UNIVERSIDAD REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INTEROPERABILIDAD ENTRE SISTEMAS DE APOYO A LA CONDUCCIÓN DE OPERACIONES MILITARES

Tesis Doctoral

Miguel Serna Cañas

Octubre 2011

UNIVERSIDAD REY JUAN CARLOS

Interoperabilidad entre Sistemas de Apoyo a la Conducción de Operaciones Militares

Tesis Doctoral

Miguel Serna Cañas

Escala de oficiales del ejército de Tierra

Octubre 2011

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES, CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

Interoperabilidad entre Sistemas de Apoyo a la Conducción de Operaciones Militares

Tesis Doctoral

Autor: Miguel Serna Cañas
Escala de oficiales del ejército de Tierra
Director: Dra. Marta Beltrán Pardo
Doctora en Informática

Octubre 2011

Título:
Interoperabilidad entre Sistemas de Apoyo a la Conducción de Operaciones Militares

Autor:
Miguel Serna Cañas

Tribunal:

Presidente :

Vocales :

Secretario :

Suplentes :

Acuerdan otorgar la calificación de:

Madrid, de de 2011

Las tropas expertas en el arte de la guerra se usan como la culebra que "responde de manera simultanea" del monte Ch'ang. Si se le golpea en la cabeza, su cola ataca; si se la golpea en la cola, su cabeza ataca, si se la golpea en el centro, tanto la cabeza como la cola atacan.
Sun Tzu

Llegar juntos es el principio; mantenerse juntos es el progreso; trabajar juntos es el éxito.
Henry Ford

Agradecimientos

Gracias Marta, por tu bien hacer, tus consejos, tu amistad, por estar siempre ahí y sobre todo por tu lealtad.

A Joaquín, gracias por tus ánimos y por haberme hecho ver que lo que estaba haciendo era importante.

Gracias Myriam, siempre me has animado no dejándome desfallecer y sin ti no podría haber conseguido nada de lo que he alcanzado en mi vida.

Gracias Miguel, por tu preocupación y hacerme participe de tus experiencias que han sido una fuente de inspiración.

Gracias María y Lucía, vuestra paciencia y cariño han sido clave para alcanzar las metas que me he propuesto.

No puedo olvidar al resto de familiares y amigos que han sabido entender el tiempo que he dedicado a esta tesis.

Índice general

1. Introducción **1**

1.1. Contexto actual 2

1.2. Hipótesis de partida 5

1.3. Objetivos 6

1.4. Metodología y planificación 7

1.5. Estructura del documento 9

2. Antecedentes **11**

2.1. Importancia de la simulación en entornos militares 11

2.2. Clasificación de simuladores en entornos militares 13

2.2.1. Simuladores administrativos 13

2.2.2. Simuladores tácticos 13

2.3. Necesidad de interoperabilidad entre simuladores de aplicación militar 16

2.4. High Level Architecture: HLA 17

2.4.1. Antecedentes 18

2.4.2. Estándares asociados a HLA 20

2.4.3. Heterogeneidad de las implementaciones del estándar IEEE1516 23

2.4.4. Situación actual de la simulación distribuida en entornos militares 24

2.5. Importancia de los sistemas de mando y control en entornos militares 25

2.6. Funcionalidades y clasificación de los sistemas de mando y control . . 27

2.7. Necesidad de interoperabilidad entre sistemas de mando y control . . 29

2.8. The Joint C3 Information Exchange Data Model: JC3IEDM 31

2.8.1. Evolución de los modelos de intercambio 31

2.8.2. Conceptos básicos acerca del modelo JC3IEDM 32
2.8.3. Estándares asociados a JC3IEDM: DEM y XML 33
2.8.3.1. Data Exchange Mechanism (DEM) 34
2.8.3.2. eXtensible Markup Language (XML) 36
2.8.4. Técnicas MDA y modelos de datos orientados a objetos 37
2.9. Necesidades de interoperabilidad entre simuladores constructivos y sistemas C2 38
2.9.1. Coalition Battle Management Language (CBML) 40
2.9.2. Military Scenario Definition Language (MSDL) 40

3. Interoperabilidad entre simuladores constructivos de aplicación militar 43

3.1. Antecedentes 44
3.1.1. Modelos de interoperabilidad de referencia en entornos industriales 44
3.1.1.1. Tipos de IRMs: Definiciones previas 45
3.1.1.2. IRM Tipo A 46
3.1.1.2.1. IRM subtipo A1. 47
3.1.1.2.2. IRM subtipo A2. 47
3.1.1.2.3. IRM subtipo A3. 48
3.1.1.3. IRM Tipo B 49
3.1.1.4. IRM Tipo C 49
3.1.1.5. IRM Tipo D 50
3.1.2. Aspectos de HLA relacionados con la gestión del tiempo . . . 50
3.1.2.1. Definiciones 51
3.1.2.2. Ordenación de los mensajes 52
3.1.2.3. Los servicios Time Management 54
3.1.2.3.1. Mecanismos de avance del tiempo. 54
3.1.2.3.2. Ordenación de los mensajes enviados. 54
3.2. Necesidad de la definición de IRMs para entornos militares 56
3.3. Definición de IRMs militares 57
3.3.1. Consideraciones previas 57

3.3.2. Necesidades específicas del entorno táctico-militar 59
3.3.3. Rol de un federado 61
3.3.4. Definición del IRM tipo A para simuladores constructivos militares 61
3.3.4.1. Definición de la entidad transferida (ETS) 62
3.3.4.1.1. Ejemplo de ETS. 63
3.3.4.2. Definición del modelo subtipo A1 para simuladores constructivos militares 64
3.3.4.2.1. Evento 1. 64
3.3.4.2.2. Evento 2. 66
3.3.4.2.3. Evento 3. 66
3.3.4.2.4. Evento 4. 67
3.3.4.2.5. Evento 5. 67
3.3.4.3. Caso práctico de aplicación del IRM A1 68
3.3.4.4. Definición del modelo subtipo A2 para simuladores constructivos militares 69
3.3.4.4.1. Evento 1. 70
3.3.4.4.2. Evento 2. 70
3.3.4.4.3. Evento 3. 70
3.3.4.4.4. Evento 4. 71
3.3.4.4.5. Evento 5. 72
3.3.4.5. Caso práctico de aplicación del IRM A2 73
3.3.4.6. Definición del modelo subtipo A3 para simuladores constructivos militares 74
3.3.4.6.1. Evento 1. 77
3.3.4.6.2. Evento 2. 78
3.3.4.6.3. Evento 3. 78
3.3.4.6.4. Evento 4. 79
3.3.4.6.5. Evento 5. 80
3.3.4.6.6. Evento 6. 80
3.3.4.7. Caso práctico de aplicación del IRM A3 81

3.3.5. Definición del IRM tipo B para simuladores constructivos militares . . . 82
3.3.5.1. Definición del recurso compartido (RSS) . . . 83
3.3.5.1.1. Ejemplo de RSS. . . . 84
3.3.5.2. Definición del modelo tipo B para simuladores constructivos militares . . . 84
3.3.5.2.1. Evento 1. . . . 84
3.3.5.2.2. Evento 2. . . . 85
3.3.5.2.3. Evento 3. . . . 85
3.3.5.2.4. Evento 4. . . . 86
3.3.5.2.5. Evento 5. . . . 87
3.3.5.3. Caso práctico de aplicación del IRM B . . . 88
3.3.6. Definición del IRM tipo C para simuladores constructivos militares . . . 89
3.3.6.1. Empleo de los servicios Time Management para gestionar la incertidumbre del suceso del IRM tipo C . . 89
3.3.6.2. Definición del evento compartido (ESS) . . . 90
3.3.6.2.1. Ejemplo de ESS. . . . 90
3.3.6.3. Definición del modelo tipo C para simuladores constructivos militares . . . 91
3.3.6.3.1. Evento 1. . . . 91
3.3.6.3.2. Evento 2. . . . 92
3.3.6.3.3. Evento 3. . . . 92
3.3.6.3.4. Evento 4. . . . 93
3.3.6.3.5. Evento 5. . . . 93
3.3.6.4. Caso práctico de aplicación del IRM C . . . 94
3.3.7. Definición del IRM tipo D para simuladores constructivos militares . . . 95
3.3.7.1. Definición de la estructura de datos compartida (DSS) 95
3.3.7.1.1. Ejemplo de DSS. . . . 96
3.3.7.2. Definición del modelo tipo D para simuladores constructivos militares . . . 97

3.3.7.2.1. Evento 1. 97
3.3.7.2.2. Evento 2. 97
3.3.7.2.3. Evento 3. 98
3.3.7.2.4. Evento 4. 98
3.3.7.2.5. Evento 5. 99
3.3.7.3. Caso práctico de aplicación del IRM D 100
3.3.8. Definición del IRM tipo E para simuladores constructivos militares 101
3.3.8.1. Marco conceptual de plan/orden 101
3.3.8.2. Definición del plan/orden (P/O) 102
3.3.8.2.1. Ejemplo de P/O. 103
3.3.8.3. Definición del modelo tipo E para simuladores constructivos militares 103
3.3.8.3.1. Fase 1: evento 1. 104
3.3.8.3.2. Fase 1: evento 2. 104
3.3.8.3.3. Fase 1: evento 3. 105
3.3.8.3.4. Fase 2: evento 4. 106
3.3.8.3.5. Fase 2: evento 5. 106
3.3.8.3.6. Fase 2: evento 6. 107
3.3.8.3.7. Fase 2: evento 7. 107
3.3.8.3.8. Fase 2: evento 8. 108
3.3.8.3.9. Evento 9. 109
3.3.8.4. Caso práctico de aplicación del IRM E 110

4. Modificación de una RTI real para la ejecución automática de los IRMs militares 111

4.1. Selección de una RTI para estudiar la aplicación práctica de los IRMs 112
4.2. Descripción de Pórtico 113
4.2.1. Conceptos generales de Pórtico 114
4.2.2. Descripción de los elementos que componen Pórtico 114
4.2.3. Diagrama de clases definido para Pórtico 116
4.3. Aplicación práctica de los IRMs en Pórtico 118

4.3.1. Funciones comunes a todos los IRMs 118
4.3.1.1. Punto de partida en la ejecución de un IRM 118
4.3.1.2. Gestión de los cambios de evento 119
4.3.1.2.1. Funciones de los componentes de Pórtico en el paso de evento 119
4.3.1.3. Identificación del IRM que se está ejecutando 122
4.3.1.3.1. Concurrencia en la ejecución de varios IRMs. 122
4.3.2. Funciones específicas del IRM subtipo A1 124
4.3.2.0.2. Funciones en el evento 1. 124
4.3.2.0.3. Funciones en el evento 2. 125
4.3.2.0.4. Funciones en el evento 3. 126
4.3.2.0.5. Funciones en el evento 4. 126
4.3.2.0.6. Funciones en el evento 5. 127
4.3.3. Funciones específicas del IRM subtipo A2 128
4.3.3.0.7. Funciones en el evento 1. 128
4.3.3.0.8. Funciones en el evento 2. 128
4.3.3.0.9. Funciones en el evento 3. 128
4.3.3.0.10. Funciones en el evento 4. 129
4.3.3.0.11. Funciones en el evento 5. 130
4.3.4. Funciones específicas del IRM subtipo A3 131
4.3.4.0.12. Funciones en el evento 1. 131
4.3.4.0.13. Funciones en el evento 2. 132
4.3.4.0.14. Funciones en el evento 3. 133
4.3.4.0.15. Funciones en el evento 4. 134
4.3.4.0.16. Funciones en el evento 5. 135
4.3.4.0.17. Funciones en el evento 6. 135
4.3.5. Funciones específicas del IRM tipo B 136
4.3.5.0.18. Funciones en el evento 1. 137
4.3.5.0.19. Funciones en el evento 2. 138
4.3.5.0.20. Funciones en el evento 3. 138
4.3.5.0.21. Funciones en el evento 4. 138

4.3.5.0.22. Funciones en el evento 5. 139
4.3.6. Funciones específicas del IRM tipo C 139
4.3.6.0.23. Funciones en el evento 1. 140
4.3.6.0.24. Funciones en el evento 2. 140
4.3.6.0.25. Funciones en el evento 3. 141
4.3.6.0.26. Funciones en el evento 4. 142
4.3.6.0.27. Funciones en el evento 5. 143
4.3.7. Funciones específicas del IRM tipo D 144
4.3.7.0.28. Funciones en el evento 1. 144
4.3.7.0.29. Funciones en el evento 2. 145
4.3.7.0.30. Funciones en el evento 3. 146
4.3.7.0.31. Funciones en el evento 4 146
4.3.7.0.32. Funciones en el evento 5. 146
4.3.8. Funciones específicas del IRM tipo E 146
4.3.8.0.33. Funciones en la fase 1: evento 1. 147
4.3.8.0.34. Funciones en la fase 1: evento 2. 147
4.3.8.0.35. Funciones en la fase 1: evento 3. 147
4.3.8.0.36. Funciones en la fase 2: evento 4. 148
4.3.8.0.37. Funciones en la fase 2: evento 5. 148
4.3.8.0.38. Funciones en la fase 2: evento 6. 148
4.3.8.0.39. Funciones en la fase 2: evento 7. 149
4.3.8.0.40. Funciones en la fase 2: evento 8. 150
4.3.8.0.41. Funciones en la fase 2: evento 9. 150

5. Interoperabilidad entre simuladores constructivos y sistemas C2 151
5.1. Antecedentes 152
5.1.1. Definiciones y conceptos básicos 152
5.1.2. Casos de interoperabilidad identificados entre sistemas C2 y simuladores constructivos 153
5.1.3. Estándares asociados a los casos de interoperabilidad 156
5.2. Necesidades de interoperabilidad 158
5.3. Definición de la arquitectura IC2A 160

5.3.1. Reglas para la definición de la IC2A 161
5.3.2. Definición de las plantillas de datos IC2D 163
5.3.2.1. Definición de ficheros 163
5.3.2.1.1. Ejemplo de definición de ficheros. 164
5.3.2.2. Definición de protocolos 165
5.3.2.2.1. Ejemplo de definición de protocolos. 166
5.3.2.3. Definición de esquemas 166
5.3.2.3.1. Ejemplo de definición de esquemas. 167
5.3.2.4. Definición de reglas de negocio 168
5.3.2.4.1. Ejemplo de definición de reglas de negocio. . 170
5.3.2.5. Definición de procesos 171
5.3.2.5.1. Ejemplo de definición de procesos. 172
5.3.2.6. Definición de estados 172
5.3.2.6.1. Ejemplo de definición de estados. 173
5.3.2.7. Definición de cambios 173
5.3.2.7.1. Ejemplo de definición de cambios. 174
5.3.3. Interfaces de la IC2A 175
5.3.3.1. Módulo base del IC2F 176
5.3.3.2. Módulo de generación de escenarios 178
5.3.3.3. Módulo de generación de líneas de acción 180
5.3.3.4. Módulo de evaluación de líneas de acción 181
5.4. Definición de los modelos de interoperabilidad de referencia: IRM tipo 1 e IRM tipo 2 183
5.4.1. IRM tipo 1: Generación de un escenario real para un simulador constructivo 183
5.4.1.1. Scenario Transfer Specification 184
5.4.1.2. Actividades definidas para el IRM tipo 1 184
5.4.2. IRM tipo 2: Evaluación de las líneas de acción planeadas en un sistema C2 186
5.4.2.1. Action Line Specification y Evaluation Matrix Specification 186
5.4.2.2. Actividades definidas para el IRM tipo 2 187

6. Aplicación práctica de la arquitectura IC2A **191**

6.1. Aspectos clave en la implementación de la arquitectura IC2A 191

6.1.1. Conceptos básicos de ESB 193

6.1.2. Justificación de la utilización del ESB como arquitectura para la implementación de la IC2A 194

6.1.3. Retos que plantea la utilización de un ESB y utilidad de los patrones SOA para superarlos con éxito 195

6.1.3.0.1. Intercambio de ficheros. 197

6.1.3.0.2. Transformación de datos. 197

6.1.3.0.3. Convergencia de protocolos. 198

6.1.3.0.4. Centralización de esquemas. 199

6.1.3.0.5. Centralización de reglas. 199

6.1.3.0.6. Centralización de procesos. 200

6.1.3.0.7. Compatibilidad de procesos. 201

6.1.3.0.8. Repositorio de estado. 201

6.1.3.0.9. Núcleo básico. 202

6.1.4. Implementación de la arquitectura IC2A sobre un ESB genérico utilizando los patrones SOA 203

6.1.4.1. Intercambio de ficheros 204

6.1.4.2. Transformación de datos 205

6.1.4.3. Convergencia de protocolos 207

6.1.4.4. Centralización de esquemas 208

6.1.4.5. Centralización de reglas 209

6.1.4.6. Centralización de procesos 210

6.1.4.7. Compatibilidad de procesos 211

6.1.4.8. Repositorio de estado 211

6.1.4.9. Núcleo básico 213

6.2. Implementación completa de la IC2A sobre JBoss ESB y validación de los modelos de interoperabilidad de referencia 213

6.2.1. Justificación de la elección de JBoss ESB 214

6.2.2. Descripción de la arquitectura de JBoss ESB 215

6.2.3. Implementación y validación de los IRMs 216

6.2.3.1. Consideraciones previas 216
6.2.3.2. Especificación de las actividades comunes en JBoss ESB 217
6.2.3.3. Especificación de las actividades para el IRM tipo 1 . 218
6.2.3.3.1. Actividades de los sistemas C2. 219
6.2.3.3.2. Actividades de los simuladores constructivos. 220
6.2.3.4. Especificación de las actividades para el IRM tipo 2 . 221
6.2.3.4.1. Actividades de los sistemas C2 para publicar líneas de acción. 221
6.2.3.4.2. Actividades de los simuladores constructivos para publicar las matrices de evaluación. 222
6.2.3.4.3. Actividades de los sistemas C2 para leer las matrices de evaluación asociadas a una línea de acción. 224

7. Conclusiones y líneas de investigación futuras 227
7.1. Conclusiones generales 227
7.2. Interoperabilidad entre simuladores constructivos de aplicación militar 228
7.3. Modificación de una RTI real para la ejecución automática de los IRMs militares 229
7.4. Interoperabilidad entre simuladores constructivos y sistemas C2 . . . 230
7.5. Aplicación práctica de la arquitectura IC2A 231
7.6. Líneas de trabajo futuro 232

A. Tabla de Acrónimos 233

B. Diagramas de tiempos de los IRMs de aplicación militar 237
B.1. Diagramas de tiempos de los IRMs Tipo A 237
B.1.1. Diagramas de tiempos del IRM subtipo A1 237
B.1.2. Diagramas de tiempos del IRM subtipo A2 239
B.1.3. Diagramas de tiempos del IRM subtipo A3 241
B.2. Diagramas de tiempos del IRM Tipo B 243
B.3. Diagramas de tiempos del IRM Tipo C 244

B.4. Diagramas de tiempos del IRM Tipo D 246
B.5. Diagramas de tiempos del IRM Tipo E 248

Bibliografía **251**

Índice de figuras

2.1. Pirámide de simulación y ejemplos concretos de simulación del ejército de tierra español ([11]) 14
2.2. Niveles de simulación constructiva dentro del ejército de tierra español ([11]) 15
2.3. Concepto CAX en el ámbito OTAN ([35]) 17
2.4. Participación en proyecto Pathfinder ([35]) 19
2.5. Fases del proceso de desarrollo de una federación: FEDEP ([48]) . . . 21
2.6. Actividades de verificación, validación y acreditación dentro del FEDEP ([49]) 22
2.7. Esquema de red basada en LINK 22 ([77]) 30
2.8. Niveles conceptuales de interoperabilidad 32
2.9. Esquema de los modelos de intercambio de las diferentes comunidades ([23]) 34
2.10. Concepto del protocolo DEM por capas ([72]) 35
2.11. Esquema de publicación/suscripción a OIGs 36
2.12. Aplicación de tecnologías SOA en redes militares 37
2.13. COI desde el punto de vista MIP ([23]) 38
2.14. Pasarela C2IS - HLA ([35]) 39
2.15. Esquema de los diferentes modelos basados en C2 ([23]) 41

3.1. Esquema del IRM tipo A 46
3.2. Esquema del IRM subtipo A2 47
3.3. Esquema del IRM subtipo A3 48
3.4. Esquema del IRM tipo B 49
3.5. Esquema del IRM tipo C 49

3.6. Esquema del IRM tipo D 50
3.7. Ejemplo de secuencia temporal con *Time Stamp Order* 53
3.8. Ejemplo de secuencia temporal con *Casual Order* 54
3.9. *Federation Time Axis* en la lista de distribución 56
3.10. Esquema de datos del JC3IEDM 58
3.11. ETS definido para el IRM tipo A 62
3.12. Transferencia por niveles de una entidad entre unidades 63
3.13. Esquema del evento 1 en el IRM subtipo A1 65
3.14. Esquema del evento 2 en el IRM subtipo A1 65
3.15. Esquema del evento 3 en el IRM subtipo A1 66
3.16. Esquema del evento 4 en el IRM subtipo A1 67
3.17. Esquema del evento 5 en el IRM subtipo A1 68
3.18. Esquema del evento 1 en el IRM subtipo A2 70
3.19. Esquema del evento 2 en el IRM subtipo A2 71
3.20. Esquema del evento 3 en el IRM subtipo A2 71
3.21. Esquema del evento 4 en el IRM subtipo A2 72
3.22. Esquema del evento 5 en el IRM subtipo A2 73
3.23. Lista de carga del federado B 75
3.24. Envío del LLD al federado B 76
3.25. Envío de la ETS al federado B 76
3.26. Esquema del evento 1 en el IRM subtipo A3 77
3.27. Esquema del evento 2 en el IRM subtipo A3 77
3.28. Esquema del evento 3 en el IRM subtipo A3 78
3.29. Esquema del evento 4 en el IRM subtipo A3 79
3.30. Esquema del evento 5 en el IRM subtipo A3 79
3.31. Esquema del evento 6 en el IRM subtipo A3 80
3.32. RSS definido para el IRM tipo B 83
3.33. Acceso por niveles al recurso compartido entre unidades 83
3.34. Esquema del evento 1 en el IRM tipo B 85
3.35. Esquema del evento 2 en el IRM tipo B 86
3.36. Esquema del evento 3 en el IRM tipo B 86

3.37. Esquema del evento 4 en el IRM tipo B 87
3.38. Esquema del evento 5 en el IRM tipo B 88
3.39. Tabla Synchronization definida para el FOM de una federación 90
3.40. Esquema del evento 1 en el IRM tipo C 91
3.41. Esquema del evento 2 en el IRM tipo C 92
3.42. Esquema del evento 3 en el IRM tipo C 92
3.43. Esquema del evento 4 en el IRM tipo C 93
3.44. Esquema del evento 5 en el IRM tipo C 94
3.45. Tabla Data Types definida para el FOM de una federación 95
3.46. Esquema del evento 1 en el IRM tipo D 97
3.47. Esquema del evento 2 en el IRM tipo D 98
3.48. Esquema del evento 3 en el IRM tipo D 99
3.49. Esquema del evento 4 en el IRM tipo D 99
3.50. Esquema del evento 5 en el IRM tipo D 100
3.51. Definición de IRM tipo E 104
3.52. Esquema de la fase 1: evento 1 en el IRM tipo E 104
3.53. Esquema de la fase1: evento 2 en el IRM tipo E 105
3.54. Esquema de la fase1: evento 3 en el IRM tipo E 105
3.55. Esquema de la fase2: evento 4 en el IRM tipo E 106
3.56. Esquema de la fase2: evento 5 en el IRM tipo E 107
3.57. Esquema de la fase2: evento 6 en el IRM tipo E 107
3.58. Esquema de la fase 2: evento 7 en el IRM tipo E 108
3.59. Esquema de la fase2: evento 8 en el IRM tipo E 109
3.60. Esquema del evento 9 en el IRM tipo E 109

4.1. Diagrama de componentes de Pórtico ([117]) 115
4.2. Diagrama de clases de Pórtico ([117]) 117
4.3. Código modificado de la clase *TimeAdvanceGrantCallbackHandler* . . 121
4.4. Diagrama de ejecución temporal lógica de los IRMs en una RTI . . . 123
4.5. RTI con objetos de diferentes IRMs publicados 123

5.1. Representación de los casos de interoperabilidad entre dominios de sistemas 152

5.2. Ejemplo de matriz de evaluación . . . 153
5.3. Esquema contextual . . . 154
5.4. Convergencia directa entre simuladores constructivos y sistemas . . . 156
5.5. Esquema general de la IC2A . . . 160
5.6. Esquema de envío de ficheros entre componentes de la IC2A . . . 164
5.7. Esquema de la convergencia de protocolos de intercambio de información entre componentes de la IC2A . . . 165
5.8. Esquema del intercambio de información basado en diferentes esquemas entre componentes del IC2A . . . 167
5.9. Esquema del estándar NDMS . . . 168
5.10. Ejemplo de esquema en formato XSD de la información de una línea de acción . . . 169
5.11. Esquema de la centralización de las reglas de negocio . . . 170
5.12. Esquema del almacenamiento de los procesos . . . 171
5.13. Esquema del almacenamiento de los estados de la información intercambiada . . . 173
5.14. Ejemplo de esquema de la información de estado de transformación de una línea de acción . . . 174
5.15. Esquema del almacenamiento de los cambios . . . 174
5.16. Esquema de las interfaces del IC2F . . . 175
5.17. Esquema del IRM tipo 1 . . . 184
5.18. Ejemplo de esquema de un escenario definido en MSDL ([19]) . . . 185
5.19. Esquema del IRM tipo 2 . . . 187
5.20. Ejemplo de esquema de una matriz de evaluación . . . 188

6.1. Esquema de un ESB estándar . . . 194
6.2. Esquema conceptual de la integración de sistemas C2 con simuladores constructivos mediante la utilización de un ESB . . . 195
6.3. Esquema general de la IC2A . . . 196
6.4. Esquema del patrón SOA File Gateway . . . 197
6.5. Esquema del patrón SOA Data Format Transformation . . . 198
6.6. Esquema del patrón SOA Protocol Bridging . . . 198
6.7. Esquema del patrón SOA Schema Centralization . . . 199

6.8. Esquema del patrón SOA Rules Centralization 200
6.9. Esquema del patrón SOA Process Centralization 201
6.10. Esquema del patrón SOA Compatible Change 202
6.11. Esquema del patrón SOA State Repository 202
6.12. Esquema del patrón SOA Service Facade 203
6.13. Esquema del punto de partida 204
6.14. Esquema de intercambio de ficheros en un ESB 205
6.15. Esquema de transformación de datos en un ESB 206
6.16. Esquema de transformación de protocolos en un ESB 207
6.17. Esquema de centralización de esquemas en un ESB 208
6.18. Esquema de centralización de reglas en un ESB 210
6.19. Esquema de centralización de procesos en un ESB 211
6.20. Esquema de compatibilidad de procesos en un ESB 212
6.21. Esquema de repositorio de estados en un ESB 212
6.22. Esquema del núcleo básico en un ESB 213
6.23. Esquema de un servicio genérico en JBoss ESB ([143]) 215
6.24. Esquema de la gestión de teatros dentro de un ESB 218
6.25. Esquema de la publicación de escenarios en un teatro IC2A 219
6.26. Esquema de la lectura de escenarios en un teatro IC2A 221
6.27. Esquema de la publicación de líneas de acción en un teatro IC2A . . 223
6.28. Esquema de la publicación de matrices de evaluación en un teatro IC2A 224
6.29. Esquema de la lectura de matrices de evaluación en un teatro IC2A . 225

B.1. Diagrama de tiempos del evento 1 en el IRM subtipo A1 237
B.2. Diagrama de tiempos del evento 2 en el IRM subtipo A1 238
B.3. Diagrama de tiempos del evento 3 en el IRM subtipo A1 238
B.4. Diagrama de tiempos del evento 4 en el IRM subtipo A1 238
B.5. Diagrama de tiempos del evento 5 en el IRM subtipo A1 239
B.6. Diagrama de tiempos del evento 1 en el IRM subtipo A2 239
B.7. Diagrama de tiempos del evento 2 en el IRM subtipo A2 239
B.8. Diagrama de tiempos del evento 3 en el IRM subtipo A2 240
B.9. Diagrama de tiempos del evento 4 en el IRM subtipo A2 240

B.10.Diagrama de tiempos del evento 5 en el IRM subtipo A2 240
B.11.Diagrama de tiempos del evento 1 en el IRM subtipo A3 241
B.12.Diagrama de tiempos del evento 2 en el IRM subtipo A3 241
B.13.Diagrama de tiempos del evento 3 en el IRM subtipo A3 241
B.14.Diagrama de tiempos del evento 4 en el IRM subtipo A3 242
B.15.Diagrama de tiempos del evento 5 en el IRM subtipo A3 242
B.16.Diagrama de tiempos del evento 6 en el IRM subtipo A3 242
B.17.Diagrama de tiempos del evento 1 en el IRM tipo B 243
B.18.Diagrama de tiempos del evento 2 en el IRM tipo B 243
B.19.Diagrama de tiempos del evento 3 en el IRM tipo B 243
B.20.Diagrama de tiempos del evento 4 en el IRM tipo B 244
B.21.Diagrama de tiempos del evento 5 en el IRM tipo B 244
B.22.Diagrama de tiempos del evento 1 en el IRM tipo C 244
B.23.Diagrama de tiempos del evento 2 en el IRM tipo C 245
B.24.Diagrama de tiempos del evento 3 en el IRM tipo C 245
B.25.Diagrama de tiempos del evento 4 en el IRM tipo C 245
B.26.Diagrama de tiempos del evento 5 en el IRM tipo C 246
B.27.Diagrama de tiempos del evento 1 en el IRM tipo D 246
B.28.Diagrama de tiempos del evento 2 en el IRM tipo D 246
B.29.Diagrama de tiempos del evento 3 en el IRM tipo D 247
B.30.Diagrama de tiempos del evento 4 en el IRM tipo D 247
B.31.Diagrama de tiempos del evento 5 en el IRM tipo D 247
B.32.Diagrama de tiempos de la fase 1: evento 1 en el IRM tipo E 248
B.33.Diagrama de tiempos de la fase 1: evento 2 en el IRM tipo E 248
B.34.Diagrama de tiempos de la fase 1: evento 3 en el IRM tipo E 248
B.35.Diagrama de tiempos de la fase 2: evento 4 en el IRM tipo E 249
B.36.Diagrama de tiempos de la fase 2: evento 5 en el IRM tipo E 249
B.37.Diagrama de tiempos de la fase 2: evento 6 en el IRM tipo E 249
B.38.Diagrama de tiempos de la fase 2: evento 7 en el IRM tipo E 250
B.39.Diagrama de tiempos de la fase 2: evento 8 en el IRM tipo E 250
B.40.Diagrama de tiempos del evento 9 en el IRM tipo E 250

Capítulo 1

Introducción

Los sistemas de apoyo a la conducción de operaciones militares se han convertido en herramientas esenciales para la realización exitosa de los diferentes tipos de misiones encomendadas a las fuerzas armadas actuales. Entre estos sistemas cabe destacar a los simuladores constructivos y a los sistemas de mando y control ([1]).

La simulación es una técnica muy poderosa y ampliamente utilizada para analizar y estudiar sistemas complejos. En la actualidad, las aplicaciones de la simulación dentro de las fuerzas armadas son muy numerosas, en el caso de los simuladores constructivos, se suelen emplear para el adiestramiento ([2]), el análisis de misiones y el planeamiento y apoyo de operaciones ([3]).

En cuanto a los sistemas de mando y control (Command and Control, C2), suelen incluir todos los procesos asociados con la recopilación de información y su interpretación para la creación de una percepción realista de la situación ([4]), así como el desarrollo de una lista de acciones basada en esta percepción.

El conjunto de estas acciones constituye la toma de decisiones que, a su vez, tiene que estar sustentada por cuatro servicios: la transmisión de las órdenes a la unidades desplegadas, la conversión de estas órdenes en señales de activación, la transmisión de las señales de control a las unidades en acción, y el acopio y procesamiento de información después del cumplimiento de estas órdenes.

La rapidez con la que cambia la situación en los conflictos actuales exige que los sistemas de mando y control no se limiten a ser sistemas de difusión y gestión de la información. Para acortar el ciclo de decisión al máximo ([5]) son necesarios también sistemas de apoyo a la decisión del mando, es decir, no sólo se trata de presentar la información de manera ordenada sino que en base a ella, se debe asesorar al mando acerca de cuál es la mejor línea de acción.

Los sistemas de mando y control podrían reducir considerablemente su ciclo de decisión gracias al uso de simuladores que, alimentados con los datos del propio sistema de mando y control evaluaran los resultados de cada una de las posibles líneas de acción, ayudando al mando a tomar una decisión de manera más rápida y teniendo en cuenta información mucho más elaborada. Y alimentando los simuladores con escenarios generados por los propios sistemas de mando y control, los resultados de las simulaciones serían mucho más útiles y ricos, aproximándose los resultados obtenidos en mucha mayor medida a la realidad.

En esta tesis doctoral se proponen modelos, metodologías y herramientas para resolver los problemas de interoperabilidad entre estos dos tipos de sistemas de apoyo a la conducción de operaciones militares.

1.1. Contexto actual

En los últimos años, el número de simuladores en servicio en las fuerzas armadas ha aumentado de manera significativa. Además, las operaciones militares actuales pueden verse afectadas por múltiples parámetros que hacen que su recreación sea muy compleja ([6]). Por este motivo se ha evolucionado de la primera generación de simuladores cuyo ámbito se reducía a un sistema específico, a una segunda generación cuyo objetivo es recrear campos de batalla completos ([7]). Esto ha dado lugar a la creación de unidades específicas que se encargan del control, desarrollo y mantenimiento de los simuladores, cada vez más complejos y sofisticados.

Para simular un campo de batalla es necesario que sistemas de simulación específicos actúen de manera conjunta afectando unos a las acciones de otros. De este modo se consiguen recreaciones muy parecidas a los escenarios reales que cumplen con los requerimientos que hoy en día las fuerzas armadas imponen a sus simuladores ([8], [9]).

Los simulación distribuida ha sido la línea de acción escogida por las fuerzas armadas de la mayor parte de naciones para alcanzar sus objetivos ([10]). Este tipo de simulación tiene varias ventajas:

- El rendimiento global de la simulación aumenta ya que se suma un mayor número de recursos hardware a su ejecución, especialmente se incrementan la capacidad de cómputo (procesadores de propósito general y GPUs) y de memoria.

- Al participar varios simuladores de ámbito diferente en la producción del resultado global, el número de parámetros que intervienen en la simulación es mayor, esto hace que los entornos simulados sean mucho más realistas y completos.

Los simuladores militares que intervienen en la recreación de un mismo espacio de combate se pueden dividir en varios niveles ([11]):

- **Nivel constructivo:** Engloba a los simuladores cuya finalidad es la dirección y control de una o varias operaciones militares.

- **Nivel virtual:** Se refiere a los simuladores cuya finalidad es el empleo y uso de un sistema de armas que puede involucrar a una o varias personas.

- **Nivel en vivo:** En este caso se trata de simuladores cuya finalidad es la instrucción individual del combatiente para mejorar una o varias habilidades específicas.

Un objetivo fundamental para las fuerzas armadas actuales es que los simuladores de un mismo nivel puedan funcionar de manera conjunta ([12]), de esta manera se podrían recrear operaciones militares completas en los niveles de abstracción que interesen. Una vez resuelto este problema, ya se podría plantear una situación ideal en la que también pudieran interoperar entre sí simuladores de diferentes niveles.

De momento la necesidad de instruir a los puestos de mando de unidades diferentes ha hecho que el funcionamiento distribuido en el nivel constructivo sea una prioridad. Además, como se mencionaba en la introducción de este capítulo, en este nivel la simulación puede ser empleada también como sistema de apoyo a la decisión del mando en una operación real. Para lograr esto es necesario que los simuladores y los sistemas de mando y control sean convergentes.

Por lo tanto, la interoperabilidad entre simuladores constructivos entre sí, y entre este tipo de simuladores y los sistemas de mando y control es una necesidad acuciante para la conducción de operaciones militares en la actualidad.

La interoperabilidad ha sido definida por la OTAN como "la capacidad que tienen los sistemas, unidades o fuerzas para suministrar y/o aceptar los servicios de otros sistemas, unidades o fuerzas y usar dichos servicios para operar conjuntamente de una forma efectiva"([13]).

Aunque no existen todavía modelos de interoperabilidad militar normalizados, sí que han tenido lugar en los últimos años varias iniciativas encaminadas a solucionar los problemas planteados por las necesidades de interoperabilidad antes descritas. Las más importantes han sido: HLA (High Level Architecture) para interoperabilidad entre simuladores ([14]) y CBML (Coalition Battle Management Language) y MSDL (Military Scenario Definition Language) para interoperabilidad entre simuladores y sistemas de mando y control. Estas iniciativas se describen y analizan en detalle en los próximos capítulos de este documento.

Se puede avanzar que el estándar HLA fue creado por el departamento de defensa de los Estados Unidos y que actualmente es mantenido por IEEE ([15], [16]). Se ha convertido en el estándar más empleado para simulación distribuida en entornos tanto civiles como militares ya que los simuladores que son compatibles con su definición pueden intercambiar información entre ellos para ejecutar una simulación conjunta.

En cuanto a CBML y MSDL, su utilización se limita exclusivamente a entornos militares. El primer estándar establece un lenguaje de batalla común entre los sistemas de mando y control y los simuladores en el nivel constructivo ([17]). CBML está basado a su vez en un estándar previo (JC3IEDM) que permite que los sistemas de mando y control interoperen entre ellos. Actualmente el JC3IEDM se encuentra en su versión III y es una puesta en común de la comunidad internacional que define toda la información de mando y control que es necesaria para la dirección y control de operaciones militares ([18]).

En cuanto a MSDL ([19]), se centra en la estandarización de la definición de los escenarios para operaciones militares. Hay que tener en cuenta que la definición de un escenario de combate supone la base para cualquier simulación distribuida militar. La definición y unificación de los mismos es una necesidad para cualquiera de los niveles de simulación ya que establece un punto de partida común para los simuladores que van a participar en un mismo ejercicio.

Aunque el estándar HLA debería garantizar, por lo menos, la interoperabilidad completa entre simuladores constructivos, la integración sencilla entre simuladores de diferentes ejércitos y naciones todavía no es una realidad. La complejidad (y en algunos casos ambigüedad) de HLA lleva a diferentes interpretaciones del estándar y a una falta de modelos de interoperabilidad que impiden en muchos casos la ejecución correcta de simulaciones distribuidas y la realización de ejercicios conjuntos.

Como se analizará más adelante, este tipo de problemas ya se están resolviendo

en entornos civiles (en concreto en los industriales), pero no así en los militares.

En cuanto a la interoperabilidad entre los simuladores y los sistemas de mando y control, la diferencia entre la arquitectura definida por el JC3IEDM y HLA hace que CBML ([20]) sea insuficiente para que los sistemas de mando y control y los simuladores constructivos sean interoperables entre ellos.

1.2. Hipótesis de partida

Del contexto presentado en la sección anterior surge directamente la hipótesis de trabajo para la presente tesis doctoral:

"Es posible proponer modelos, metodologías y herramientas que resuelvan los problemas de interoperabilidad entre simuladores constructivos de aplicación militar entre sí, y entre estos y los sistemas de mando y control."

Si se demuestra esta hipótesis de partida resolviendo los problemas de interoperabilidad ya mencionados, será posible que los sistemas de mando y control reduzcan su ciclo de decisión gracias al uso de simuladores constructivos que, alimentados con los datos del propio sistema de mando y control puedan proporcionar los resultados de cada una de las posibles líneas de acción, ayudando al mando a tomar una decisión de manera más rápida y teniendo en cuenta información mucho más elaborada. Además, será posible realizar ejercicios en los que un conjunto de simuladores constructivos recreen el teatro de operaciones virtual para el sistema de mando y control.

En este punto hay que señalar que la intención de esta tesis doctoral es conseguir interoperabilidad real entre los sistemas de apoyo a la conducción de operaciones militares. Es decir, no se trata de proponer la utilización de un middleware poco acoplado que actúe sólo como pasarela para el intercambio de informes y órdenes. Como se analizará en profundidad en el capítulo 2 de esta tesis, ese tipo de soluciones ya se han propuesto en trabajos previos pero imponen multitud de limitaciones en los entornos de trabajo actuales de las fuerzas armadas.

Además, hay que destacar que se pretende que las soluciones propuestas en esta tesis sean compatibles con el modelo JC3IEDM, que no sólo es un modelo relacional para intercambiar datos entre sistemas de mando y control sino que también especifica un conjunto de funciones y requerimientos (operativos y de sistema) asociados a estos datos y que plantean unos casos de uso aceptados por la comunidad internacional ([21], [22]).

1.3. Objetivos

Los objetivos generales de esta tesis doctoral surgen directamente de la hipótesis de partida planteada:

1. Definir modelos, metodologías y herramientas que posibiliten la interoperabilidad real entre sistemas de apoyo a la conducción de operaciones militares, en concreto, entre simuladores constructivos y sistemas de mando y control.
2. Validar la utilidad de las soluciones propuestas en escenarios reales.

Estos objetivos generales pueden desglosarse en otros más específicos divididos en dos grandes grupos: los objetivos asociados a la interoperabilidad de simuladores constructivos de aplicación militar entre sí, y los relacionados con la interoperabilidad entre estos simuladores constructivos y los sistemas de mando y control.

1. Interoperabilidad entre simuladores constructivos de aplicación militar entre sí.
    - Estudiar las limitaciones del estándar HLA en la consecución de interoperabilidad real entre simuladores de aplicación militar.
    - Analizar si estas limitaciones aparecen en otro tipo de aplicaciones (empresariales, industriales, médicas, etc); y si es así, estudiar las soluciones propuestas para determinar si es posible tomarlas como punto de partida para esta tesis doctoral.
    - Evaluar las necesidades específicas de interoperabilidad en entornos de simulación militar y determinar las características específicas de los simuladores constructivos en estos entornos.
    - Proponer modelos de interoperabilidad de referencia basados en HLA para simuladores constructivos de aplicación militar que permitan la integración automática y sencilla de este tipo de simuladores.
    - Comprobar que los modelos propuestos incluyen todos los casos de intercambio de información definidos en el JC3IEDM (estándar de interoperabilidad entre sistemas de mando y control).
    - Validar la utilidad de los modelos propuestos en un escenario real.
2. Interoperabilidad entre simuladores constructivos y sistemas de mando y control.

- Estudiar las limitaciones de las soluciones propuestas hasta el momento para resolver los problemas de interoperabilidad entre simuladores constructivos y sistemas de mando y control.
- Analizar si estas necesidades aparecen en otro tipo de aplicaciones que empleen sistemas de apoyo a la decisión (empresariales, industriales, médicos, etc); y si es así, estudiar las soluciones propuestas para determinar si es posible tomarlas como punto de partida para esta tesis doctoral.
- Evaluar las necesidades específicas de interoperabilidad en entornos de aplicación militar y determinar las características específicas de los simuladores constructivos y los sistemas de mando y control en estos entornos.
- Proponer modelos, metodologías y herramientas que permitan la interoperabilidad entre simuladores constructivos de aplicación militar y sistemas de mando y control.
- Comprobar que los modelos y herramientas propuestos incluyen todos los casos de intercambio de información definidos en el JC3IEDM y que son compatibles con los modelos propuestos en la primera parte de la tesis para la interoperabilidad entre simuladores constructivos entre sí.
- Validar la utilidad de los modelos propuestos en un escenario real.

1.4. Metodología y planificación

Ya que en esta tesis se abordan áreas de trabajo relacionadas con la informática (arquitectura de computadores, simulación distribuida y sistemas de información) y diferentes disciplinas militares, para proponer una metodología de trabajo adecuada que permita conseguir todos los objetivos propuestos en la sección anterior, se han utilizado aquellas etapas del método científico que generalmente son respetadas en la construcción y desarrollo de nuevas teorías sea cual sea el área de trabajo. Estas etapas son las siguientes:

1. **Observación:** El primer paso consiste en la observación de fenómenos dentro de una muestra. En el caso de la presente investigación, se han observado los entornos de aplicación militar en los que resulta necesaria la interoperabilidad entre simuladores constructivos entre sí o entre estos simuladores y los sistemas de mando y control. En concreto la observación ha sido un proceso participativo, involucrándose el autor de esta tesis con los hechos e interactuando con

las personas y situaciones objeto de estudio. Esta labor de observación se ha realizado a lo largo de un gran periodo de tiempo (años) gracias a la labor profesional del investigador. Se trata por tanto de una observación simple, realizada por una persona de cualificación adecuada para la misma, de forma consciente y sin prejuicios. Además en muchos casos ha sido una observación no participativa, de esta manera se ha intentado que los resultados de la observación fueran objetivos.

