST con Escenarios que se Cumplen y otros que No.

En rojo el evento con latencia fuera de su deadline, y en verde los eventos con latencia cumpliendo su deadline.

Se pueden ver con detalle las ventanas de los marcos de razonamiento: (Figura 342 y Figura 343)

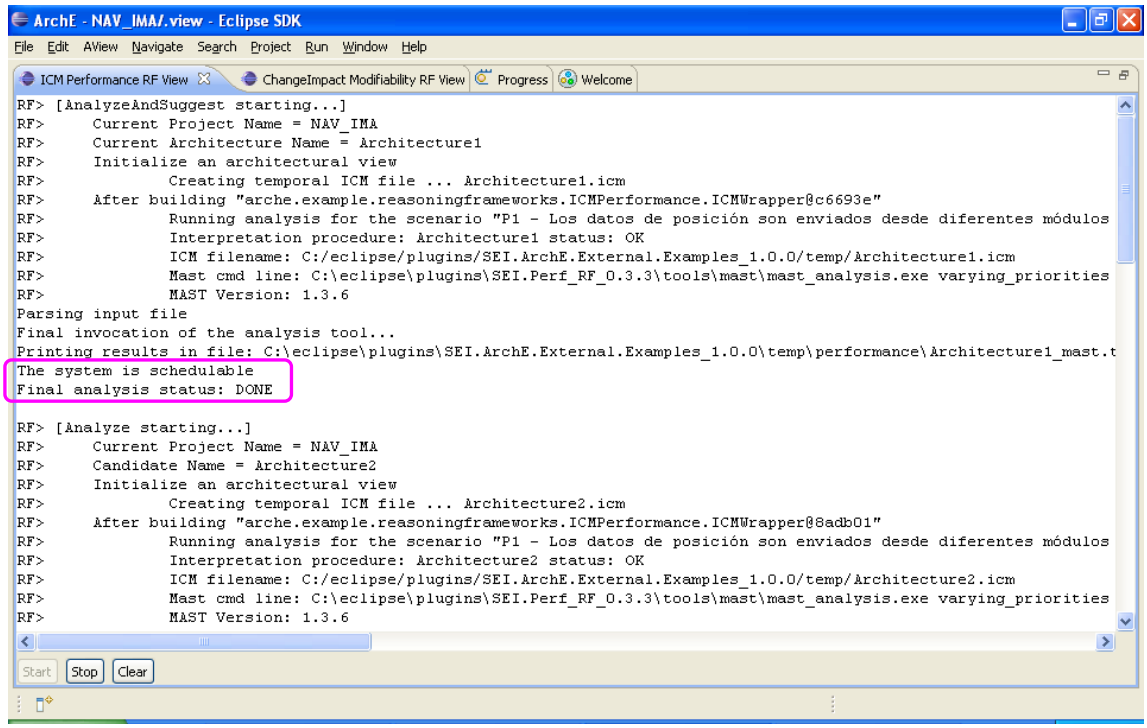


Figura 342. Ventana de Ejecución del Marco de Razonamiento ICM Performance.

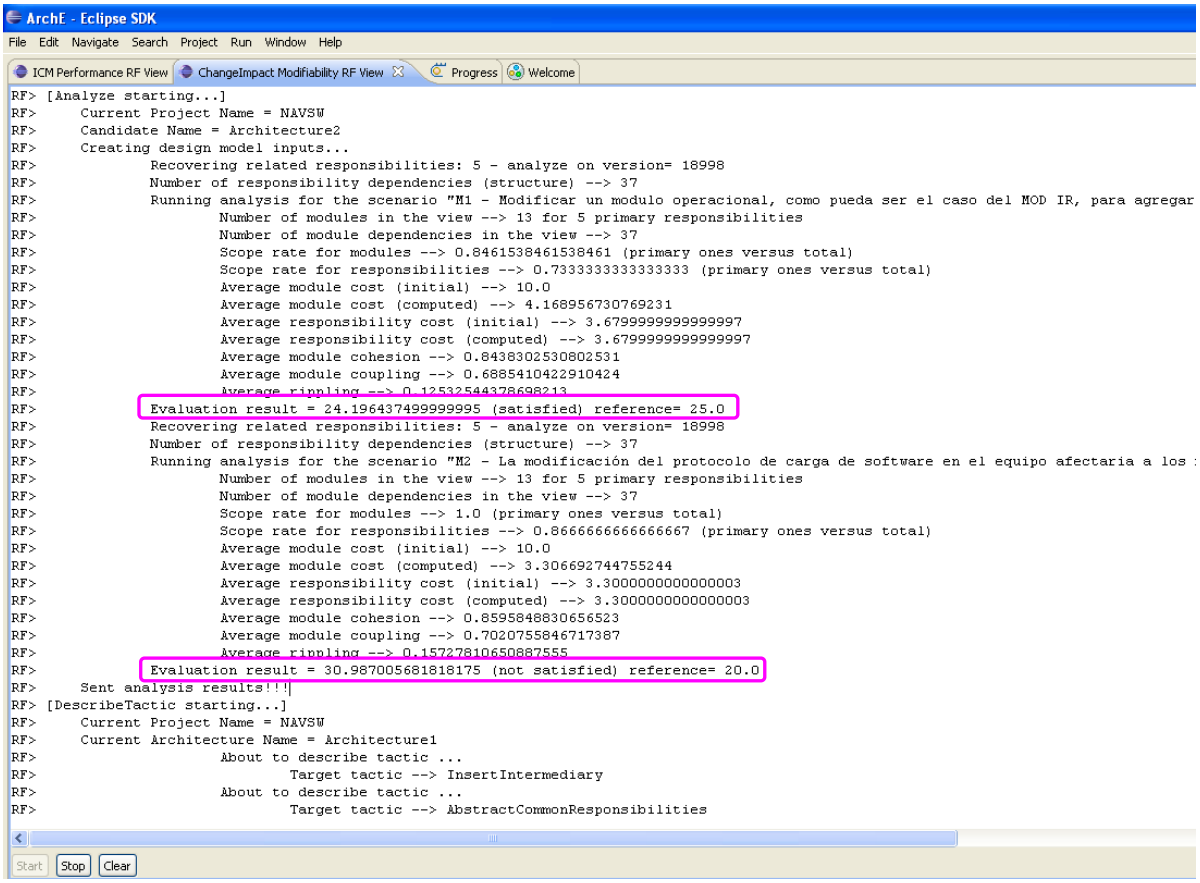


Figura 343. Ventana de Ejecución del Marco de Razonamiento ChangeImpact Modifiability.

14.2 Notas y Limitaciones de ArchE

Durante la realización de este trabajo, se han podido constatar las siguientes notas y limitaciones en el uso de la herramienta ArchE:

- Las responsabilidades tienen que tener un tiempo de ejecución entre 0 y 10 ms, lo cual obliga a escalar los posibles escenarios y las arquitecturas a esos órdenes de magnitud.
- Hasta que no hay creado al menos un escenario de modificabilidad y se aplique a al menos una responsabilidad, no se muestra el diagrama ULM de las responsabilidades.
- El diagrama UML muestra las relaciones de *dependency*, no las de *reaction*.
- Para que funcione el *ICM Performance*, se han de cumplir las siguientes condiciones:
  - Las responsabilidades deben tener un tiempo de ejecución
  - Los escenarios de *ICM Performance* han de estar definidos
  - Mapeo de escenario ICM Performance a responsabilidades establecido.
  - Relaciones tipo "*reaction*" entre responsabilidades asignadas a escenarios de ICM Performance
- Para que funcione el *ChangeImpact Modificability*:
  - Las responsabilidades deben de tener un coste del cambio
  - Los escenarios de *ChangeImpact Modificability* han de estar definidos
  - Mapeo de escenario *ChangeImpact Modificability* a responsabilidades
  - Relaciones tipo "*dependency*" entre responsabilidades asignadas a escenarios de *ChangeImpact Modificability*
- En *ICM Performance*, cuando hay varios escenarios de este tipo, y uno de ellos no se cumple, ArchE no realiza la comprobación para el resto.
- Cuando un escenario de performance no se cumple, se pone un círculo rojo. Pero cuando sí se cumple, no se pone verde, solo en gris: (Figura 344 y Figura 345)