2. **Descripción:** El segundo paso consiste en realizar una detallada descripción del fenómeno observado. Para realizar esta descripción se han utilizado los resultados de la fase de observación y se ha recurrido también a técnicas de documentación. Se han completado los resultados de la observación previa con un exhaustivo análisis de todos los antecedentes y planteamientos previos conocidos. Esta etapa ha supuesto un trabajo de gabinete y biblioteca, consultando fuentes bibliográficas de las diferentes áreas de conocimiento antes mencionadas.

3. **Inducción:** En esta etapa se ha realizado la extracción de los principios y características generales implícitos en los fenómenos y situaciones observados y /o documentados. Gran parte de los resultados de esta etapa de la investigación se presentan en los capítulos siguientes de este documento.

4. **Planteamiento de la hipótesis:** A partir de la inducción se ha llegado a la hipótesis que sirve como punto de partida para el trabajo de investigación y que se ha expuesto en la sección 1.2.

5. **Discusión y razonamiento estructurado/Experimentación:** Comprobación de la hipótesis de partida con los medios apropiados para llegar a su demostración o refutación. Junto con los resultados de la etapa de Inducción, los resultados de esta fase constituyen la mayor parte de los contenidos de este documento.

6. **Comparación universal:** Constante contraste de la hipótesis con la realidad una vez que se ha demostrado (o refutado).

Esta metodología general sirve de guía a lo largo de todo el trabajo de investigación realizado pero debe traducirse en una planificación más concreta. Como se puede observar en la sección anterior, existe un paralelismo bastante claro entre las dos partes de esta tesis doctoral (la primera se centra en la interoperabilidad entre

simuladores y la segunda en la interoperabilidad entre simuladores y sistemas de mando y control), lo que ha permitido realizar una planificación similar para ambas:

- Justificación de la necesidad del tipo de interoperabilidad estudiada en entornos militares.
- Estudio y análisis exhaustivo de los antecedentes en la resolución de ese tipo de problemas de interoperabilidad, en entornos militares y en otros típicos como los empresariales o los industriales.
- Identificación de las características y necesidades específicas de los entornos de conducción de operaciones militares.
- Definición de modelos, metodologías y herramientas específicas para estos entornos, compatibles con HLA, la especificación JC3IEDM, los estándares CBML y MSDL, y con otros estándares de facto en integración como XML, las arquitecturas orientadas a servicios, etc. Además, las soluciones propuestas deben basarse en la federación de los sistemas entre sí y no en la utilización de un middleware poco acoplado que actúe sólo como pasarela para el intercambio de datos. Por último deben tener en cuenta los mecanismos de tolerancia a fallos, críticos en el tipo de entorno de investigación considerado.
- Verificación y validación de las soluciones propuestas, así como de su viabilidad en escenarios reales.

1.5. Estructura del documento

El resto del presente documento se estructura de la siguiente manera.

En el Capítulo 2 se resumen los resultados de las fases de Observación y Descripción de esta tesis doctoral, proporcionando el marco teórico de la investigación realizada. En concreto se presentan los modelos, soluciones, estándares y herramientas que se han empleado en el pasado para resolver los problemas de interoperabilidad en entornos militares.

En el Capítulo 3 se comienza con la primera parte de la tesis doctoral, por lo que se definen los modelos de interoperabilidad de referencia para simuladores constructivos de aplicación militar, todos ellos compatibles con la especificación JC3IEDM ([23]).

A continuación, en el Capítulo 4, se demuestra la validez y viabilidad de estos nuevos modelos modificando una implementación real de HLA de manera que pueda asegurar la perfecta ejecución de los modelos definidos en el capítulo anterior.

En el Capítulo 5 se propone una arquitectura para interoperabilidad entre simuladores constructivos y sistemas de mando y control, pasando por tanto a la segunda parte de la tesis. Para ello se tienen en cuenta la especificación JC3IEDM ([23]), el estándar IEEE1516 ([15]) y los modelos propuestos en el Capítulo 3.

Posteriormente, en el Capítulo 6, se demuestra de nuevo la validez y viabilidad de la arquitectura definida en el capitulo anterior aplicando todas las definiciones en un escenario militar real.

Por último en el Capítulo 7 se resumen las principales conclusiones de esta investigación así como las líneas de trabajo futuro más interesantes que surgen de ella.

Esta tesis doctoral incorpora además dos anexos documentales en forma de apéndice que sirven para clarificar y/o ampliar ciertos aspectos concretos surgidos a lo largo del documento.

Capítulo 2

Antecedentes

Las unidades militares pueden verse envueltas en la actualidad en diferentes misiones de naturaleza muy diversa, lo que conlleva una composición heterogénea de las mismas para dar cobertura a todo su ámbito de actuación. Esto hace que el mando y control de unidades militares sea muy complejo, lo que supone que los cuarteles generales y las planas mayores de mando de los mandos militares estén compuestos por un gran número de secciones que se encargan de temas distintos, pero interrelacionados de manera muy estrecha entre sí. Y por tanto se requiere un alto grado de coordinación para lograr una óptima conducción de las actividades de la unidad, no sólo entre las diferentes unidades sino también entre los distintos sistemas de información que manejan.

En este capítulo se resumen los conceptos más importantes relacionados con interoperabilidad entre simuladores de aplicación militar, entre sistemas de mando y control (C2), y entre simuladores y sistemas de mando y control.

También se presentan las soluciones que hasta el momento se han utilizado para conseguir la integración de este tipo de sistemas así como las iniciativas que en la actualidad se están llevando a cabo en esta dirección.

2.1. Importancia de la simulación en entornos militares

La instrucción de los ejércitos en campos de maniobras, necesaria para conseguir un alto grado de coordinación en la conducción de operaciones, es muy costosa tanto en recursos humanos como materiales, y además difícilmente aporta el grado

de realismo deseado. Esto es así porque este tipo de instrucción necesita largos tiempos de planeamiento (normalmente ciclos anuales para coordinar el uso del campo de maniobras entre unidades y hacer previsión de créditos), lo que implica que el entrenamiento de operaciones no previstas sea muy laborioso y la mayoría de las veces, inviable ([1]).

Por estos motivos el empleo de otros métodos para poder instruir a las unidades de manera más realista, con mayor tolerancia a imprevistos y que además tengan un coste menor, es una capacidad imprescindible para los ejércitos modernos ([8], [24]).

A continuación se detallan las ventajas que tiene el uso de simuladores frente a la instrucción clásica en el campo de maniobras ([3], [25]):

- Aunque la simulación supone un coste inicial elevado, se amortiza en poco tiempo.

- La tolerancia a imprevistos es mucho mayor. Con el uso de simuladores se permite la instrucción de la unidad dentro de la propia base, introduciendo todas las variables que se puedan encontrar en una situación real de combate. De esta forma se reduce el tiempo de planeamiento y preparación de ejercicios.

- Se perfeccionan la toma de decisiones y los procedimientos empleados por las unidades mediante la evaluación posterior de los mismos gracias a todos los datos e información almacenados durante la simulación. El concepto de realismo siempre es discutible. Si bien la existencia de enemigo (sea humano, sea máquina) produce una mejor evaluación de las decisiones tomadas, en la realización de ejercicios, sean del tipo que sean, nunca se llegará a alcanzar el realismo de una situación de combate real.

Por las razones expuestas hasta el momento, el uso de simuladores en las fuerzas armadas se ha visto incrementado en los últimos años, lo que ha supuesto una mejora en la calidad de la instrucción de las unidades, preparándolas para tener un rendimiento y coordinación elevado en situaciones de combate real.

2.2. Clasificación de simuladores en entornos militares

Si se investiga en las fuentes que abordan el tema de la simulación, se encontrarán diferentes tipos de clasificaciones teniendo en cuenta diversos criterios (por su finalidad, por su composición, etc.). Cada organización toma como referencia la que más se adapta a su estructura, misión y objetivos.

El criterio de clasificación de más interés para el desarrollo de esta investigación es el que se sigue para dividir los simuladores dedicados a temas administrativos y los empleados para ámbitos tácticos. De esta forma podemos dividir los simuladores de aplicación militar en dos categorías.

2.2.1. Simuladores administrativos

Los simuladores administrativos son todos aquellos que no son empleados para la simulación de funciones de combate. Se utilizan en aquellas actividades necesarias para el funcionamiento administrativo de la organización militar ([26], [27]).

En esta división se encuentran aquellos sistemas encargados de evaluar el impacto que tienen en la organización determinadas decisiones administrativas, por ejemplo, para optimizar los procedimientos empleados o para evaluar sus aspectos sociológicos. Un ejemplo dentro de las fuerzas armadas de este tipo de simuladores son los simuladores de evolución de personal ([1], [28]).

Los simuladores administrativos son empleados en muchas otras organizaciones, por lo que su ámbito de aplicación no es exclusivo de las fuerzas armadas.

2.2.2. Simuladores tácticos

Dentro de este conjunto se engloban aquellos simuladores que tienen como objeto las funciones de combate.

Aunque estas actividades son exclusivas de los ejércitos, es verdad que se pueden encontrar ciertas similitudes con las realizadas por otras organizaciones ajenas a éstos (por ejemplo, simuladores de la función logística de abastecimiento).

Este tipo de simuladores se puede subdividir en tres subtipos (figura 2.1) cuyo criterio de clasificación coincide con los niveles de instrucción definidos en la fuerzas armadas (instrucción de puestos de mando, instrucción de sistemas de armas e

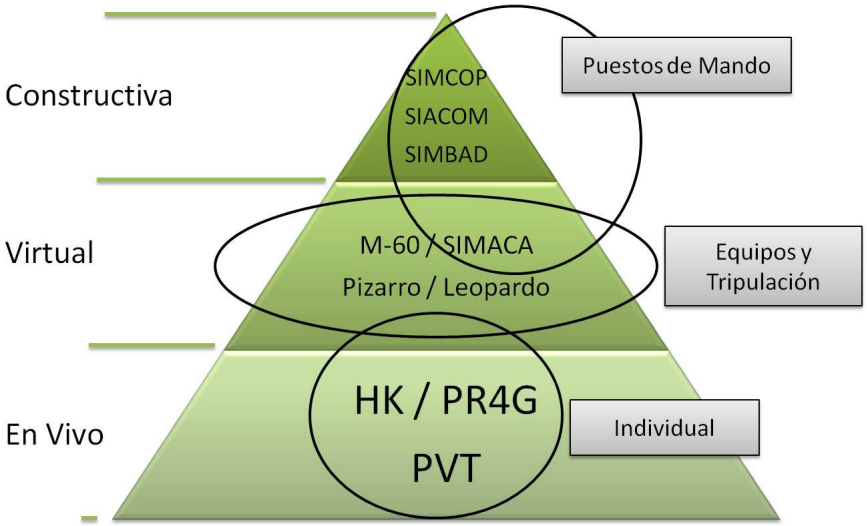

Figura 2.1: Pirámide de simulación y ejemplos concretos de simulación del ejército de tierra español ([11])

instrucción individual [11], [29]).

- **Simuladores constructivos.** Se define un simulador constructivo como "el conjunto de sistemas y modelos en los que actores simulados operan sistemas simulados de manera que las personas estimulan el sistema pero no intervienen en la elaboración de los resultados ([11])".

  Este es el tipo de simulador que se emplea dentro de las fuerzas armadas para recrear las actividades realizadas por los cuarteles generales y planas mayores de mando. Es decir, dentro de esta categoría se encuentran todos aquellos simuladores cuyo objetivo es el mando y control de las operaciones militares. El uso de este tipo de simulación es muy importante para entrenar a los puestos de mando en el planeamiento y la conducción de operaciones, ya que a través de las evaluaciones de las mismas se puede determinar si se ha tomado o no la mejor decisión en cada caso.

  Dentro de este conjunto se pueden realizar subdivisiones, dependiendo del ámbito al que vayan dirigidos. Por ejemplo en la figura 2.2 se puede observar la subdivisión que existe en el ejército de tierra español.

  Como se introdujo en el capítulo 1, este es el tipo de simulador en el que se centra la investigación realizada en esta tesis doctoral.

- **Simuladores virtuales.** Son aquellos simuladores que tienen como objetivo

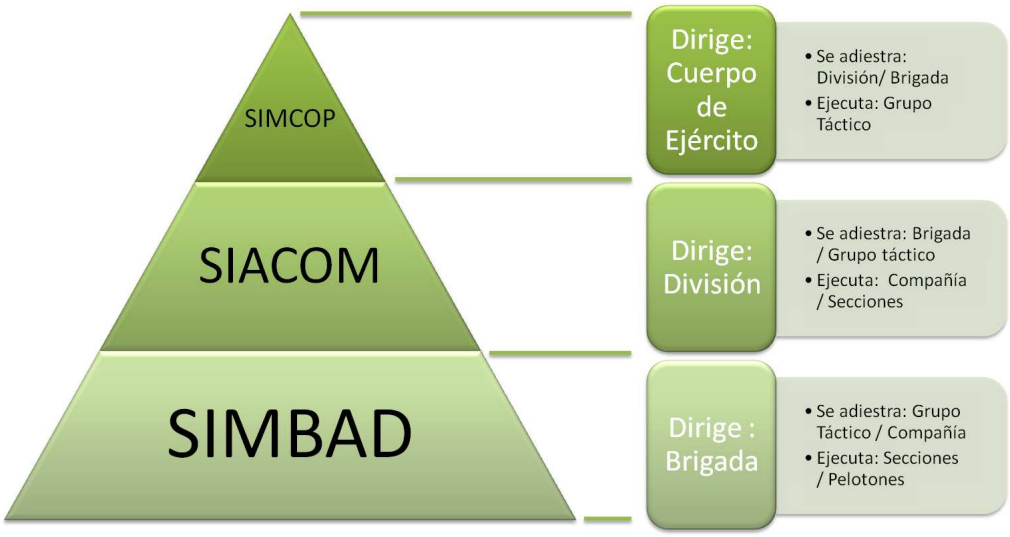

Figura 2.2: Niveles de simulación constructiva dentro del ejército de tierra español ([11])

la instrucción de tripulaciones y equipos. Por lo tanto en esta categoría se enmarcan las acciones colectivas más básicas realizadas en unidades de combate, de apoyo al combate y apoyo logístico al combate.

Su objeto es la instrucción en un tipo determinado de sistemas de armas, por lo tanto, su marco de actuación aparece en pequeñas unidades cuyo armamento requiere de una especialización por parte de sus operadores.

Sería un error identificar este tipo de simuladores con el término pequeño, dado que existen sistemas de armas que involucran a un equipo muy importante de personas en diferentes localizaciones. Por ejemplo el SIMACA o Simulador de Artillería de Campaña, es un simulador de tiro de Artillería de Campaña e interviene todo el personal de una unidad militar tipo batería ([30]).

- **Simuladores en vivo.** Su objetivo es la instrucción individual del combatiente. Suelen ser ejecutados de manera aislada puesto que el objetivo es enseñar al usuario actividades de instrucción individual. Un ejemplo de este tipo de simuladores, sería el módulo de tiro del simulador VBS2 ([31]).

2.3. Necesidad de interoperabilidad entre simuladores de aplicación militar

En un principio los simuladores recreaban sistemas individuales sin tener en cuenta que otros sistemas externos podrían afectar a sus resultados. Esto era debido a las limitaciones de capacidad de cómputo y memoria en los sistemas informáticos disponibles para las simulaciones y a la falta de infraestructura para que diferentes simuladores pudieran trabajar interconectados.

Hoy en día no existen estas limitaciones, y el trabajo conjunto de varios simuladores individuales obtiene como resultado recreaciones de sistemas mucho más complejos compuestos por varios sencillos ([32]), y una potencia de cómputo y memoria mucho mayor al unir varios sistemas a una misma simulación, que permite que los resultados obtenidos sean más completos y realistas.

Por estos motivos el empleo de arquitecturas distribuidas en el campo de la simulación se ha visto incrementado en los últimos años ([33]) aumentando la calidad de los resultados obtenidos.

En el ámbito militar no sólo es necesaria la interoperabilidad entre simuladores para la consecución de los objetivos anteriormente mencionados. La participación de los ejércitos en operaciones conjuntas en las que las fuerzas armadas de diferentes países emplean simuladores diferentes, hace necesario que los sistemas sean interoperables entre sí para entrenar operaciones militares usando los métodos de la simulación distribuida ([5]).

Por ejemplo, la OTAN realiza ejercicios CAX (Computer Assisted eXercises, [34]) para la instrucción de sus unidades tipo NRF (NATO Response Force), en los que se sustituye el despliegue real de las tropas por una simulación. El uso de simuladores interoperables de los diferentes países participantes en un núcleo NRF aumenta la calidad de dichos ejercicios, evaluando y optimizando los procedimientos utilizados en la fase de planeamiento y ejecución de operaciones internacionales. En la figura 2.3 se puede observar la concepción de un ejercicio CAX en el ámbito OTAN ([35]). Aquí se definen un escenario y varios simuladores federados que simulan células de respuestas pertenecientes a un NRF.

La necesidad de un funcionamiento convergente de los simuladores militares en entornos de simulación distribuida ha provocado el desarrollo de varios estándares cuyo máximo exponente es HLA (High Level Architecture, [14]). Este tipo de arquitectura, que actualmente es la más difundida en entornos militares pero también en

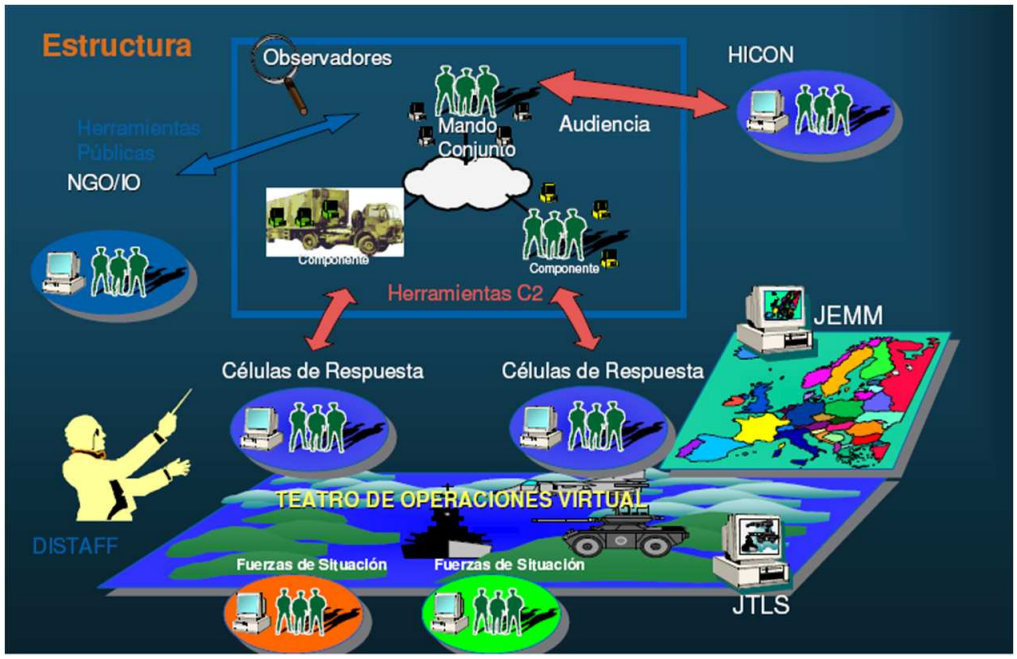

Figura 2.3: Concepto CAX en el ámbito OTAN ([35])

entornos civiles, ha generado alrededor de ella un gran número de nuevos estándares, el más destacado, el estándar IEEE1516 ([15]).

En organizaciones militares como la OTAN, HLA ha adquirido un lugar preferente en aquellos grupos que estudian la interoperabilidad entre simuladores de aplicación militar (por ejemplo, el RTO NATO Modelling and Simulation Group, [36]), contando actualmente con un alto grado de apoyo por parte de todos los países miembros y siendo, de hecho, un estándar de facto dentro de las fuerzas armadas.

2.4. High Level Architecture: HLA

HLA es la arquitectura más difundida dentro del ámbito de la interoperabilidad entre simuladores. Conocer sus orígenes, su evolución y su situación actual, es imprescindible para el desarrollo de cualquier investigación relacionada con la interoperabilidad entre o con sistemas de simulación.

2.4.1. Antecedentes

DIS (Distributed Interactive Simulation) fue un protocolo muy difundido (creado por DARPA en 1983) que llegó a ser un estándar de IEEE, DoD (Department of Defense en los Estados Unidos) y OTAN ([37]). Este protocolo permitía que varias simulaciones interactivas funcionaran de manera coordinada definiendo para ello vehículos, sensores, colisiones y eventos. El inconveniente principal de este protocolo era que estaba basado en datagramas para el intercambio de información y los simuladores miembros de la simulación distribuida recibían toda la información contenida en la PDU (Protocol Data Unit) ([38]), mucha de la cual no era de interés para ellos ([39]). En consecuencia, el envío de información no era selectivo y no se realizaba una utilización eficiente de los recursos de cómputo y comunicaciones involucrados en la simulación.

Por este motivo en 1995 el Departamento de Defensa de los Estados Unidos creó una arquitectura llamada HLA ([40]), que fue rápidamente aceptada por la comunidad internacional hasta convertirse en uno de los estándares de simulación más difundidos. HLA pronto relegó a un segundo plano a DIS, mejorando las características menos optimizadas de este primer protocolo.

En la actualidad, la mayoría de los trabajos realizados en el ámbito de la interoperabilidad entre simuladores se basan en HLA. La última versión, desarrollada por IEEE, es una evolución de la versión 1.3 ([14]), y se compone de cinco especificaciones: IEEE1516, IEEE1516.1, IEEE1516.2, IEEE1516.3, IEEE1516.4. En estos momentos el estándar se encuentra en revisión para añadir extensiones que permitan su empleo sobre tecnologías orientadas a servicios ([41]).

Aunque HLA fue creado con fines claramente militares, en la actualidad su uso se ha extendido a todos los ámbitos donde el trabajo conjunto entre diferentes simuladores es necesario. Organizaciones como la OTAN (referente dentro del ámbito militar), pero también civiles, están realizando un gran esfuerzo por impulsar estudios que desarrollen este estándar hasta alcanzar el nivel de interoperabilidad deseado de manera transparente al usuario ([26]).

Dentro del entorno militar un ejemplo claro es el proyecto Pathfinder ([11], figura 2.4), escenario en el que varios países han establecido una red de simulación usando la arquitectura HLA (en su versión 1.3). La falta de un modelo de interoperabilidad normalizado para los FOM (Federation Object Model, [16]) de simuladores constructivos ha sido una de las lecciones aprendidas más importantes del mismo ([42]). Por este motivo, el grupo de trabajo MSG-58 (perteneciente al grupo de modelado

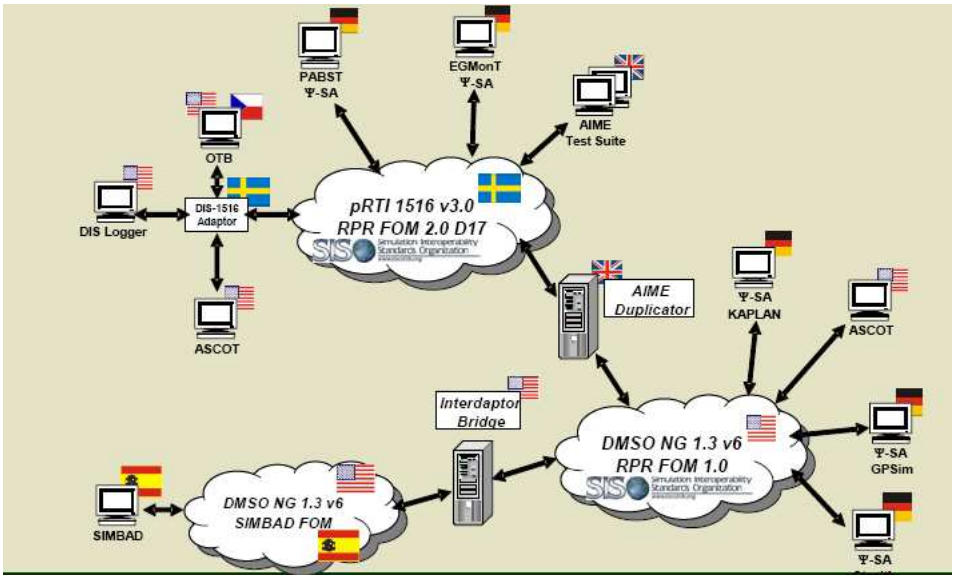

Figura 2.4: Participación en proyecto Pathfinder ([35])

de simulación del RTO, [36]) está elaborando un FOM estándar para su uso dentro del ámbito de la OTAN.

Este tipo de ejercicios y proyectos convierten a HLA en un estándar vivo y dinámico que todavía debe mejorarse en muchos aspectos. Por ello las unidades OTAN siguen sin realizar ejercicios de simulación distribuida basados en simuladores nacionales, algo que es posible en el plano teórico gracias a HLA, pero que presenta en estos momentos multitud de problemas en el plano más práctico. Probablemente uno de los más críticos sea la falta de diseños de interoperabilidad comunes.

En el entorno civil la referencia está en la organización SISO (Simulation Interoperability Standards Organization, [43]) cuyo ámbito de actividad es la interoperabilidad entre simuladores.

SISO, a través de sus diferentes grupos de trabajo, estudia y mantiene estándares relacionados con la interoperabilidad entre simuladores ([44]) de diferente aplicación: militar, empresarial, industrial, etc. Algunos de ellos son de especial relevancia para el modelado y desarrollo de sistemas de simulación basados en la arquitectura HLA ([45]), y en los diferentes escenarios en los que estos estándares se han puesto en funcionamiento se han encontrado los mismos problemas y limitaciones ya mencionados para el ámbito militar.

2.4.2. Estándares asociados a HLA

La generalización del empleo de HLA en sistemas de simulación tanto civiles como militares ha hecho que, finalmente, el mantenimiento y desarrollo de sus estándares asociados recaiga sobre una organización global como IEEE. En el ámbito militar, la OTAN ha adoptado sin problemas los productos desarrollados por IEEE como referencia (STANAG 4603: [46], [47]).

El estándar HLA definido por IEEE se divide en las siguientes especificaciones:

- **HLA framework and rules (IEEE std. 1516, [14]).** En esta especificación se definen diez reglas (cinco para federaciones y cinco para federados) para enmarcar las responsabilidades de federados y federaciones y así asegurar su correcta interacción.

- **HLA federate interface specification (IEEE std. 1516.1, [15]).** Esta especificación es la más voluminosa porque en ella se definen los servicios e interfaces que debe tener implementada una RTI (IEEE1516 se basa en el concepto de Messages Oriented Middleware y la RTI es el elemento que se encarga de sincronizar la transferencia de mensajes). De esta manera se asegura el correcto funcionamiento de las llamadas por parte de los federados a la RTI y las *callback* de ésta hacia los federados. También se define el MOM (Management Object Model), OMT (Object Model Template) usado por la RTI por defecto.

- **HLA object model template (OMT-IEEE std. 1516.2, [16]).** En esta especificación se describe el OMT, modelo que siguen los datos que se intercambian entre los miembros de una federación. Aquí queda definido el SOM (Simulation Object Model), que es el modelo de objetos de un federado y el FOM (Federation Object Model), modelo de objetos de una federación y por lo tanto dependiente de los SOMs de los federados que forman parte de ella.

- **Federation development and execution process (FEDEP-IEEE std. 1516.3, [48]).** Esta especificación define el FEDEP, que es el proceso recomendado para desarrollar una federación ([47]). En él se identifican siete fases, que sirven como guía a la hora de desarrollar una federación. Cabe destacar la importancia de este proceso de referencia a la hora de analizar la interoperabilidad entre simuladores, puesto que el estudio de ésta afecta a cada una de las fases que componen el FEDEP (figura 2.5).

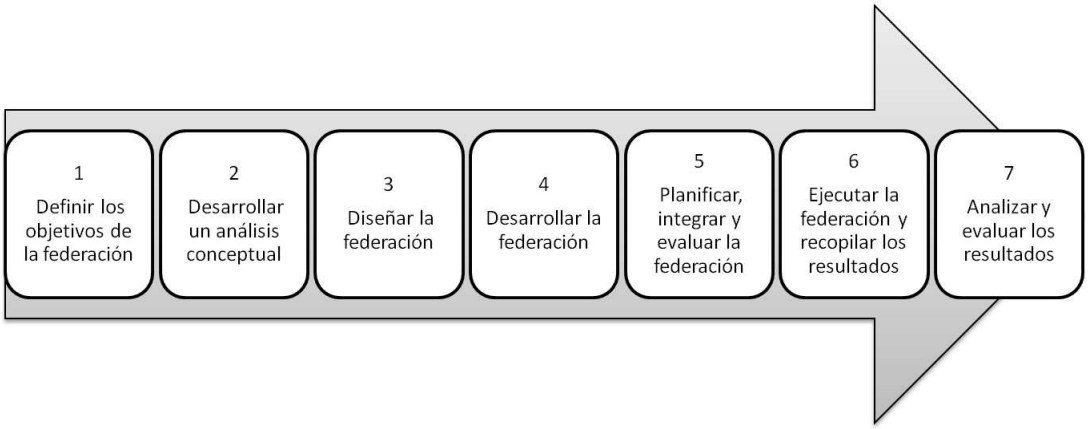

Figura 2.5: Fases del proceso de desarrollo de una federación: FEDEP ([48])

- **Verification, validation and accreditation of a federation (IEEE std. 1516.4, [49]).** Esta especificación es la más reciente y define el proceso recomendado para la verificación, validación y acreditación de una federación. Estas actividades deben ser realizadas a lo largo de todo el proceso FEDEP (figura 2.6), referido en el estándar anterior. Este proceso es vital para describir y descartar federados que no cumplan el estándar y por lo tanto, que produzcan un problema de interoperabilidad en la federación. También cuando se trabaja entre varias federaciones, este proceso sirve para comprobar si sus RTI cumplen el estándar de manera que se asegure su integración.

El impacto que las nuevas tecnologías orientadas a servicios podrían tener en la arquitectura HLA, está siendo objeto de estudio en la actualidad ([50]). La ampliación del estándar IEEE1516 para que se vea mejorado con el uso de estas tecnologías está siendo considerado seriamente por parte de IEEE ([50], [51], [52]).

Estos estándares son el núcleo de definición de HLA, pero como se comentaba con anterioridad, conseguir que la interoperabilidad entre simuladores sea una realidad exige una constante actualización y mejora de esta arquitectura inicial.

SISO, como organización cuyo objetivo es estudiar la interoperabilidad entre simuladores, mantiene en la actualidad varios estándares asociados a este problema, siendo los más importantes para esta investigación:

- **Based Object Model (BOM, [45]).** Los datos que se intercambian en una federación siguen un modelo de objetos OMT. En esta especificación se

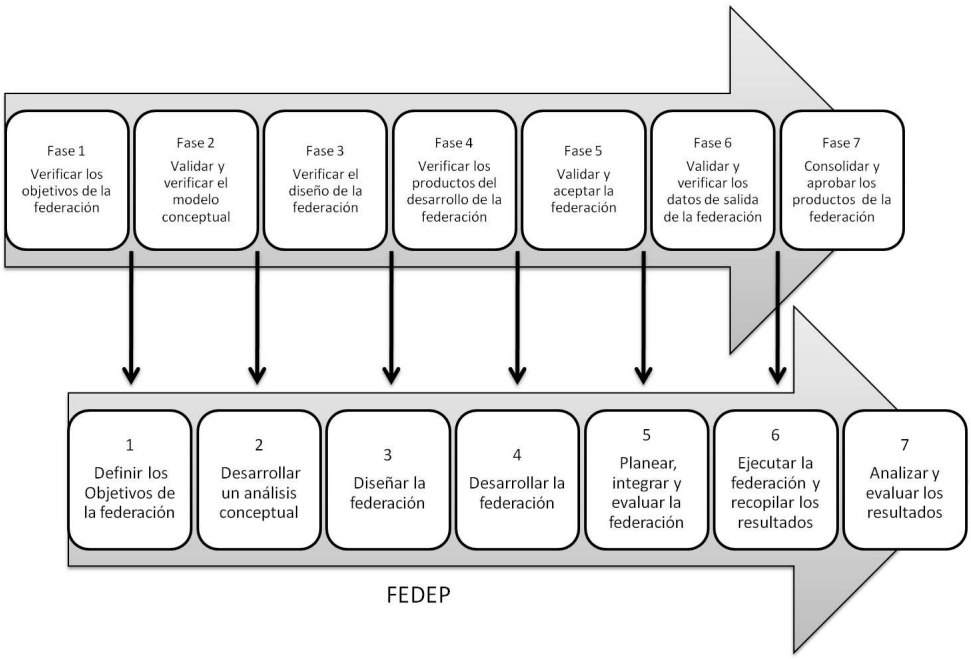

Figura 2.6: Actividades de verificación, validación y acreditación dentro del FEDEP ([49])

desarrolla el estándar IEEE1516.2 y se definen los tipos y categorías de los elementos que pueden formar un BOM ([53], [54]).

- **Battle Management Language (BML).** Este tipo de especificación tiene por objeto el intercambio de información entre simuladores y sistemas de mando y control en entornos militares ([55], [56]). Se trata de definir un lenguaje de batalla común (CBML, [57]), ampliando los modelos de datos de intercambio de información entre sistemas de mando y control de la OTAN.

- **Military Scenario Definition Language (MSDL, [19], [55]).** Este estándar es una referencia para la creación de escenarios en sistemas tácticos militares ([58]), con el objeto de establecer un escenario inicial común para todos los sistemas participantes en un mismo ejercicio sea cual sea su origen.

- **Commercial Off-the-Shelf (COTS) Simulation Package Interoperability Reference Models (CSPI, [59]).** El ámbito de este estándar son los simuladores basados en sistemas COTS dirigidos generalmente a la simulación industrial. En el estándar se definen cuatro modelos de interoperabilidad de referencia para automatizar el funcionamiento conjunto de varios simuladores dentro de una simulación distribuida.

2.4.3. Heterogeneidad de las implementaciones del estándar IEEE1516

Hoy en día existen multitud de implementaciones del estándar IEEE1516. Cada organización, para desarrollar sus sistemas de simulación utiliza una RTI (Run Time Infrastructure) ya desarrollada o crea la suya propia. Aunque la mayoría de ellas siguen el estándar (lo cual debería asegurar que fueran interoperables), las interpretaciones del mismo pueden diferir dadas sus ambigüedades y complejidad, lo que se traduce en problemas de interoperabilidad. Esta problemática, mencionada anteriormente, ha sido objeto de numerosos estudios intentando buscar una solución estándar global todavía no encontrada.

Las implementaciones del estándar HLA se suelen componer de una RTI y sus interfaces de acceso, basados en unas librerías que dan soporte a uno o varios lenguajes de programación. Es inevitable pensar que la implementación de la RTI elegida impone los lenguajes de desarrollo de los simuladores e incluso, en algunos casos, las plataformas sobre las que pueden funcionar.

La mayoría de las librerías de las RTI implementadas dan soporte para los lenguajes de programación Java, C++ y algunas de ellas, también para C# ([60]). La gran parte de los simuladores militares están desarrollados en estos lenguajes de programación, de forma que estas librerías dan una amplia cobertura para estos sistemas.

Sin embargo, en otros sectores donde la simulación es una práctica habitual, como pueda ser el industrial, los simuladores no suelen estar programados en este tipo de lenguaje de alto nivel sino que emplean habitualmente paquetes de simulación de tipo COTS (Commercial Off-The-Shelf). Ejemplos de este tipo de paquetes son Arena, Extend, SLX, Simplex 3, QUEST ó IGRIP ([32], [60]). La mayoría de estos paquetes implementan HLA para que sea posible construir con ellos simulaciones distribuidas de manera que sea lo más transparente posible para el programador. Muchas veces se añaden capas intermedias que abstraen al desarrollador del estándar IEEE1516 ([61]), de manera que una única sentencia de un módulo HLA en estos paquetes puede englobar en realidad a varias llamadas de las definidas en el estándar.

Las diferentes interpretaciones en el desarrollo de las RTI existentes han generado problemas de compatibilidad entre ellas ([62]), igual que las diferencias existentes entre la utilización de HLA cuando los simuladores están programados en lenguajes de propósito general o utilizando paquetes COTS. Por este motivo grupos de trabajo de organizaciones como SISO o NATO Modelling and Simulation Group (RTO), han

estudiado cómo hacer converger estas diferentes implementaciones para encontrar una verdadera interoperabilidad.

En resumen, se intenta definir un manera estándar de utilizar el estándar HLA para que la interoperabilidad entre simuladores sea prácticamente automática y transparente a los desarrolladores y usuarios. El objetivo es alcanzar simulaciones distribuidas de tipo Plug-and-Play ([60]) en las que los diferentes simuladores puedan conectarse y desconectarse en caliente y de forma muy sencilla.

Este objetivo está más cerca en el ámbito industrial gracias al estándar CSPI de SISO, que define cuatro formas estándar de utilizar HLA en las situaciones de interoperabilidad más comunes dentro de la simulación industrial basada en paquetes COTS, pero el problema todavía no está resuelto en otros entornos como pueda ser el militar.

2.4.4. Situación actual de la simulación distribuida en entornos militares

Como se ha mencionado con anterioridad, el desarrollo de operaciones multinacionales donde las unidades militares que intervienen están formadas por fuerzas de varios países, ha generado la necesidad de una instrucción conjunta de las operaciones. Los sistemas de simulación como herramientas de ayuda a la instrucción, no son ajenos a esta necesidad. Es por ello que la interoperabilidad entre los simuladores de diferentes ejércitos se ha convertido en una necesidad que ha llevado a la OTAN a estandarizar HLA ([29], [63]).

Como arquitectura distribuida, HLA permite siempre que se utiliza, aumentar el rendimiento global de la simulación y hacer que varios simuladores individuales puedan intercambiar información a través de la RTI, y por lo tanto, que sean interoperables ([64]).

En el seno de la OTAN se han realizado ejercicios de integración de los simuladores constructivos de diferentes países usando la arquitectura HLA ([35]) para evaluar su utilidad en el entrenamiento de las unidades pertenecientes a esta organización. La conclusión siempre ha sido que al estándar todavía le falta madurez para resolver de manera automática, sencilla y transparente la interoperabilidad entre diferentes simuladores.

La simulación distribuida en entornos militares no sólo tiene como aplicación el entrenamiento en ambientes multinacionales ([65]). Dentro de un mismo ejército

existen también necesidades de interoperabilidad entre los diferentes simuladores que se tienen en servicio, por ejemplo, para los diferentes ejércitos. Pero los problemas son exactamente los mismos que en entornos multinacionales, al final casi siempre es necesario modificar el desarrollo de los simuladores y finalizar la integración entre ellos de forma manual para que ésta sea correcta.

HLA no es la única arquitectura distribuida sobre la que se han realizado pruebas de interoperabilidad para simuladores constructivos. Sistemas de simulación que emplean los servicios web para intercambiar información entre ellos en formato JBML (implementación del estándar CBML en lenguaje Java) también han sido probados entre varios países de la OTAN ([55], [66]). Pero todavía no existe ningún estándar asociado a este tipo de integración, que además parece que está lejos de ser soportada por todos los simuladores ya que la orientación a servicios no está todavía muy extendida en este tipo de aplicaciones.

De los resultados obtenidos en entornos militares, tanto nacionales como multinacionales, se deduce que existe una necesidad imperiosa de definir modelos de interoperabilidad de referencia para simuladores constructivos de aplicación militar de la misma manera que el estándar CSPI de SISO ([59]) lo ha hecho para simuladores de aplicación industrial. Las principales diferencias con estos modelos son las situaciones específicas en las que esta interoperabilidad es necesaria, y que los simuladores de aplicación militar no están programados utilizando paquetes COTS sino lenguajes de propósito general.

2.5. Importancia de los sistemas de mando y control en entornos militares

Los procedimientos para el mando y control (Command and Control ó C2) de unidades militares han existido desde siempre en los ejércitos. La diferencia entre los tradicionales y los actuales radica en que antes, con sistemas procedimentales se conseguían los objetivos deseados dada la naturaleza de los combates que tenían lugar. En la actualidad se necesitan sistemas de información que organicen el gran volumen de datos generado, en detrimento de los procedimientos, que no resultarían efectivos en los nuevos escenarios ([67]).

El concepto de combate simétrico, que consiste en dos o varios ejércitos con modelos estratégicos similares que se enfrentan de manera directa utilizando unidades del tipo acorazadas/mecanizadas en su mayoría, ha cambiado en los últimos años

debido al nuevo concepto geoestratégico. Las operaciones militares actuales consisten casi siempre en pequeñas unidades dispersas en el terreno, sin un enemigo visible y que pueden verse expuestas a situaciones muy cambiantes en poco tiempo ([68]). Por este motivo los procedimientos tradicionales para ejercer el mando sobre las unidades han necesitado una revisión profunda.

El resultado ha sido que la unidad que tiene el ciclo de decisión más rápido adquiere ventaja sobre las demás, lo que conlleva en la mayoría de los casos, imponerse al resto ([69]) en el combate. Para aumentar la velocidad del ciclo de decisión, el mando debe disponer de la máxima información posible acerca de su espacio de batalla, de manera ordenada y jerarquizada. El nuevo marco de operaciones hace que el mando acceda fácilmente a multitud de información pero descontextualizada, lo que provoca que la línea de acción seleccionada tenga en cuenta pocos parámetros, en muchos casos obsoletos y sin relacionar con los demás.

Para tener en cuenta información actualizada y dentro de un contexto, que en la mayor parte de los casos puede ser vital, se han comenzado a emplear nuevos procedimientos ([5]). Estos se basan en sistemas de información específicos, sistemas de mando y control, que permiten que la difusión de la información sea lo más rápida y ordenada posible, con el objetivo de hacer este intercambio de manera eficaz, de convertir datos en bruto en información relacionada, valiosa y en última instancia, de reducir el ciclo de decisión mediante sistemas que puedan asesorar a los mandos.

La primera generación de estos sistemas consistía en bases de datos que replicaban datos entre sí, de tal manera que el mando de la unidad recibía toda la información de sus unidades subordinadas de forma ordenada. En un primer momento este concepto de sistema de mando y control dio solución a las nuevas necesidades de los ejércitos.

Sin embargo, la rapidez con la que cambia la situación en los combates actuales, demanda un ciclo de decisión aún más rápido que el conseguido hasta ese momento ([69]). Por este motivo la segunda generación de sistemas añadió a la anterior sistemas de apoyo a la decisión del mando, es decir, no sólo se trata de presentar la información de manera ordenada, sino de asesorar al mando en función de esta información y de su contexto global acerca de la mejor línea de acción posible en ese momento.

2.6. Funcionalidades y clasificación de los sistemas de mando y control

El diferente concepto de la defensa que cada una de las naciones maneja desemboca en unas necesidades diferentes en lo que se refiere a sistemas de información, por lo que los diseños los sistemas de mando y control pueden variar mucho. De hecho, dentro de un mismo país puede haber diferentes tipos de sistemas para cada uno de los ejércitos, al ser sus misiones y objetivos distintos.

Teniendo en cuenta la misión a la que da cobertura el sistema de mando y control se puede realizar una primera clasificación dividiéndolos en los siguientes grupos:

- **Sistemas de mando y control para el ejército de tierra.** Este tipo de sistemas están diseñados para el mando y control de operaciones terrestres, lo que no implica que sean de uso exclusivo para el ejército de tierra.
- **Sistemas de mando y control para el ejército del aire.** Este tipo de sistemas van dirigidos a la conducción de operaciones aéreas (en este grupo se incluyen los sistemas para la gestión del espacio de batalla aérea de las unidades antiaéreas del ejército de tierra).
- **Sistemas de mando y control para la armada.** En este tipo de sistemas se incluyen aquellos dirigidos a mando y control de operaciones marítimas, anfibias y aquellas que desde tierra van dirigidas a apoyarlas.

Si se considera el nivel de la operación a la que va dirigido el sistema, se pueden distinguir los siguientes grupos:

- **Sistemas de nivel táctico.** El ámbito de este tipo de sistemas son las operaciones de más bajo nivel ([69]). Por tanto son empleados por unidades de tipo batallón o inferiores, y normalmente su finalidad última es controlar las actividades de alguna función de combate. En consecuencia, este tipo de sistemas suele estar ligado a las actividades concretas del tipo de unidad al que van dirigidos.
- **Sistemas de nivel operacional.** Estos sistemas están en un nivel superior al anterior. Son muy genéricos y su finalidad es el mando operacional de las unidades a las que sirven.

Como se puede deducir de las clasificaciones anteriores, debido a las diferencias existentes entre las finalidades y objetivos de los distintos sistemas de mando y control, sus funcionalidades y diseño pueden variar mucho. Pero hay cierto tipo de requisitos que la comunidad internacional ha decidido que sean comunes a todos los sistemas. Tomando como referencia el modelo de intercambio internacional JC3IEDM en su versión 3.1 ([23]), se pueden agrupar estos requisitos en cuatro grandes módulos que siempre están presentes:

- **Sistema de mensajería (protocolo MEM de MIP, [70]).** En este grupo se encuentra toda la gestión de mensajería electrónica dentro del campo de batalla.

- **Sistema de planes y órdenes.** Incluye toda la gestión necesaria para la publicación y suscripción a los planes y las órdenes. Las naciones que forman parte de la OTAN siguen el STANAG 2014 ([71]).

- **Sistema de gestión de objetos (protocolo DEM de MIP, [72]).** Es el núcleo principal de las funcionalidades de los sistemas de mando y control. En él se incluye toda la información referente a los objetos de interés del campo de batalla. Estos se pueden agrupar en:

  - Servicios (Facility).
  - Características (Feature).
  - Material (Material).
  - Organización (Organization).
  - Personas (Person).

  La representación de los objetos mediante símbolos se suele realizar siguiendo el estándar OTAN APP-6A ([73]) y el estándar del ejército de Estados Unidos, MIL-STD-2525B ([74]).

  La representación en un plano de todos los objetos de batalla forman la denominada RGP (Recognized Ground Picture) en el ámbito terrestre o COP (Common Operational Picture) en el ámbito conjunto ([22]).

- **Sistema de gestión de trabajos (protocolo DEM de MIP).** Incluye toda la información referente a la interacción de los objetos en el campo de batalla. Existen dos tipos de acciones:

- Acciones de trabajo (action-task).
- Acciones de evento (action-event).

2.7. Necesidad de interoperabilidad entre sistemas de mando y control

Las operaciones en las que intervienen los ejércitos modernos suelen desarrollarse en ambientes multinacionales, es decir, son ejecutadas por unidades compuestas por varios ejércitos. Los diferentes diseños de los sistemas de mando y control para cada uno de ellos, hacen que sea necesaria una solución para que puedan acometerse misiones conjunto-combinadas usando este tipo sistemas.

La OTAN, como organización militar que reúne a un gran número de países, ha estudiado en los últimos años cómo conseguir que los sistemas de los países miembro sean interoperables entre ellos. Esto ha dado lugar a la elaboración de varias especificaciones que han tenido una gran aceptación dentro de la organización ([67]).

Un ejemplo es LINK 16, un estándar OTAN para intercambio de información muy difundido entre los sistemas de mando y control del ejército del aire y la armada ([75]). La mayoría de los sistemas en servicio están diseñados siguiendo este estándar (o por su predecesor, LINK 11 ([76])). La última versión de este estándar, LINK 22 (figura 2.7), extiende el anterior para el intercambio de información en redes wireless ([77]).

Por otro lado, MIP (Multilateral Interoperability Program) es un programa asociado a OTAN en el que varios países miembro están desarrollando un estándar general para el intercambio de datos tácticos. Dentro de este programa los países contribuyen al desarrollo de un modelo de referencia común, llevando a cabo las siguientes tareas, divididas entre working groups ([78]):

- **Operational Working Group (OWG).** Este grupo se dedica a la definición de las necesidades operativas.

- **Systems Engineering and Architecture Working Group (SEAWG).** Este grupo se dedica al desarrollo de los requisitos de sistema y del protocolo de intercambio de datos DEM (Data Exchange Mechanism).

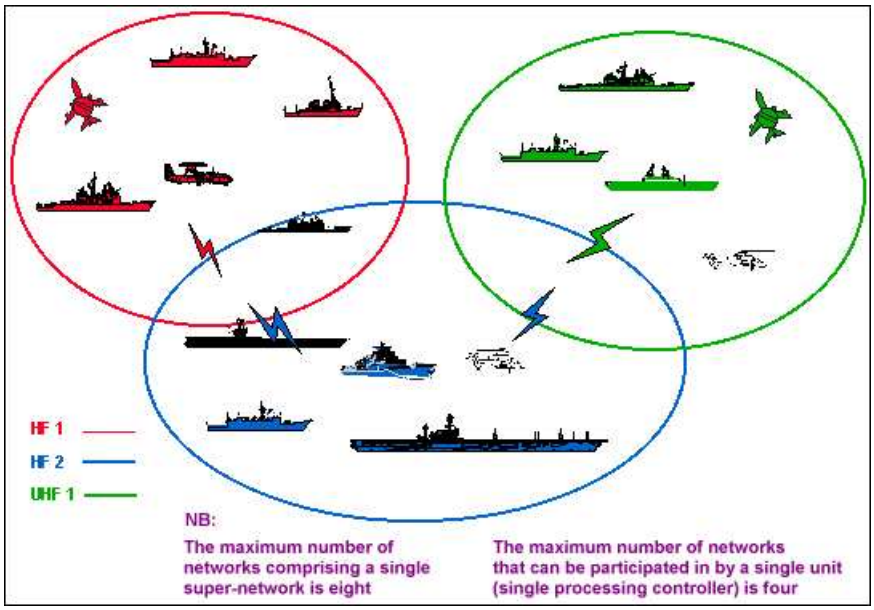

Figura 2.7: Esquema de red basada en LINK 22 ([77])

- **Data Modelling Working Group (DMWG).** Este grupo se dedica a la elaboración de la especificación del modelo de datos JC3IEDM.
- **Test Evaluation Working Group (TEWG).** Este grupo se encarga de la programación, elaboración y coordinación de los diferentes niveles de pruebas que se definen en MIP:
  - MIP system level test 1 (MSLT1). Pruebas del protocolo DEM (réplica de datos).
  - MIP system level test 2 (MSLT2). Pruebas de base de datos y reglas de negocio implementadas en ellas.
  - MIP system level test 3 (MSLT3). Pruebas de funcionalidades completas del sistema.
  - MIP operational level test (MOLT). Pruebas de operaciones completas en las que se programa un tema táctico, similar al que tendría lugar en un ejercicio real.

El grupo de Data Modelling (DMWG) trabaja en la elaboración de un modelo de datos común (su versión actual es JC3IEDM 3.1) que completado con reglas de negocio e implementación, permita el intercambio de información entre todos los

sistemas de mando y control. Este modelo se explica en profundidad en la siguiente sección.

2.8. The Joint C3 Information Exchange Data Model: JC3IEDM

El JC3IEDM es la ultima evolución de los modelos de intercambio C2, en realidad C3, ya que se ha añadido la capacidad de comunicación (Communication). Es necesario estudiar este modelo, y el camino seguido por la comunidad internacional para llegar a él, para comprender los problemas de interoperabilidad que ya se han solventado y los que todavía no ha sido posible resolver.

2.8.1. Evolución de los modelos de intercambio

El programa ATCCIS MIP fue el primer paso que dio la OTAN para estudiar el intercambio de información táctica ([79]). Este programa tenía definidas cinco fases:

- **Fase I (1980-1983)**. El objetivo era el desarrollo y análisis de los objetivos que se pretendían alcanzar con el programa ATCCIS.
- **Fase II (1985-1990)**. Consistía en identificar los conceptos militares que eran susceptibles de ser intercambiados como información C2 ([80]).
- **Fase III (1992-1997)**. Consistía en la demostración de la viabilidad del sistema, probando que era posible el intercambio de información automatizada entre los sistemas C2 de varios países ([81]).
- **Fase IV (1997-1999)**. El objetivo de esta fase fue la implantación del sistema dentro de un entorno operativo.
- **Fase V (2000-2002)**. En esta fase se estudiaron y analizaron los resultados obtenidos para proponer las ampliaciones necesarias en el futuro ([82]).

La especificación LC2IEDM (Land C2 Information Exchange Data Model), fue el resultado del programa ATCCIS ([81]), conocido dentro del programa MIP como Bloque I.

En Octubre de 2002 el programa ATCCIS se convirtió en el programa MIP, que fue el responsable del desarrollo de la nueva especificación. El C2IEDM (C2

Modelo LCMI Levels of Conceptual Models Interoperability	Nivel 5 Conceptual	Visión del mundo compartida. Los sistemas suelen intercambiar posiciones	Modelos conceptuales de sistemas, MDA
	Nivel 4 Pragmático/ Dinámico	Se intercambia información, su modo de empleo y su aplicación	Modelos de referencia de procesos, documentación, UML
	Nivel 3 Semántico	Se intercambian datos de contexto	RPR-FOM, C2IEDM
	Nivel 2 Sintáctico	Se intercambian datos en formatos acordados	OMT HLA, WSDL, DIS, IDL
	Nivel 1 Técnico	Establecimiento de conexión física	Conexiones físicas, conectividad de red
	Nivel 0 Aislamiento	Sistemas aislados. No hay interoperabilidad	No hay conectividad

Figura 2.8: Niveles conceptuales de interoperabilidad

Information Exchange Data Model), cuya versión 6.1.5e ([83]) es la que se encuentra en servicio en la actualidad, ampliaba el modelo de intercambio basándose en los estudios realizados en la fase V y mejorando el protocolo de réplica ([84]).

El bloque III es el que está en desarrollo en estos momentos, siendo el JC3IEDM 3.1 ([23]) (Joint C3 Information Exchange Data Model) la versión actual del modelo de datos. Esta versión es una evolución de la anterior mejorando algunos defectos comprobados tras su utilización y con el añadido de todo el módulo de gestión de planes y órdenes en operaciones.

2.8.2. Conceptos básicos acerca del modelo JC3IEDM

La especificación del modelo de intercambio JC3IEDM está dividida en varios documentos. A continuación se describen los que son de mayor interés para esta tesis doctoral:

- **JC3IEDM main** ([23]). En este documento se define el modelo de datos lógico y físico, describiendo las entidades, sus atributos y las relaciones entre ellos.

- **JC3IEDM metamodel specification** ([85]). En este documento se desarrolla la estructura del metamodelo, que es esencial para la realización de los trabajos de ingeniería y desarrollo.

- **JC3IEDM extensible markup language (XML) references schemas and implementation guidance** ([86]). En este documento, realizado por primera vez en esta versión del modelo, se desarrolla el intercambio de información C2 en formato XML, cuyo campo de aplicación se extiende a la interoperabilidad con otros tipos de sistemas y al uso de tecnologías basadas en la filosofía SOA (Services Oriented Architecture).

- **JC3IEDM compendium of text business rules** ([87]). Este documento tiene como objetivo elevar el nivel de interoperabilidad a pragmática ([33]). Esta especificación es la más complicada de implementar, pero a su vez es fundamental para conseguir que los sistemas C2 se comuniquen entre sí en escenarios reales. Las reglas de negocio definidas son las responsables de que el JC3IEDM no sea únicamente una pasarela para intercambiar datos, sino que se logre una unidad semántica dentro de la comunidad C2.

- **JC3IEDM compendium of coded business rules** ([87]). Este documento define dominios de datos para los atributos definidos dentro del modelo JC3IEDM que así lo requieran.

- **JC3IEDM glossary** ([88]). Debido al gran número de siglas usadas dentro del ámbito de la OTAN, es necesario definir en este tipo de documentos su significado. Es la única manera de manejar un lenguaje unificado dentro del programa MIP.

Hay que señalar que el modelo de intercambio JC3IEDM no es sólo un modelo de datos común aceptado por la comunidad internacional para intercambiar datos C2. Lo que se pretende es alcanzar un nivel de interoperabilidad mayor, una integración semántica entre todos los países miembros del programa (figura 2.8).

Tampoco se puede limitar el JC3IEDM a la interoperabilidad entre sistemas C2, este modelo de intercambio pretende ser el punto de partida de otras comunidades dentro de OTAN que emplean la información táctica como parte de sus especificaciones (figura 2.9, [86]).

2.8.3. Estándares asociados a JC3IEDM: DEM y XML

Para lograr un intercambio de información C2 entre distintos sistemas de mando y control no basta con emplear modelos de datos comunes entre ellos, un requisito imprescindible es la utilización de protocolos de intercambio de información. Por eso

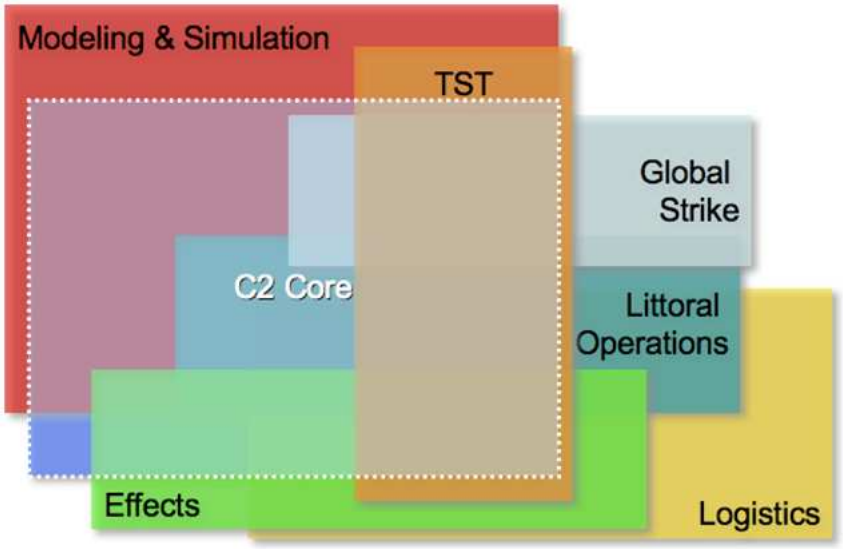

Figura 2.9: Esquema de los modelos de intercambio de las diferentes comunidades ([23])

en esta sección se va a analizar la importancia de los protocolos DEM ([72]) y XML (eXtensible Markup language) ([86]) dentro de los escenarios de interoperabilidad C2.

2.8.3.1. Data Exchange Mechanism (DEM)

DEM es el protocolo de réplica desarrollado por MIP para el intercambio de información C2 y está basado en los protocolos de comunicación TCP/IP (figura 2.10, [87]).

DEM se basa en la publicación y suscripción de OIGs (Operational Information Groups) por parte de los diferentes sistemas. Una OIG es un grupo de información del JC3IEDM, relacionada con un mismo concepto, que está predefinida en la especificación y se intercambia con otro sistema.

Un sistema publica OIGs para otros sistemas de manera que se puedan suscribir a ellas y acceder a la información que contienen. Con este protocolo se consigue un intercambio de la información de manera estructurada, muy importante para el desarrollo de operaciones militares, en las que la jerarquía es un aspecto de vital importancia (figura 2.11).

Los tipos de OIGs definidas por MIP son seis ([21]):

- Globally significant.

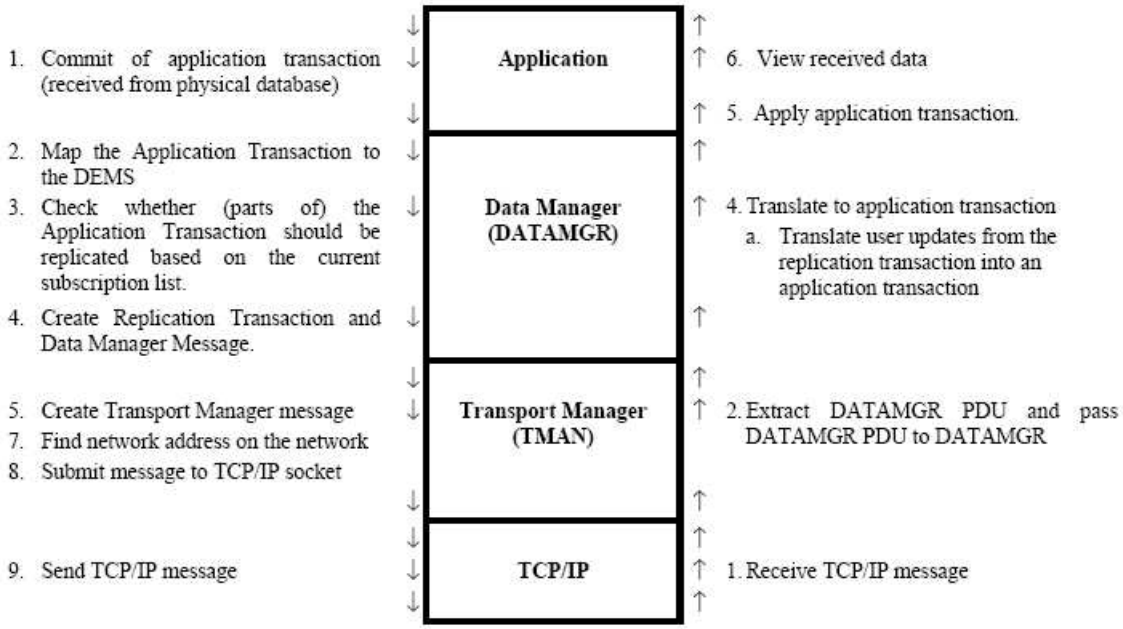

Figura 2.10: Concepto del protocolo DEM por capas ([72])

- Composed plan.
- Correlated enemy and unknown.
- Uncorrelated enemy and unknown.
- Friendly and neutral (Org).
- Friendly and neutral (Non-aggregable).

Cada objeto que se crea pude asociarse a una o ninguna OIG, dependiendo de si se quiere o no replicar con otros sistemas, es decir, hacer accesible la información que contiene el objeto para ellos. Y a excepción de la OIG Composed plan, cada sistema puede crear sólo un tipo de OIG.

Resumiendo, DEM es un protocolo que tiene la ventaja de permitir el intercambio de información C2 de manera estructurada, sin embargo tiene la limitación de ser únicamente soportado por redes de comunicaciones TCP/IP, no siempre disponibles en el campo de batalla.

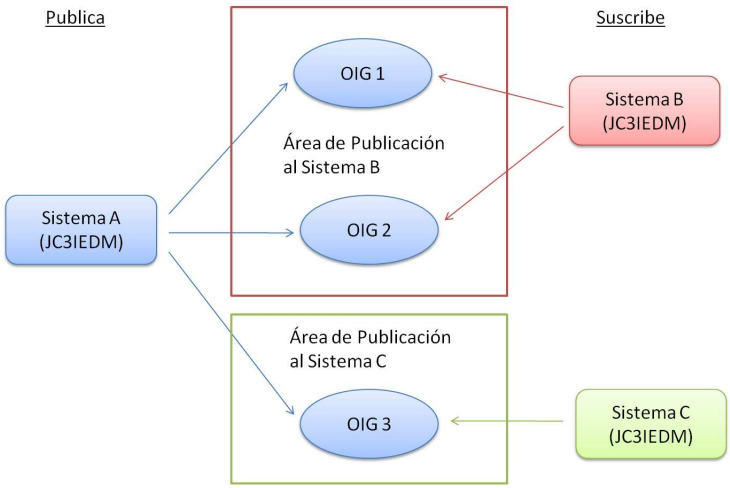

Figura 2.11: Esquema de publicación/suscripción a OIGs

2.8.3.2. eXtensible Markup Language (XML)

Muchas de las redes de comunicaciones militares en servicio en la actualidad no soportan el protocolo TCP/IP. Esto acentúa la limitación de los protocolos, que como DEM, dependen de estos protocolos de comunicaciones.

Por esta razón, el uso de las nuevas tecnologías SOA, basadas en XML, tienen un gran protagonismo dentro del programa MIP (figura 2.12). Hay que tener en cuenta que los servicios web emplean el protocolo de comunicación http, que es soportado por la gran mayoría de las redes de comunicaciones actuales ([89]). Es por ello que la definición de la información C2 que se debe intercambiar entre sistemas de mando y control en formato XML ha sido una prioridad en el programa MIP para intentar solucionar la limitación del protocolo DEM en redes que no soportan TCP/IP ([66]).

Aunque las redes de comunicaciones no son el único motivo por el cual se emplea XML como protocolo de intercambio de información. Al ser JC3IEDM el referente de otras comunidades, es necesario definir un mecanismo de intercambio de información, lo mas estándar posible, para que los datos comunes del COI (Community Of Interest) entre distintos tipos de sistemas sean intercambiables ([90]). Esto se resume en la figura 2.13.

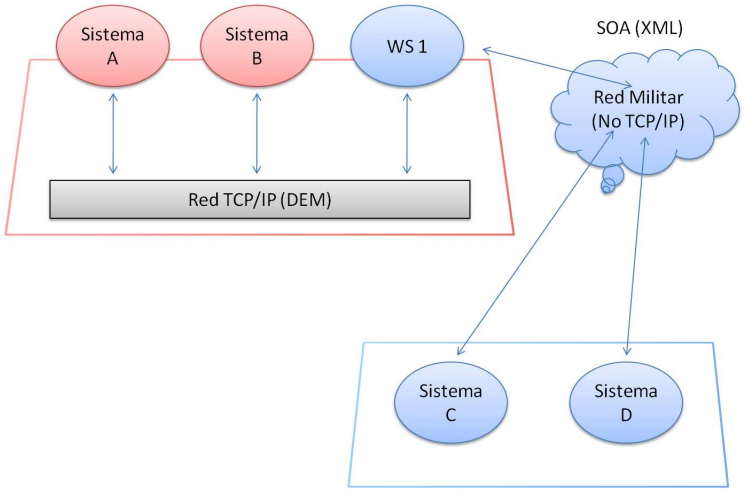

Figura 2.12: Aplicación de tecnologías SOA en redes militares

2.8.4. Técnicas MDA y modelos de datos orientados a objetos

Al ser el modelo de datos JC3IEDM un referente para multitud de comunidades dentro de la OTAN, dentro del propio programa MIP se realizan estudios para facilitar la integración entre todas ellas.

En concreto, los estudios realizados dentro de la working party de MIP, MDA (Model Driven Architecture), buscan automatizar los procesos de ingeniería y generación de código mediante la creación de modelos definidos mediante técnicas MDA ([91]) así como desarrollar herramientas que permitan que, de manera automática, se realicen las diferentes tareas de ingeniería que habitualmente se completan de manera manual.

El modelo de datos JC3IEDM es un modelo relacional, sin embargo la working party MDA estudia su adaptación a un modelo orientado a objetos y la definición de business rules en formato OCL (Object Constraint Language, [92]).

La arquitectura propuesta por el MDA es fundamental para la convergencia entre la arquitectura de los sistemas C2 y cualquiera compatible con la arquitectura HLA. El modelado de datos de la información intercambiada dentro de una federación se realiza siguiendo OMT, que en algunos casos puede ser transformable de manera directa con un modelo C2 orientado a objetos ([92]).

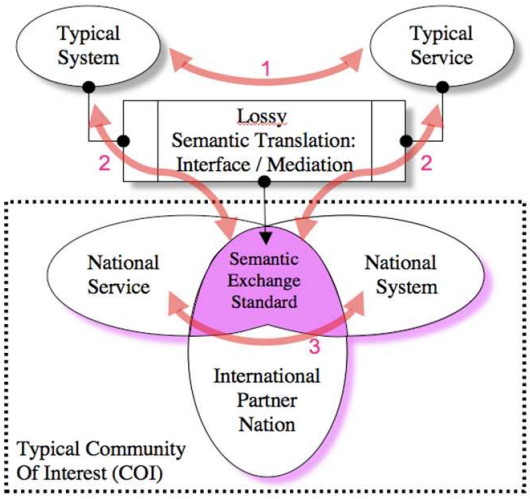

Figura 2.13: COI desde el punto de vista MIP ([23])

2.9. Necesidades de interoperabilidad entre simuladores constructivos y sistemas C2

Como ya se ha explicado, los sistemas de mando y control tienen como objetivo principal el intercambio de información y su presentación organizada para que el mando militar tome la línea de acción más acertada en cada situación y en cada momento. Pero la información C2 organizada que se presenta al mando para la toma de decisiones es, en la mayoría de los casos actuales, muy superior a la que la mente humana puede procesar ([28]).

Por tanto, la dirección de las operaciones militares exige hoy en día sistemas que, teniendo en cuenta toda la información, no sólo la presenten de manera organizada sino que además puedan asesorar al mando en la toma de decisiones ([93]). Una solución muy potente para asesorar al mando en la dirección de operaciones es el uso de simuladores constructivos que, alimentados con los datos del sistema C2, evalúen los resultados de cada una de las líneas de acción posibles.

Por este motivo la interoperabilidad entre estos dos tipos de sistemas es una nueva necesidad demandada por todos los ejércitos ([94]). Pero intentar que arquitecturas tan diferentes sean realmente interoperables no es una labor sencilla. Hasta el momento se han dado pasos en dos direcciones diferentes.

En la primera los ejércitos de las diferentes naciones y la misma OTAN resuelven la interoperabilidad de manera artesanal, definiendo pasarelas particulares entre

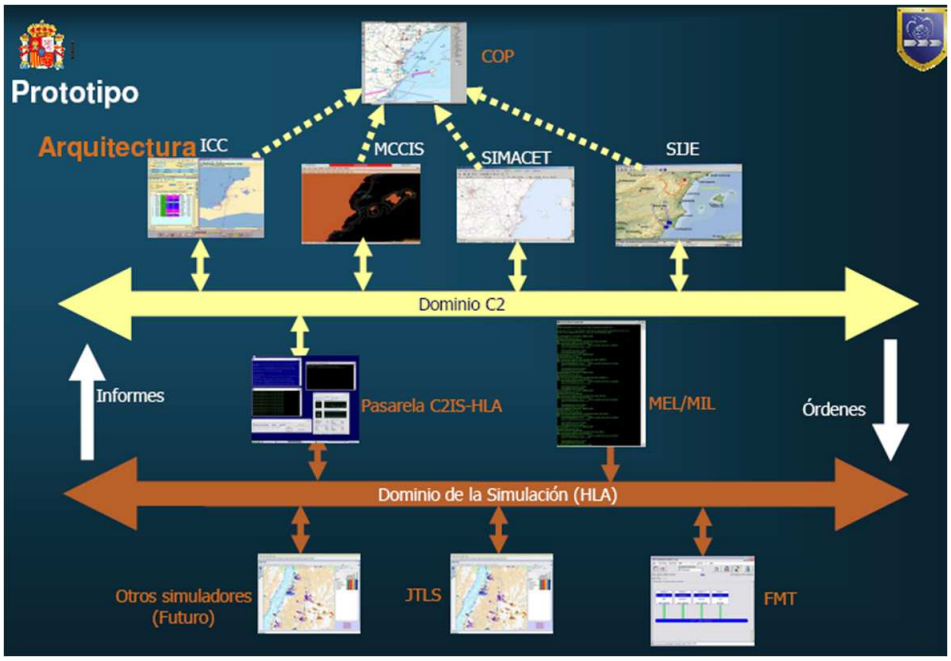

Figura 2.14: Pasarela C2IS - HLA ([35])

ambos tipos de sistemas que requieren de mucha parametrización e intervención manual, pero que de momento facilitan el trabajo del día a día (figura 2.14) aunque no sea de una forma estandarizada.

Por este motivo en la segunda dirección, la OTAN como organización militar, junto con otras organizaciones civiles como SISO, están trabajando en la definición de estándares como CBML (Coalition Battle Management Language) o MSDL (Military Scenario Definition Language), que resuelven la interoperabilidad ([55]) de manera parcial y en aspectos muy concretos.

Las soluciones propuestas hasta el momento, aunque son un buen punto de partida, no son suficientes. Se dan las herramientas para conseguir que el intercambio de datos y la unidad semántica entre sistemas sea posible. Pero la integración no es aún una realidad, la falta de una arquitectura de referencia estandarizada que cubra todas las necesidades de interoperabilidad hace que no se alcance el grado de integración deseado ([93]).

Aún así, antes de terminar este capítulo, es interesante analizar con algo más de profundidad estos dos estándares como primer paso hacia una interoperabilidad real entre ambos tipos de sistemas.

2.9.1. Coalition Battle Management Language (CBML)

La necesidad de la definición de un lenguaje de batalla común surge como una adaptación del espacio de batalla real al espacio de batalla virtual.

En el espacio de batalla real una unidad ejerce el mando y control sobre sus unidades subordinadas mediante la trasmisión de órdenes. Estas órdenes son las que se pretenden modelar en un espacio de combate virtual (BML o Battle Management Language).

CBML es uno de los primeros estándares de ámbito especifico militar (desarrollado por SISO y RTO) cuyo objetivo es modelar un estándar ([17], [20]) de órdenes a intercambiar entre simuladores constructivos, y entre estos y sistemas C2 ([95], [96], [97]).

Para la definición de las órdenes, CBML toma como referencia el estándar C2IEDM desarrollado por MIP ([98]). Aunque es necesario realizar extensiones al mismo ya que la simulación constructiva tiene unas necesidades de información que difieren en algunos puntos de la de los sistemas C2 como se puede observar en la figura 2.15 ([90], [99]).

Aun así, algunas implementaciones de CBML como JBML ([66]) han concluido que la definición de órdenes estándar es un gran paso para la convergencia entre estos sistemas pero no es suficiente. La diferencia de arquitecturas y estándares asociados a simuladores constructivos y sistemas C2 es un problema que desemboca en una falta de interoperabilidad entre estos sistemas ([100]).

A pesar de todo hay que hacer hincapié en que CBML supone un gran avance en este campo. Estandarizar las órdenes que se transmiten en el campo de batalla y que éstas estén basadas en un estándar como el C2IEDM (en una futura versión, en el JC3IEDM) es un gran paso para conseguir un alineación en la estructura de la información intercambiada entre ambos tipos de sistemas ([58], [86], [101]).

2.9.2. Military Scenario Definition Language (MSDL)

Las operaciones militares objeto de la simulación constructiva tienen lugar en un escenario (espacio de batalla) que tiene unas condiciones iniciales y va evolucionando con el desarrollo de las operaciones militares. Por lo tanto el punto de partida de toda simulación constructiva queda definido por un escenario inicial.

Hasta el momento los tipos de datos y las estructuras para la definición de un escenario inicial dependía de cada tipo de simulador o de los operadores. MSDL es

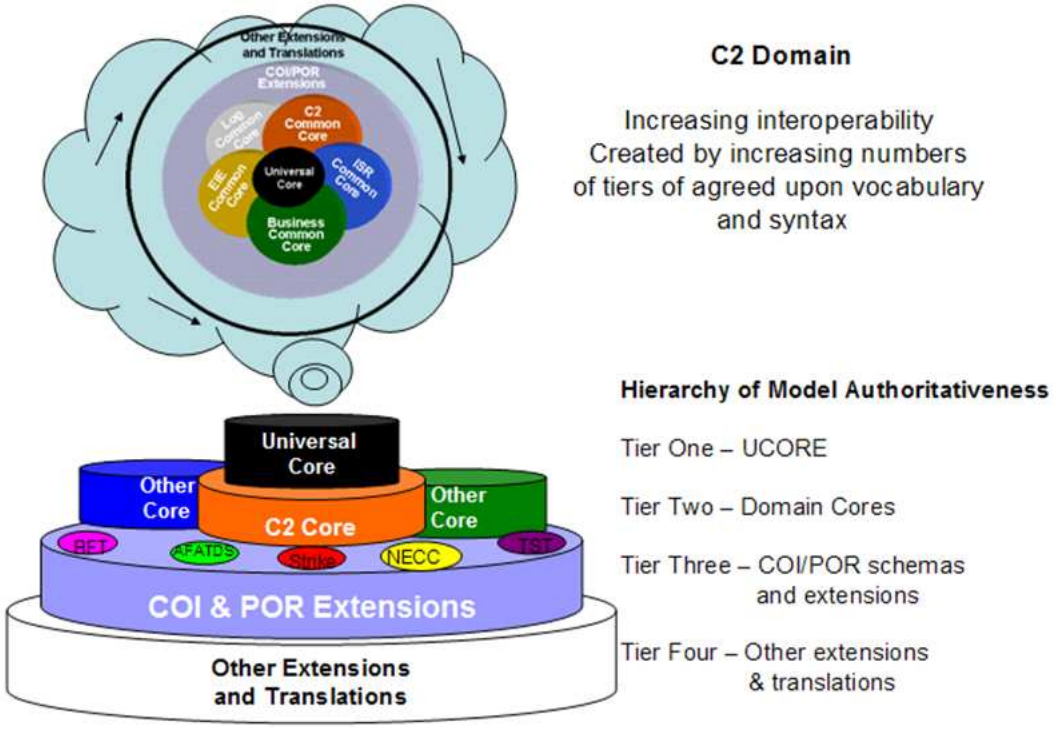

Figura 2.15: Esquema de los diferentes modelos basados en C2 ([23])

un estándar cuyo objetivo es estandarizar los datos y la estructura para la creación de escenarios iniciales en simulaciones de operaciones militares de tal manera que puedan ser intercambiables entre simuladores diferentes ([19], [102]).

De esta manera el punto de partida de los simuladores constructivos que participan en una simulación es común y entendible por todos ellos ([102]).

La información de un escenario inicial definido en MSDL está estructurada siguiendo un esquema XSD (XML Schema Definition) especificado en el estándar y dividido en los siguientes elementos principales:

- ScenarioID.
- Options.
- Environment.
- ForcesSides.
- Organizations.

- Overlays.
- Installations.
- TacticalGraphics.
- MOOTWGraphics.

Al tomar como base el JC3IEDM, este estándar proporciona un punto de partida común a todos los simuladores constructivos que lo implementen ([93]), y consigue una convergencia esencial entre los simuladores constructivos y los sistemas C2.

Capítulo 3

Interoperabilidad entre simuladores constructivos de aplicación militar

La mayoría de las soluciones propuestas hasta el momento para resolver el problema de la interoperabilidad entre simuladores constructivos de aplicación militar se basan en la definición de arquitecturas de comunicación y/o de estructuras de información estándar. Pero la experiencia ha demostrado que estas definiciones, que en teoría deberían ser suficientes para poder interoperar de forma automática, son insuficientes en la práctica. De hecho en las pruebas realizadas en este sentido una de las lecciones aprendidas ha sido el caos que genera la ausencia de unas reglas que establezcan los "pasos a seguir"para conseguir la interoperabilidad de manera sencilla entre los sistemas que componen una simulación distribuida ([103], [104]).

Como se ha introducido en el capítulo anterior, en el campo de la simulación industrial, SISO (Simulation Interoperability Standards Organization) ha solucionado este problema estandarizando unos modelos de interoperabilidad de referencia (IRMs) que describen de manera genérica cómo se deben manejar las diferentes situaciones en las que los simuladores industriales necesitan interoperar siguiendo la arquitectura HLA ([59]).

Lo que se pretende en este capítulo es definir de la misma manera unos modelos de interoperabilidad de referencia que describan todos los casos donde sea necesaria la interoperabilidad entre simuladores constructivos de aplicación militar.

Para ello ese analizan los IRMs definidos por SISO y se estudia si es viable su adaptación a los simuladores constructivos militares. A lo largo de este estudio es

necesario considerar las restricciones y diferencias que tiene el ámbito táctico-militar con respecto al de la simulación industrial teniendo en cuenta el modelo JC3IEDM, ya que este estándar engloba toda la información táctico-militar que es necesario intercambiar en este tipo de entornos y que por lo tanto, debe ser cubierta por los nuevos IRMs de aplicación militar que se definan.

3.1. Antecedentes

3.1.1. Modelos de interoperabilidad de referencia en entornos industriales

En las primeras generaciones de simuladores de aplicación industrial el ámbito era un sistema industrial específico que funcionaba de manera aislada y los factores externos a éste se calculaban empleando la estadística. Con el objeto de conseguir más realismo en las simulaciones realizadas se pasó a una segunda generación de simuladores donde el objetivo era que los sistemas aislados actuaran de manera conjunta simulando así un sistema completo. Para conseguir esto se han aprovechado los avances en simulación distribuida ([32]) gracias a la aparición del estándar HLA.

La simulación distribuida dentro del ámbito industrial ha avanzado de la mano de los paquetes de simulación COTS (Commercial-Off-The-Self), ya que la mayor parte de los simuladores de aplicación industrial se desarrollan utilizando este tipo de paquetes (SLX, Simul8, Arena ó Extend son algunos ejemplos de paquetes COTS). Hoy en día todos estos paquetes son compatibles con el estándar IEEE1516 y suelen incluir una serie de módulos que, desde el punto de vista del desarrollador, abstraen de la complejidad de HLA agrupando varios interfaces definidos por el estándar en una función más simple y entendible.

Con el paso del tiempo en los entornos industriales se ha comprendido que, aunque esta abstracción es una gran ventaja en las etapas de desarrollo de los simuladores, debido a la gran variedad de sistemas COTS que existen en el mercado (cada uno de ellos con su módulo HLA específico), son necesarios unos modelos estándar que complementen al estándar IEEE1516 para conseguir interoperabilidad real en entornos de simulación distribuidos. Es decir, se ha trabajado en la definición de un estándar que especifique cómo se debe utilizar el estándar IEEE1516 en ciertas situaciones típicas de entornos industriales.

SISO, como organización internacional competente en esta materia, creó un gru-

po de trabajo llamado CSPI PDG (COTS Simulation Packages Interoperability Product Development Group) para definir unos modelos de interoperabilidad de referencia aplicables en entornos industriales típicos.

Actualmente el estándar desarrollado por este grupo se encuentra en su versión draft 2.0 ([59]) y en él se definen cuatro tipos de IRMs, que se describirán con más detalle en las secciones siguientes.

3.1.1.1. Tipos de IRMs: Definiciones previas

El estándar SISO-STD-006-2007 versión draft 2.0 ([59]) define cuatro tipos de IRMs para su empleo en entornos industriales. La finalidad de los IRMs es modelar todos los casos posibles en los que es necesaria la interoperabilidad entre dos o más simuladores industriales en un entorno distribuido. De este modo sistemas COTS diferentes que tengan implementados los IRMs pueden ser realmente interoperables entre ellos ya que resuelven las mismas situaciones y problemas exactamente de la misma manera.

Antes de comenzar con la definición de los IRMs de uso industrial es necesario definir una serie de conceptos básicos. Estos son:

- **Modelo.** Un modelo representa un sistema real ejecutado bajo un entorno CSP (COTS Simulation Package).
- **Federado.** Un federado es un modelo que se ejecuta en un único ordenador.
- **Tiempo.** Representa el tiempo de simulación de un modelo específico. Es un valor entero positivo y el estándar no define unidades para él.
- **Actividad.** Es un proceso que se lleva a cabo o realiza en un sistema y que tiene una duración conocida.
- **Entidad.** Es un elemento procesado por un sistema y que pasa a través de las actividades y colas que lo componen.
- **Cola.** Es una cola de entidades que sigue una política de gestión determinada, por ejemplo FIFO (First In First Out) o LIFO (Last In First Out).
- **Interacción.** Es una relación que tiene lugar entre dos o varios federados.
- **Recurso.** Es un elemento necesario para realizar una actividad.

Figura 3.1: Esquema del IRM tipo A

- **Evento.** Situación que produce que el sistema sufra una transición de un estado a otro.
- **Estructura de datos.** Es un elemento parecido a un recurso, pero de composición más flexible, que puede ir desde una única variable, a un registro completo.

En un entorno industrial típico estas definiciones son suficientes para modelar el sistema de producción, ya que los procesos de producción son actividades, las máquinas y operarios son recursos, las materias primas y productos son entidades, los inventarios son colas y las órdenes de fabricación y recetas, por poner sólo dos ejemplos, son estructuras de datos.

Teniendo esto en cuenta, los IRMs definidos por el estándar SISO-STD-006-2007 versión draft 2.0 son cuatro:

- **Tipo A.** Transferencia de entidad.
- **Tipo B.** Recurso compartido.
- **Tipo C.** Evento compartido.
- **Tipo D.** Estructura de datos compartida.

3.1.1.2. IRM Tipo A

De manera genérica este IRM (figura 3.1) describe la transferencia de una entidad desde un federado (federado A) a otro (federado B).

La entidad que se va a transferir es el resultado de una actividad A que se produce en un tiempo de simulación T1. Esta entidad es recibida por el federado B que la almacena en una cola B en un tiempo T2 mayor que T1.

Dentro del IRM tipo A se definen tres subtipos que se explican a continuación:

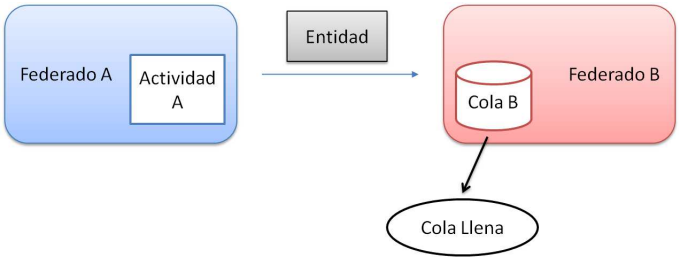

Figura 3.2: Esquema del IRM subtipo A2

- **IRM subtipo A1.** Transferencia de entidad general.
- **IRM subtipo A2.** Cola receptora llena.
- **IRM subtipo A3.** Prioridad de entradas múltiples.

3.1.1.2.1. IRM subtipo A1. Este subtipo describe el caso más básico de transferencia de una entidad entre dos federados. El federado A envía una entidad que es recibida por el federado B sin la influencia de ningún factor.

Un ejemplo práctico de este tipo de IRM es el de la cadena de producción de una factoría de coches. El federado A es un simulador que simula el ensamblado del coche y el federado B es un simulador que simula el pintado del mismo (dos actividades diferentes dentro del sistema de producción, cada una de ellas simulada con su propio modelo en un ordenador diferente y puede que programado con paquetes COTS diferentes).

Cuando el federado A termina de montar el coche transfiere la entidad (en este caso el coche montado) al federado B para proceder a su pintado.

Este subtipo es la base para la definición de los otros dos subtipos como casos derivados de él.

3.1.1.2.2. IRM subtipo A2. Este subtipo describe un caso excepcional del subtipo A1 donde la entidad transferida no puede ser recibida por el federado B porque la cola en la que la va a almacenar está llena (figura 3.2). Es decir, se tienen en cuenta inventarios de planta realistas en los que la capacidad de almacenamiento es limitada.

La situación descrita en este subtipo supone una parada en las actividades y un tiempo muerto en la cadena de producción. Por este motivo debe ser evitada o por

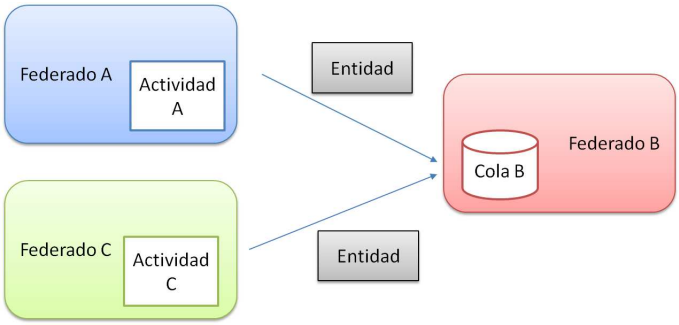

Figura 3.3: Esquema del IRM subtipo A3

lo menos, se debe intentar minimizar el tiempo de espera de recepción de la entidad por parte del federado B.

Siguiendo con el ejemplo de la fábrica de coches descrito anteriormente, en este caso se tiene en cuenta el número de coches que la sección de pintura puede tener en una sala de espera (cola B).

En este caso no puede pasar ningún coche (entidad) de la cadena de montado (federado A) a la sección de pintura (federado B) hasta que no exista espacio en su sala de espera para recibirlo.