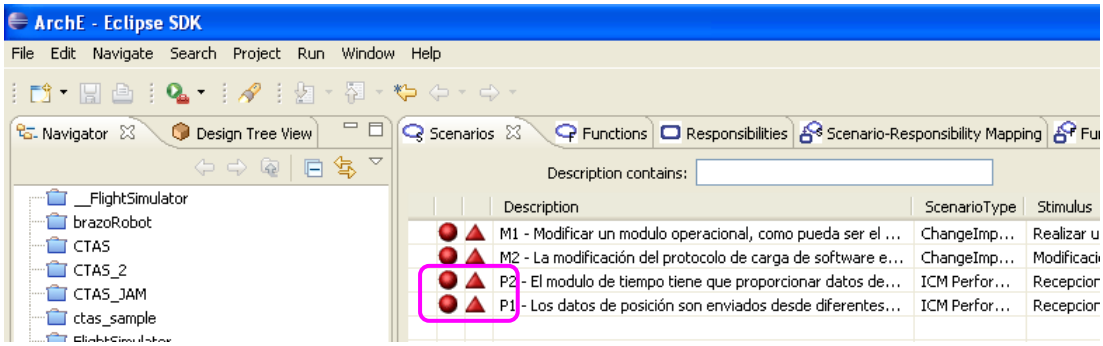


Figura 344. Resultado de ArchE – Caso de No Cumplimiento de los Escenarios de Rendimiento.

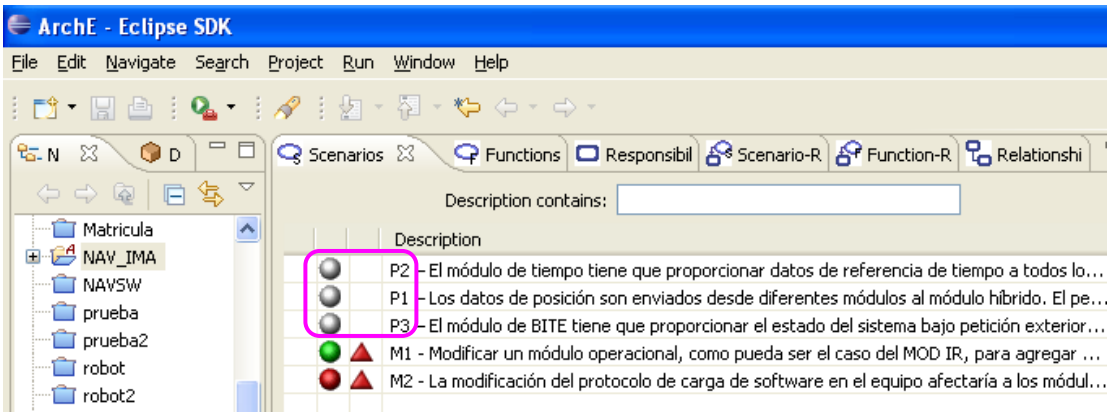


Figura 345. Resultado de ArchE – Caso de Cumplimiento de los Escenarios de Rendimiento.

- Para cumplir los escenarios de performance, hay que actuar sobre todas las responsabilidades afectadas por escenarios de *ICM Performance*, así como sobre las *deadlines* y los períodos de los escenarios de *ICM Performance*.
- No se permite establecer prioridades tanto a las responsabilidades como a las tareas (escenarios + responsabilidades) en *ICM Performance*
- La definición de responsabilidades es sensible a los caracteres no ingleses (por ejemplo, acentos). El programa no arranca MAST si se detecta algo así. Esto se ha visto sólo una vez durante una de las pruebas.
- Si los marcos de razonamiento no están cargados, no se pueden definir relaciones (de *dependency* o *reaction*) entre responsabilidades, ni elegir el tipo de escenario (*ICM Performance* o *ChangeImpact Modificability*) cuando se está definiendo.
- ArchE realiza las siguientes tareas en los atributos de calidad:
  - *ChangeImpact Modificability*: Análisis, tácticas
  - *ICM Performance*: Análisis; pero **ArchE no proporciona tácticas para *ICM Performance*.**
- El programa xmlBlaster ha de ser ejecutado (con la opción "r" al final de su ejecución) antes de ejecutar ArchE, puesto que xmlBlaster es el que gestiona las comunicaciones entre ArchE y los marcos de razonamiento, como se ha visto en el Anexo 1. Si no se ejecuta, los marcos de razonamiento no funcionarán apropiadamente.

10 BIBLIOGRAFÍA

[01] Vogel, Olivier; Arnold, Ingo; Chughtai, Arif; Kehrer, Timo. *Software Architecture - A Comprehensive Framework and Guide for Practitioners*. Heidelberg (Germany): Springer, 2009. ISBN: 978-3-642-19735-2.

[02] Shaw, Mary; Garlan, David. *Software architecture: Perspectives on an emerging discipline*. Pearson, 1996. ISBN-10: 0131829572.