3.1.1.2.3. IRM subtipo A3. La situación descrita por este subtipo tiene como participantes a tres federados en vez de sólo dos como los subtipos descritos anteriormente. Este IRM describe a dos federados (simulador A y simulador C) que envían una entidad cada uno a un federado B y ambas llegan al mismo tiempo (figura 3.3).

La gestión por parte del federado B de la entidad que pasa primero (gestión de prioridades) es un punto crítico para la ejecución de este modelo. Una elección equivocada supondría una alteración de las prioridades del proceso de elaboración pudiendo tener un impacto negativo en el flujo normal de la cadena de producción.

Siguiendo con el ejemplo de la fábrica de automóviles en este caso existen dos cadenas de montaje simuladas por dos simuladores diferentes (federado A y federado C).

Este modelo describe la situación en la que los dos simuladores envían cada uno un coche montado a la sección de pintura (federado B). En este caso se debe priorizar cual de los dos coches pasa primero ya que no es posible que los dos coches entren simultáneamente a la sala de espera de la sección de pintura.

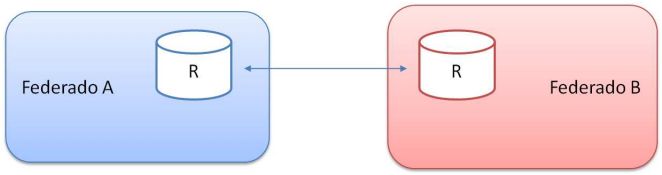

Figura 3.4: Esquema del IRM tipo B

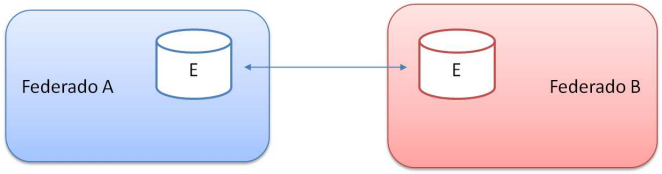

Figura 3.5: Esquema del IRM tipo C

3.1.1.3. IRM Tipo B

Este modelo describe a dos federados (federado A y federado B) que comparten un mismo recurso (figura 3.4).

Las actividades pertenecientes a ambos federados pueden emplear el recurso compartido dando como resultado la modificación del mismo. Esto supone una interconexión entre las actividades de los federados siendo el recurso compartido el nexo de unión.

Un ejemplo de este tipo de IRM en la fábrica de coches aparecería si hubiera dos líneas de pintado de coches.

En este caso existen dos cabinas de pintado simuladas por dos simuladores diferentes (federado A y federado B). En una de las cabinas (federado A) existe un tanque de pintura (recurso) que debe suministrar pintura simultáneamente a las dos cabinas de pintado.

3.1.1.4. IRM Tipo C

Este modelo describe a dos federados (federado A y federado B) que comparten un mismo evento (figura 3.5).

Un evento produce un cambio de estado en un sistema, entonces si dos federados comparten un mismo evento cambiarán de estado de manera simultánea. Esto es muy habitual en simulación distribuida, ya que se definen eventos compartidos como

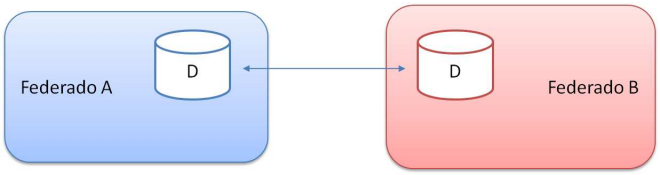

Figura 3.6: Esquema del IRM tipo D

elementos de coordinación para los federados que los comparten.

Siguiendo con el ejemplo de la fábrica de coches un ejemplo de este tipo de IRM sería si el simulador de ensamblado de coches (federado A) y el simulador de pintado (federado B) tuvieran que parar su línea de producción simultáneamente debido a un descanso de media hora (evento) para realizar el cambio de turno.

3.1.1.5. IRM Tipo D

Este modelo describe a dos federados (federado A y federado B) que comparten una misma estructura de datos (figura 3.6).

Aunque este IRM se asemeja al tipo B existen diferencias semánticas y estructurales entre ellos. La estructura de datos tiene una definición mucho más flexible que un recurso (cuya estructura está cerrada normalmente para definir un operario o una máquina).

Un ejemplo de este tipo de IRM sería un caso particular del ejemplo expuesto para el IRM B. En este caso existen dos cabinas de pintado simuladas por dos simuladores independientes (federado A y federado B).

Una cabina tiene varios tanques de pintura cada uno de un color que comparte con la otra cabina. Sería necesario ejecutar el IRM tipo D si lo que se quiere compartir no es el tanque de pintura sino un listado con las mezclas de colores (estructura de datos).

3.1.2. Aspectos de HLA relacionados con la gestión del tiempo

La secuencia temporal en la ejecución de los IRMs tiene influencia en el resultado final obtenido tras la simulación, por ejemplo, los modelos de interoperabilidad definidos por SISO para entornos industriales se desarrollan de manera lineal en el

tiempo. En general, todo lo relacionado con la gestión del tiempo especificada por el estándar IEEE1516 es esencial para la definición de IRMs en cualquier ámbito, y en esta sección se analizan los aspectos más importantes relacionados con el entorno táctico-militar.

La primera consideración que debe realizarse respecto a esta gestión del tiempo tiene que ver con el tiempo real en el marco de referencia de los simuladores constructivos.

Un sistema en tiempo real (STR) se define como: "Aquel sistema digital que interactúa activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad, controlabilidad y alcanzabilidad" ([105], [106]).

Tiempo real no equivale a rapidez o inmediatez, sino a cumplir con unas restricciones temporales, con una latencia mínima. En el estándar IEEE1516.1 ([15]) se define el módulo *Time Management* para gestionar estas restricciones temporales. Es por ello que HLA puede considerarse como una arquitectura que, en general, permite el tiempo real.

En el entorno militar, y dependiendo del tipo de simulador que se estudie, las restricciones de tiempo pueden ser más o menos estrictas. En el caso de la simulación constructiva los tiempos de ejecución son muy largos ya que el desarrollo de una operación militar (objeto de este tipo de simuladores) suele ser en el orden de minutos y horas. Y el estándar IEEE1516 no tiene ningún problema en soportar latencias de este orden de magnitud, por lo que permite conseguir el tiempo real en este tipo de simulaciones.

El resto de consideraciones relacionadas con la gestión del tiempo en el tipo de simulador considerado en esta investigación se desarrollan en las siguientes secciones con detalle.

3.1.2.1. Definiciones

Para comprender los conceptos definidos por el módulo *Time Management* es necesario comprender las diferentes acepciones del tiempo dentro de una federación ([107]). Por este motivo se definen a continuación los siguientes conceptos:

- **Time Management.** Conjunto de servicios y mecanismos que controlan el avance del tiempo dentro de un federado durante la ejecución de la federación.

- **Logical Time.** Es el tiempo actual de la línea temporal lógica en la que se encuentra el federado. Por ejemplo, si un federado se encuentra en un tiempo lógico T, todos los mensajes que reciba desde la RTI estarán etiquetados con un tiempo lógico inferior o igual a T.
- **Wallclock Time.** Medida del tiempo real de un federado. Esta medida se suele realizar desde un reloj hardware.
- **Federate Time.** Es el tiempo lógico del federado calculado en la escala de tiempos lógica de la RTI o del *wallclock* del federado.
- **Federation Time Axis.** Secuencia de valores ordenados en el que cada uno representa un instante de tiempo del sistema que se modela.
- **Evento.** Es el cambio en el valor de un atributo, una interacción entre objetos, una instanciación de un nuevo objeto o el borrado de un objeto creado. Cada evento tiene asociado un *Time Stamp*.
- **Time Stamp (de un evento).** Es la representación gráfica de un evento (dentro del *Federation Time Axis*) que indica cuándo ocurre.
- **Logical Time Axis.** Es un conjunto de puntos (dentro de la federación) ordenados dentro de una línea temporal lógica para establecer una relación cronológica entre eventos.

3.1.2.2. Ordenación de los mensajes

La arquitectura HLA se puede englobar dentro de las de tipo MOM (Message Oriented Middleware, [108]) al estar basada en el intercambio de mensajes entre sus componentes. El orden en el que los mensajes son liberados dentro de una federación es uno de los factores que permite gestionar el módulo *Time Management*.

Por lo tanto la RTI (que es el componente de la federación que coordina la transferencia de mensajes) se encarga de ordenar los mensajes recibidos por los federados y liberarlos a los destinatarios en el tiempo establecido.

La interfaz *Time Management* define una serie de protocolos para la ordenación de los mensajes que son:

- **Orden de recepción.** Los mensajes se ordenan siguiendo el orden en el que la RTI los recibe.

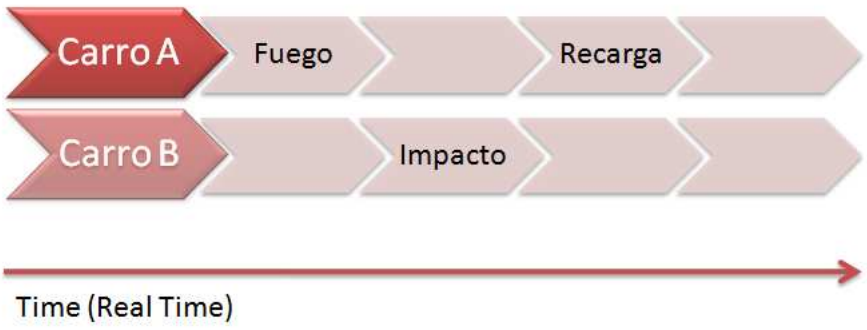

Figura 3.7: Ejemplo de secuencia temporal con *Time Stamp Order*

- **Orden de Prioridad.** Los mensajes se ordenan en una cola siguiendo un protocolo de prioridades previamente establecido por la RTI.

- **Time Stamp Order.** Los mensajes se ordenan según el tiempo (*Federation Time Axis*) en el que han sido liberados.

  Si se emplea este protocolo para ordenar los mensajes liberados en una secuencia de fuego de un carro A sobre otro B el orden sería: fuego del carro A, impacto del proyectil en el carro B y recarga de munición en el carro A (figura 3.7).

- **Casual Order.** Los mensajes son ordenados siguiendo un orden parcial entre ellos. Es decir, si el mensaje A es anterior al B y C es anterior a A entonces se deduce que el mensaje C es anterior a B, de esta manera se establece un orden relativo entre los mensajes.

  Por lo tanto, empleando este modo de ordenación si un carro A localiza un carro B y realiza fuego sobre él entonces un carro C que lo divise con posterioridad a la acción de fuego lo verá destruido (figura 3.8).

La principal diferencia entre el tipo *Time Stamp Order* y el tipo *Casual Order* es la menor latencia de este último. Sin embargo, si el número de federados que intervienen en la simulación es muy elevado, el tipo *Casual Order* podría perder eficacia llegando a producirse incoherencias en la ordenación.

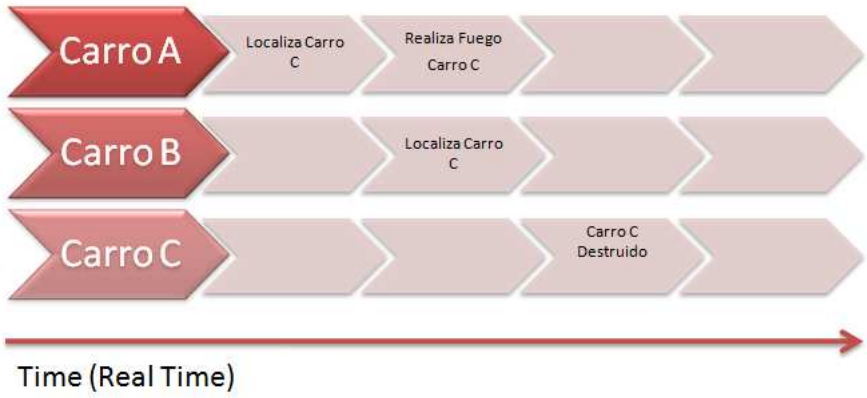

Figura 3.8: Ejemplo de secuencia temporal con *Casual Order*

3.1.2.3. Los servicios Time Management

Para realizar una correcta gestión del tiempo dentro de una federación es necesario definir cómo avanza el tiempo y la influencia que tiene sobre los federados. Por ello en el módulo *Time Management* se definen los conceptos de mecanismo de avance del tiempo y ordenación de los mensajes enviados.

3.1.2.3.1. Mecanismos de avance del tiempo. Los federados pueden avanzar el tiempo lógico de la federación con respecto a dos paradigmas, que son:

- **Time-Stepped (periódico).** El avance en el tiempo se realiza mediante saltos de tiempo (*Time Step*) de duración constante.
- **Event-Driven (por eventos).** El avance en el tiempo se realiza mediante etiquetas. El federado procesa los mensajes y avanza su reloj lógico hasta el tiempo de la siguiente etiqueta (*Stamp*).

3.1.2.3.2. Ordenación de los mensajes enviados. El módulo *Time Management* define dos roles diferentes en los que se puede encontrar un federado. Estos roles son:

- **Time Constrained.** En este caso, el federado está sometido a las restricciones de tiempo definidas para la federación siendo su tiempo lógico el definido para ella.

- **Time Regulating.** Un federado bajo este rol puede gestionar el tiempo lógico de la federación ordenando avances en el tiempo lógico de la misma.

Dependiendo de los roles que tenga asignado un federado se puede clasificar en:

- **Tiempo Lógico Sincronizado.** Cuando el federado es *Time Constrained* y *Time Regulating*.
- **Tiempo Sincronizado Externamente.** Cuando el federado no tiene ninguno de los dos roles anteriormente descritos.
- **Tiempo Lógico Pasivo.** El federado es *Time Constrained* pero no *Time Regulating*.
- **Tiempo Lógico Agresivo.** El federado es *Time Regulating* pero no es *Time Constrained*.

De esta manera un federado puede gestionar el tiempo lógico del resto de los federados o dejarse gestionar por otros.

La gestión del tiempo de la federación es de gran importancia ya que cuando se produce un avance en el tiempo lógico de la federación la RTI libera los mensajes con un tiempo lógico menor o igual al que se ha avanzado. La gestión interna de la RTI para liberar los mensajes se realiza mediante listas de distribución.

Los federados envían mensajes a la RTI con su propio tiempo de ejecución. La RTI los procesa y almacena los mensajes que deben ser liberados a los federados en su lista de distribución con el tiempo de ejecución anteriormente marcado. Por lo tanto cuando se produce un avance de tiempo lógico de la federación, la RTI envía aquellos mensajes de su lista que cumplen los criterios de tiempo anteriormente explicados. Los mensajes almacenados en las colas de la RTI que son enviados a los federados se llaman *callback*.

Las diferentes entradas que puede tener una lista de distribución son:

- **Add(F,A,T).** Esta entrada especifica que el federado F se ha suscrito al atributo A en un tiempo lógico T. Por lo tanto el federado F recibirá las *callback* de las actualizaciones del atributo A con un tiempo lógico mayor a T.
- **Delete(F,A,T).** Esta entrada especifica que el federado F deja de estar suscrito al atributo A en un tiempo lógico T. Por lo tanto el federado F no recibirá las *callback* de las actualizaciones del atributo A con un tiempo lógico mayor a T.

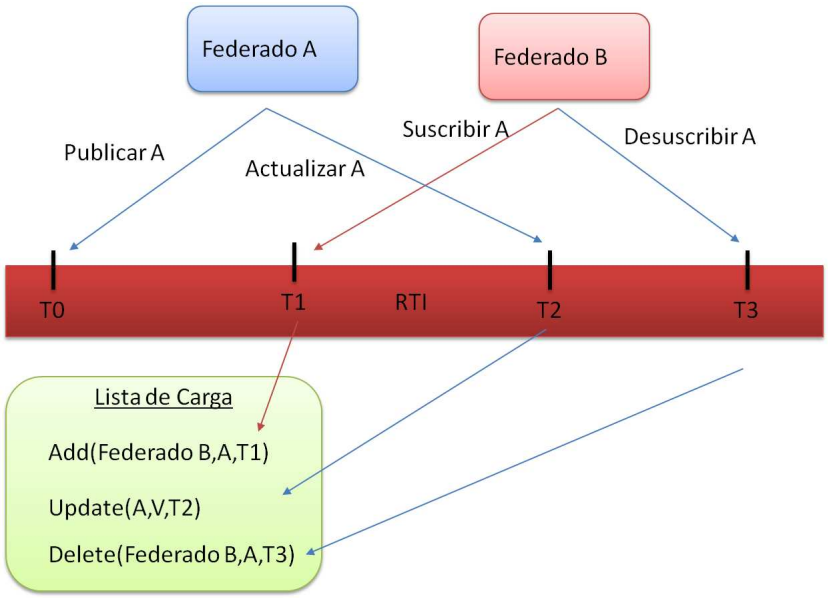

Figura 3.9: *Federation Time Axis* en la lista de distribución

- **Update(A,V,T).** Esta entrada especifica que se realiza una actualización del atributo A, a un valor V, en un tiempo lógico T. Por lo que los federados suscritos a ese atributo en un tiempo lógico menor o igual que T recibirán las *callback* de esa actualización.

De las entradas de la lista de distribución se deduce que un federado en el rol de *Time Constrained* sólo recibe actualizaciones del atributo al que se ha suscrito en un tiempo T si son realizadas en un tiempo T1 mayor que T. En el caso contrario no las recibirá (figura 3.9).

3.2. Necesidad de la definición de IRMs para entornos militares

La complejidad del estándar IEEE1516 ([14]) se traduce en diferentes implementaciones del mismo que dan como resultado sistemas e infraestructuras (RTI) que no son realmente interoperables entre sí.

La arquitectura HLA permite especificar un caso concreto de interoperabilidad de muchas maneras diferentes, todas ellas correctas. En ejercicios de simulación distribuida en el entorno militar esto se ha traducido, tradicionalmente, en una

falta de convergencia que complica enormemente la ejecución de las simulaciones. Normalmente se conoce el objetivo pero no hay una línea de acción común para llegar a él, dejando reducida la interoperabilidad a un hecho de suerte (elección de una línea de acción común por parte de todos los miembros participantes en el ejercicio).

Es necesario un marco de referencia que defina cuales son las líneas de acción que se deben tomar para cada caso de interoperabilidad definido en un entorno concreto. Este marco se ha definido para el entorno industrial mediante los cuatro tipos de IRMs ya explicados al principio de este capítulo, pero todavía no se ha avanzado en su definición para el entorno militar.

Parece que la combinación de los modelos de interoperabilidad de referencia definidos para entornos industriales y del estándar JC3IEDM ([23],[35]), que resume las necesidades de interoperabilidad en el entorno táctico-militar, puede ser un buen punto de partida para proponer unas guías de interoperabilidad en el entorno de la simulación militar de tipo constructivo.

3.3. Definición de IRMs militares

3.3.1. Consideraciones previas

La OTAN define la simulación constructiva como: "Modelos y simuladores cuyo ámbito son los usuarios que operan los sistemas simulados introduciendo datos pero sin intervenir en los datos de salida"([36]).

Si se tiene en cuenta la definición anterior se pueden identificar como simuladores constructivos (en el ámbito de las operaciones militares) aquellos que tienen como finalidad la simulación del proceso de mando y control (Command and Control o C2) de las operaciones militares.

Los sistemas de mando y control son los que dan soporte a este tipo de operaciones) y el JC3IEDM ([21]) es el estándar internacionalmente aceptado para el intercambio de información táctica militar entre sistemas C2.

En consecuencia, es necesario analizar de manera conjunta la especificación del JC3IEDM y los IRMs definidos por SISO ([59]) para decidir si estos modelos son de aplicación directa en el ámbito militar o si es necesaria la definición de otros nuevos para este contexto.

Para realizar este análisis se han identificado aquellos conceptos que definen el

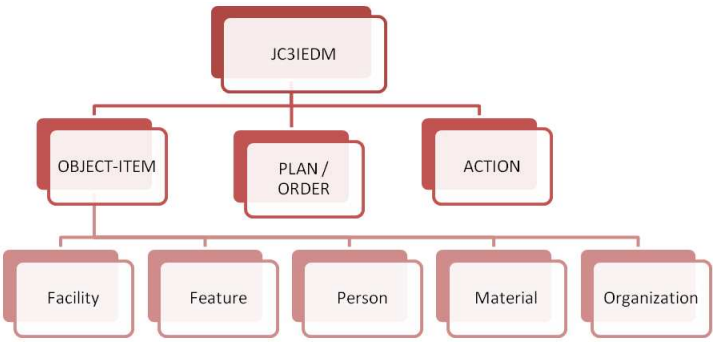

Figura 3.10: Esquema de datos del JC3IEDM

marco de trabajo del JC3IEDM (figura 3.10), ya que los modelos de interoperabilidad de referencia definidos para este entorno deberían dar cobertura a todos estos conceptos. En los siguientes puntos se resumen las deducciones del análisis realizado:

- **Object-Item.** Este concepto define aquellos objetos del campo de batalla que son de interés para la conducción de operaciones (cada objeto tiene un tipo *Object-Type*) y pueden ser compartidos o intercambiados. Un objeto puede ser: una persona (*person*), una organización (*organization*, aquí se incluyen a las unidades militares), un material (*material*), una construcción (*facility*) o una característica (*feature*). Este concepto puede relacionarse con la entidad, la estructura de datos y el recurso definidos en los IRMs industriales de SISO. Por lo tanto queda cubierto, en principio, con IRMs equivalentes a los de tipo A, B y D.

- **Action.** Este concepto se define como una interacción entre dos o varios *Object-Item* en el campo de batalla y puede ser planeada (*Action-Task*) o no planeada (*Action-Event*). Una *Action* hace que un sistema evolucione de un estado a otro ya que produce un efecto en los *Object-Item* que participan en ella (concepto de evento definido en los IRMs de SISO). Por lo tanto este concepto queda cubierto por un IRM equivalente al tipo C de SISO.

- **Plan and Order.** Este concepto define la agrupación de varios *Object-Item* o *Actions* que tienen una misma finalidad previamente establecida y planeada (de un plan pueden derivar varias órdenes). Este concepto no es trazable con ningún IRM definido por SISO para el entorno industrial ya que es muy característico de los entornos militares. Se podría tratar de encajar el Plan

and Order con una estructura de datos, pero en el IRM de tipo D sólo se contempla su compartición, mientas que en simulación constructiva también será necesaria su transferencia. Así que en este caso sería necesario definir un IRM completamente nuevo.

De este análisis se concluye que los IRMs definidos por SISO pueden ser adaptados para entornos táctico-militares pero que no son suficientes para cubrir todo el espectro del JC3IEDM, por lo que es muy probable que haya que definir un nuevo tipo de IRM que se denominará tipo E. Para realizar la adaptación propuesta hay que tener en cuenta las diferencias existentes entre el ámbito industrial y el táctico-militar, con las necesidades tremendamente específicas de este último.

3.3.2. Necesidades específicas del entorno táctico-militar

Es necesario adaptar los IRMs industriales definidos por SISO a las necesidades impuestas por el contexto operativo militar. Para lograr este objetivo lo primero es identificar las características del ámbito táctico-militar que influyen en la definición de los nuevos modelos:

1. Las actividades realizadas por los sistemas que participan en la ejecución de un IRM tienen una influencia directa sobre la ejecución del mismo. Por lo tanto cada sistema que intervenga en un IRM debe tener disponibles los resultados de las actividades que realizan el resto (en ese mismo tiempo lógico) antes de seguir con su ejecución.

2. En las operaciones militares terrestres el tiempo real siempre supera los minutos e incluso las horas. Esto implica que se pueden emplear protocolos para la coordinación de la ejecución del IRM que cumplan con estas latencias.

3. La tolerancia a fallos es un factor crítico para estos sistemas ya que un fallo en la ejecución de un IRM supone un fallo de coordinación y puede conllevar la pérdida de vidas humanas (siempre un coste demasiado elevado).

Estas características justifican el empleo del módulo *Time Management* definido para el estándar IEEE1516 ya que permite coordinar y sincronizar las actividades de los federados con una latencia que no supone un problema en este ámbito. Ahora bien, hay que especificar cómo se va a emplear este módulo ya que permite un amplio abanico de posibilidades. Los criterios de empleo de este módulo propuestos en esta investigación para el contexto táctico-militar son:

- Hasta que todos los federados no requieran un avance en el tiempo, el tiempo lógico de la federación no puede avanzar (de esta manera se puede cumplir la característica 1). Para ello el criterio de avance debería ser para cada federado *Time Regulating* y *Time Constrained* pero con alguna salvedad respecto a la definida en el estándar. Si el tiempo avanza en cuanto algún federado lo solicite, esto puede provocar que ciertos federados no realicen todas sus actividades, teniendo repercusiones sobre el resultado de la ejecución del IRM. Por lo tanto, el avance en el tiempo debe ser solicitado por todos y cada uno de los federados participantes en el IRM.

- Las actividades de los federados dan como resultado el envío de mensajes hacia la RTI. La pregunta es ¿cuál es el criterio de ordenación más adecuado para estos mensajes? En esta tesis doctoral se proponen dos alternativas dependiendo del número de federados participantes en la simulación distribuida. Si son pocos los federados (menos de cinco suele ser el límite, aunque depende en gran medida de la complejidad que se capaz de gestionar cada implementación de RTI), el empleo del criterio *Casual Order* sería el más adecuado. Sin embargo si el número de federados aumenta por encima de cinco, sería mejor optar por el criterio *Time Stamp Order*.

- La ejecución de los IRMs en este ámbito es una secuencia de actividades que ejecutan los federados. Para su definición este conjunto de actividades se deben agrupar de manera secuencial en eventos.

  Un evento agrupa a un conjunto de actividades que deben realizar los federados participantes en el IRM y que no requieren sincronización entre si. Por lo tanto en este caso el tipo de sincronía *Event-Driven* parece la más adecuada.

  Esta sincronización basada en eventos es posible ya que el tiempo real para este tipo de simuladores es de minutos u horas (característica 2).

- La información enviada y recibida por los federados durante la ejecución de los IRMs es un factor crítico en la consecución de resultados correctos (característica 3). Por este motivo los eventos que definen los IRMs deben asegurar que la información que se envía y la que se recibe es la misma, incluyendo todas las comprobaciones que sean necesarias para cumplir este propósito.

3.3.3. Rol de un federado

Los IRMs definidos por SISO para simuladores industriales especifican la participación de dos federados (tres en el caso del IRM A3) en cada modelo de interoperabilidad. Esto no es eficiente en la mayoría de los casos de aplicación militar, ya que es muy común que la ejecución de un IRM pueda requerir cuatro o más federados (hay que tener en cuenta que una unidad suele tener tres o más unidades subordinadas en la mayoría de los casos).

Por lo tanto para la definición de los IRMs militares es necesario incluir el concepto de rol, que engloba a varios federados bajo una misma denominación. De esta manera un IRM que para entorno industrial define dos federados en el entorno militar define dos roles (A y B) que agrupan a varios federados que realizan las mismas actividades durante la ejecución del modelo.

En la ejecución de cualquier IRM debe haber, como mínimo, un federado por cada rol definido en el modelo. Por otro lado el número máximo de federados por rol es un aspecto que se debe estar limitado por la escalabilidad del sistema (RTI, infraestructura de red, etc).

3.3.4. Definición del IRM tipo A para simuladores constructivos militares

El IRM tipo A básico para simuladores militares constructivos consiste en un rol A que envía un *Object-Item* a otro rol B. A diferencia del IRM tipo A definido por SISO ([59]) la entidad transferida no es el resultado de un proceso de elaboración que se propaga dentro de una cadena de producción sino que es un elemento del campo de batalla.

En muchos casos la transferencia de un *Object-Item* tiene como objetivo la asignación del mismo a otra unidad diferente. Sin embargo, en otros casos tiene sólo un carácter de transferencia de la responsabilidad sobre la información.

Al igual que el IRM tipo A definido por SISO, el de aplicación militar contempla tres subtipos que definen situaciones idénticas pero con los matices mencionados anteriormente.

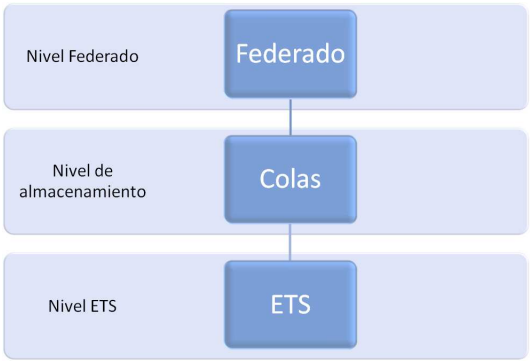

Figura 3.11: ETS definido para el IRM tipo A

3.3.4.1. Definición de la entidad transferida (ETS)

La entidad que se va a transferir en este IRM es un *Object-Item* cuya definición debe realizarse según el formato OMT (Object Model Template) definido para HLA ([16]) y no según el relacional definido en el JC3IEDM.

El formato OMT define un FOM (Federation Object Model) para cada federación dentro del cual se definen objetos, atributos y las relaciones jerárquicas entre ellos (aquí están los *Object-Item* que pueden participar en este IRM) de tal manera que los objetos subordinados hereden los atributos de los objetos superiores.

Esta estructura de datos jerarquizada se adapta bien al entorno militar para definir diferentes áreas de información o interés llamados niveles (figura 3.11). Cada nivel puede componerse a su vez de varios niveles de jerarquía de objetos bajo un mismo concepto o interés. De esta manera un federado con el rol B sólo recibe el nivel de información de la entidad transferida que necesita. Con este tipo de definición se consigue un envío de información selectivo y se utilizan los recursos de manera más eficiente, en concreto, el ancho de banda de la infraestructura de comunicaciones, que suele ser un recurso crítico en el entorno de aplicación militar.

Por lo tanto la especificación de la ETS (Entity Transfer Specification) para este IRM queda definida por los siguientes niveles:

- **Nivel federado.** Este nivel contiene aquella información necesaria para la ejecución del IRM como, por ejemplo, los federados que intervienen y el rol que tiene asignado cada uno de ellos.
- **Nivel de almacenamiento.** Este nivel contiene aquella información necesaria para la gestión de las colas de recepción de los federados con rol B. Por ejemplo,

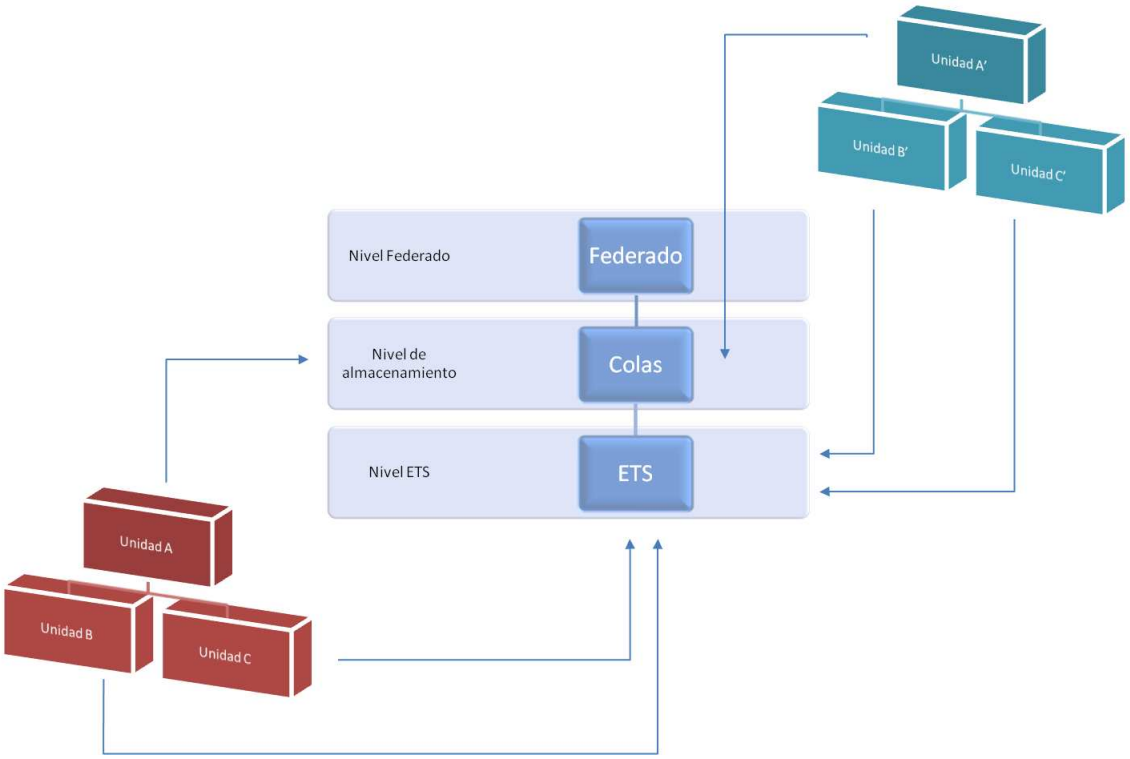

Figura 3.12: Transferencia por niveles de una entidad entre unidades

aquí se puede incluir el estado de las colas o el espacio disponible en las mismas.

- **Nivel ETS.** Este nivel contiene aquella información específica de la entidad que se transfiere. Por lo tanto este nivel es el menos genérico y el mas específico, ya que está ligado a un *Object-Item* concreto.

En el nivel ETS los objetos definidos tienen los atributos de este nivel y los heredados del resto de niveles (esto se debe a la herencia de atributos definida para OMT). En consecuencia el nivel ETS es el más específico de los tres siendo de utilidad a aquellos federados que representan a unidades de menor nivel jerárquico (donde es necesaria la información más detallada para realizar las simulaciones).

En conclusión, dependiendo de la necesidad de información que va ligada a un nivel jerárquico en la organización, el federado recibirá información de la entidad transferida en un nivel u otro (figura 3.12).

3.3.4.1.1. Ejemplo de ETS. En el ámbito táctico-militar una ETS puede estar basada en un *Object-Item* definido en el JC3IEDM. En este ejemplo se va a

definir una ETS que contiene la información relacionada con una unidad militar (organization en el JC3IEDM).

En este caso la entidad está compuesta por tres clases, cada una en un nivel de ETS diferente y jerárquicamente dependientes entre sí. Estas clases son:

- **Federate Information.** Esta clase tiene los siguientes atributos:
  - El atributo Rol cuyo valor especifica el rol que tiene en el IRM el federado que la publica.
  - El atributo IRM cuyo valor especifica en qué tipo de IRM está participando la ETS.
- **Queue Information.** Esta clase tiene los siguientes atributos:
  - El atributo Estado Cola cuyo valor indica si la cola receptora del federado que la ha publicado está llena o no.
  - El atributo Volumen cuyo valor indica el volumen disponible en la cola receptora del federado que la ha publicado.
  - El atributo Estado Transferencia cuyo valor indica el federado autorizado para realizar la transferencia de la entidad a la cola receptora del federado que ha publicado la ETS.
- **Organization.** En esta clase se añaden todos aquellos atributos que especifican las características de una unidad militar (como por ejemplo tipo, estado operativo, localización).

3.3.4.2. Definición del modelo subtipo A1 para simuladores constructivos militares

Este subtipo consiste en dos roles (A y B) que intercambian una ETS sin que exista ningún factor externo que influya en la transferencia.

Este subtipo se define mediante cinco eventos en los que los federados que intervienen en el IRM deben ejecutar las actividades definidas para ellos.

3.3.4.2.1. Evento 1. En este evento el rol A publica en la RTI la entidad que quiere transferir (ETS A). En el entorno de la simulación constructiva militar será un *Object-Item* definido con anterioridad en el FOM de la federación.

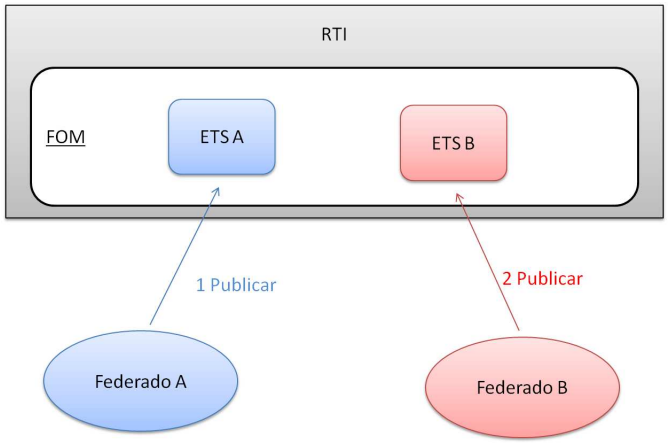

Figura 3.13: Esquema del evento 1 en el IRM subtipo A1

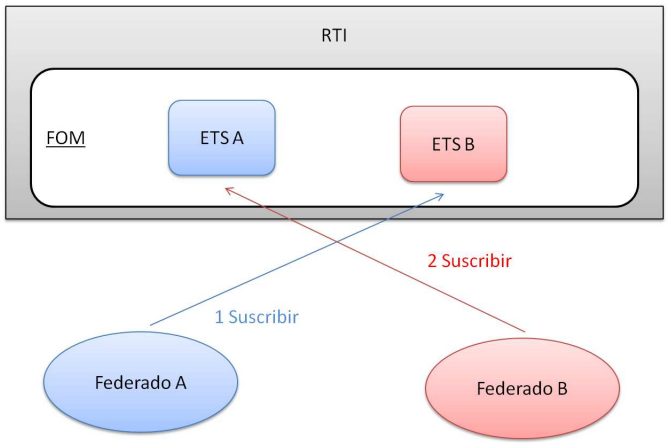

Figura 3.14: Esquema del evento 2 en el IRM subtipo A1

El rol B publica a su vez otra entidad (ETS B) idéntica a la que publica el rol A. El objetivo de esta entidad es ser un mecanismo de tolerancia a fallos, puesto que como se describe en otros eventos a continuación, el rol A tiene que comprobar durante la ejecución del IRM que la entidad que le llega al rol B llega de manera íntegra y es exactamente la que se ha enviado (figura 3.13).

En el diagrama de tiempos (figura B.1 en el apéndice B de este documento) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1). Todos los diagramas de tiempos de todos los eventos se han incluido en el apéndice B de esta tesis para mejorar la legibilidad de este capítulo.

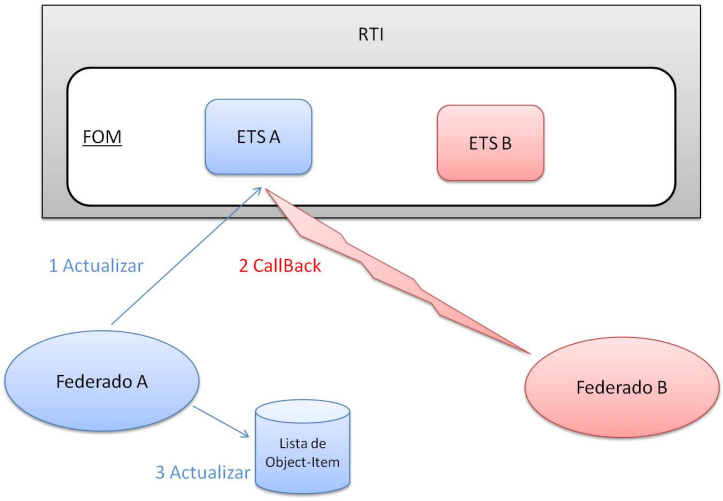

Figura 3.15: Esquema del evento 3 en el IRM subtipo A1

3.3.4.2.2. Evento 2. En este evento el rol B se suscribe a la entidad publicada por el rol A en el evento anterior (ETS A). Así cuando el rol A actualiza sus atributos el rol B los recibe haciéndose la transferencia de entidad entre ambos roles (en esta actividad el rol B se suscribe al nivel de ETS A que corresponde a su necesidad de información).

El rol A se suscribe a su vez a la entidad publicada en el evento anterior por el rol B (ETS B). De esta manera puede comprobar que la entidad publicada por el rol B coincide con la publicada por el rol A. Esta comprobación de la ETS se realiza siempre por niveles de información (figura 3.14).

En el diagrama de tiempos (figura B.2) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.4.2.3. Evento 3. En este evento el rol A actualiza los atributos de la ETS A publicada en el evento 1. Por lo tanto el rol B recibe los datos de los atributos actualizados al final del evento (hay que aclarar que el rol B sólo recibe los datos de los atributos que correspondan a los objetos del nivel suscrito en el evento 2).

Este evento representa la transferencia de la entidad desde el punto de vista del rol A. Por lo tanto una vez actualizados los atributos de ETS A el rol A debe actualizar su lista interna de *Object-Item* (figura 3.15).

En el diagrama de tiempos (figura B.3) se representan las actividades realizadas

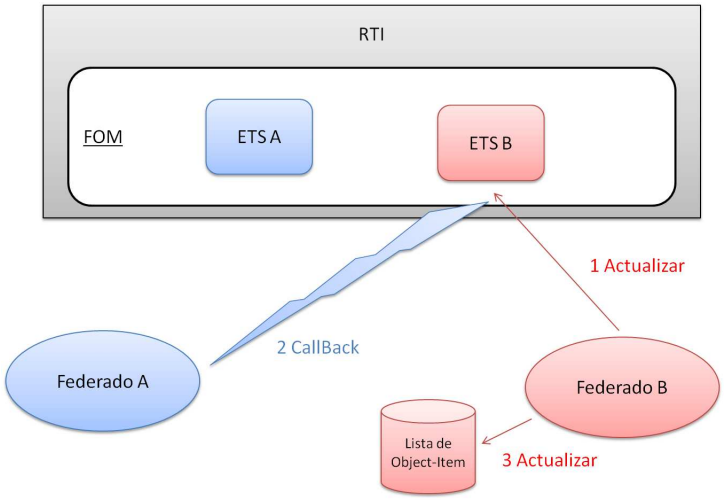

Figura 3.16: Esquema del evento 4 en el IRM subtipo A1

por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.4.2.4. Evento 4. En este evento el rol B actualiza los datos de la ETS B publicada en el evento 1 con los datos que ha recibido de la actualización de la ETS A en el evento anterior (figura 3.16).

El rol A recibe al final del evento los datos actualizados de la ETS B pudiendo de esta manera comprobar que son los mismos con los que actualizó la ETS A en el evento anterior (de nuevo una comprobación que asegura la tolerancia a fallos del modelo).

En el diagrama de tiempos (figura B.4) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

3.3.4.2.5. Evento 5. Para finalizar la ejecución del IRM el rol A despublica la entidad (ETS A) que publicó en el evento 1. De la misma manera el rol B despublica la entidad (ETS B) publicada en el mismo evento (figura 3.17).

En el diagrama de tiempos (figura B.5) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

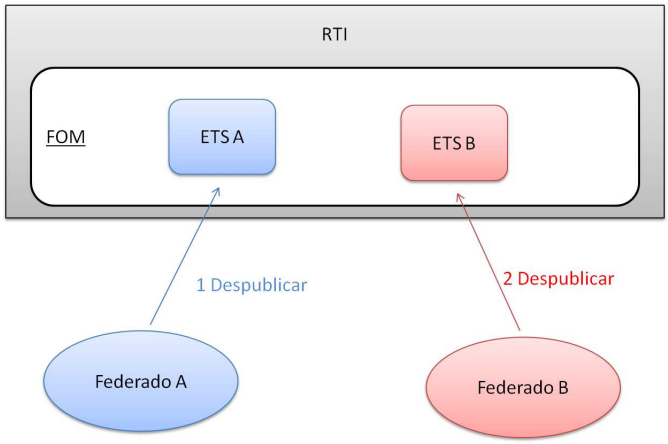

Figura 3.17: Esquema del evento 5 en el IRM subtipo A1

3.3.4.3. Caso práctico de aplicación del IRM A1

En esta sección se aplica este modelo de referencia a una necesidad de interoperabilidad entre dos simuladores constructivos. Primero se describe el contexto en el que se va a aplicar el IRM para poder identificar los diferentes elementos que lo definen:

"La composición de las unidades de combate militares puede variar durante el transcurso de una operación. Esta variación se adapta a las necesidades momentáneas del combate, lo que permite una flexibilidad en la dirección de las mismas.

De esta manera si la brigada X pasa de posiciones de retaguardia a posiciones de vanguardia, necesita más apoyos de fuego de los que estaba requiriendo hasta el momento. Por ese motivo la brigada XI, que pasa a posiciones de retaguardia y deja su sitio a la brigada X, le agrega su grupo de artillería de campaña (grupo de artillería XI). De esta manera se refuerzan los apoyos de fuego de la brigada X que es la que pasa a combatir en primera línea."

En este contexto se pueden identificar los diferentes elementos definidos para el IRM A1 en su aplicación militar. Estos son:

- **Federado A.** Es el sistema que simula las actividades de la brigada XI. Este sistema tiene dentro de su relación de unidades subordinadas el grupo de artillería de campaña XI.
- **Federado B.** Es el sistema que simula las actividades de la brigada X. Este sistema en un tiempo T5 (fin del IRM) tiene en su relación de unidades subor-

dinadas el grupo de artillería de campaña XI y podrá programar actividades con él.

- **ETS.** Es la especificación del grupo de artillería de campaña (en el formato JC3IEDM sería un *Object-Item* del tipo *organization*). Este grupo en un tiempo T0 pertenece a la brigada XI y en un tiempo T5 a la brigada X. Es decir, se trata de la entidad transferida entre los dos simuladores durante la ejecución del IRM.

- **Cola B.** Son las unidades subordinadas a la brigada X (en el formato JC3IEDM es el conjunto de *organizations* que tienen una relación de subordinación con la brigada X).

Con este ejemplo sencillo se observa que a pesar de las diferencias conceptuales existentes entre las entidades para los simuladores industriales y simuladores constructivos, con la adaptación propuesta se puede modelar una situación muy común en operaciones militares aplicando este modelo de referencia.

3.3.4.4. Definición del modelo subtipo A2 para simuladores constructivos militares

Este subtipo consiste en dos roles (A y B) que intercambian una ETS teniendo en cuenta que la capacidad de la cola de recepción del rol B (cola B) no es ilimitada. La situación descrita por este IRM es crítica en un entorno militar, puesto que si se gestiona mal, se puede traducir en unidades esperando a que se libere la cola B. En muchas situaciones esto puede hacer aparecer un objetivo rentable para el enemigo, lo que pone en peligro la supervivencia de las unidades.

Para evitar la situación descrita anteriormente la definición del IRM A2 debe cumplir los siguientes requisitos:

- El rol A comprueba si hay espacio en la cola B antes de enviar la entidad. Esta información está dentro del nivel de almacenamiento de la definición de la ETS.

- Si hay espacio, se procede a realizar una reserva para garantizar que seguirá existiendo cuando llegue la entidad transferida.

- En el caso que no haya espacio en la cola B, hay que bloquear la transferencia esperando a que exista espacio para reanudarla.

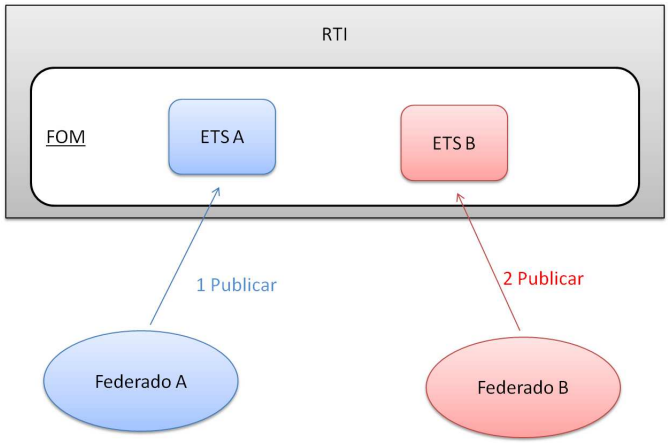

Figura 3.18: Esquema del evento 1 en el IRM subtipo A2

Este subtipo se divide en cinco eventos específicos de este IRM y tres que son idénticos a eventos ya especificados en el modelo A1 donde los federados que intervienen deben ejecutar las actividades definidas para ellos.

3.3.4.4.1. Evento 1. Este evento es idéntico al evento 1 definido para el IRM A1. El rol A publica en la RTI una entidad (ETS A) idéntica a la que publica el rol B (ETS B, figura 3.18).

En el diagrama de tiempos (figura B.6) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1).

3.3.4.4.2. Evento 2. Este evento es idéntico al evento 2 definido para el IRM A1. El rol A se suscribe a la entidad (ETS B) publicada en el evento 1 por el rol B y el rol B se suscribe a la entidad (ETS A) publicada por el rol A en el evento 1 (figura 3.19).

En el diagrama de tiempos (figura B.7) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.4.4.3. Evento 3. En este evento se definen las actividades necesarias para realizar una reserva de espacio en la cola B de manera que la entidad transferida pueda ser almacenada a su llegada. El objetivo es evitar que el rol A envíe una entidad que no pueda ser almacenada en la cola B (figura 3.20).

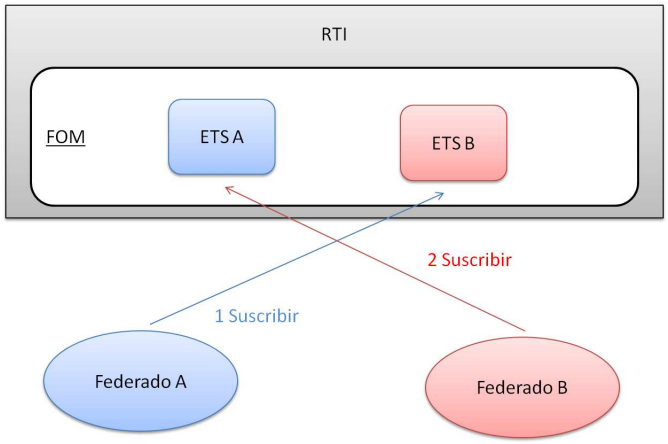

Figura 3.19: Esquema del evento 2 en el IRM subtipo A2

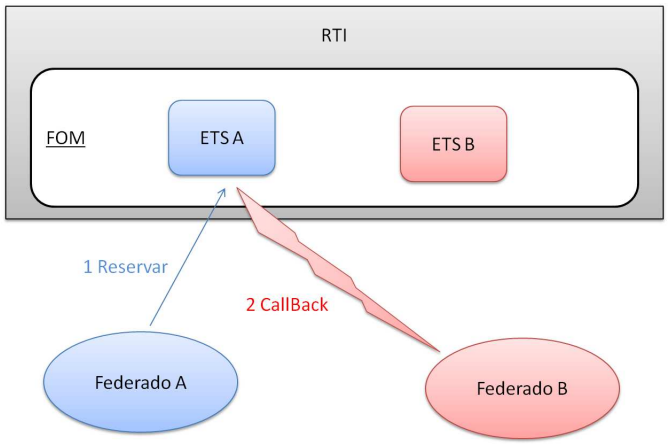

Figura 3.20: Esquema del evento 3 en el IRM subtipo A2

Para realizar la reserva el rol A actualiza el nivel de almacenamiento de la ETS A publicada con los datos de ocupación de espacio de la entidad transferida. De esta manera el rol B, suscrito a ella (en el evento 2), recibirá las *callback* con los datos de la reserva.

En el diagrama de tiempos (figura B.8) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.4.4.4. Evento 4. Si la cola receptora no tiene espacio suficiente para almacenar la entidad transferida, en este evento se ejecutan las actividades necesarias

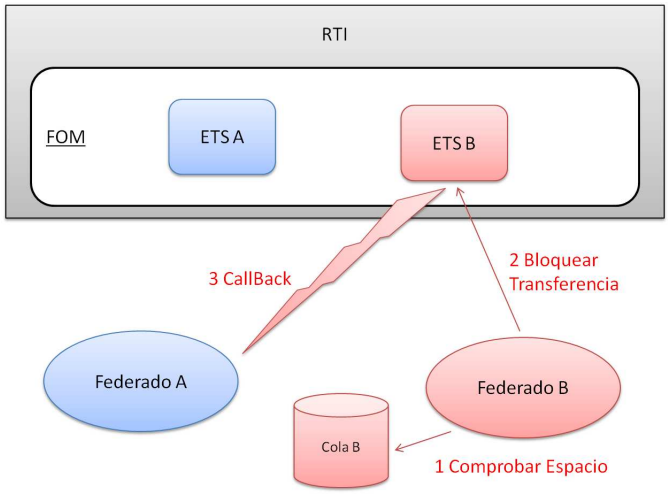

Figura 3.21: Esquema del evento 4 en el IRM subtipo A2

para bloquear la transferencia (figura 3.21).

El rol B analiza las *callback* recibidas en el evento 3 y comprueba que no hay espacio suficiente en la cola B para la entidad que el rol A desea transferir. En este caso, actualiza el nivel de almacenamiento de la ETS B (publicada en el evento 1) para bloquear el proceso de transferencia de la entidad. El rol A suscrito a ETS B recibe las *callback* de esta actualización y bloquea el proceso de transferencia hasta que la cola B tenga espacio suficiente.

En el diagrama de tiempos (figura B.9) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

3.3.4.4.5. Evento 5. En este evento se definen las actividades que realizan los dos roles cuando la cola B tiene espacio suficiente para almacenar la entidad transferida. En este caso se desbloquea la transferencia de la entidad (figura 3.22).

Por lo tanto el rol B comprueba el espacio libre en la cola B y cuando este es suficiente, actualiza el nivel de almacenamiento de la ETS B para desbloquear la transferencia. El rol A suscrito a ETS B recibe las *callback* y transfiere la entidad al rol B.

La ejecución de este IRM a partir de este evento es la misma que la del subtipo A1 a partir del evento 3.

En el diagrama de tiempos (figura B.10) se representan las actividades realizadas

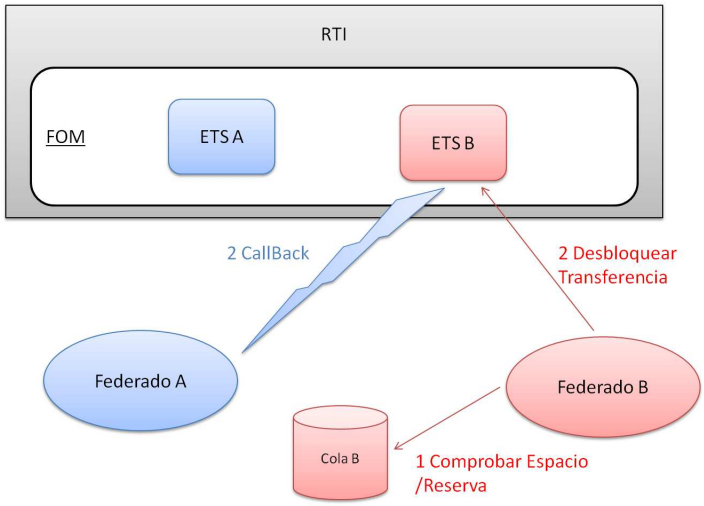

Figura 3.22: Esquema del evento 5 en el IRM subtipo A2

por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

3.3.4.5. Caso práctico de aplicación del IRM A2

El objeto de este caso práctico es aplicar el modelo de interoperabilidad subtipo A2 a una situación real de necesidad de interoperabilidad entre simuladores militares constructivos. Para ello el contexto escogido es el siguiente:

"Una unidad de combate tipo brigada tiene una unidad logística que apoya a sus unidades subordinadas (apoyo directo). Una de las funciones del grupo logístico es el abastecimiento de recursos a las unidades de la brigada siguiendo las prioridades del jefe de ésta.

Uno de los recursos que abastece el grupo logístico es el combustible para los vehículos, que se almacena en unos depósitos de grandes dimensiones antes de repartirlo a las unidades. Cuando una unidad necesita combustible lo solicita al grupo logístico y éste, a través de camiones cisterna, lo lleva físicamente a la unidad que lo ha pedido y lo descarga en los depósitos que tiene la unidad para almacenar su propio combustible.

De esta manera el batallón I, perteneciente a la brigada X, pide combustible al grupo logístico X pero sus depósitos (los del batallón I) están llenos. Por lo tanto los camiones cisterna del grupo logístico X no salen hasta que el batallón I tenga espacio suficiente para almacenar el combustible que ha pedido."

En este caso se observa claramente el sentido de la definición de este subtipo del IRM tipo A. Si una columna de camiones cisterna saliera para entregar combustible y tuviera que esperar en un punto del terreno a que el batallón I tuviera espacio suficiente en sus depósitos, sería un objetivo muy rentable (estático y desprotegido) para un potencial ataque de las fuerzas enemigas.

En este contexto se puede identificar los diferentes elementos definidos para el IRM A2 en su aplicación militar. Estos son:

- **Federado A.** Es el sistema que simula las actividades del grupo logístico X.
- **Federado B.** Es el sistema que simula las actividades del batallón I.
- **ETS.** Es la especificación del combustible que se va a enviar al batallón I (en el formato JC3IEDM sería un *Object-Item* del tipo *material*).
- **Cola B.** Son los depósitos que tiene el batallón I para almacenar el combustible que está a su cargo.

En este ejemplo se observa la viabilidad de modelar una situación de apoyo logístico común con este modelo de referencia. También se demuestra lo importante que es tener en cuenta ciertos aspectos críticos en una situación táctico-militar para la definición de los modelos de interoperabilidad.

3.3.4.6. Definición del modelo subtipo A3 para simuladores constructivos militares

Este subtipo consiste en tres roles (A, B y C) donde el rol A y el rol C envían entidades al rol B y éstas llegan en el mismo tiempo real (*wallclock*). Al igual que se explicó en el anterior IRM, esta situación en el contexto táctico-militar es crítica, ya que supone que una entidad debe esperar a que la otra sea recibida.

Para evitar esto hay que realizar una gestión basada en listas de reserva o de carga donde un rol realiza una reserva antes de transferir su entidad. Estas reservas permiten al rol receptor gestionar los envíos de las entidades de tal manera que autorice las transferencias para que no haya superposición entre ellas minimizando los tiempos de espera (incluso anulándolos si es posible).

Para ello el rol B (en este IRM, el que recibe las entidades) emplea un protocolo de prioridades que le permite ordenar las solicitudes de reserva que le llegan (en este orden el rol B va autorizando las transferencias).

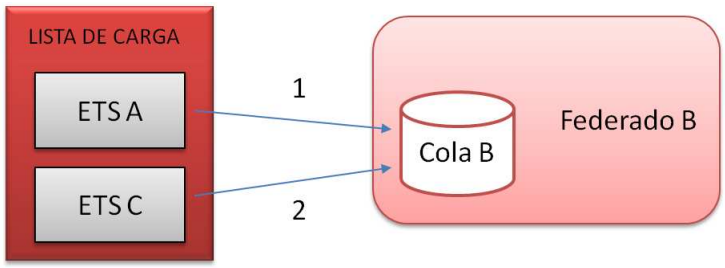

Figura 3.23: Lista de carga del federado B

La base de este protocolo está en el concepto de reserva, ya que el rol que envía una entidad debe realizar una reserva al rol receptor para que éste pueda especificar el momento en el que se debe realizar la transferencia.

El rol B va ordenando las reservas en una lista de carga según el orden de prioridad que se ha establecido. Esta lista de carga le permite ir autorizando las transferencias por orden (figura 3.23).

Un rol realiza una reserva enviando un vector al rol B llamado LLD (Load List Data) cuyos parámetros son Fa, V, TR y P (figura 3.24 y 3.25):

- **Federado origen (Fa).** Información del federado que va a enviar la entidad.
- **Volumen ocupado (V).** El volumen que ocupa la entidad que va a ser enviada en la cola de recepción.
- **Tiempo de reserva (TR).** El tiempo lógico de la federación en el que el federado origen hace la reserva.
- **Prioridad (P).** Prioridad que se le asigna a la entidad enviada.

Los vectores de reserva serán ordenados dentro de la lista de carga siguiendo los siguientes criterios:

1. Los de prioridad más alta.
2. Los que su tiempo de reserva sea más bajo.
3. Los que ocupen un mayor volumen.

En caso de que los criterios de clasificación fueran iguales para las dos entidades, se ordenarían por orden alfabético del nombre del federado origen.

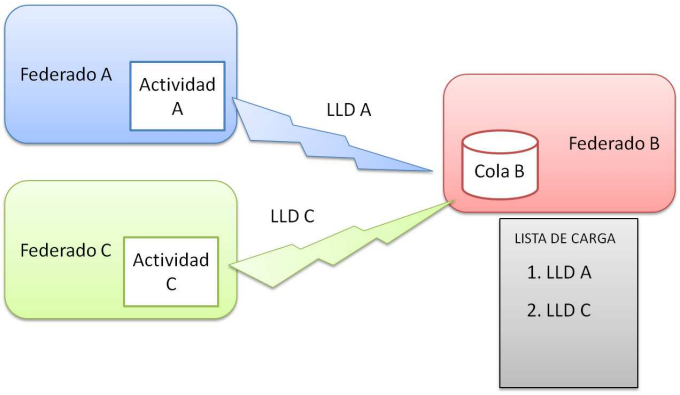

Figura 3.24: Envío del LLD al federado B

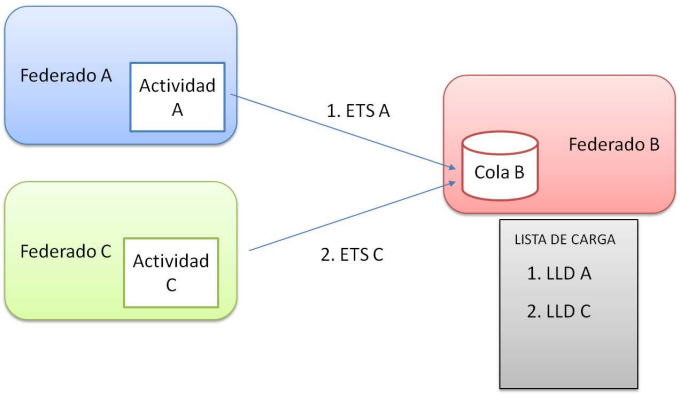

Figura 3.25: Envío de la ETS al federado B

La justificación de la lista de prioridades está basada en las siguientes características de las operaciones militares:

- La prioridad establecida por el jefe de una unidad militar es el criterio que prevalece por encima de todo.
- A igualdad de prioridad, el tiempo de reserva más bajo debe prevalecer. Esto se debe a que si dos entidades enviadas llegan al mismo tiempo la que antes sale tiene un tiempo de recorrido (y por lo tanto, de exposición a un potencial ataque) mayor que la otra. Por lo tanto para mantener el equilibrio entre las entidades hay que dar prioridad a la que antes sale y por lo tanto marca un tiempo de reserva menor.
- A igualdad de prioridad y tiempo lógico, la entidad de mayor volumen debe

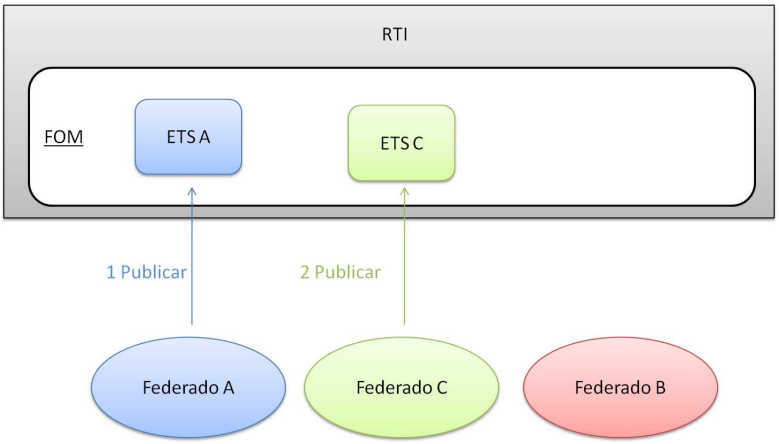

Figura 3.26: Esquema del evento 1 en el IRM subtipo A3

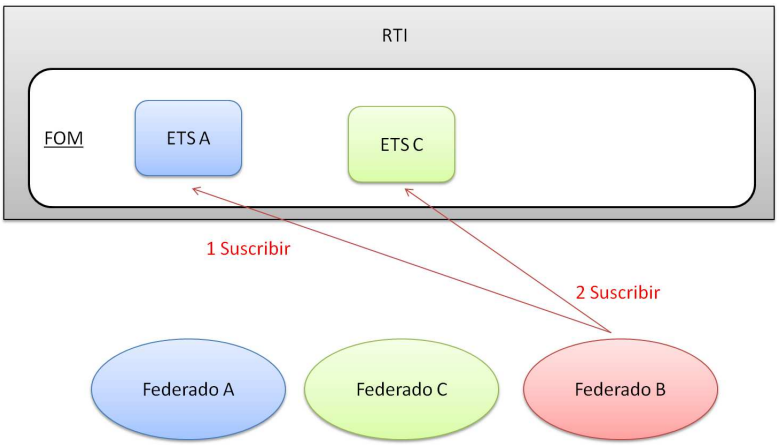

Figura 3.27: Esquema del evento 2 en el IRM subtipo A3

ser la primera en entrar en la cola de recepción. Esto se debe a que a mayor volumen, la posibilidad de ocultación sobre el terreno es menor y por tanto es un objetivo con mejor adquisición.

En los siguientes párrafos se describen las actividades que deben realizarse durante los eventos que definen este subtipo de IRM.

3.3.4.6.1. Evento 1. Este evento define las actividades necesarias para que los roles A (ETS A) y C (ETS C) publiquen en la RTI la entidad que van a transferir al rol B (figura 3.26).

En el diagrama de tiempos (figura B.11) se representan las actividades realizadas

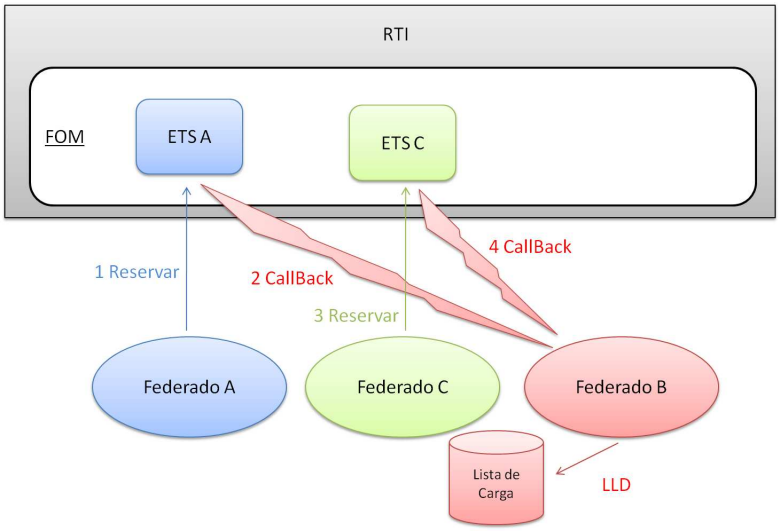

Figura 3.28: Esquema del evento 3 en el IRM subtipo A3

por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1).

3.3.4.6.2. Evento 2. En este evento el rol B se suscribe a las entidades que publicaron el rol A y C en el evento anterior. De esta manera los roles A y C podrán mandar su vector de reserva actualizando sus entidades y el rol B las recibirá como *callback* (figura 3.27).

En el diagrama de tiempos (figura B.12) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.4.6.3. Evento 3. En este evento se describen las actividades mediante las cuales el rol A y C envían el vector de reserva al rol B (figura 3.28).

Para ello actualizan el nivel de almacenamiento de las entidades publicadas en el evento 1 con los parámetros definidos para el vector reserva. El rol B, como está suscrito a estas entidades, recibe las *callback* de los vectores y las ordena en su lista de carga.

En el diagrama de tiempos (figura B.13) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

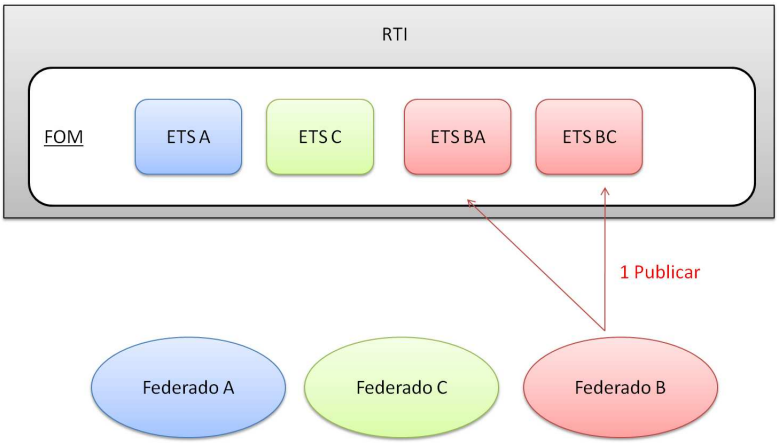

Figura 3.29: Esquema del evento 4 en el IRM subtipo A3

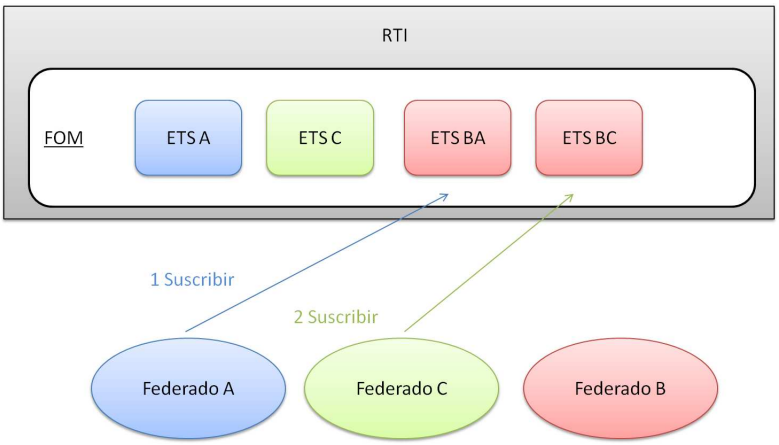

Figura 3.30: Esquema del evento 5 en el IRM subtipo A3

3.3.4.6.4. Evento 4. Una vez enviados los vectores de reserva el rol B publica en la RTI dos entidades (ETS BA y ETS BC) cada una de ellas idéntica a las que publicaron el rol A y el rol C en el evento 1. De esta manera el rol B se prepara para realizar una transferencia de entidad (de rol a rol) dependiendo del orden de prioridad en su lista de carga (figura 3.29).

En el diagrama de tiempos (figura B.14) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

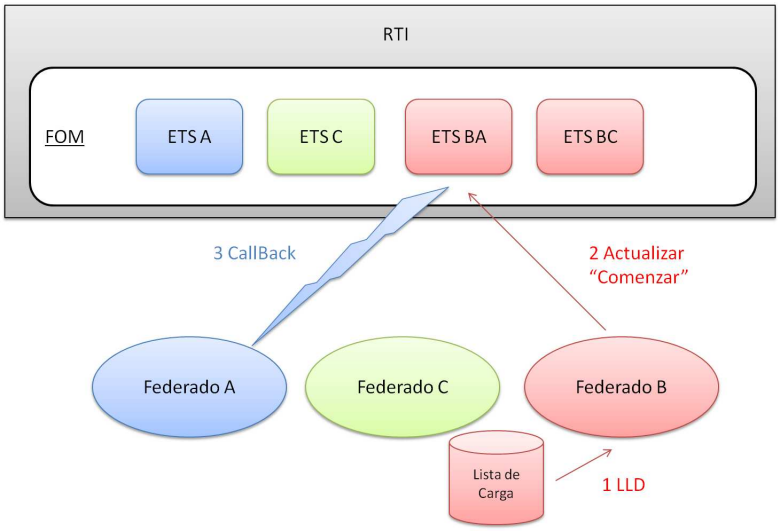

Figura 3.31: Esquema del evento 6 en el IRM subtipo A3

3.3.4.6.5. Evento 5. Para poder realizar la transferencia de su entidad, el rol A se suscribe en este evento a la entidad ETS BA y el rol C, a la ETS BC. De esta manera los roles A o C podrán recibir las *callback* de las actualizaciones de esa entidad. Este modo es el empleado por el rol B para comunicar a los roles que empiecen a transferir su entidad (figura 3.30).

En el diagrama de tiempos (figura B.15) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

3.3.4.6.6. Evento 6. En este evento el rol B consulta su lista de carga y empieza la transferencia de la entidad que está en la posición más alta de la lista (figura 3.31).

Para ello actualiza el nivel de almacenamiento de la entidad ETS BA o ETS BC (dependiendo de las prioridades) donde ordena el comienzo de la transferencia. En este caso el rol C o el A (dependiendo del elegido) recibe la *callback* y comienza el envío de la entidad hacia el rol B.

En el diagrama de tiempos (figura B.16) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T5 y T6).

A partir de este evento, para transferir las dos entidades se seguirá el mismo esquema que en el IRM subtipo A1 a partir del evento 3. Es decir se transfiere la ETS BC o la ETS BA siguiendo el IRM A1 a partir del evento 3 y cuando se

termina se ejecuta el evento 6 de este IRM (es decir se consulta la lista de carga) y se transfiere la otra entidad ETS BA o ETS BC siguiendo el IRM A1 a partir del evento 3.

3.3.4.7. Caso práctico de aplicación del IRM A3

El objeto de este apartado es demostrar que el modelo de interoperabilidad subtipo A3 es aplicable a una situación real donde la interoperabilidad entre simuladores constructivos es necesaria. Para ello el contexto seleccionado es el siguiente:

"Siguiendo con el apoyo directo que realiza un grupo logístico a las unidades de una brigada, en este caso se va escoger como referencia la función de mantenimiento que realiza el mismo.

El grupo logístico realiza una labor de reparación o mantenimiento de los vehículos de las unidades de la brigada. Para ello calcula una carga de trabajo de los diferentes talleres de los que dispone. De esta manera puede prever para cada conjunto vehículo/avería el tiempo que va a emplear, permitiéndole gestionar las cargas de trabajo.

En este caso el grupo logístico X tiene que atender dos solicitudes de mantenimiento simultáneas que provienen del batallón I y del batallón II pero que exceden su carga de trabajo."

En este caso, y siguiendo el IRM A3, una de las dos solicitudes tiene prioridad sobre la otra. Por lo tanto un vehículo pasa a la cadena de mantenimiento antes que el otro, realizando así una gestión de la cadena correcta y no dejando tiempos muertos en los que un vehículo esté parado en el taller.

En este contexto se pueden identificar los diferentes elementos definidos para el IRM A3 en su aplicación militar. Estos son:

- **Federado A.** Es el sistema que simula las actividades del batallón I.
- **Federado B.** Es el sistema que simula las actividades de mantenimiento del grupo logístico X.
- **Federado C.** Es el sistema que simula las actividades del batallón II.
- **ETS A.** Es el vehículo que envía el batallón I a reparar (según el JC3IEDM *Object-Item* tipo *material*).
- **ETS C.** Es el vehículo que envía el batallón II a reparar (según el JC3IEDM *Object-Item* tipo *material*).

- **Cola B.** Es la sala de recepción de vehículos antes de entrar a la cadena de mantenimiento. Normalmente suelen ser naves o aparcamientos donde hay una capacidad máxima de vehículos establecida.

- **LLD.** Es la lista en la que se almacenan las órdenes de trabajo que envían el batallón I y el batallón II para solicitar la reparación de los vehículos.

En este ejemplo se modela una acción logística habitual en unidades de combate y se puede observar la importancia del protocolo de prioridades para la aplicación de este subtipo de IRM en el campo táctico-militar ya que la optimización de la cadena de mantenimiento va a depender de la gestión de las órdenes de trabajo.

3.3.5. Definición del IRM tipo B para simuladores constructivos militares

Este modelo de interoperabilidad consiste en un rol A que comparte un recurso con otro rol B. En el ámbito táctico-militar un recurso (tomando como referencia el JC3IEDM) puede ser un *Object-Item* de los siguientes tipos:

- Personal.

- Organización.

- Material.

- Construcción.

Una de las diferencias principales entre este IRM y el tipo A radica en la distinción conceptual que existe en el ámbito militar entre compartir un recurso y transferir una entidad.

Un recurso es un concepto estático, es decir, algo que no deja de pertenecer al federado durante toda ejecución del IRM (por ese motivo no se transfiere). Sin embargo una entidad es un concepto más dinámico, está dentro de una cadena (por ejemplo la logística) por lo que fluye por ella según se ha establecido en los planes de operaciones (documento donde el mando de una unidad militar plasma su concepto de maniobra).

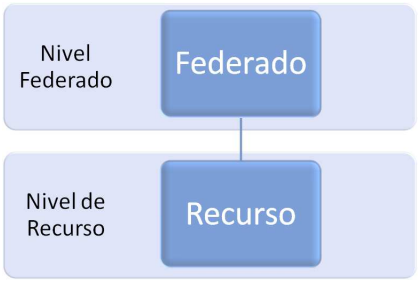

Figura 3.32: RSS definido para el IRM tipo B

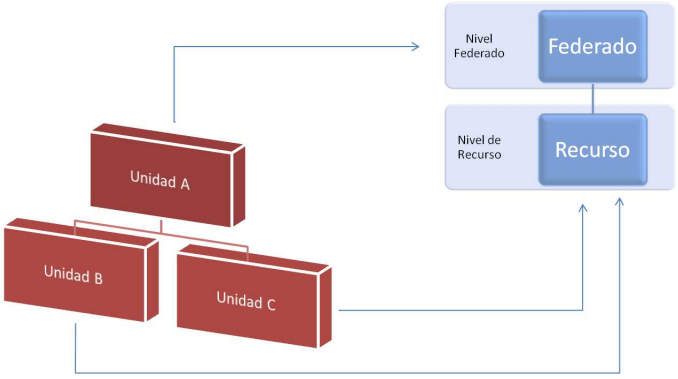

Figura 3.33: Acceso por niveles al recurso compartido entre unidades

3.3.5.1. Definición del recurso compartido (RSS)

Al igual que la definición de entidad, la RSS (Resource Shared Specification) sigue la estructura de datos definida en OMT. Dentro de esta estructura se definen los siguientes niveles lógicos (figura 3.32):

- **Nivel federado.** Este nivel especifica la información referente a los federados que comparten el recurso.
- **Nivel recurso.** Este nivel especifica la información detallada de un recurso concreto.

Los niveles lógicos pueden englobar varios niveles jerárquicos definidos en el OMT (figura 3.33). Así, dependiendo del nivel de información requerido para ese recurso, un federado se podrá suscribir a un nivel u otro.

De esta manera un federado que representa a una unidad A con nivel jerárquico superior a una unidad B se puede suscribir al nivel federado, ya que necesita una

información de mayor nivel. Sin embargo la unidad B que necesita información en más detalle, se suscribe al nivel recurso.

3.3.5.1.1. Ejemplo de RSS. En el ámbito táctico-militar una RSS puede estar basada en un *Object-Item* definido en el JC3IEDM. En este ejemplo se define una RSS que contiene la información relacionada con una construcción militar (*facility* en el JC3IEDM).

En este caso el recurso (basándose en el OMT) está compuesto por dos clases, cada una en un nivel de RSS diferente, y jerárquicamente dependientes entre sí. Estas clases son:

- **Federate Information.** Esta clase tiene los siguientes atributos:
  - El atributo Rol cuyo valor especifica el rol que tiene en el IRM el federado que la publica.
  - El atributo IRM cuyo valor especifica en qué tipo de IRM está participando la RSS.
- **Facility.** En esta clase se añaden todos aquellos atributos que especifican una *facility* (como por ejemplo tipo, estado operativo o localización).

3.3.5.2. Definición del modelo tipo B para simuladores constructivos militares

Este IRM consiste en dos roles (A y B) que comparten un mismo recurso entre los tiempos lógicos T2 y T4. En este periodo de tiempo ambos roles pueden actualizar la RSS de tal manera que el otro reciba las actualizaciones realizadas. Para la definición del modelo se ha estandarizado que el rol A es el que realiza las actualizaciones.

Este IRM se compone de cinco eventos:

3.3.5.2.1. Evento 1. En este evento el rol A publica un recurso (RSS A) en la RTI y el rol B publica un recurso (RSS B) idéntico al publicado por A (figura 3.34).

El recurso publicado por ambos roles es el que va a compartir durante la ejecución del IRM. De esta manera ambos roles podrán actualizar el recurso ya que un federado sólo puede actualizar el objeto del cual es el propietario.

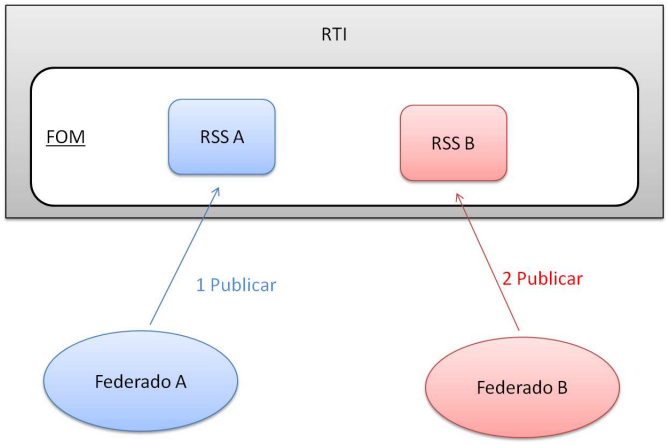

Figura 3.34: Esquema del evento 1 en el IRM tipo B

En el diagrama de tiempos (figura B.17) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1).

3.3.5.2.2. Evento 2. En este evento el rol A se suscribe al recurso RSS B publicado por el rol B en el evento anterior y el rol B se suscribe al recurso RSS A publicado por el rol A en el evento anterior.

El objeto de este IRM es que el recurso compartido sea único. Por lo tanto cualquier actualización realizada sobre el mismo por parte de un rol debe ser conocida por el otro rol (figura 3.35).

De esta manera cuando un rol realice una actualización sobre el recurso que ha publicado el otro, al estar suscrito, recibe las *callback*, siendo conocedor así de esa actualización.

En el diagrama de tiempos (figura B.18) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.5.2.3. Evento 3. A partir de este evento el recurso comienza a ser compartido realmente. Por lo tanto, el rol A empieza a realizar actualizaciones sobre el recurso (RSS A) que publicó en el evento 1 (figura 3.36).

El rol A realiza una actualización sobre la RSS A y el rol B recibe en T3 la *callback* de dicha actualización.

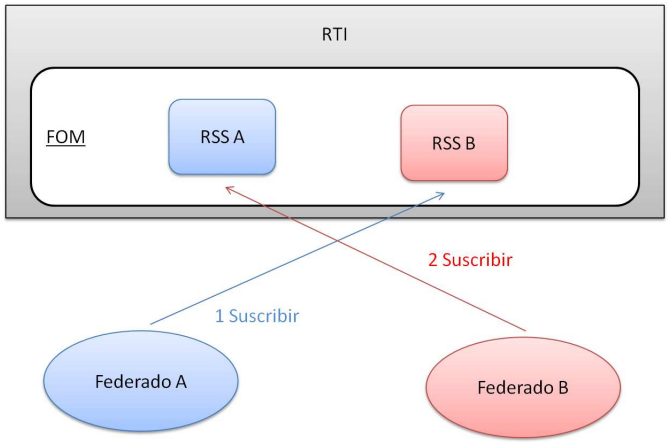

Figura 3.35: Esquema del evento 2 en el IRM tipo B

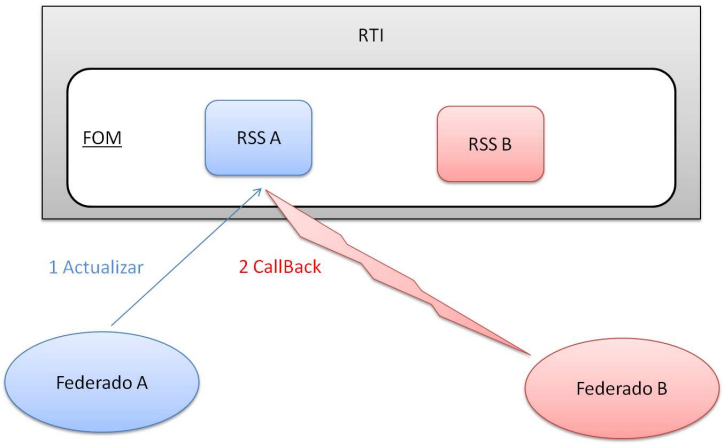

Figura 3.36: Esquema del evento 3 en el IRM tipo B

En el diagrama de tiempos (figura B.19) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.5.2.4. Evento 4. Con el objeto de que los recursos publicados en la RTI, RSS A y RSS B, sean iguales entre si y por lo tanto mantengan la coherencia entre ellos, el rol B debe actualizar la RSS B con los datos recibidos en el evento anterior (figura 3.37).

Por lo tanto el rol B actualiza RSS B con los datos que recibe en la *callback* del evento anterior. En T4 el rol A recibe las *callback* de esta actualización comprobando

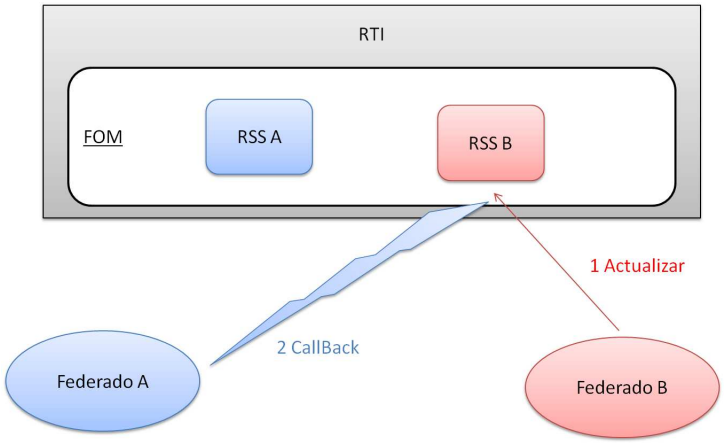

Figura 3.37: Esquema del evento 4 en el IRM tipo B

que coinciden con las actualizaciones que realizó en el evento anterior. Esto garantiza la tolerancia a fallos.

En el diagrama de tiempos (figura B.20) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

3.3.5.2.5. Evento 5. Los eventos 3 y 4 se repiten cada vez que el federado con rol A quiera hacer una actualización del recurso compartido. Para acabar la ejecución de este IRM (es decir, para dejar de compartir el recurso) se ejecuta este evento en el cual el rol A despublica el recurso publicado en el evento 1 (RSS A) y el rol B hace lo propio con RSS B (figura 3.38).

Como se puede observar el rol A es el que toma la iniciativa para finalizar el IRM.

En el diagrama de tiempos (figura B.21) se representan las actividades desarrolladas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

Cuando se deja de compartir un recurso, éste no puede dejar de existir sin más, por lo tanto el federado con rol A que inicia el IRM debe hacerse cargo del mismo y asumirlo como propio (de esta manera no desaparece). Por este motivo hay que definir en el nivel federado del RSS quién es el propietario del recurso.

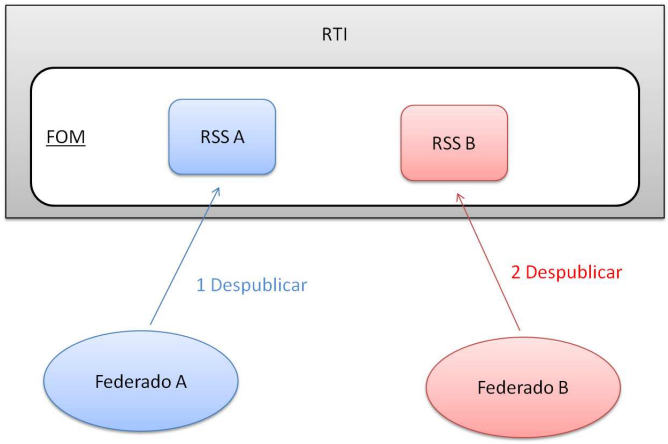

Figura 3.38: Esquema del evento 5 en el IRM tipo B

3.3.5.3. Caso práctico de aplicación del IRM B

A continuación se ejemplifica la aplicación del IRM tipo B en una situación de necesidad de interoperabilidad real entre simuladores constructivos.

El contexto escogido es el siguiente:

"Cuando se planifica una operación militar se definen las unidades que van a participar y las relaciones existentes entre ellas (ORBAT u orden de batalla). Cada una de estas unidades tiene su propio armamento ligero, que sumados, forman el armamento ligero total de la unidad jerárquicamente superior.