[03] Schmidt, Richard F. *Software Engineering - Architecture-driven Software Development*. Elsevier, 2013. ISBN 978-0-12-407768-3.

[04] Gorton, Ian. *Essential Software Architecture*. Springer, 2011 [2ª Edición]. ISBN 978-3-642-19175-6.

[05] Bass, Len; Clements, Paul; Kazman, Rick. *Software Architecture in Practice*. Addison-Wesley, 2003. ISBN: 0-321-15495-9.

[06] Rozanski, Nick; Woods, Eoin. *Software Systems Architecture*. Addison-Wesley, 2005. ISBN 0-321-11229-6.

[07] Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, 1995. ISBN: 978-0-471-95869-7.

[08] Garlan, David; Shaw, Mary. *An Introduction to Software Architecture*. CMU-CS-94-166. 1994.

[09] *Introduction to Software Architecture Evaluation*. Universidad Politécnica de Valencia, de: http://users.dsic.upv.es/~jagonzalez/IST/files/IntroductionArchitectureEvaluation.pdf

[10] Dobrica, L.; Niemela, E. *A Survey on Software Architecture Analysis Methods*. 2002. En: Software Engineering, IEEE Transactions on Software Engineering (Volume:28 , Issue: 7). IEEE. ISSN: 0098-5589

[11] Koziolek, Heiko; Schlich, Bastian; Bilich, Carlos. *A Large-Scale Industrial Case Study on Architecture-based Software Reliability Analysis*. 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE).

[12] Clements, Paul. *Current Best Practices in Software Architecture Session 4: Evaluating Software Architectures*. 13rd October 2005. Software Engineering Institute. Carnegie Mellon University.

[13] Wojcik, Rob; Bachmann, Felix; Bass, Len; Clements, Paul; Merson, Paulo; Nord, Robert; Wood, Bill. *Attribute-Driven Design (ADD), Version 2.0*. November 2006. Technical Report. CMU/SEI-2006-TR-023. ESC-TR-2006-023.

[14] RTCA DO-178C *Software Considerations in Airborne Systems and Equipment Certification*. 2011

[15] Rierson, Leanna K. *Using The Software Capability Maturity Model For Certification Projects*. Washington. 1998. FAA.

[16] Buter, Andrew; Stienstra, Curt; VanderLeest, Steven H. *Agile for Aerospace*. DornerWorks. GLSEC 2008.

[17] Casals, David. *Aeronautical Software Course: Real Time Software Architectures*. Mission Systems Department.Airbus Defense and Space. 2014

[18] *System Development Modular Approach Eases Avionics Certification Challenges* - COTS Journal online, de: http://www.cotsjournalonline.com/articles/view/101451

[19] ARINC Report 651-1. *Design Guidance For Integrated Modular Avionics*. 1997

[20] DOT/FAA/AR-03/77. *Commercial Off-The-Shelf Real-Time Operating System and Architectural Considerations*. 2004. FAA.

[21] ARINC SPECIFICATION 653-1. *Avionics Application Software Standard Interface*. 2003.

[22] Rufino, José; Craveiro, Joao; *Robust Partitioning and Composability in ARINC 653 Conformant Real-Time Operating Systems*.

[23] Wind River VxWorks 653. 2010, de:

http://www.windriver.com/products/product-overviews/PO_VxWorks653_Platform_0210.pdf

[24] LynxOS-178 RTOS for DO-178B Software Certification, de:

http://www.lynx.com/products/real-time-operating-systems/lynxos-178-rtos-for-do-178b-software-certification/

[25] Green Hills Software - Safety Critical Products: INTEGRITY®-178B RTOS, de:

http://www.ghs.com/products/safety_critical/integrity-do-178b.html

[26] Mejia Alvarez, Pedro; Cova Suazo Nancy, Noemí; Pérez Reséndiz Marisol; *Arquitectura de Software*. CINVESTAV – IPN. Sección de Computación. De:

http://slideplayer.es/slide/1057374/

[27] XPort1020 Freescale MPC8270 Processor-Based Multi-Protocol Twelve-Port Serial 6U cPCI Module, de:

http://www.xes-inc.com/products/view/xport1020/

[27] http://www.xes-inc.com/products/view/xport1020/

[28] http://comps.canstockphoto.es/can-stock-photo_csp26053611.jpg

[29] http://www.futureplatone.com/img/simulador-vuelo-valencia.png

[30] http://www.fancyicons.com/free-icons/108/occupations/png/256/pilot_female_light_256.png