Por lo tanto, los el armamento ligero de una unidad subordinada pertenecen también a la unidad superior en el ORBAT, debiendo ser este un recurso compartido por ambas unidades"

Con lo que el IRM B es de aplicación directa a esta situación. Los elementos del IRM B que se identifican en este contexto son:

- **Federado A.** Es el sistema que simula las actividades de la unidad jerárquicamente superior definida en el *ORBAT* para esa operación.
- **Federado B.** Es el sistema que simula las actividades de la unidad jerárquicamente subordinada a la unidad simulada por el federado A definida en el *ORBAT*.
- **Recurso (RSS).** Es el armamento ligero asignado a la unidad definida en el federado B.

En este caso se observa que compartir un recurso es una situación habitual en operaciones militares terrestres ya que los recursos de una gran unidad son la suma de los recursos de las unidades que la forman.

3.3.6. Definición del IRM tipo C para simuladores constructivos militares

Este IRM consiste en dos roles (A y B) que comparten un mismo evento. En el entorno táctico-militar un evento se puede definir como una interacción entre dos o varios *Object-Item* (concepto de *Action-Event* del JC3IEDM explicado en la sección 3.3.1).

Por lo tanto este IRM consiste en dos roles que comparten la información relacionada con una *Action* porque es de interés para ambos. La diferencia con los IRMs definidos hasta el momento es que en el caso de una *Action-Event* no se puede saber de antemano el tiempo lógico de simulación en el que va a ocurrir, pues depende de factores que no son programables por los federados que intervienen en el IRM.

3.3.6.1. Empleo de los servicios Time Management para gestionar la incertidumbre del suceso del IRM tipo C

Los federados que participan en este IRM lo pueden hacer por dos motivos: por estar afectados directamente por la *Action* o porque la *Action* está dentro de su ámbito de información (por ejemplo, una unidad de inteligencia). Lo que es común a ambos motivos es su incertidumbre acerca del tiempo en el que sucede la *Action*.

Mediante el empleo del módulo *Time Management* definido en el estándar IEEE1516 ([15]) se puede minimizar el impacto de esta incertidumbre en la ejecución del IRM (es decir, el tiempo lógico en el que comienza).

Lo habitual es que este tiempo se conozca en el momento justo en el que comienza el IRM, momento en el que todos los federados participantes en el IRM deberán avanzar su tiempo lógico al tiempo en el que la *Action* ha sucedido.

Este punto inicial debe estar etiquetado con un nombre, de tal manera que el avance no sea a un tiempo concreto sino a una etiqueta (puesto que el tiempo es conocido en el momento de producirse la *Action* y no con antelación). Para ello las acciones deben estar definidas como puntos de sincronización en la tabla *Synchronization Table* definida en el FOM (figura 3.39) de la federación para tal cometido.

Label	Tag Type	Capability	Semantics

Figura 3.39: Tabla Synchronization definida para el FOM de una federación

3.3.6.2. Definición del evento compartido (ESS)

Para la definición de la ESS (Event Shared Specification) se sigue el modelado OMT definido para la arquitectura HLA. La diferencia con los IRMs anteriores es que no se define en base a las tablas de objetos y atributos, sino a la de interacciones y parámetros.

Es decir, un evento se define como una interacción en el OMT. Esto es debido a que una interacción define la relación entre dos objetos justo como pasa con el concepto *Action*, que no es más que una actividad producida por dos *Object-Item*.

Un evento se define en dos niveles:

- **Nivel federado.** Este nivel especifica la información referente a los federados que comparten el evento.
- **Nivel estructura de datos.** Este nivel especifica la información detallada del evento.

3.3.6.2.1. Ejemplo de ESS. Un ejemplo en el ámbito militar puede ser un IED (Improvised Explosive Device):

- **Federate Information.** Esta interacción tiene los siguientes atributos:
  - El atributo Rol cuyo valor especifica el rol que tiene en el IRM el federado que la publica.

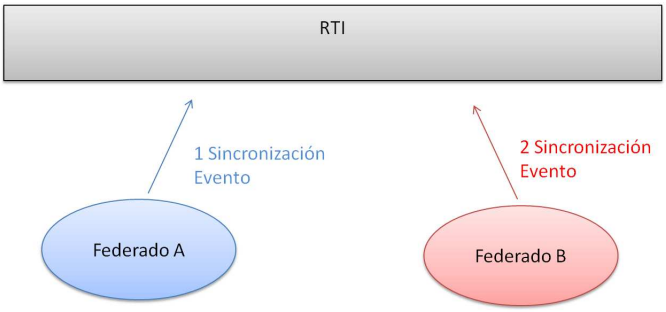

Figura 3.40: Esquema del evento 1 en el IRM tipo C

- El atributo IRM cuyo valor especifica en qué tipo de IRM está participando el ESS.

- **IED.** En esta interacción se añaden todos aquellos atributos que especifican las características de un IED concreto (como por ejemplo tipo, potencia, localización, composición, estado).

3.3.6.3. Definición del modelo tipo C para simuladores constructivos militares

Este IRM consiste en dos roles (A y B) que comparten la información de un evento en el momento en que éste sucede (instante E). Se ha estandarizado para este modelo que el rol A es el generador de información del evento y el rol B es el receptor de la información.

El modelo se ejecuta en cinco eventos:

3.3.6.3.1. Evento 1. En este evento ambos roles avanzan su tiempo lógico al tiempo que se ha etiquetado como comienzo del IRM (tiempo que coincide con el suceso del evento).

Al final de este evento ambos roles tienen un mismo tiempo de suceso del evento (E) y están situados en el mismo punto de la *Federation Time Axis* (figura 3.40).

En el diagrama de tiempos (figura B.22) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y E).

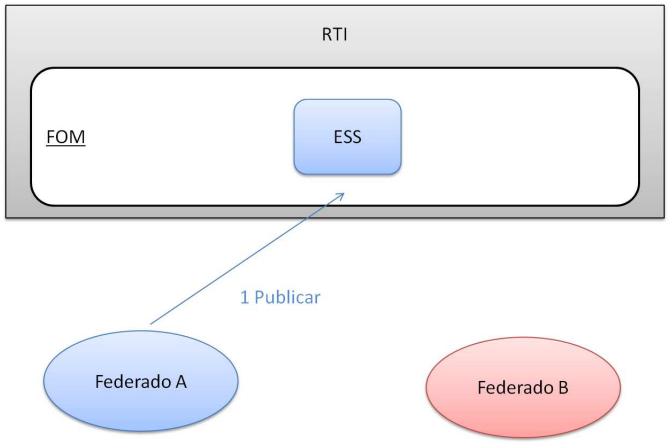

Figura 3.41: Esquema del evento 2 en el IRM tipo C

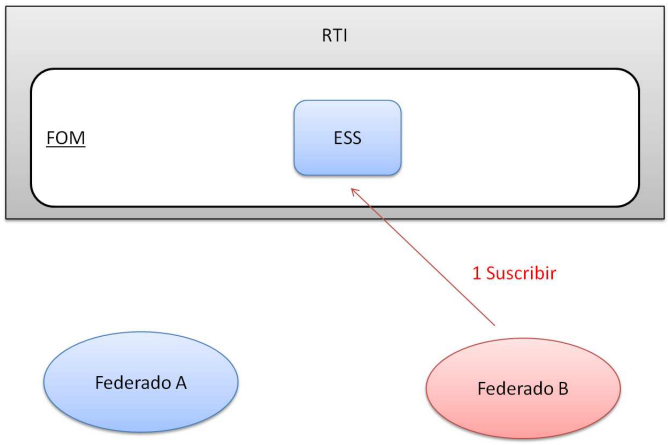

Figura 3.42: Esquema del evento 3 en el IRM tipo C

3.3.6.3.2. Evento 2. En este evento el rol A publica el evento compartido (ESS) con todos los parámetros que lo definen. A partir de este momento el rol B se puede suscribir a la ESS (figura 3.41).

En el diagrama de tiempos (figura B.23) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos E y T1).

3.3.6.3.3. Evento 3. En este evento el rol B se suscribe al evento publicado por el rol A. De esta manera puede recibir las *callback* de las actualizaciones de los parámetros del evento (figura 3.42).

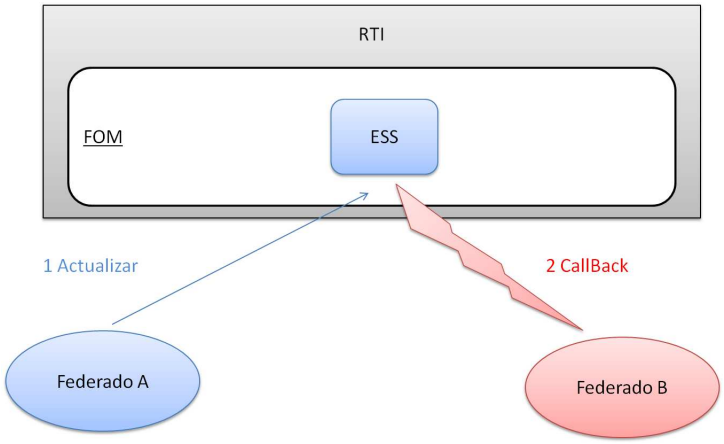

Figura 3.43: Esquema del evento 4 en el IRM tipo C

En el diagrama de tiempos (figura B.24) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.6.3.4. Evento 4. El rol A actualiza los parámetros del evento publicado en el evento 2 y el rol B recibe las *callback* (en un tiempo lógico T3) al estar suscrito a él .

De esta manera el rol B recibe la información del evento que necesita o es de su área de interés (figura 3.43).

En el diagrama de tiempos (figura B.25) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.6.3.5. Evento 5. El rol A despublica el evento publicado en el evento 2. Un vez que se acaba la *Action* y se ha informado al rol B, el IRM acaba mediante la ejecución de este evento (figura 3.44).

En el diagrama de tiempos (figura B.26) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

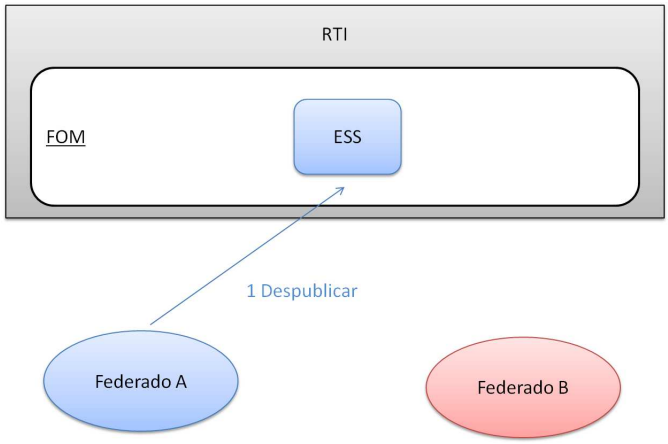

Figura 3.44: Esquema del evento 5 en el IRM tipo C

3.3.6.4. Caso práctico de aplicación del IRM C

El objeto de esta sección es ilustrar la aplicación de este tipo de IRM en una situación táctico-militar. Para ello se describirá un contexto en el que su aplicación resuelva la interoperabilidad entre dos simuladores constructivos militares:

"En una misión de mantenimiento de la paz una de las tareas más frecuentes de las fuerzas armadas es patrullar las zonas de responsabilidad asignadas a una unidad con el objetivo de que no haya disturbios. En alguna ocasión (sobre todo en zonas hostiles) una patrulla puede verse afectada por un IED (Improvised Explosive Device, explosivos de fabricación casera y no catalogados).

*Por lo tanto en este caso una patrulla (*Object-Item*) es afectada por un evento (*Action-Event*) y debe informar a su unidad superior. La rapidez en la difusión de esta información, además de salvar vidas, puede ser fundamental para tener éxito en las operaciones realizadas en esa zona."*

Por lo tanto si aplicamos a este contexto un IRM tipo C podemos identificar los siguientes elementos:

- **Federado A.** Es el sistema que simula las actividades de la patrulla que sufre un ataque mediante IED.
- **Federado B.** Es el sistema que simula las actividades de la unidad superior a la que pertenece la patrulla. También pueden incluirse aquí las unidades de inteligencia receptoras de este tipo de información.
- **ESS.** Es el informe que manda la patrulla sobre el IED (efectos, posición...).

Name	Size In Bits	Interpretation	Endian	Encoding

Figura 3.45: Tabla Data Types definida para el FOM de una federación

En este ejemplo se puede observa que la información de los eventos es fundamental para la supervivencia de las unidades militares y que el IRM propuesto permite su compartición entre dos o más simuladores constructivos.

3.3.7. Definición del IRM tipo D para simuladores constructivos militares

El IRM tipo D consiste en dos roles (A y B) que comparten una misma estructura de datos (en un tiempo de simulación T2-T4). En el entorno táctico-militar una estructura de datos puede seguir una estructura no normal (es decir, que no se especifica en el modelo JC3IEDM) de un *Object-Item*. Esto posibilita una mayor flexibilidad a la hora de compartir información entre federados al permitir una mayor variedad de estructuras para los *Object-Item*.

Por lo tanto este IRM permite compartir, desde una colección de *Object-Item* hasta una parte de ellos, pudiendo ser una estructura de datos formada por una simple variable o por una colección de estructuras más complejas.

La única limitación que se impone a las estructuras de datos es la tabla *Data Types* definida en el FOM de la federación ya que ésta define los tipos de datos que se pueden emplear para las estructuras de datos de la federación de simuladores (figura 3.45).

3.3.7.1. Definición de la estructura de datos compartida (DSS)

Aunque una DSS (Data structure Shared Specification) no sigue la estructura especificada en el JC3IEDM, debe estar conforme a la OMT (estructura que siguen

los datos en HLA) que impone las siguientes restricciones:

- Los tipos de datos que forman la estructura de datos deben estar definidos en la tabla *Data Types* del FOM de la federación.
- La estructura de datos debe ser una selección de los atributos de un objeto que se define en la tabla *Object* del FOM de la federación.
- Una estructura de datos debe estar basada en los *Object-Item* definidos en el JC3IEDM. Esto quiere decir que pueden ser una parte o una colección de *Object-Item*.

La DSS está formada por dos niveles de información (que pueden englobar varias clases jerárquicamente dependientes entre ellas) que son:

- **Nivel federado.** Este nivel especifica la información referente a los federados que comparten la estructura de datos.
- **Nivel estructura de datos.** Este nivel especifica la información detallada de la estructura de datos.

3.3.7.1.1. Ejemplo de DSS. Un caso típico de estructura de datos sería el informe que incluye el nombre de las unidades, y el nombre y empleo del personal que pertenece a ellas. En este ejemplo la estructura de datos tendría dos clases:

- **Federate Information.** Esta clase tiene los siguientes atributos:
  - El atributo Rol cuyo valor especifica el rol que tiene en el IRM el federado que la publica.
  - El atributo IRM cuyo valor especifica en qué tipo de IRM está participando la DSS.
- **Encuadramiento de personal.** En esta clase se añaden los atributos de nombre de unidad, y empleo y apellidos del personal que pertenece a ella.

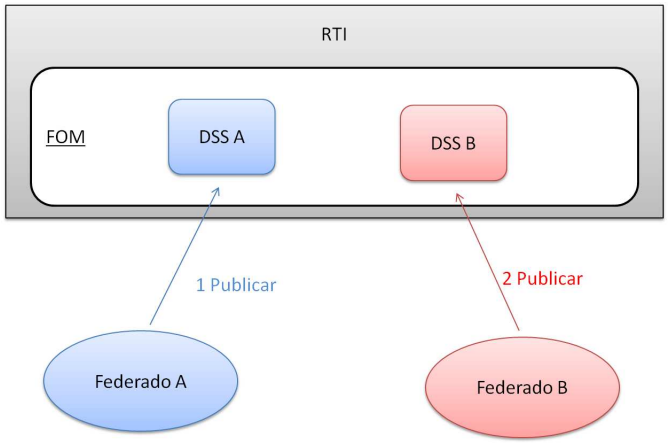

Figura 3.46: Esquema del evento 1 en el IRM tipo D

3.3.7.2. Definición del modelo tipo D para simuladores constructivos militares

Este IRM consiste en dos roles (A y B) que comparten una misma estructura de datos entre los tiempos lógicos T2 y T4. En este periodo de tiempo ambos roles pueden actualizar la DSS de tal manera que el otro recibe las actualizaciones realizadas. Para la definición del modelo se ha estandarizado que el rol A es el que realiza las actualizaciones.

Este IRM se define a través de cinco eventos:

3.3.7.2.1. Evento 1. En este evento el rol A publica en la RTI una estructura de datos (DSS A) y el rol B publica otra estructura de datos idéntica a la anterior (DSS B, figura 3.46).

De esta manera el rol A puede actualizar la estructura de datos compartida si así lo requiere. La necesidad de que cada rol publique un DSS es que cada federado puede actualizar el objeto del cual es propietario (un federado es propietario de un objeto que publica).

En el diagrama de tiempos (figura B.27) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1).

3.3.7.2.2. Evento 2. En este evento el rol A se suscribe a la estructura de datos publicada por el rol B en el evento 1 (DSS B) y el rol B se suscribe a la estructura

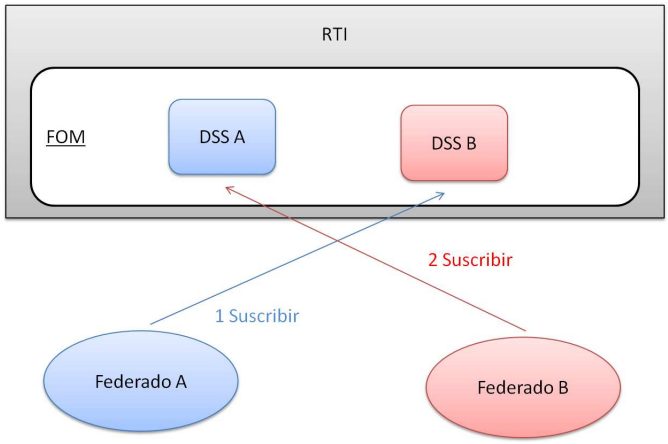

Figura 3.47: Esquema del evento 2 en el IRM tipo D

de datos publicada por el rol A en el evento 1 (DSS A, figura 3.47).

De esta manera los roles recibirán las actualizaciones que realiza el otro rol sobre la estructura de datos compartida (siendo conocedores de los datos actualizados).

En el diagrama de tiempos (figura B.28) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.7.2.3. Evento 3. Este evento comienza cuando el rol A realiza una actualización de la estructura de datos compartida (figura 3.48).

El objetivo de este evento es que el rol A actualice la estructura de datos compartida (DSS A) y que el rol B reciba las *callback* (y por lo tanto, conozca las actualizaciones realizadas) en el tiempo de simulación T3.

En el diagrama de tiempos (figura B.29) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.7.2.4. Evento 4. En este evento el rol B actualiza su estructura de datos publicada en el evento 1 (DSS B) con los datos recibidos en la *callback* en el tiempo de simulación T3. Por lo tanto el rol A suscrito a DSS B recibe la *callback* de esta actualización en un tiempo de simulación T3 (figura 3.49).

De esta manera las dos estructuras de datos publicadas en la RTI (DSS A y DSS B) mantienen una coherencia entre ellas. Este es el mecanismo de tolerancia a fallos,

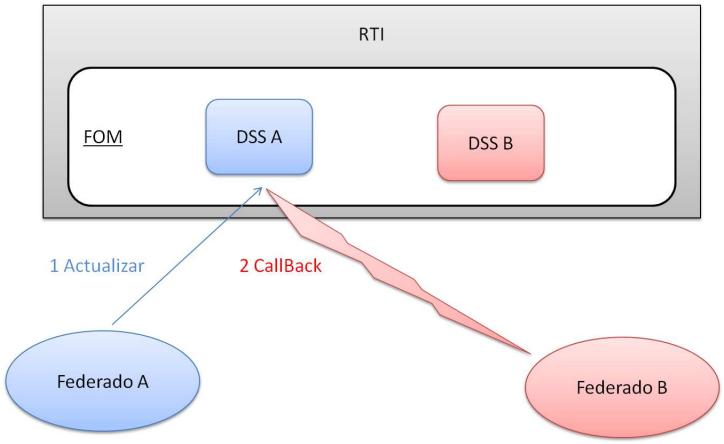

Figura 3.48: Esquema del evento 3 en el IRM tipo D

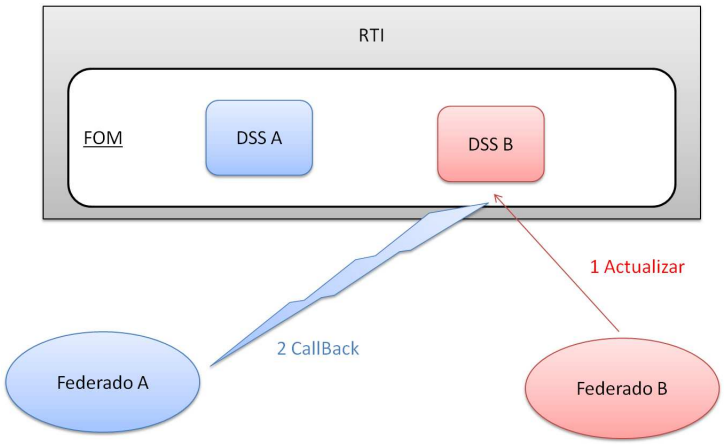

Figura 3.49: Esquema del evento 4 en el IRM tipo D

ya que el rol A, al recibir las *callback* al final del evento, comprueba si coinciden con las actualizaciones que realizó en el evento anterior.

En el diagrama de tiempos (figura B.30) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

3.3.7.2.5. Evento 5. Los eventos 3 y 4 se repiten cada vez que un rol tenga que actualizar la estructura de datos compartida. Este evento se ejecuta cuando se quiere finalizar la ejecución del IRM (figura 3.50).

Para ello el rol A y el rol B despublican las estructuras de datos que publicaron

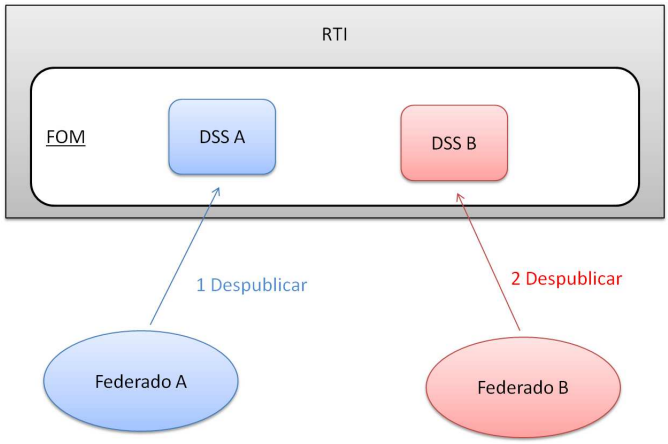

Figura 3.50: Esquema del evento 5 en el IRM tipo D

en el evento 1.

En el diagrama de tiempos (figura B.31) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

Cuando se deja de compartir una estructura de datos, ésta no puede dejar de existir sin más, por lo tanto el federado con rol A que inicia el IRM debe hacerse cargo de la misma y asumirla como propia (de esta manera no desaparece). Por este motivo hay que definir en el nivel federado del DSS quién es el propietario de la estructura de datos.

3.3.7.3. Caso práctico de aplicación del IRM D

Es esta sección se va aplicar el IRM tipo D a una situación táctico-militar donde es necesaria la interoperabilidad entre simuladores constructivos. Para ello el contexto seleccionado es el siguiente:

"Durante el planeamiento de una operación militar se elabora una lista de objetivos que están previstos de antemano para las unidades de artillería. La unidad que planea esto es el FSE (Fire Support Element) de una brigada, que comparte la lista elaborada con el grupo de artillería, unidad encargada de dirigir y realizar el fuego sobre esos objetivos planificados.

Por varios motivos (relacionados con la flexibilidad en el combate) la lista de objetivos puede ser modificada o ampliada por parte del FSE o del grupo de artillería siendo necesario un flujo bidireccional de las modificaciones a la misma."

Basándose en el modelo JC3IEDM un objetivo suele ser un *Object-Item* (normalmente catalogado como enemigo), pero en este caso práctico existen varias diferencias.

Para la lista de objetivos mencionada no es necesaria toda la información que se especifica en el JC3IEDM, solamente se necesitan las coordenadas de los objetivos y su nivel de resistencia. Por este motivo esta lista de objetivos emplea de manera parcial la estructura de datos *Object-Item*.

Teniendo en cuenta lo anterior los elementos especificados en el IRM tipo D e identificados en este contexto son:

- **Federado A.** Es el sistema que simula las actividades del FSE de la brigada que elabora la lista de objetivos previstos.
- **Federado B.** Es el sistema que simula las actividades del puesto de mando del grupo de artillería que apoya a la brigada mediante el fuego.
- **DSS.** Es la estructura de datos con la lista de objetivos previstos. En este caso contiene para cada objetivo las coordenadas (X,Y,Z) y el nivel de resistencia.

Se comprueba que este IRM permite modelar de manera flexible situaciones que se pueden dar dentro del ámbito táctico-militar y que no requieren tanta rigidez en la estructura de los datos como en los modelos anteriores.

3.3.8. Definición del IRM tipo E para simuladores constructivos militares

Como se expuso en el análisis del JC3IEDM realizado al principio de este capítulo, los IRMs definidos por SISO no dan cobertura a todos los conceptos definidos en el mismo.

El concepto de plan y orden definido en la especificación JC3IEDM debe ser modelado definiendo un nuevo IRM para tal fin. Por este motivo en esta sección se va a definir un nuevo modelo de interoperabilidad llamado tipo E que refleje las situaciones táctico-militares donde se gestionen planes/órdenes.

3.3.8.1. Marco conceptual de plan/orden

Desde el punto de vista del JC3IEDM ([21]) una operación militar se puede definir como un conjunto de *Object-Item* y *Actions* unidos entre si bajo una misma

nomenclatura (plan/orden).

De esta manera se rompe con el concepto estático de *Object-Item* y *Action* para evolucionar a un concepto mas dinámico de los mismos donde estos elementos interactúan entre si para conseguir un mismo objetivo dando lugar a una operación militar ([47]).

Dentro de un plan/orden se definen los siguientes conceptos ([26]):

- **ORBAT.** Se define como el orden de batalla establecido. En él se especifican las unidades que intervienen y la relaciones existentes entre ellas (por ejemplo, subordinación o agregación).
- **Task Force.** Se define como un conjunto de unidades que van a realizar una acción concreta dentro de una operación militar.
- **Holding Force.** Se define como el conjunto de recursos materiales asignados a una unidad militar para la realización de una operación.

Un plan/orden se desarrolla para diseñar el flujo de acciones que va a realizar una unidad militar dentro de una operación. Por lo tanto el concepto operación lleva agregado uno o varios planes/órdenes que lo definen.

Este concepto no está contemplado en la versión actual del JC3IEDM por lo tanto se puede afirmar que este estándar es monoperación (concepto que hay que tener en cuenta para el desarrollo de este IRM).

3.3.8.2. Definición del plan/orden (P/O)

Un plan/orden define las líneas de acción y el concepto que tiene el mando de una unidad militar para realizar una operación.

Un plan se compone de un cuerpo y varios anexos donde se detallan aquellas partes del plan específicas de una o varias unidades subordinadas a la que planea la operación (por ejemplo, temas de artillería e ingenieros).

Por lo tanto la composición de un plan/orden además, de la estructura textual definida en el STANAG 2014 ([109]), incluye otros conceptos como *task forces*, *holding forces* y *ORBAT* definidos para el mismo.

En este caso la especificación de P/O se puede dividir en dos niveles diferentes:

- **Nivel Federado.** En este nivel se especifica la información de los federados que publican un P/O concreto.

- **Nivel P/O.** En este nivel se especifica la información que compone el cuerpo y los anexos del P/O.

3.3.8.2.1. Ejemplo de P/O. Un caso típico de P/O con dos clases cada una perteneciente a un nivel sería:

- **Federate Information.** Esta clase tiene los siguientes atributos:
  - El atributo Rol cuyo valor especifica el rol que tiene en el IRM el federado que publica el P/O.
  - El atributo IRM cuyo valor especifica en que tipo de IRM está participando la P/O.
- **Plan.** En esta clase se añaden todos aquellos atributos que especifican un plan (como por ejemplo cabecera, cuerpo, anexos).

3.3.8.3. Definición del modelo tipo E para simuladores constructivos militares

Este IRM se compone de dos roles (A y B siendo el rol A de rango jerárquico superior al rol B) con el objetivo de modelar el ciclo de planeamiento definido para las unidades militares.

El rol A elabora un plan (P/O A) que comparte con el rol B que elabora otro plan (P/O B) basado en el plan compartido por el rol A y lo envía a este rol para la modificación y actualización del P/O A original.

En este IRM el concepto de compartición difiere de los IRMs anteriores. Es decir, el rol A comparte un plan que sólo puede modificar él mismo (al contrario de lo que sucede con el recurso y la estructura de datos en los IRMs B y D que pueden ser modificados por los dos roles). Del mismo modo el rol B sólo puede modificar el P/O A enviando al rol A planes (P/O B) que lo actualicen y modifiquen (figura 3.51).

Para la definición de este IRM se divide en dos fases: la primera donde el rol A publica y da a conocer su plan y una segunda fase donde el rol B realiza su plan basado en el publicado por el rol A y lo envía a este con la finalidad de modificar el plan original. Por lo tanto este IRM es un modelo mixto entre la compartición y la transferencia.

Este IRM se compone de nueve eventos (divididos en dos fases), que son los siguientes:

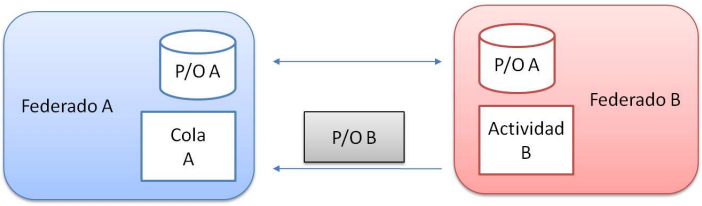

Figura 3.51: Definición de IRM tipo E

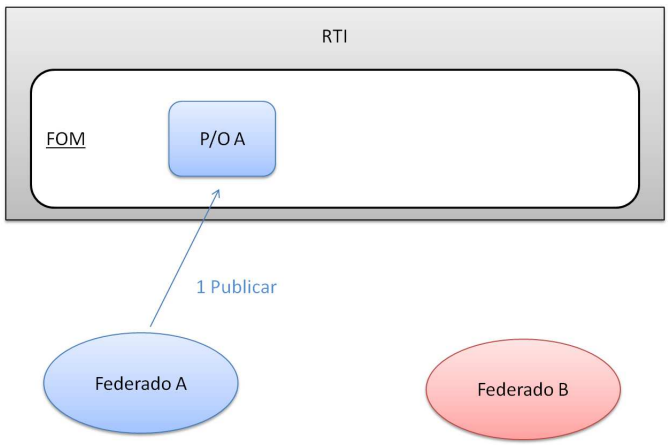

Figura 3.52: Esquema de la fase 1: evento 1 en el IRM tipo E

3.3.8.3.1. Fase 1: evento 1. El objeto de la fase 1 es el envío del plan/orden elaborado por el rol A al rol B para que pueda comenzar a elaborar sus planes/órdenes específicos que complementan al original (figura 3.52). Es decir, el inicio de la compartición.

Este evento comienza una vez que el rol A termina la elaboración de su plan/orden y necesita enviarlo al rol B. Para ello el rol A publica el plan/orden (P/O A) en la RTI para que de esta manera pueda actualizarlo y los federados suscritos al P/O A reciban esas actualizaciones.

En el diagrama de tiempos (figura B.32) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T0 y T1).

3.3.8.3.2. Fase 1: evento 2. En este evento el rol B se suscribe al P/O A publicado por el rol A en el evento anterior. De esta manera puede recibir las futuras actualizaciones que el rol A realice sobre él (figura 3.53).

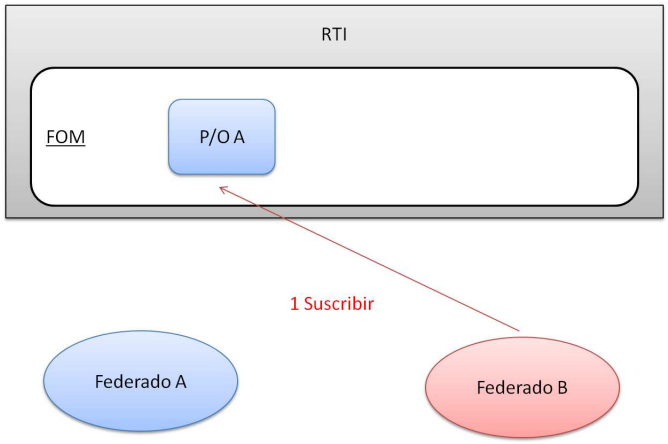

Figura 3.53: Esquema de la fase1: evento 2 en el IRM tipo E

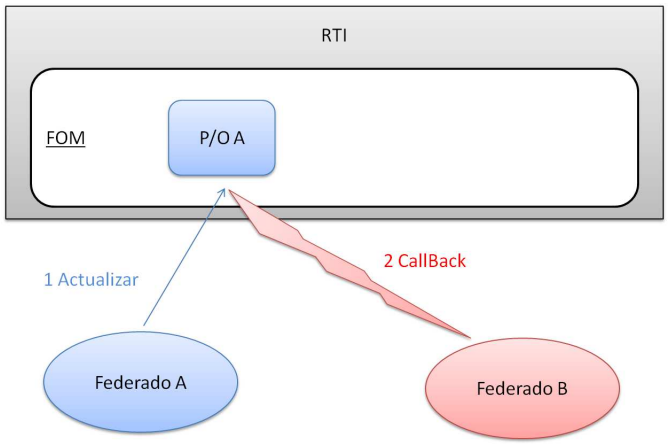

Figura 3.54: Esquema de la fase1: evento 3 en el IRM tipo E

En el diagrama de tiempos (figura B.33) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T1 y T2).

3.3.8.3.3. Fase 1: evento 3. En este evento el rol A actualiza el P/O A publicado en el evento 1 con los datos del plan/orden elaborado. El rol B suscrito a él recibe en T3 las *callback* de las actualizaciones (figura 3.54).

Mediante este evento se envía completamente el plan/orden elaborado por el rol A al rol B pudiendo este último rol comenzar a elaborar sus planes/órdenes específicas dando así comienzo la fase 2 del IRM.

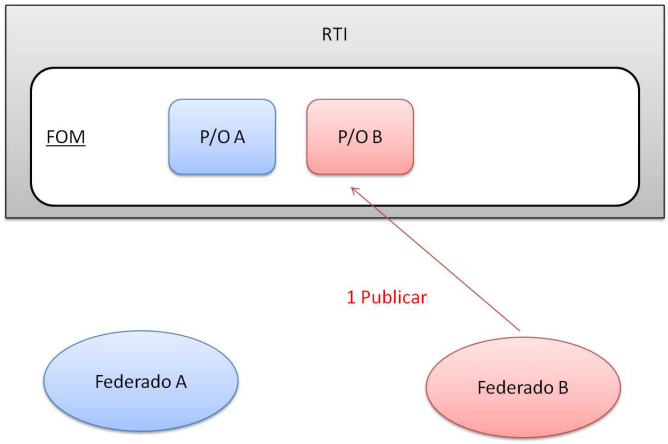

Figura 3.55: Esquema de la fase2: evento 4 en el IRM tipo E

En el diagrama de tiempos (figura B.34) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T2 y T3).

3.3.8.3.4. Fase 2: evento 4. En este evento comienza la ejecución de la fase 2 del IRM. El objeto de esta fase es el envío por parte del rol B de los planes/órdenes que ha realizado basadas en el plan/orden enviado en la fase anterior al rol A (figura 3.55). Es decir, la realización de la transferencia.

Para ello en este evento el rol B publica un objeto tipo plan/orden (P/O B) con la finalidad de que todos los federados que se suscriban a él puedan recibir las actualizaciones que sufra.

En el diagrama de tiempos (figura B.35) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T3 y T4).

3.3.8.3.5. Fase 2: evento 5. En este evento el rol A se suscribe al P/O B publicado por el rol B en el evento anterior. De esta manera puede recibir las *callback* de las actualizaciones realizadas sobre P/O B (figura 3.56).

En el diagrama de tiempos (figura B.36) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T4 y T5).

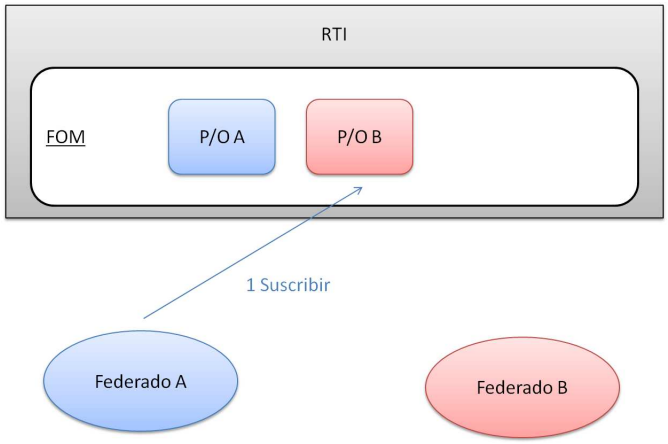

Figura 3.56: Esquema de la fase2: evento 5 en el IRM tipo E

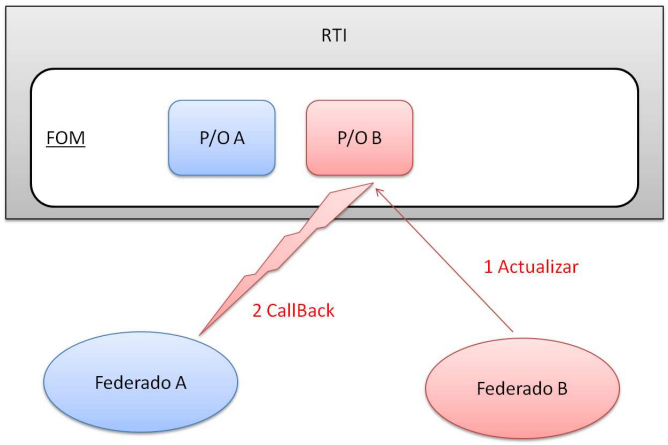

Figura 3.57: Esquema de la fase2: evento 6 en el IRM tipo E

3.3.8.3.6. Fase 2: evento 6. En este evento el rol B actualiza el P/O B publicado con los datos del plan/orden que ha desarrollado basado en el plan/orden recibido por el rol A. En T6 el rol A recibe las *callback* de las actualizaciones y por tanto el plan/orden elaborado por el rol B (figura 3.57).

En el diagrama de tiempos (figura B.37) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T5 y T6).

3.3.8.3.7. Fase 2: evento 7. Como se dijo anteriormente un plan/orden es modificado por otro. Por este motivo al recibir el rol A el plan/orden del rol B en

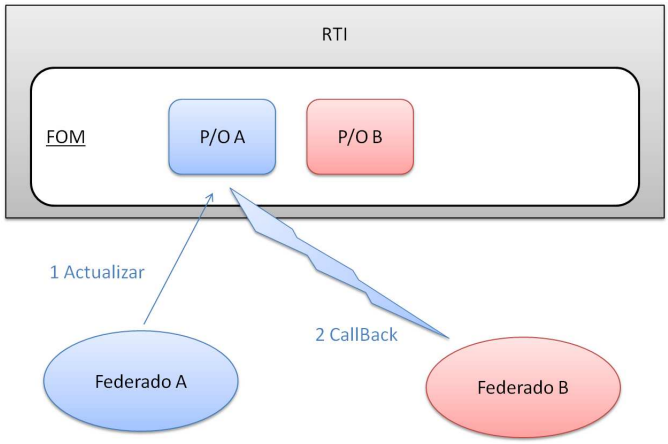

Figura 3.58: Esquema de la fase 2: evento 7 en el IRM tipo E

T6 (P/O B), debe modificarlo (figura 3.58).

En este evento el rol A actualiza el P/O A con los datos recibidos de las *callback* de la actualización del P/O B. El rol B por lo tanto al estar suscrito a P/O A recibe (en T7 tiempo fin del evento) las *callback* de las actualizaciones del P/O A. De esta manera el rol B certifica que el plan/orden que ha enviado al rol A ha sido aprobado por el jefe de la unidad militar (ya que el rol A representa a la unidad que genera el plan/orden inicial y por lo tanto, jefe de la unidad militar).

En el diagrama de tiempos (figura B.38) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T6 y T7).

3.3.8.3.8. Fase 2: evento 8. Una vez que el rol B ha transferido el plan/orden creado, finaliza la ejecución de la fase 2 del IRM.

Por lo tanto, en este evento el rol B despublica el objeto P/O B publicado al principio de la fase. Y da por concluida la transferencia del plan/orden creado (figura 3.59).

En el diagrama de tiempos (figura B.39) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T7 y T8).

La fase 2 se repite tantas veces como planes/órdenes nuevos basados en P/O A tenga que realizar el rol B.

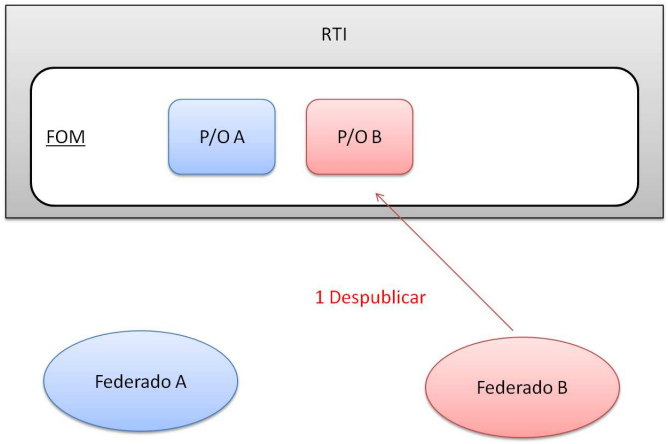

Figura 3.59: Esquema de la fase2: evento 8 en el IRM tipo E

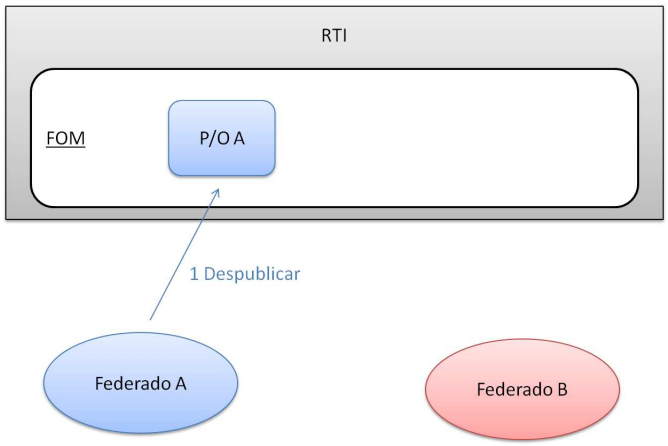

Figura 3.60: Esquema del evento 9 en el IRM tipo E

3.3.8.3.9. Evento 9. Este evento se ejecuta para finalizar la ejecución del IRM. Para ello el rol A despublica el objeto P/O A publicado en la fase 1 de este IRM. De esta manera el jefe de la unidad militar marca la pérdida de vigencia del plan/orden (figura 3.60).

En el diagrama de tiempos (figura B.40) se representan las actividades realizadas por ambos roles durante el desarrollo de este evento (que tiene lugar entre los tiempos T8 y T9).

3.3.8.4. Caso práctico de aplicación del IRM E

Es esta sección se aplica el IRM tipo E a una situación táctico-militar donde es necesaria la interoperabilidad entre simuladores constructivos. Para ello el contexto seleccionado es el siguiente:

"Cuando una brigada de infantería mecanizada planea una operación militar elabora un plan genérico donde el mando expresa cual es su concepto de la operación. Este plan es especificado por las unidades subordinadas de la brigada mediante anexos quedando de esta manera elaborado el plan.

En este contexto específico el grupo de artillería que pertenece a la brigada recibe un plan de la misma para realizar una operación militar. Basándose en este plan el grupo de artillería elabora el anexo de apoyo de fuegos (misión principal del grupo de artillería de la brigada) enviándolo a la brigada para su aprobación y posterior actualización del plan original."

En este contexto se identifican los siguientes elementos del IRM tipo E:

- **Federado A.** Es el sistema que simula las actividades del cuartel general de la brigada de infantería, que es el órgano del apoyo al mando, estando entre sus misiones el planeamiento.
- **Federado B.** Es el sistema que simula las actividades del grupo de artillería, que es la unidad perteneciente a la brigada que elabora el anexo de apoyo de fuegos que actualizará al plan original.
- **P/O A.** Es el plan original que elabora el cuartel general de la brigada y que en un principio no incluye los anexos específicos de las unidades de la brigada.
- **P/O B.** Es el anexo de apoyo de fuegos que elabora el grupo de artillería.

En este caso se demuestra que siguiendo el IRM tipo E es posible modelar una situación concreta dentro del ámbito táctico-militar donde es necesario el de planeamiento de una operación.

Capítulo 4

Modificación de una RTI real para la ejecución automática de los IRMs militares

En el capítulo anterior se han definido cinco tipos diferentes de modelos de interoperabilidad de referencia para garantizar la interoperabilidad real entre simuladores constructivos de aplicación militar en entornos de simulación distribuida. Esta definición se ha realizado mediante la propuesta de las estructuras de datos que se deben transferir y/o compartir en cada IRM, coherentes en cada caso con el modelo militar JC3IEDM, y estableciendo una serie de eventos para cada modelo que resumen la secuencia de actividades que cada federado debe realizar.

Pero esta definición, eminentemente teórica, no está completa sin demostrar que es posible implementar estos modelos con las RTIs existentes en la actualidad para el estándar IEEE1516. En este capítulo se verifica la definición de los modelos ejecutándolos sobre una RTI real y se valida su utilidad en entornos reales.

Para ello el primer paso es seleccionar una de las RTIs disponibles en la actualidad. A continuación, se demostrará que es posible modificar esta RTI, analizando la función que debe realizar cada uno de sus componentes para asegurar la correcta ejecución de cada uno de los eventos de cada IRM definido en el capítulo anterior.

De esta forma no sólo se demuestra la validez de los modelos propuestos sino que también se ilustra la manera en la que deben utilizarse en un entorno de simulación distribuida real.

4.1. Selección de una RTI para estudiar la aplicación práctica de los IRMs

Para seleccionar la RTI que mejor se adapta a los objetivos pretendidos en este capítulo es necesario establecer unos criterios de evaluación. La RTI seleccionada debe cumplir las siguientes características para que sea de utilidad:

- **Código abierto.** La aplicación práctica de los IRMs requiere realizar cambios en los componentes de la RTI seleccionada.

  Por este motivo una RTI de código abierto parece la mejor opción, ya que permite modificar el código fuente e introducir los cambios que sean necesarios.

  Además, las conclusiones del análisis realizado en una RTI de código abierto serán mucho más completas si se puede observar sin limitaciones el comportamiento de todos sus componentes (evitando cajas negras).

- **RTI genérica.** La definición de los IRMs está basada en el estándar IEEE1516. Por lo tanto para su aplicación la RTI debe permitir el empleo de las interfaces del estándar tal cual están definidas en el mismo.

  Esto descarta aquellas RTI que para facilitar su empleo unen varias interfaces en una única función (ya que la función no es estándar y puede no ajustarse a la definición de los IRMs realizada).

- **Compatibilidad con las interfaces Time Management.** La interfaz *Time Management* es la base de la gestión de los eventos en la definición de los IRMs de aplicación militar.

  Por este motivo la RTI seleccionada debe tener implementadas todas las interfaces *Time Management* definidas en el estándar IEEE1516.

- **Modularidad.** La división de la RTI en diferentes módulos permite el estudio de los eventos de un IRM desde diferentes puntos de vista.

  De esta manera se obtiene una visión lo más amplia posible (en varias partes de la RTI) de la ejecución de los eventos.

- **Facilidad de modificación.** Las modificaciones que deben realizarse en la RTI para la ejecución de los IRMs pueden ser muy laboriosas y costosas si el código fuente de la RTI no es fácilmente legible y modificable, por lo que este criterio es muy importante desde el punto de vista del tiempo de desarrollo.

Teniendo en cuenta los criterios de evaluación anteriormente expuestos y estudiando las RTIs disponibles en este momento, Pórtico parece la candidata ideal para probar la implementación de los modelos militares frente a otras como PITCH pRTI, HLA direct o RTI NG pro ([110], [111], [112], [113]).

Pórtico es una RTI open source ya que se distribuye bajo los términos de la licencia CDDL (Common Developer and Distribution Licence, [114]). Además, permite el empleo de las interfaces tal cual están definidas en el estándar IEEE1516 y tiene implementadas las interfaces de *Time Management*.

La identificación de los componentes que forman la RTI Pórtico es casi inmediata porque la definición de su arquitectura es completamente modular y las funciones que realizan estos componentes pueden ser modificadas de manera sencilla a través de la definición de nuevos *handlers* (elementos que desarrollan funciones específicas de los componentes de Pórtico).

Además de satisfacer los criterios antes expuestos, Pórtico presenta una ventaja más como RTI de referencia en esta investigación: Pórtico es una RTI que se emplea en simuladores militares de tipo constructivo y actualmente se encuentra en dotación en algunos ejércitos ([31]). Esto refuerza a Pórtico como una RTI óptima para realizar el análisis deseado, ya que las conclusiones obtenidas en este capítulo pueden ser de directa aplicación a los simuladores constructivos conectados mediante esta infraestructura.

Por todos estos motivos, la RTI escogida para la implementación real de los IRMs definidos en el capítulo anterior es la versión 1.0.1 de Pórtico.

4.2. Descripción de Pórtico

Una de las características principales de Pórtico es su modularidad ([115]), está compuesto por diferentes elementos que realizan una serie de actividades que sumadas, dan como resultado las funciones que el estándar IEEE1516 define para cada una de las interfaces.

La arquitectura definida para Pórtico agrupa a sus componentes en dos tipos diferentes:

- Componentes situados en la parte del federado (llamados locales o LRC).
- Componentes situados en la parte de la RTI (llamados de servidor o RTI).

4.2.1. Conceptos generales de Pórtico

Antes de comenzar con la descripción de las funciones de los diferentes componentes que forman Pórtico es necesario definir una serie de términos y conceptos básicos que van muy ligados a las funciones que realizan:

- **Queue.** Es un elemento que sirve como almacén de llamadas o respuestas entre componentes que necesitan ser guardadas temporalmente para un posterior procesamiento o envío.

  El objetivo de almacenar las llamadas es para su procesamiento y envío asíncrono o impuesto por el módulo de gestión del tiempo de la federación, *Time Management*.

- **FederateAmbassador.** Es un elemento cuya función es servir como el embajador de un federado. Por lo tanto, este elemento sirve para comunicar al federado las llamadas o mensajes que le envía la RTI.

- **RTIAmbassador.** Este elemento tiene la función contraria al anterior, es decir, es el embajador de una RTI en un federado. Por lo tanto las llamadas realizadas desde un federado a la RTI son enviadas a través de él.

- **Callback.** Son los mensajes enviados desde la RTI a uno o varios federados. El estándar IEEE1516 define que estas llamadas son selectivas, dirigidas solamente a los federados que han informado de su interés en recibirlas.

- **Handlers.** Son módulos que realizan funciones determinadas de los componentes. Por lo tanto, si se varía un *handler* o se desarrolla uno nuevo, se puede modificar o incrementar la funcionalidad de un componente ([116]).

- **Framework de comunicación.** Las llamadas entre los federados y la RTI deben ser soportadas por un framework que permita el envío y recepción de las mismas. Por lo tanto, el framework de comunicación es el soporte de comunicación entre los federados y la RTI.

4.2.2. Descripción de los elementos que componen Pórtico

Pórtico se desglosa en cuatro componentes principales (dos en la parte local y otros dos en la parte de la RTI) y el elemento de comunicación que une la parte del federado con la RTI.

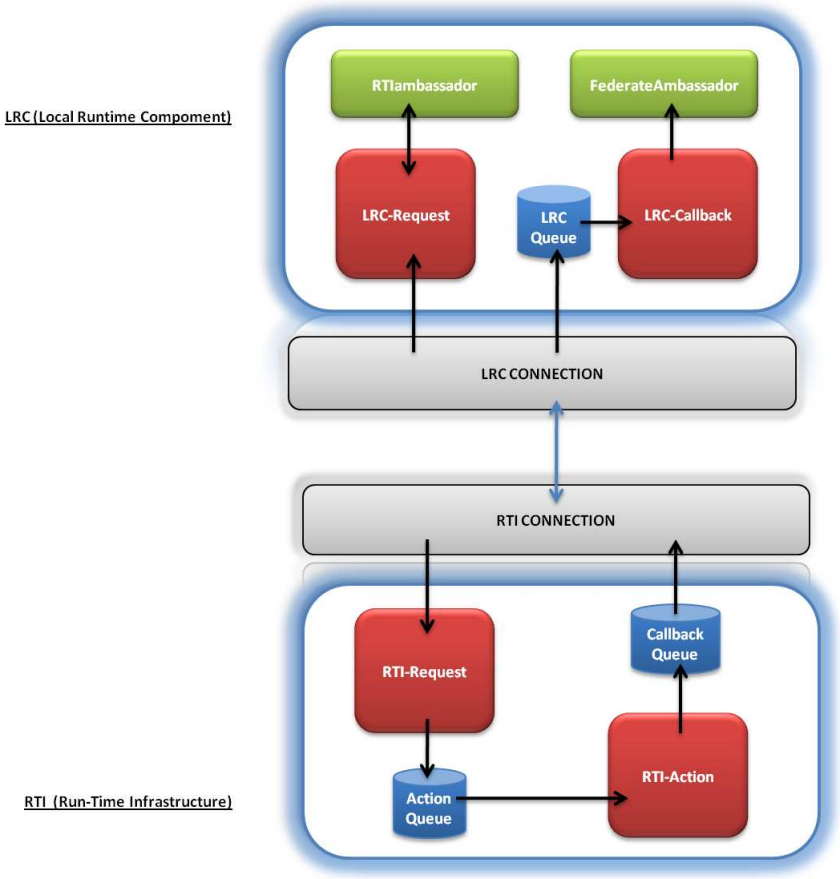

Figura 4.1: Diagrama de componentes de Pórtico ([117])

A estos elementos hay que sumarles el *RTIAmbassador* y el *FederateAmbassador* que no son componentes específicos de Pórtico ya que están definidos en el estándar IEEE1516.

Por lo tanto, los componentes que forman Pórtico (figura 4.1) y sus funciones principales se pueden resumir de la siguiente manera:

- **Componentes LRC (Local Runtime Component).**

  - **LRC-Request.** Este componente recibe las llamadas del *RTIAmbassador* y comprueba que están en el formato aceptado por la RTI Pórtico. En el caso en que la llamada sea admitida, la envía a la RTI. En el caso contrario, no deja que la llamada pase a la RTI.

  - **LRC-Callback.** Este componente lee de la *LRC-Queue* las *callback* que

envía la RTI al federado. Las procesa y se las entrega al *FederateAmbassador* en un formato entendible por él.

- **Componentes RTI (Runtime Infrastructure).**

  - **RTI-Request.** Este componente recibe las llamadas que son enviadas por el *LRC-Request*, las procesa y genera un conjunto de acciones que tiene que realizar la RTI como consecuencia de esa llamada.

    Las acciones resultado del *RTI-Request* son almacenadas en la *Action-Queue*.

  - **RTI-Action.** Lee las acciones almacenadas en la *Action-Queue* y las procesa para generar las *callback* que son almacenadas en la *Callback-Queue*.

- **LRC/RTI Connection.** Este elemento es el que comunica los componentes LRC con los de la RTI.

  De esta manera, se encarga de que las llamadas enviadas de forma directa desde el *LRC-Request* lleguen al *RTI-Request*. Y de que las *callback* almacenadas en la *Callback-Queue* lleguen a la *LRC-Queue*.

Por lo tanto, son tres los elementos de tipo *Queue* que intervienen en la arquitectura de Pórtico: la *LRC-Queue*, la *Action-Queue* y la *Callback-Queue*.

En el caso de Pórtico, el framework de comunicación puede ser cualquiera (la arquitectura de Pórtico es independiente del framework empleado). Aunque por defecto, Pórtico tiene implementado el framework nativo de Java y el definido por Jgroups ([118], [119]).

4.2.3. Diagrama de clases definido para Pórtico

El diagrama de clases de Pórtico define (al más bajo nivel) las clases que dan soporte a cada uno de los componentes y qué tipo de relación existe entre ellas ([117]).

El estudio detallado del diagrama de clases (figura 4.2) ayuda también a entender la composición interna de Pórtico y las llamadas realizadas entre los componentes (como llamadas entre clases).

A continuación se describe cual es la función de las principales clases existentes en Pórtico ([120]).

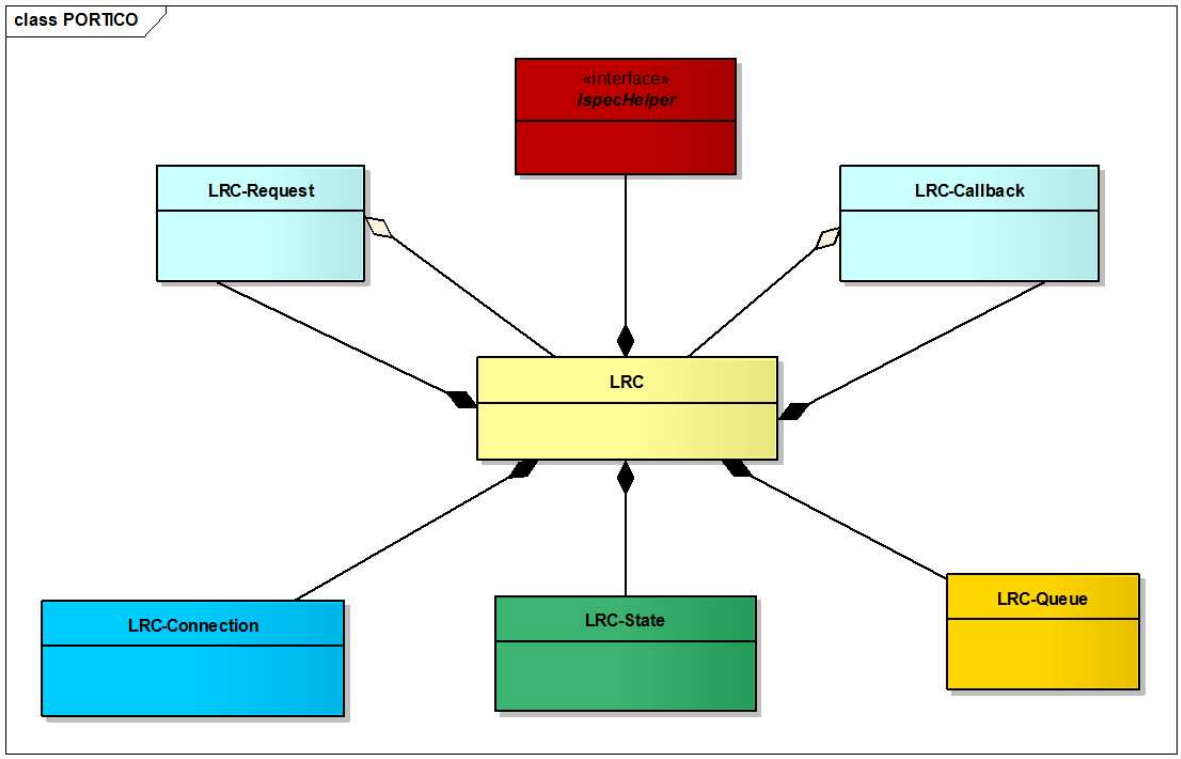

Figura 4.2: Diagrama de clases de Pórtico ([117])

- **LRC.** Es una de las clases más importantes dentro de la arquitectura de Pórtico. Esta clase es creada cuando se instancia un *RTIAmbassador*, ya que es la responsable de gestionar todo el entorno relacionado con un federado específico.

  En consecuencia, esta clase es contenedora de gran parte de las clases que dan cobertura a los componentes locales de Pórtico.

- **LRC-Request.** Esta clase forma parte de la clase LRC y da soporte a toda la funcionalidad del *LRC-Request*.

- **LRC-Callback.** Al igual que la anterior, esta clase está agregada en la clase LRC y es responsable de la funcionalidad del *LRC-Callback*.

- **LRC-State.** Esta clase forma parte de la clase LRC y gestiona toda la información temporal necesaria para que la clase LRC realice sus cometidos.

- **LRC-Queue.** Esta clase forma parte de la LRC y gestiona toda la funcionalidad relacionada con la *LRC-Queue*.

- **LRC-Connection.** Esta clase forma parte de la LRC y además está relacionada con aquellas clases que gestionan componentes de Pórtico locales que necesitan comunicarse con elementos de la RTI (*LRC-Request* y *LRC-Queue*).
- **ISpecHelper.** Esta clase no se puede instanciar (es decir, es abstracta) y es definida para que Pórtico acepte varias implementaciones diferentes del *FederateAmbassador*.

4.3. Aplicación práctica de los IRMs en Pórtico

Para implementar los modelos de interoperabilidad de referencia definidos para simuladores constructivos de aplicación militar con Pórtico, se toman como punto de partida los componentes principales que componen esta RTI (*LRC-Request*, *RTI-Request*, *RTI-Action* y *LRC-Callback*) y se definen las funciones principales que deben realizar cada uno de ellos para asegurar la correcta ejecución de los modelos según su definición.

Las funciones que realizan los componentes de Pórtico en cada IRM se dividen en dos grupos, las que son comunes en la ejecución de todos los IRMs y las que son específicas de cada uno de ellos.

4.3.1. Funciones comunes a todos los IRMs

En todos los IRMs definidos en el capítulo anterior hay una serie de actividades que son comunes, independientemente del tipo de modelo del que se trate. Estas actividades requieren que los componentes de Pórtico realicen una serie de funciones que se denominarán en adelante funciones comunes.

Las actividades o estados comunes en la ejecución de todos los modelo son:

- Punto de partida en la ejecución de un IRM.
- Gestión de los cambios de evento.
- Identificación del IRM que se está ejecutando.

4.3.1.1. Punto de partida en la ejecución de un IRM

Para la ejecución de un IRM es necesario que se cumplan unas condiciones previas y es la misión de todos los componentes de Pórtico comprobar que antes de que

comience la ejecución de un IRM, todos los participantes las cumplan:

- Existe una federación creada que es la que da soporte a la ejecución del IRM.
- Los sistemas participantes (federados) deben estar unidos a la federación anteriormente citada.
- Las configuraciones para la gestión del tiempo en la federación (descritas en la especificación de los IRMs militares realizada en el capítulo anterior) están definidas.

4.3.1.2. Gestión de los cambios de evento

El paso de la ejecución de un evento a otro en un IRM es uno de los puntos más críticos en el desarrollo del mismo. Esto se debe a que los federados y los componentes de Pórtico dejan de desarrollar unas funciones (específicas del evento que está en ejecución) para ejecutar las que requiere el nuevo evento. Si el cambio de evento no está debidamente coordinado se pueden encontrar elementos diferentes ejecutando diferentes eventos (lo cual no asegura la correcta ejecución del IRM).

Aunque la teoría es sencilla su aplicación práctica es algo más compleja. Cuando un federado solicita un avance en el tiempo lógico de la federación toda la federación cambia su tiempo lógico de ejecución ([15]). Por lo tanto cuando un federado termina un evento y solicita el paso al siguiente todos los federados pasan al siguiente evento con independencia de que hayan terminado el evento o no, creando una inconsistencia en la ejecución del IRM.

Por esto es necesario que los elementos que forman Pórtico gestionen el cambio de evento de manera diferente a la definida por el estándar IEEE1516 para el avance en el tiempo, ya que un evento no se puede dar por finalizado hasta que todos los componentes que participan en un IRM no soliciten un avance hacia el siguiente evento. En otras palabras, no se puede avanzar el tiempo lógico de la RTI hasta que todos los federados hayan solicitado un *Time Advance Request* porque hayan finalizado las actividades que debían ejecutarse en el evento del IRM en el que se encontraban.

A continuación se describen las funciones que tienen que desarrollar cada uno de los componentes de Pórtico para que el paso de evento se realice de manera correcta.

4.3.1.2.1. Funciones de los componentes de Pórtico en el paso de evento

- **LRC-Request.** Este componente es el que permite que se envíe una solicitud de avance de tiempo desde el federado a la RTI. Una vez que el federado solicita un avance en el tiempo (es decir, un avance al siguiente evento del IRM) este componente no permite que se envíen más mensajes a la RTI hasta que el cambio de evento sea efectivo. Esto se debe a que un federado que solicita un avance hacia el siguiente evento no debe realizar ninguna actividad más en ese evento puesto que ya lo ha terminado.

- **RTI-Request.** Este componente tiene que ejecutar una de las funciones más importantes dentro de la actividad de cambio de evento, ya que este componente recibe todas las peticiones de avance de tiempo de todos los federados que ejecutan el IRM. Por lo tanto, su función es no procesar un avance de tiempo en la RTI hasta que lo hayan solicitado todos los federados que participan en el IRM, lo contrario generaría los problemas de consistencia en la ejecución del modelo ya mencionados.

- **RTI-Action.** Este componente, al procesar las acciones y convertirlas en *callback* para los federados, puede comunicar a todos ellos que hay un avance hacia el siguiente evento. Por eso su función es leer de la *Action-Queue* la solicitud de avance de tiempo lógico de la federación y generar una *callback* a cada federado que comunica que se avanza hacia el siguiente evento del IRM.

  Esta función requiere que este componente no procese ninguna acción más hasta que haya comunicado a los federados que el evento ha terminado. Esto se debe a que las acciones enviadas antes de que llegue la confirmación de cambio de evento al federado éste las tomara como del evento anterior (y sin embargo son del evento posterior).

- **LRC-Callback.** Este componente comunica al *FederateAmbassador* y al *LRC-Request* el paso al siguiente evento. De esta manera, este último componente reconoce que está en un nuevo evento y permite que se envíen las acciones del nuevo evento a la RTI.

Aunque el objeto de este capitulo es explicar cómo se debería modificar una RTI real para implementar los IRMs propuestos (y así demostrar que es posible hacerlo), no implementar todas las modificaciones propuestas en Pórtico, puede resultar clarificador mostrar un ejemplo de modificación de la RTI. Se ha escogido para ello la gestión del cambio de evento discutida en esta sección.

```
public class TimeAdvanceGrantCallbackHandler extends LRCMessageHandler
{

        //----------------------------------------------------------
        //             INSTANCE VARIABLES
        //----------------------------------------------------------

        private Impl1516Helper helper;
        public int Federados_IRM; // Numero de Federados que participan en el IRM

        //----------------------------------------------------------
        //             INSTANCE METHODS
        //----------------------------------------------------------

        public void initialize( Map<String,Object> properties )
        {
                super.initialize( properties );
                this.helper = (Impl1516Helper)lrc.getSpecHelper();
                Federados_IRM = Control_Federados_IRM.Federados_Activos;

        }// Fin del metodo initialize

        //Federados_Listos Numero de Federados que han solicitado avance de tiempo

        public void process( MessageContext context, int Federados_Listos) throws Exception
        {
                if (Federados_IRM <= Federados_Listos){
                TimeAdvanceGrant grant = context.getRequest( TimeAdvanceGrant.class, this );
                if( logger.isTraceEnabled() )
                        logger.trace( "CALLBACK timeAdvanceGrant(time="+grant.getTime()+")" );
                helper.getFederateAmbassador().timeAdvanceGrant( new DoubleTime(grant.getTime()) );
                context.success();
                }//Fin 1º if

        }//Fin metodo process

}// Fin de la clase TimeAdvanceGrantCallbackHandler
```

Figura 4.3: Código modificado de la clase *TimeAdvanceGrantCallbackHandler*

Se trata de mostrar el código modificado del *handler* que pertenece al componente de Pórtico *RTI-Action*. El nombre de la clase sobre la que se implementa este *handler* es *TimeAdvanceGrantCallbackHandler*. En la figura 4.3 se puede observar en negrita (color rojo) el código que se ha tenido que añadir y que a continuación se explica brevemente.

Existen dos variables nuevas cuyos valores son fundamentales. La primera es *Federados_IRM*, cuyo valor es el número de federados que están participando en la ejecución del IRM. Su valor se asigna al instanciar la clase y lo toma del campo *Federados_Activos* de la clase *Control_Federados_IRM* (al inicio del IRM se instancia un objeto de esta clase, que es el que controla el IRM).

La otra variable es *Federados_Listos*, que es el número de federados que han solicitado el avance en el tiempo para pasar al siguiente evento. Este parámetro se le

pasa al método *process*, que se ejecuta cada vez que un federado solicita un avance en el tiempo.

Este método también se ha modificado añadiendo una sentencia *if* (if Federados_IRM ¡= Federados_Listos) de tal manera que no se ejecute una *callback* si no han solicitado un avance en el tiempo todos los federados del IRM.

4.3.1.3. Identificación del IRM que se está ejecutando

Dependiendo del IRM que se este ejecutando las funciones realizadas por cada uno de los componentes de Pórtico son diferentes. Por este motivo es necesario que la RTI conozca qué tipo de IRM se está ejecutando en una situación de interoperabilidad. De esta manera la RTI puede asegurar que se realizan las actividades requeridas por cada uno de los eventos correctamente.

Existen diferentes maneras de solucionar este problema, aunque para mantener en todo momento la compatibilidad con el estándar IEEE1516, la mejor opción es resolverlo empleando los elementos definidos por el propio estándar.

Por ello se ha decidido etiquetar el comienzo del IRM con el nombre del mismo. De esta manera los federados que desean participar en un IRM, solicitan un avance en el tiempo lógico de la federación (T0) a la etiqueta que lleva el nombre del IRM (con lo que esta acción lleva implícito el conocimiento del tipo de IRM a ejecutar por parte del federado).

Para poder definir etiquetas con el nombre de los IRMs dentro del FOM de la federación, hay que definir en la tabla de sincronización las etiquetas con los nombres de los diferentes tipos de modelos.

Es importante reseñar que todos los federados que participan en el IRM van a solicitar un avance en el tiempo a una etiqueta con el nombre del mismo. Esta información es recogida por el elemento de Pórtico *LRC-Request* y es utilizada en la acción de cambio de evento, puesto que este componente necesita saber cuántos federados participan para evaluar el momento en el que se realiza el cambio de evento: cuando se ha recibido de cada uno de ellos la solicitud de avance en el tiempo.

4.3.1.3.1. Concurrencia en la ejecución de varios IRMs. La solución adoptada anteriormente sería suficiente si sólo fuera posible ejecutar un único IRM en cada momento. Pero lo habitual es que varios IRMs se ejecuten de manera concurrente en una RTI. Por lo tanto, cuando un federado descubra los objetos publicados en la RTI, no tiene ningún criterio para poder identificar cual pertenece a su IRM

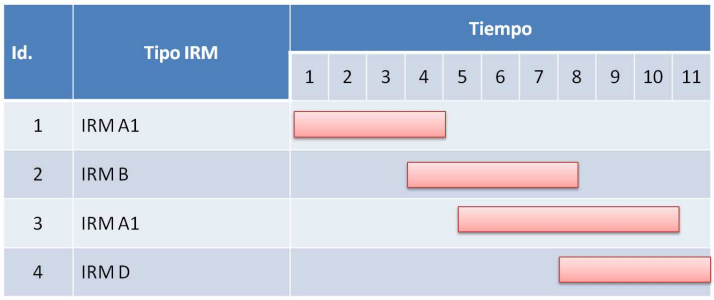

Figura 4.4: Diagrama de ejecución temporal lógica de los IRMs en una RTI

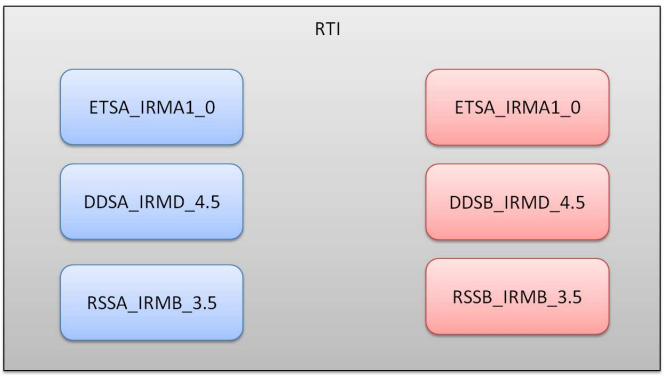

Figura 4.5: RTI con objetos de diferentes IRMs publicados

y cual no (figura 4.4).

Por ello es necesario que cada uno de los objetos publicados en la RTI se identifique con un IRM concreto, o lo que es lo mismo, con un patrón o situación de interoperabilidad concreta. De esta manera los federados que ejecutan ese mismo IRM los podrán diferenciar del resto de objetos (figura 4.5).

La solución propuesta en esta investigación se basa en estandarizar la nomenclatura de los objetos publicados. Para identificar en qué IRM concreto se ejecuta un objeto se definen dos parámetros: el nombre del IRM y el tiempo de federación en el que empieza su ejecución.

De esta manera el nombre de un objeto es: Nombre de objeto + _ + tipo IRM + _ + tiempo de comienzo del IRM. Por ejemplo Soldados_IRMA1_6 significa que el objeto Soldados se publica por el IRM A1 que comienza en un tiempo de ejecución 6.

4.3.2. Funciones específicas del IRM subtipo A1

En esta sección se van a estudiar las funciones que deben realizar los componentes principales de Pórtico para cada uno de los eventos que definen el IRM A1 de manera que éste se pueda ejecutar de forma automática y transparente al usuario.

Aunque se han realizado pruebas exitosas donde la modificación del código permite la ejecución automática del IRM no es objeto de este capítulo mostrar el código fuente desarrollado y/o modificado para la RTI, sino identificar los puntos donde es necesario realizar las modificaciones.

Para la correcta ejecución de este IRM, la entidad transferida en el mismo debe tener un atributo llamado Rol que identifique el tipo de federado que la ha publicado en la RTI (y que por lo tanto tiene los valores A ó B).

4.3.2.0.2. Funciones en el evento 1. En este evento los componentes de Pórtico deben asegurar que se publican dos entidades equivalentes en la RTI y que se actualiza el atributo Rol de ambas con el valor A ó B según corresponda.

Por lo tanto las funciones de cada componente de Pórtico en este evento son las siguientes:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son para publicar una entidad y actualizar su atributo Rol con el valor A ó B.

  En el caso en el que un federado no publique una entidad y/o no actualice a continuación el atributo Rol antes de solicitar un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente recibe los mensajes enviados a la RTI y comprueba que las entidades que se van a publicar son equivalentes (según la especificación ETS).

  En tal caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario, bloquea la ejecución del IRM por entidades incoherentes.

- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones que se deben ejecutar y comprueba que, como mínimo, se publican dos entidades:

una con el atributo Rol con valor A y la otra con valor B.

En el caso en el que no se cumpla esto, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee de la *LRC-Queue* las *callback*, y como los federados son *Time Constrained*, este componente no comunica las *callback* al *FederateAmbassador* hasta que todos los federados no envían un *Time Advance Request*.

4.3.2.0.3. Funciones en el evento 2. En este evento los componentes de Pórtico deben asegurar que el federado A queda suscrito a la entidad que publicó el federado B en el evento anterior y viceversa.

Por lo tanto las funciones que cada componente de Pórtico debe realizar en este evento son:

- **LRC-Request.** Este componente comprueba que los mensajes que manda el federado a la RTI son para suscribirse a una entidad publicada en la misma.

  En el caso en el que un federado no se suscriba a una entidad y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI durante este evento son para suscribirse a las entidades publicadas en el evento 1 de este IRM.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario, se expulsa de la ejecución del IRM al federado que se ha suscrito a una entidad no publicada en el evento 1 y ha requerido un *Time Advance Request*.

- **RTI-Action.** Este componente comprueba que, como mínimo, existe un federado suscrito a una entidad con el atributo Rol con valor A y otro federado suscrito a otra entidad con el atributo Rol con valor B.

  En el caso en el que no se cumpla este mínimo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Ejecuta las mismas acciones que en el evento 1 de este IRM.

4.3.2.0.4. Funciones en el evento 3. En este evento los componentes de Pórtico deben asegurar que el federado A actualiza la entidad que publica en el evento 1 de este IRM.

Por lo tanto las funciones de los elementos de Pórtico para este evento son las siguientes:

- **LRC-Request.** Este componente es el encargado de comprobar que los mensajes que envía el federado a la RTI son exclusivamente para actualizar una entidad publicada.

- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son para actualizar la entidad que publicó el federado A en el evento 1 de este IRM.

  En caso afirmativo, genera las acciones oportunas para la RTI y las almacena en la *Action-Queue*. En caso contrario, no se permite que los mensajes sean almacenados en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones almacenadas en la *Action-Queue* y comprueba que son las necesarias para actualizar los datos de una entidad cuyo atributo Rol es A.

  En tal caso, ejecuta las acciones en la RTI y genera sus *callback* correspondientes, que son almacenadas en la *Callback-Queue*. Si no existe ninguna acción para actualizar como mínimo una entidad cuyo atributo Rol es A, se bloquea la ejecución del IRM.

- **LRC-Callback.** Ejecuta las mismas acciones que en el evento 1 de este IRM.

4.3.2.0.5. Funciones en el evento 4. En este evento los componentes de Pórtico deben asegurar que el federado B actualiza la entidad que publicó en el evento 1 con los mismos datos con los que el federado A actualizó su entidad publicada en el evento anterior (el evento 3).

En este sentido las funciones principales de los componentes de Pórtico para este evento son las siguientes:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son para actualizar los atributos de una entidad publicada.

- **RTI-Request.** Este componente comprueba que los mensajes que recibe la RTI son para actualizar la entidad que publicó el federado B en el evento 1 de este IRM.

  En este caso, genera las acciones para realizar esta función y las almacena en la *Action-Queue*. En caso contrario, no se permite que los mensajes sean almacenados en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones almacenadas en la *Action-Queue*, comprueba que indican la actualización de una entidad con el atributo Rol con valor B y que los datos de actualización son los mismos con los que el federado A actualizó la entidad en el evento anterior.

  En este caso, ejecuta las acciones en la RTI y genera las correspondientes *callback* para almacenarlas en la *Callback-Queue*. Si no existe ninguna acción para actualizar como mínimo una entidad cuyo atributo Rol es B, bloquea la ejecución del IRM.

- **LRC-Callback.** Ejecuta las mismas acciones que en el evento 1 de este IRM.

4.3.2.0.6. Funciones en el evento 5. En este evento los componentes de Pórtico deben asegurar que el federado A y el federado B despublican la entidad que publicaron en el evento 1 de este IRM.

En consecuencia las funciones principales de los componentes de Pórtico para este evento son las siguientes:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son para despublicar entidades.

  En el caso en el que un federado no despublique la entidad publicada en el evento 1 o en el evento 2 y solicite un *Time Advance Request*, queda bloqueado y no se le permite finalizar el IRM ni solicitar un *Time Advance Request* (por lo tanto queda bloqueado en el último tiempo lógico en el que ha estado).

- **RTI-Request.** Este componente comprueba que los mensajes recibidos por la RTI en este evento son para despublicar las entidades que se publicaron en el evento 1 de este IRM.

  En este caso, genera las acciones para llevar a cabo esta función y las almacena en la *Action-Queue*. En caso contrario, se bloquea la ejecución del IRM impidiendo que se solicite un *Time Advance Request* por parte de los federados.

- **RTI-Action.** Este componente lee las acciones almacenadas en la *Action-Queue* y comprueba que son las necesarias para despublicar, como mínimo, una entidad con el atributo Rol con valor A y otra con valor B.

  En este caso ejecuta las acciones en la RTI y genera sus *callback* correspondientes almacenándolas en la *Callback-Queue*. En caso contrario se bloquea la ejecución del IRM impidiendo que se solicite un *Time Advance Request* por parte de los federados.

- **LRC-Callback.** Ejecuta las mismas acciones que en el evento 1 de este IRM.

4.3.3. Funciones específicas del IRM subtipo A2

Para ejecutar este subtipo de IRM con Pórtico las entidades que están definidas en el FOM de la federación deben cumplir ciertos requisitos:

1. Tener un atributo llamado Rol que identifique si el federado que la ha publicado es del tipo A ó B (al igual que ocurría en el caso del subtipo A1).

2. Tener un atributo llamado Reserva. Este atributo informa al federado con el rol A en qué estado se encuentra una solicitud de reserva en la cola de destino: solicitado, bloqueada o permitida.

3. Tener un atributo llamado Volumen. Este atributo informa del volumen que se desea reservar para almacenar la entidad que se va a transferir.

4.3.3.0.7. Funciones en el evento 1. En este evento ambos federados publican la entidad que se va a transferir. Las funciones de cada componente de Pórtico son las mismas que se describieron para el evento 1 del IRM A1.

4.3.3.0.8. Funciones en el evento 2. En este evento el federado A se suscribe a la entidad publicada por el federado B y viceversa. Con lo cual, las acciones de cada componente de Pórtico son idénticas a las descritas para el evento 2 del IRM A1.

4.3.3.0.9. Funciones en el evento 3. En este evento el federado A realiza una solicitud de reserva de espacio en la cola del federado B para posteriormente transferir la entidad. Así que las actividades realizadas por los componentes de Pórtico para este evento son:

- **LRC-Request.** Este componente comprueba que el federado envía los mensajes necesarios para actualizar los atributos Reserva y Volumen del objeto que publicó en el evento 1 (el valor del atributo Reserva es Solicitada).

  En el caso en el que un federado no envíe estos mensajes y solicite un *Time Advance Request*, queda bloqueado y no se permite pasar al siguiente evento (quedando fuera del IRM).

- **RTI-Request.** Este componente comprueba que la RTI sólo recibe mensajes para la actualización los atributos Reserva y Volumen de los objetos publicados que tienen el atributo Rol con el valor A.

  Si la comprobación es positiva, crea las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario, no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente comprueba que al menos se han actualizado los atributos de Reserva y Volumen, en al menos un objeto cuyo valor del atributo Rol sea A.

  En este caso, ejecuta las acciones solicitadas en la RTI y genera las *callback* que son almacenadas en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee de la *Require-Queue* las *callback* y como los federados son *Time Constrained*, este componente no comunica las *callback* al *FederateAmbassador* hasta que todos los federados no envían un *Time Advance Request*.

4.3.3.0.10. Funciones en el evento 4. En este evento el federado B bloquea la transferencia ya que la cola receptora no tiene espacio suficiente para almacenar la entidad transferida.

Para ello el atributo Reserva de la entidad publicada por el federado B en el evento 1 es actualizada al valor Bloqueada (el federado A al recibir la *callback* entiende que no puede enviar la entidad).

Las actividades realizadas por los componentes de Pórtico para este evento son:

- **LRC-Request.** Este componente comprueba que el federado sólo envía los mensajes necesarios para actualizar el atributo Reserva del objeto publicado en el evento 1 (el valor del atributo Reserva es Bloqueada).

- **RTI-Request.** Este componente comprueba que sólo llegan a la RTI mensajes de actualización para el atributo Reserva al valor Bloqueada para los objetos cuyo atributo Rol tiene el valor B.

  Si se cumple esta condición, crea las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente comprueba que al menos se ha actualizado el atributo de Reserva en al menos un objeto cuyo valor del atributo Rol sea B.

  En este caso, ejecuta las acciones solicitadas en la RTI y genera las *callback* que son almacenadas en la *Callback-Queue*. En el caso contrario, se bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee de la *Require-Queue* las *callback* y como los federados son *Time Constrained*, este componente no comunica las *callback* al *Federate Ambassador* hasta que todos los federados no envían un *Time Advance Request*.

4.3.3.0.11. Funciones en el evento 5. Este evento comienza cuando la cola receptora tiene suficiente espacio para poder almacenar la entidad transferida.

Por lo tanto, el federado B actualiza el atributo Reserva de la entidad que publica en el evento 1 al valor Permitida.

- **LRC-Request.** Este componente comprueba que el federado sólo envía los mensajes necesarios para actualizar el atributo Reserva del objeto publicado en el evento 1 (el valor del atributo Reserva es Permitida).

- **RTI-Request.** Este componente comprueba que sólo llegan a la RTI mensajes de actualización para el atributo Reserva al valor Permitida para los objetos cuyo atributo Rol tiene el valor B.

  Si se cumple esta condición, crea las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Ejecuta las mismas acciones que el evento 4 de este mismo IRM.
- **LRC-Callback.** Ejecuta las mismas acciones que el evento 4 de este mismo IRM.

A partir de este evento se sigue la misma secuencia que a partir del evento 3 del IRM A1, así que no son necesarias nuevas modificaciones en los componentes de Pórtico.

4.3.4. Funciones específicas del IRM subtipo A3

Para ejecutar este subtipo de IRM con Pórtico las entidades que están definidas en el FOM de la federación deben cumplir ciertos requisitos:

1. Tener un atributo llamado Rol que identifique si el federado que la ha publicado es del tipo A, B ó C (al igual que ocurría en el caso del subtipo A1).
2. Tener unos atributos que permitan contener la información definida en el protocolo de prioridades (sección 3.3.4.6) para el vector de reserva (LLD).
3. Tener un atributo llamado Reserva. Este atributo informa en qué estado se encuentra la transferencia de la entidad en la cola destino: Desconocida, Permitida.

4.3.4.0.12. Funciones en el evento 1. En este evento los componentes de Pórtico deben asegurar que los federados A y C publican en la RTI la entidad que quieren transferir al federado B.

Por lo tanto las funciones que deben realizar los componentes de Pórtico para este evento son:

- **LRC-Request.** Este componente comprueba que el federado envía los mensajes necesarios para publicar la entidad que se desea trasferir y actualizar el atributo Rol.

  En el caso en el que un federado no publique una entidad, no actualice el atributo Rol y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que a la RTI sólo llegan los mensajes necesarios para la publicación de entidades a transferir y para la actualización del atributo Rol con el valor A ó C.

  Si la comprobación es positiva, crea las acciones correspondientes y las pasa a la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Comprueba que en la *Action-Queue* existen las acciones necesarias para publicar, como mínimo, una entidad con el atributo Rol con valor A y otra con el valor C.

  En este caso, ejecuta las acciones en la RTI y genera las *callback* de las mismas y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee las *callbacks* almacenadas en la *LRC-Queue* pero como los federados son *Time Constrained* este componente no libera las *callback* al *FederateAmbassador* hasta que todos los federados no envían el *Time Advance Request*.

4.3.4.0.13. Funciones en el evento 2. En este evento los componentes de Pórtico deben asegurar que al final del mismo el federado B queda suscrito a las entidades publicadas en la RTI en el evento anterior.

Por lo tanto las funciones que deben realizar los componentes de Pórtico en este evento son las siguientes:

- **LRC-Request.** Este componente comprueba que el federado envía a la RTI los mensajes necesarios para suscribirse a entidades publicadas en el evento anterior.

  En el caso en el que un federado no se suscriba a una entidad y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son los necesarios para suscribirse a las entidades con el atributo Rol con el valor A y C.

Si la comprobación es positiva crea las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Comprueba que en la *Action-Queue* se almacenan, como mínimo, las acciones necesarias para suscribirse a una entidad con el atributo Rol A y a otra con el atributo Rol C.

  En este caso, ejecuta las acciones en la RTI y genera las *callback* y las almacena en la *Callback-Queue*. En el caso contrario, se bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.4.0.14. Funciones en el evento 3. En este evento los componentes de Pórtico deben realizar las funciones necesarias para asegurar que el federado B recibe los vectores de reserva (LLD) de los federados A y C.

Por lo tanto las funciones de cada componente de Pórtico para este evento son:

- **LRC-Request.** Este componente sólo permite al federado enviar aquellos mensajes de actualización de los atributos que forman los datos contenidos en el vector de reserva.

- **RTI-Request.** Este componente recibe los mensajes del componente anterior y comprueba que las actualizaciones se realizan sobre entidades cuyo atributo Rol tiene el valor de A y C.

  En caso afirmativo genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente comprueba que las acciones que lee de la *Action-Queue*, por lo menos, son para actualizar los atributos del vector de reserva de una entidad con el atributo Rol con valor A y otra con el valor C.

  En este caso ejecuta las acciones sobre la RTI y genera las *callback* que las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo

impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.4.0.15. Funciones en el evento 4. En este evento los componentes de Pórtico deben asegurar que los federados publican dos entidades equivalentes a las que publicaron los federados A y C en el evento 1.