[31] http://comps.canstockphoto.es/can-stock-photo_csp18061284.jpg // http://st2.depositphotos.com/1429923/5516/v/950/depositphotos_55164017-Flat-illustration-of-expert-with-control-panel.-Analytics-and-management.jpg

[32] http://st2.depositphotos.com/1000244/5869/v/450/depositphotos_58693619-sound-speaker-icon.jpg

[33] http://ubuntuarte.com/wordpress/wp-content/uploads/2008/04/sun_server_familyubuntu.jpg

[34] http://www.sdm.es/Web/wp-content/uploads/2014/09/servidores.png

[35] http://previews.123rf.com/images/scanrail/scanrail1205/scanrail120500028/13877506-Hard-disk-and-database-icon-isolated-on-white-background-Stock-Photo.jpg

[36] Bachmann, Felix; Bass, Len; Klein, Mark; *Preliminary Design of ArchE: A Software Architecture Design Assistant*. September 2003. Technical Report. CMU/SEI-2003-TR.

[37] Bachmann, Felix; Bass, Len; Bianco, Philip; Klein, Mark. *Using ArchE in the Classroom: One Experience.* September 2007. Technical Note. CMU/SEI-2007-TN-001.

[38] Bachmann, Felix; Bass, Len; Klein, Mark. *Illuminating the Fundamental Contributors to Software Architecture Quality*. August 2002. Technical Report. CMU/SEI-2002-TR-025. ESC-TR-2002-025.

[39] Bachmann, Felix; Klein, Mark. *Methodical Design of Software Architecture Using an Architecture Design Assistant (ArchE)*. 2005 by Software Engineering Institute - Carnegie Mellon University. Pittsburgh.

[40] Bachmann, Felix; Bass, Len; Bianco, Phil. *Software Architecture Design with ArchE*. Marzo de 2007. Software Engineering Institute. Carnegie Mellon University. Pittsburgh.

[41] Bass, Len. *ArchE – An Architecture Design Assistant*. August 2, 2007. Software Engineering Institute. Carnegie Mellon University. Pittsburgh.

[42] Bianco, Phil; Diaz-Pace, Andres. *Current SEI SAT Initiative Technology Investigations*. May 1st, 2008. Software Engineering Institute. Carnegie Mellon University. Pittsburgh.

[43] Moreno, Gabriel A.; Hansen, *Jeffrey. Overview of the Lambda-* Performance Reasoning Frameworks*. February 2009. Technical Report. CMU/SEI-2008-TR-020. ESC-TR-2008-020.

[44] Diaz-Pace, Andres ; Kim, Hyunwoo; Bass, Len; Bianco, Phil; Bachmann, Felix. *Integrating Quality-attribute Reasoning Frameworks in the ArchE Design Assistant*. Software Engineering Institute. Carnegie Mellon University. Pittsburgh.

[45] Champagne, Roger; Gagné, Sébastien. *Towards automation of architectural tactics application – an example with ArchE*. 2011. Dept. of Software and IT Engineering. ÉTS (University of Québec). Montréal, Canada.

[46] Lee, Jinhee; Bass, Len. *Elements of a Usability Reasoning Framework*. September 2005. Software Architecture Technology Initiative. Technical Note. CMU/SEI-2005-TN-030.

[47] Clements, Paul; Bass, Len. Relating Business Goals to Architecturally Significant Requirements for Software Systems. May 2010. Technical Note. CMU/SEI-2010-TN-018.

[48] Gaitán Peña, Carlos Alberto. *Arquitecturas Software: Gestión de los atributos de calidad y su incorporación en ArchE para la mejora del análisis, evaluación y soporte en la toma de decisiones durante el desarrollo arquitectónico*. Enero de 2014. Trabajo Fin de Master del Máster Universitario En Investigación En Ingeniería De Software Y Sistemas Informáticos – UNED. Curso 2012/2013.

[49] Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Merson, Paulo; Nord, Robert; Stafford, Judith. *Documenting Software Architectures. Views and Beyond*. Pearson, 2011 [2ª Edición]. ISBN-10: 0321552687 / ISBN-13: 9780321552686.

[50] Malveau, Raphael; Mowbray, Thomas J. *Software Architect Bootcamp*. Prentice Hall, 2000. ISBN: 0-13-027407-0