Por lo tanto las funciones de cada componente de Pórtico para este evento son:

- **LRC-Request.** Este componente comprueba que el federado sólo envía mensajes para publicar dos entidades y actualizar su atributo Rol con el valor B.

  En el caso en el que un federado no publique una entidad, no actualice el atributo Rol a B y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que sólo llegan los mensajes necesarios para la publicación de dos entidades por federado B y para la actualización del atributo con valor B.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Comprueba que las *action* almacenadas en la *Action-Queue* son las necesarias para publicar, al menos, dos entidades equivalentes a las publicadas por los federados A y C en el evento 1.

  En tal caso genera las *callback* y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este requisito, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.4.0.16. Funciones en el evento 5. En este evento los componentes de Pórtico deben asegurar que los federados A y C se suscriben a las entidades que publicó el federado B en el evento anterior.

Por lo tanto las acciones que debe realizar cada componente de Pórtico son las siguientes:

- **LRC-Request.** Este componente comprueba que el federado envía mensajes a la RTI para suscribirse a una entidad publicada en la RTI.

- **RTI-Request.** Este componente comprueba que los mensajes que recibe son de suscripción para entidades cuyo atributo Rol tenga un valor B.

  En este caso, transforma estos mensajes en acciones y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente comprueba que las acciones almacenadas en la *Action-Queue* permiten que un federado A ó C se suscriba a una entidad con el valor del atributo Rol en B.

  En este caso ejecuta las acciones en la RTI y genera las *callback* de las mismas y las almacena en la *Callback-Queue*. En caso contrario bloquea la ejecución del IRM e impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.4.0.17. Funciones en el evento 6. Llegados a este evento las transferencias de entidades desde el federado A y C están bloqueadas.

En este evento los componentes de Pórtico deben asegurar que se desbloquea la transferencia del federado cuyo vector de reserva esté mejor ubicado en la lista de reserva de la cola del federado B.

Para ello las acciones realizadas por cada componente de Pórtico son las siguientes:

- **LRC-Request.** Este componente comprueba que el federado sólo envía a la RTI mensajes de actualización del atributo Reserva al valor Permitida.

- **RTI-Request.** Este componente comprueba que los mensajes de actualización provienen de un federado B y se realizan sobre un atributo de una entidad con el atributo Rol con el valor B.

  En este caso genera las acciones actualizar este valor en la RTI y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Lee de la *Action-Queue* las acciones almacenadas y comprueba que se actualiza solamente el atributo de una entidad BA ó BC.

  En este caso ejecuta las acciones en la RTI y genera su *callback* correspondientes y las almacena en la *Callback-Queue*. En caso contrario bloquea la ejecución del IRM e impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

A partir de este evento se sigue la misma secuencia que en el IRM A1 empezando por el evento 3, por lo que no son necesarias nuevas modificaciones en los componentes de Pórtico, que ya están preparados para realizar las actividades que les corresponden.

4.3.5. Funciones específicas del IRM tipo B

Para la ejecución de este modelo en Pórtico, los recursos compartidos por los federados deben cumplir las siguientes condiciones:

1. Los recursos deben tener definido un atributo llamado Rol cuyo valor identifica el tipo de federado que los publica (los valores son A y B).

2. Mientras dos federados comparten un recurso, los eventos 3 y 4 se repiten de manera cíclica cada vez que el federado A actualice el recurso.

   Para controlar el flujo de ejecución de este IRM (es decir, para saber si desde el evento 4 se vuelve al evento 3 o si por el contrario, se debe pasar al evento 5 para terminar el IRM) se define un atributo en el recurso compartido llamado Control IRMB.

Este atributo tiene como valor el evento que sigue al evento 4. De esta manera inicialmente tiene el valor 3 (indicando que al finalizar el evento 4 se ejecuta el evento 3) pero cuando se finaliza el IRM, el valor de este atributo es 5.

4.3.5.0.18. Funciones en el evento 1. En este evento los componentes de Pórtico deben asegurar que al final del mismo el federado A y el federado B han publicado recursos equivalentes en la RTI.

Para ello las funciones de cada componente de Pórtico durante la ejecución de este evento son:

- **LRC-Request.** Comprueba que el federado envía únicamente mensajes a la RTI para publicar un recurso y actualizar el atributo Rol (con valor A ó B según corresponda).

  En el caso en el que un federado no publique un recurso, no actualice su atributo Rol y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que los mensajes recibidos por la RTI son para la publicación de recursos equivalentes (según la especificación RSS).

  Una vez comprobado este requisito, genera las acciones para realizar esta publicación y las almacena en la *Action-Queue*. Si no se cumple, se bloquea la ejecución del IRM por encontrarse en la RTI recursos incoherentes.

- **RTI-Action.** Comprueba que las acciones almacenadas en la *Action-Queue* son las necesarias para, como mínimo, publicar dos recursos: uno con el valor del atributo Rol A y el otro con el valor B.

  En caso afirmativo ejecuta las acciones en la RTI y genera sus *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso negativo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee las *callbacks* almacenadas en la *LRC-Queue* pero como los federados son *Time Constrained*, este componente no libera las *callback* al

FederateAmbassador hasta que todos los federados no envían el *Time Advance Request*.

4.3.5.0.19. Funciones en el evento 2. En este evento el Federado A se suscribe a la entidad publicada por el Federado B y viceversa. Esto implica que las acciones de cada componente serán idénticas a las descritas para el evento 2 del IRM A1.

4.3.5.0.20. Funciones en el evento 3. La función de los componentes de Pórtico en este evento es asegurar que el federado A actualiza el recurso publicado en el evento 1 y el federado B recibe esas actualizaciones (en estas actualizaciones se incluye la del atributo Control IRMB).

Por lo tanto las funciones de cada componente de Pórtico durante la ejecución de este evento son:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son de actualización del recurso publicado en el evento 1.
- **RTI-Request.** Este componente comprueba que los mensajes de actualización que llegan a la RTI son para el recurso cuyo atributo Rol tiene el valor A.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.
- **RTI-Action.** Este componente lee las acciones de la *Action-Queue* y comprueba que son las necesarias para actualizar los atributos de, al menos, un recurso con el atributo Rol con valor A.

  En tal caso ejecuta las acciones en la RTI y genera las *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).
- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.5.0.21. Funciones en el evento 4. Durante la ejecución de este evento los componentes de Pórtico deben asegurar que el federado B actualiza el recurso que

publicó en el evento 1 con los mismos datos que el federado A ha utilizado para actualizar su recurso publicado en el evento anterior.

Por lo tanto las funciones de cada componente de Pórtico durante este evento son:

- **LRC-Request.** Este componente comprueba que los mensajes que el federado envía a la RTI son de actualización del recurso publicado.
- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son para actualizar un recurso publicado con el atributo Rol con valor B.

  En este caso genera las acciones necesarias para ejecutarlo en la RTI y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.
- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que son para actualizar un recurso con los mismos datos que se utilizaron para actualizar el recurso en el evento anterior.

  En este caso ejecuta las acciones en la RTI y genera sus correspondientes *callback* que las almacena en la *Callback-Queue*. En el caso en el que no se cumpla esta condición, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).
- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.5.0.22. Funciones en el evento 5. En este evento las funciones de cada uno de los componentes de la arquitectura de Pórtico son equivalentes a las que se describieron en el evento 5 del IRM A1.

4.3.6. Funciones específicas del IRM tipo C

Los componentes de Pórtico deben asegurar que se ejecutan correctamente los eventos definidos en la especificación de este IRM propuesta en el capítulo anterior.

Para que la ejecución de este IRM se pueda desarrollar correctamente en Pórtico, la especificación del evento compartido (ESS) debe contener un atributo llamado Rol que indica qué federado publica ese evento (en este caso sólo puede tomar el valor A).

4.3.6.0.23. Funciones en el evento 1. En este evento los componentes de Pórtico deben asegurar que los federados A y B avanzan su tiempo lógico al tiempo T0 marcado como el tiempo inicial de ejecución del IRM.

Por lo tanto las funciones de cada componente de Pórtico para desarrollar este evento son:

- **LRC-Request.** Este componente comprueba que los federados sólo envían mensajes de *Time Advance Request* a la RTI.

  En el caso en el que el federado no envíe un *Time Advance Request*, queda fuera en la ejecución del IRM, no pudiendo enviar mas mensajes a la RTI sobre este IRM específico.

- **RTI-Request.** Este componente recibe los mensajes que llegan a la RTI y comprueba que los mensajes de avance de tiempo de los federados son hacia un tiempo lógico común (T0, que es el punto de partida del IRM e indica el momento en el que tiene lugar el evento).

  En caso afirmativo genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario, no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones almacenadas en la *Action-Queue* y comprueba que, como mínimo, existen dos federados que envían la solicitud de avance de tiempo lógico.

  En tal caso ejecuta las acciones en la RTI y las *callback* correspondientes almacenándolas en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee las *callbacks* almacenadas en la *LRC-Queue* pero como los federados son Time Constrained, este componente no libera las *callback* al *FederateAmbassador* hasta que todos los federados no envían el *Time Advance Request*.

4.3.6.0.24. Funciones en el evento 2. En este evento los componentes de Pórtico deben asegurar que el federado A publica un evento en la RTI.

Por lo tanto las funciones de los componentes de Pórtico para la ejecución de este evento son:

- **LRC-Request.** Este componente debe comprobar que el federado envía mensajes a la RTI para publicar un evento y actualizar el parámetro Rol.

  En el caso en el que un federado no publique un evento, no actualice en atributo Rol y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que los mensajes que recibe la RTI son para publicar un evento y actualizar su parámetro Rol al valor A.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que, al menos, se publica un evento con el parámetro Rol con valor A.

  En este caso ejecuta las acciones en la RTI y genera las *callback* correspondientes. En el caso contrario, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.6.0.25. Funciones en el evento 3. En este evento los componentes de Pórtico deben asegurar que el federado B se suscribe al evento que publicó el federado A en el evento anterior.

Para alcanzar este fin las funciones de cada componente de Pórtico para este evento son las siguientes:

- **LRC-Request.** Comprueba que los mensajes que envía el federado a la RTI son de suscripción al evento publicado en la RTI.

  En el caso en el que un federado no se suscriba a un evento o no actualice el atributo Rol y solicite un *Time Advance Request*, queda fuera de la ejecución

del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este componente comprueba que los mensajes que recibe la RTI son para suscribirse a un evento cuyo parámetro Rol tiene el valor A.

  En este supuesto genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que son las necesarias para, como mínimo, que un federado B quede suscrito a un evento publicado en la RTI.

  En este caso ejecuta las acciones en la RTI y genera las *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla esta condición se bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.6.0.26. Funciones en el evento 4. En este evento los componentes de Pórtico tienen que asegurar que el federado A actualiza el evento publicado en el evento 2 del modelo con los datos correspondientes al mismo.

Por lo tanto las funciones de cada componente de Pórtico para este evento son:

- **LRC-Request.** Este componente analiza que los mensajes que manda el federado a la RTI son de actualización del evento publicado anteriormente.

- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son de actualización de un evento ya publicado cuyo parámetro Rol tienen el valor A.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones de la *Action-Queue* y comprueba que al menos un evento publicado en la RTI es actualizado.

En este caso ejecuta las acciones en la RTI y genera las *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.6.0.27. Funciones en el evento 5. En este evento los componentes de Pórtico tienen como finalidad principal asegurar que el federado A despublica el evento que publicó en el evento 2.

Por lo tanto las funciones de cada componente de Pórtico durante este evento son:

- **LRC-Request.** Este elemento comprueba que los mensajes que envía el federado a la RTI son los necesarios para despublicar un evento.

  En el caso en el que un federado no se despublique el evento publicado en el evento 1 y solicite un *Time Advance Request*, queda bloqueado y no se le permite finalizar el IRM, por lo tanto no se avanza su tiempo lógico.

- **RTI-Request.** Este componente comprueba que los mensajes que recibe la RTI son para despublicar un evento con el parámetro Rol con valor A.

  En tal caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que son las necesarias para, como mínimo, despublicar el evento publicado en el evento 2 de este IRM.

  Es este caso ejecuta las acciones en la RTI y las correspondientes *callback* y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla esta condición, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.7. Funciones específicas del IRM tipo D

La función principal de los componentes de Pórtico para la ejecución de este tipo de IRM es permitir que los federados A y B puedan compartir la información de una estructura de datos.

Para la correcta ejecución del IRM en Pórtico la especificación de la estructura de datos (DSS) debe cumplir las siguientes condiciones:

1. Debe existir un atributo llamado Rol que identifica al federado que ha publicado la estructura de datos en la RTI (por lo que puede tomar los valores A y B).

2. Debe existir un atributo llamado Control IRMD cuyo valor por defecto es 3 e indica qué evento se va a ejecutar después del evento 4. De esta manera se controla el flujo de ejecución del IRM al igual que se hizo en el caso del IRM de tipo B.

4.3.7.0.28. Funciones en el evento 1. En este evento los componentes de Pórtico deben asegurar que los federados que participan en este IRM publican en la RTI una DSS idéntica.

Para ello las funciones que ejecutan cada componente de Pórtico durante la ejecución de este evento son:

- **LRC-Request.** Comprueba que los mensajes que se envían a la RTI son los necesarios para publicar una DSS.

  En el caso en el que un federado no publique una DSS, no actualice su atributo Rol y solicite un *Time Advance Request*, queda fuera de la ejecución del IRM. Eso significa que este componente bloquea todos los mensajes (en relación a este IRM) enviados a la RTI por parte de este federado.

- **RTI-Request.** Este elemento comprueba que los mensajes que llegan a la RTI son los necesarios para publicar las estructuras de datos y actualizar su atributo Rol con un valor A ó B, según corresponda en cada caso.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Lee de la *Action-Queue* las acciones y comprueba que son las necesarias para publicar al menos dos estructuras de datos, una con el atributo Rol con valor A y la otra con el valor B.

  En este caso ejecuta las acciones en la RTI, genera sus *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee las *callbacks* almacenadas en la *LRC-Queue*, pero como los federados son *Time Constrained*, este componente no libera las *callback* al *FederateAmbassador* hasta que todos los federados no envían el *Time Advance Request*.

4.3.7.0.29. Funciones en el evento 2. En este evento los componentes de Pórtico deben asegurar que el federado A se suscribe a la estructura de datos publicada por el federado B en el evento anterior y viceversa.

Con lo cual las acciones de cada componente de Pórtico para este evento son:

- **LRC-Request.** Este componente comprueba que los federados sólo envían mensajes de suscripción a la RTI.

- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son únicamente para suscribirse a las estructuras de datos publicadas en el evento anterior.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones de la *Action-Queue* y comprueba que son las necesarias para, como mínimo, suscribirse a una estructura de datos con atributo Rol con valor A y otra con valor B.

  En este caso ejecuta las acciones en la RTI y genera sus correspondientes *callback* y las almacena en la *Callback-Queue*. En el caso en el que no se cumpla esta condición, bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1 de este IRM.

4.3.7.0.30. Funciones en el evento 3. En este evento los componentes de Pórtico deben asegurar que el federado A actualiza la estructura de datos que publicó en la RTI en el evento 1.

Esto significa que las acciones de cada componente de Pórtico son equivalentes a las descritas para el evento 3 del IRM tipo B (cambiando el recurso por estructura de datos).

4.3.7.0.31. Funciones en el evento 4 En este evento los componentes de Pórtico deben asegurar que el federado B actualiza la estructura de datos que publicó en la RTI en el evento 1 con los mismos datos que actualizó el federado A la estructura de datos en el evento anterior.

Por lo que las acciones de cada componente de Pórtico son equivalentes a las descritas para el evento 4 del IRM tipo B.

4.3.7.0.32. Funciones en el evento 5. En este evento los componentes de Pórtico deben asegurar que el federado A y B despublican la estructura de datos que publicaron en el evento 1 de este IRM. Por lo que las acciones de cada componente de Pórtico son equivalentes a las descritas para el evento 5 del IRM A1.

4.3.8. Funciones específicas del IRM tipo E

Para la aplicación práctica de este IRM el Pórtico los componentes del mismo deben asegurar que los federados participantes en el mismo pueden compartir y modificar un plan/orden siguiendo la especificación para este IRM realizada en el capítulo anterior.

Es necesario realizar unas consideraciones previas que afectan, principalmente, a la especificación del Plan/Orden (P/O):

1. El P/O debe tener un atributo llamado Rol que identifica qué tipo de federado lo ha publicado en la RTI (por lo que puede tomar los valores A ó B).

2. El evento 3 es el ultimo evento de la fase 1. Esta fase se mantiene en este evento hasta que finaliza la ejecución del IRM.

Para anunciar cuándo finaliza esta fase (y por lo tanto el IRM) el P/O tiene un atributo llamado Control fase1 con el valor En vigor por defecto.

Cuándo se cambia el valor de este atributo a Derogado el IRM finaliza.

3. La fase 2 del IRM se repite tantas veces como el federado B tenga que hacer actualizaciones del P/O.

   Para controlar el flujo de esta fase el P/O tiene un atributo llamado Control fase2 con el valor Borrador.

   La fase 2 finaliza si al finalizar el evento 9 este atributo tiene el valor de Aprobado.

4.3.8.0.33. Funciones en la fase 1: evento 1. En este evento los componentes de Pórtico deben asegurar que el federado A publica un P/O en la RTI.

Por lo tanto las acciones de los componentes de Pórtico para este evento son equivalentes a las del evento 2 del IRM C (cambiando evento por P/O).

4.3.8.0.34. Funciones en la fase 1: evento 2. En este evento los componentes de Pórtico deben asegurar que el federado B se suscribe al P/O publicada por el federado A en el evento anterior.

Con lo que las acciones de los componentes de Pórtico son equivalentes a las del evento 3 del IRM C (cambiando evento por P/O).

4.3.8.0.35. Funciones en la fase 1: evento 3. En este evento los componentes de Pórtico deben asegurar que el federado A actualiza el P/O publicado en el evento 1 de este IRM.

Para ello las acciones de los componentes de Pórtico para este eventos son las siguientes:

- **LRC-Request.** Este componente comprueba que los mensajes que manda el federado a la RTI son los necesarios para actualizar el P/O publicado.

- **RTI-Request.** En este evento este componente comprueba que la RTI sólo recibe mensajes de actualización para un P/O cuyo atributo Rol tiene el valor A.

En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.

- **RTI-Action.** Este componente lee las acciones de la *Action-Queue* y comprueba que son las necesarias para publicar al menos un P/O con el atributo Rol con valor A.

  En este caso ejecutas las acciones en la RTI y genera sus *callback* correspondientes. En el caso que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permite almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Lee las *callback* almacenadas en la *LRC-Queue* pero como los federados son *Time Constrained* este componente no libera las *callback* al *FederateAmbassador* hasta que todos los federado no envían el *Time Advance Request*.

4.3.8.0.36. Funciones en la fase 2: evento 4. En la ejecución de este evento los componentes de Pórtico deben asegurar que el federado B publica un P/O en la RTI.

Por lo tanto las acciones de los componentes de Pórtico son equivalentes a las ejecutadas en el evento 2 del IRM C (las únicas diferencias es que se publica un P/O en vez de un evento y que el valor del atributo Rol es B).

4.3.8.0.37. Funciones en la fase 2: evento 5. En este evento los componentes de Pórtico deben asegurar que el federado A se suscribe al P/O publicado por el federado B en el evento anterior.

En consecuencia las acciones de los componentes de Pórtico son equivalentes a las realizadas en el evento 3 del IRM C (las únicas diferencias es que se publica un P/O en vez de un evento y que el valor del atributo Rol es B).

4.3.8.0.38. Funciones en la fase 2: evento 6. En este evento los elementos que forman Pórtico deben asegurar el federado B actualiza el P/O que publicó en el evento anterior.

Por lo tanto las acciones que los componentes de Pórtico deben realizar durante este evento son las siguientes:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son para actualizar el P/O que ha publicado anteriormente.
- **RTI-Request.** Este componente comprueba que los mensajes que llegan a la RTI son para la actualización de un P/O publicado cuyo valor del atributo Rol es B.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.
- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que se actualiza al menos un P/O publicado por el federado B en la fase 2.

  En este caso ejecuta las acciones en la RTI y genera las *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permita almacenar ningún mensaje en la *Callback-Queue*).
- **LRC-Callback.** Su función es similar a la que realiza en el evento 3.

4.3.8.0.39. Funciones en la fase 2: evento 7. En este evento los componentes de Pórtico deben asegurar que el federado A actualiza el P/O publicado con los mismos datos que actualizó el federado B el P/O en el evento anterior.

Por lo tanto las actividades realizadas por cada componente de Pórtico durante este evento son:

- **LRC-Request.** Este componente comprueba que los mensajes que envía el federado a la RTI son para actualizar los necesarios para actualizar un P/O.
- **RTI-Request.** Este componente comprueba que los mensajes que recibe la RTI son para actualizar un P/O cuyo atributo Rol tiene el valor A.

  En este caso genera las acciones correspondientes y las almacena en la *Action-Queue*. En caso contrario no se permite que se almacenen esas acciones en la *Action-Queue*.
- **RTI-Action.** Este componente lee de la *Action-Queue* las acciones y comprueba que son las necesarias para actualizar al menos un P/O con los mismos datos que el federado B actualizo el P/O en el evento anterior.

En este caso ejecuta las acciones en la RTI, genera las *callback* correspondientes y las almacena en la *Callback-Queue*. En el caso que no se cumpla este mínimo bloquea la ejecución del IRM hasta que se cumpla. Esta acción de bloqueo impide que se pase al siguiente evento (no permita almacenar ningún mensaje en la *Callback-Queue*).

- **LRC-Callback.** Su función es similar a la que realiza en el evento 1.

4.3.8.0.40. Funciones en la fase 2: evento 8. Este evento se ejecuta cuando el valor del atributo Control fase2 toma el valor de Aprobado.

En este caso los componentes de Pórtico deben asegurar que el federado B despublica el P/O que publicó anteriormente.

En ese caso las acciones de los componentes de Pórtico para este evento son equivalentes a las del evento 5 del IRM C (las únicas diferencias es que se publica un P/O en vez de un evento y que el valor del atributo Rol es B).

4.3.8.0.41. Funciones en la fase 2: evento 9. Este evento se ejecuta cuando el atributo Control fase1 toma el valor Derogado (en caso contrario se ejecutaría el evento 4).

En consecuencia los componentes de Pórtico tienen como finalidad principal asegurar que el federado A despublica el P/O que publicó anteriormente.

En ese caso las acciones de los componentes de Pórtico para este evento son equivalentes a las del evento 5 del IRM C (las únicas diferencias es que se publica un P/O en vez de un evento y que el valor del atributo Rol es B).

Capítulo 5

Interoperabilidad entre simuladores constructivos y sistemas C2

El objetivo de los simuladores constructivos militares es recrear un espacio de batalla virtual mientras que el objetivo de los sistemas de información C2 es representar un espacio de batalla real. Por este motivo la interoperabilidad entre ambos tipos de sistemas puede ser tan útil para las fuerzas armadas actuales.

Un ejemplo del interés de la comunidad internacional en este tipo de interoperabilidad son los ejercicios organizados por la OTAN orientados a la convergencia entre simuladores constructivos federados siguiendo el estándar IEEE1516 y sistemas C2 que siguen el JC3IEDM ([65], [121], [122]).

Los resultados obtenidos hasta el momento en este tipo de ejercicios se han considerado insuficientes ya que la convergencia necesaria entre ambos tipos de sistemas es algo más que la simple trazabilidad entre esquemas diferentes de información conseguida hasta el momento ([58]).

Hay que tener en cuenta que cada tipo de sistema tiene un ámbito diferente, lo que se traduce en arquitecturas, necesidades y objetivos diferentes. Por lo tanto, hacer que converjan de manera total resulta prácticamente inviable. Sin embargo, en esta tesis doctoral se parte de la hipótesis de que esta convergencia sí es posible si se tienen en cuenta los puntos en común entre los dos dominios (simulación constructiva y mando y control) y sólo se persigue en casos de interoperabilidad concretos (figura 5.1).

En este capítulo se toman como punto de partida los casos de interoperabilidad

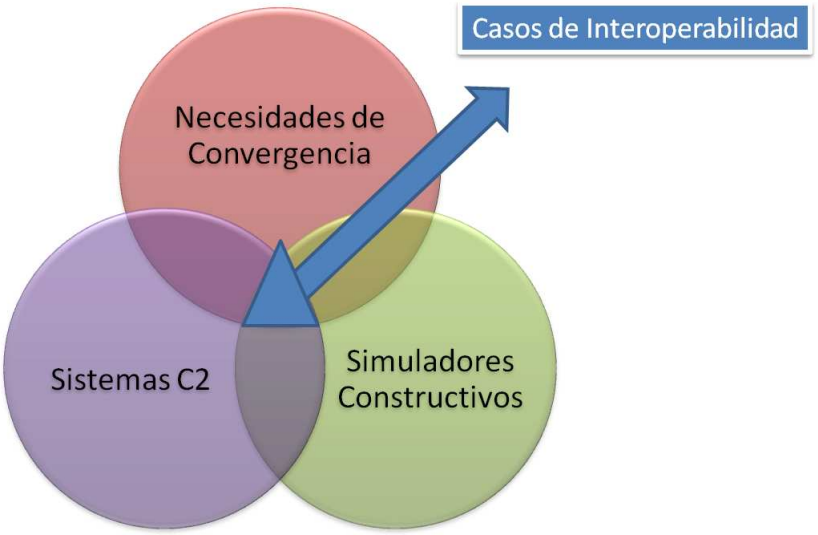

Figura 5.1: Representación de los casos de interoperabilidad entre dominios de sistemas

entre simuladores constructivos y sistemas C2 identificados como una necesidad actual para definir una arquitectura estándar que permita la convergencia parcial entre ambos tipos de sistema. Sobre esta arquitectura se proponen modelos de interoperabilidad de referencia que permitan que todas las fuerzas armadas que empleen la arquitectura definida, resuelvan los casos de interoperabilidad de la misma manera.

5.1. Antecedentes

5.1.1. Definiciones y conceptos básicos

Antes de comenzar con la discusión de cualquier otro concepto es necesario conocer dos definiciones básicas en el ámbito táctico-militar y que están directamente involucradas en el tipo de interoperabilidad que se pretende resolver en esta tesis doctoral:

- **Línea de acción**. Se define como un conjunto de acciones planeadas para que las lleve a cabo una determinada unidad militar ([68]). Desde el punto de vista de modelo JC3IEDM, se trata de una secuencia de *action-task* planificadas para ser realizadas por un *object-item* de tipo *organization*.
- **Matriz de evaluación**. Se define como una tabla que resume los resultados de la evaluación realizada a una determinada línea de acción. Para crear esta

Línea de Acción: Bravo

	Unidad A	Unidad B	Unidad C	Unidad D
Bajas humanas propias				
Bajas humanas enemigas				
Bajas carros de combate propios				
Gasto de munición 155mm				
Gasto de munición 5.56				
Gasto de combustible				
Longitud máxima logística				

Figura 5.2: Ejemplo de matriz de evaluación

matriz se definen los parámetros o magnitudes que se consideran de interés para medir la bondad de la línea de acción. A continuación, se identifica a las unidades militares que intervienen en esa línea. En la matriz, cada fila corresponde a uno de los criterios de evaluación y cada columna a una de estas unidades, de manera que en cada celda se escribe el resultado de evaluar un criterio para una unidad (ejemplo en la figura 5.2). Obviamente la manera óptima de construir estas matrices, en cuanto a velocidad y fiabilidad, es la simulación.

5.1.2. Casos de interoperabilidad identificados entre sistemas C2 y simuladores constructivos

La interoperabilidad entre dos o varios sistemas se puede concebir desde varios puntos de vista diferentes. Si dos sistemas pueden converger en todas la funcionalidades comunes a ellos se alcanza una interoperabilidad general. Este caso es el más completo pero muchas veces es inviable, técnica y económicamente.

Por ese motivo la interoperabilidad entre sistemas debe ser más selectiva acotando aquellas funcionalidades o casos donde es necesario que los sistemas converjan (interoperabilidad parcial). Es en esta situación en la que se definen los casos de interoperabilidad como aquellas situaciones donde es necesaria una convergencia entre los sistemas.

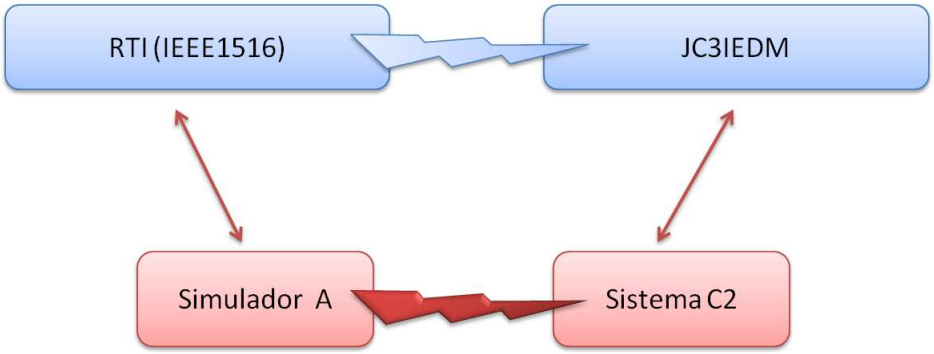

Figura 5.3: Esquema contextual

El contexto en el que se va a trabajar consiste en varios simuladores constructivos federados siguiendo el estándar IEEE1516 y un sistema C2 basado en el JC3IEDM (figura 5.3). En un escenario como este, los casos de interoperabilidad se pueden identificar tomando como referencia los siguientes puntos de vista:

- Desde el punto de vista de un simulador constructivo los sistemas C2 tienen la información que describe la situación actual del espacio de batalla. Esta situación describe un escenario que sería el punto de entrada para el inicio de una simulación constructiva. De esta manera la simulación estaría basada en un escenario real siendo la calidad de la misma superior a la creada artificialmente por el director de ejercicio si no existe interoperabilidad entre ambos tipos de sistemas.
- Desde el punto de vista de un sistema C2 los simuladores constructivos son un apoyo para evaluar las diferentes líneas de acción en la fase de planeamiento de una operación. De esta manera se obtendrían del simulador unas matrices de evaluación basadas en más parámetros de los que puede tener en cuenta la mente humana ([123]).

Los puntos de vista considerados están basados en la direccionalidad de la convergencia, es decir, en lo que necesita un tipo de sistema del otro. De ellos surgen directamente los casos de interoperabilidad que se tienen en cuenta en esta tesis doctoral para conseguir la convergencia parcial entre el dominio de la simulación constructiva y el del mando y control:

1. **Generación de un escenario real para un simulador constructivo.** Un escenario se puede definir como el punto de partida para un ejercicio de

simulación constructiva ([19], [29]). En la actualidad los escenarios para un ejercicio de simulación los crea un equipo de personas designado para tal fin ([34]).

Esto genera ciertos problemas que suelen concluir en una falta en la calidad de los ejercicios. El motivo es que estos equipos de simulación normalmente crean escenarios imaginarios y muchas veces (esto depende de la experiencia de los equipos) poco veraces o lejos de la realidad. Con lo que las unidades se entrenan en situaciones que no se parecen a las que se van a encontrar en la realidad y por lo tanto, poco útiles para mejorar sus habilidades.

La solución a este problema es realizar ejercicios de simulación basados en escenarios reales en los que las unidades que se están instruyendo van a participar. Como se puede observar la información requerida para la generación de los escenarios virtuales está en los sistemas C2 que se emplean para dirigir esa operación concreta.

Por este motivo la generación de un escenario virtual (basado en un escenario real) para su empleo en los simuladores constructivos se considera una necesidad de interoperabilidad entre ambos sistemas.

2. **Evaluación de las líneas de acción planeadas en un sistema C2.** Los sistemas de ayuda a la decisión son un punto de apoyo al jefe de cualquier organización ya que son capaces de tener en cuenta muchos más parámetros para evaluar una alternativa de los que la mente humana puede procesar ([124], [125]).

   Desde el punto de vista táctico-militar, en la fase de planeamiento de una operación se evalúan varias líneas de acción de tal manera que se selecciona la que mejor se adapte a las prioridades del mando de la unidad militar. El resultado de las evaluaciones se suele expresar en forma de matrices comparativas, las matrices de evaluación ([5], [68]) definidas en la sección anterior.

   Por este motivo la construcción de las matrices de evaluación se considera una necesidad de interoperabilidad entre ambos tipos de sistemas, ya que los simuladores constructivos pueden realizar la evaluación de las líneas de acción propuestas desde el sistema C2 de manera óptima, fiable y eficiente.

Figura 5.4: Convergencia directa entre simuladores constructivos y sistemas

5.1.3. Estándares asociados a los casos de interoperabilidad

Una vez definidas las situaciones o casos donde es necesario que converjan ambos tipos de sistemas es necesario estudiar los estándares que están asociados a los mismos y que son de aplicación a las situaciones descritas.

Los sistemas objeto de estudio de este capítulo se pueden definir como débilmente acoplados ya que las arquitecturas de ambos son muy diferentes. De hecho, la integración directa entre ellos no es posible (figura 5.4).

Los estándares asociados a ambos tipos de sistemas también tienen grandes e importantes diferencias. Pero deben ser el marco de referencia para la definición de una arquitectura de interoperabilidad común ([126]).

Hay que recordar que los estándares que resultan de interés para el desarrollo de esta tesis doctoral (todos ellos introducidos en capítulos anteriores) son:

- **Estándares asociados a simuladores constructivos:**
  - MSDL (Military Scenario Definition Language).
  - CBML (Coalition Battle Management Language).
  - HLA (High Level Architecture).
- **Estándares asociados a sistemas C2:**
  - Modelo de datos JC3IEDM.
  - DEM (Data Exchange Mechanism).

¿Cómo deberían tenerse en cuenta todos estos estándares en la definición de una nueva arquitectura de interoperabilidad? Como se ha analizado en la sección anterior, los dos casos de interoperabilidad que debe resolver esta nueva arquitectura

son dos, la generación de un escenario real (que provenga del sistema C2) para un simulador constructivo y la evaluación de las líneas de acción (por parte de uno o varios simuladores constructivos) como apoyo a la decisión del mando en un sistema C2.

En el primer caso de interoperabilidad, la información que llega al simulador constructivo para la generación del escenario debe seguir el estándar MSDL para la definición de escenarios de manera que pueda ser interpretada. El resto de estándares no tienen una aplicación tan directa a los casos de interoperabilidad, pero es imprescindible tenerlos en cuenta para hacer posible el intercambio de información entre ambos tipos de sistemas ([127], [128]).

HLA (estándar de arquitectura para los simuladores constructivos) y el modelo JC3IEDM (estándar de arquitectura para los sistemas C2) son tremendamente distintos tanto en su filosofía y concepción como en sus detalles de implementación a más bajo nivel.

El estándar CBML ha sido un primer paso en la convergencia de estos dos dominos que puede servir como punto de partida para esta tesis doctoral. De hecho uno de los objetivos de CBML es buscar una semántica común entre ambos tipos sistemas para un ámbito muy concreto, el de las órdenes de combate. Pero esta convergencia semántica no es suficiente para alcanzar una interoperabilidad real que resuelva los dos casos identificados, sólo puede servir como punto de partida.

En los dos casos de interoperabilidad que hay que resolver es necesario que los simuladores y los sistemas C2 intercambien distintos tipos de información. Pero la estructura de los datos que emplean no es compatible, HLA emplea un esquema basado en OMT con dependencia jerarquizada entre entidades y, por el contrario, el JC3IEDM emplea un esquema relacional. Otra diferencia principal relacionada con el intercambio de datos es que HLA emplea un sistema de suscripción/publicación y el JC3IEDM emplea un protocolo definido por MIP (DEM) y que difiere mucho del anterior. Además a todo esto se suma que el JC3IEDM impone unas restricciones a los datos intercambiados (*business rules*) muy numerosas y que definen una semántica común a los sistemas C2 que no cumplen los simuladores constructivos.

Todos estos retos son los que tiene que superar la arquitectura que se proponga en este capítulo.

5.2. Necesidades de interoperabilidad

Las necesidades de interoperabilidad son la especificación a un nivel inferior de abstracción, mucho más detallado, de los casos de interoperabilidad. Es decir, el desarrollo del estudio combinado de los casos de interoperabilidad y los estándares asociados a ellos dan como resultado aquellos aspectos necesarios para que los sistemas sean convergentes en los puntos especificados en los casos de interoperabilidad.

Se han identificado las siguientes necesidades de interoperabilidad, y es imprescindible que se resuelvan para obtener la convergencia parcial deseada:

- **Intercambio de ficheros.** Esta es la alternativa escogida en todos los casos para el paso de información entre los simuladores constructivos y los sistemas C2, por lo tanto debe garantizarse que es posible realizar intercambio de ficheros entre ambos tipos de sistemas en formatos estándar comprensibles para los dos. Por ejemplo, el estándar MSDL para generación de escenarios contempla que el escenario inicial sea intercambiado empleando ficheros con esquemas XSD ([19]).

- **Transformación de datos.** Los estándares asociados a cada tipo de sistema emplean una estructura de datos diferente. Para que la convergencia entre ambos sea posible cada sistema debe recibir la información estructurada en el formato que define el estándar empleado por él.

  Por ejemplo, si un sistema C2 envía una línea de acción lo hace con la estructura definida en el JC3IEDM, y sin embargo, el simulador constructivo que la recibe la espera recibir en el formato OMT definido por HLA.

- **Convergencia de protocolos.** La estructura de los datos convergente no es el único factor que influye para que se puedan enviar datos entre dos sistemas. El modo en el que se envían también es un aspecto que puede hacer que un sistema C2 y un simulador constructivo que intercambian datos no se entiendan entre ellos.

  Por lo tanto es necesaria una convergencia entre protocolos que permita que, por ejemplo, un sistema C2 envíe una línea de acción empleando el protocolo DEM y que ésta pueda ser recibida por un simulador constructivo que la espera recibir empleando el protocolo IEEE1516.

- **Centralización de esquemas.** Los estándares para intercambio de información asociados a los sistemas C2 y a los simuladores constructivos definen unas

estructuras que deben seguir los datos. Estas estructuras de datos deben estar centralizadas de tal manera que puedan ser empleadas como metadatos por las necesidades de interoperabilidad.

Por lo tanto es imprescindible crear un repositorio de esquemas, de esta manera los datos intercambiados entre los sistemas C2 y los simuladores constructivos siguen siempre uno de los esquemas almacenados en esta estructura centralizada.

- **Centralización de reglas.** El JC3IEDM especifica que los datos intercambiados entre sistemas, además de seguir un esquema, respeten un conjunto de reglas.

  Para que la gestión en el intercambio de esta información entre sistemas C2 y simuladores constructivos sea eficiente, de nuevo existe la necesidad de un almacén centralizado, el repositorio de reglas, que se apliquen a los esquemas de información definidos.

- **Centralización de procesos.** Muchos de los procesos necesarios para dar cobertura a los casos de interoperabilidad pueden ser parecidos entre ellos, o incluso estar repetidos. Para evitar la redundancia de los procesos y optimizar la resolución de los casos de interoperabilidad debe existir un control centralizado.

  Un repositorio de procesos permite evitar la redundancia, facilita la reutilización y la gestión óptima de las versiones.

- **Compatibilidad de procesos.** Los elementos que permiten resolver los casos de interoperabilidad pueden coexistir bajo diferentes versiones. Por lo tanto es necesaria la trazabilidad de las diferentes versiones que existen de todos los elementos de la arquitectura de interoperabilidad, y la definición de aquellos aspectos que pueden provocar incompatibilidades.

- **Centralización del estado.** Los procesos que dan cobertura a los casos de interoperabilidad generan información temporal que es necesaria para la ejecución de otros procesos.

  Por este motivo es necesario un repositorio central, el repositorio de estados, donde pueda ser almacenada de forma temporal toda esta información.

- **Núcleo básico.** Las funciones básicas que gestionan el proceso de convergencia entre sistemas C2 y simuladores constructivos son la base sobre la que

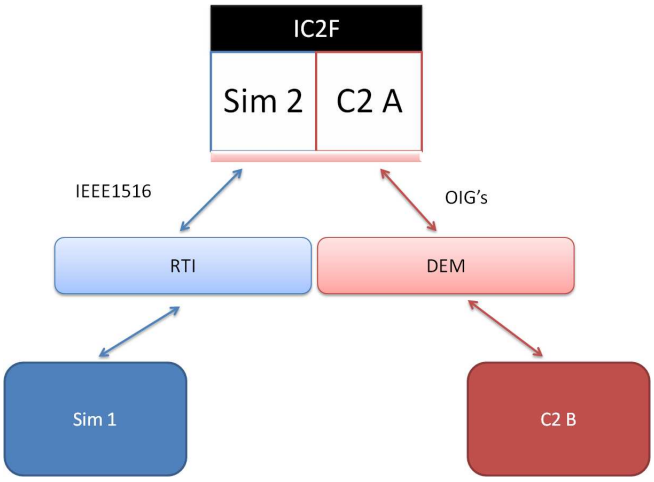

Figura 5.5: Esquema general de la IC2A

se construyen el resto de necesidades. Por ello es necesario definir toda esta funcionalidad bajo la denominación de núcleo básico, imprescindible para el funcionamiento de la arquitectura de compatibilidad.

Todas estas necesidades identificadas deben ser resueltas por la arquitectura de interoperabilidad propuesta si se desea la convergencia parcial entre sistemas C2 y simuladores constructivos que permita solucionar los dos casos de interoperabilidad considerados.

5.3. Definición de la arquitectura IC2A

La ausencia de arquitecturas estándar que traten la interoperabilidad entre la arquitectura HLA y la definida por JC3IEDM hace que sea necesario definir una nueva arquitectura que garantice su convergencia.

Esta nueva arquitectura debe tener como finalidad la interoperabilidad de los sistemas basados en IEEE1516 y los sistemas basados en JC3IEDM en las situaciones en que ésta sea requerida. Por este motivo los casos y necesidades de interoperabilidad identificados en las secciones anteriores son la base sobre la que debe estar construida la nueva arquitectura (figura 5.5).

La denominación seleccionada para esta nueva arquitectura de interoperabilidad es *Interoperability C2 Architecture* (IC2A). Y se compone de los siguientes elementos:

- **Sistemas de Simulación.** Sistemas de simulación constructiva que interoperan entre ellos bajo el estándar IEEE1516.
- **Sistemas C2.** Sistemas de mando y control (C2) que interoperan entre ellos empleando el estándar JC3IEDM.
- **IC2F (Interoperability C2 Framework).** Elemento que permite la comunicación entre los simuladores constructivos y los sistemas C2.

El IC2F tiene como finalidad permitir la convergencia parcial (sólo en los dos casos de interoperabilidad identificados) entre los sistemas que participan en la IC2A. Uno de los objetivos en su diseño es que permita un fácil escalado para incluir nuevos casos de interoperabilidad que puedan ser de interés para las fuerzas armadas en el futuro.

El IC2F está compuesto a su vez por dos elementos:

- **Interoperability C2 Interfaces (IC2I).** Son las interfaces a través de las cuales los sistemas C2 y los sistemas de simulación constructiva se comunican con el IC2F.
- **Interoperability C2 Data (IC2D).** Es el componente que define las estructuras de datos con las que puede trabajar el IC2F.

5.3.1. Reglas para la definición de la IC2A

Para definir la nueva arquitectura de interoperabilidad es necesario proponer unas reglas básicas que formen el marco conceptual de funcionamiento.

Existen dos conceptos básicos a partir de los cuales se definen las reglas para los elementos participantes en un escenario de interoperabilidad:

- **Teatro.** Es una agrupación lógica de varios sistemas identificada con un mismo nombre. Los sistemas que interoperan bajo un mismo teatro tienen un fin común que no es compartido por otros sistemas que no están unidos a él.
- **Actor.** Es un sistema C2 o de simulación constructiva que puede pertenecer a uno o varios teatros.

Un teatro o un actor que formen parte de una arquitectura IC2A concreta deben cumplir las siguientes reglas:

- **Teatro:**

  1. Una IC2A concreta puede tener definido sólo un IC2D.
  2. Una IC2A puede tener varios teatros definidos de manera simultánea.
  3. Un teatro debe tener como mínimo un actor (un actor de tipo simulador constructivo o de tipo sistema C2).
  4. Un teatro pude tener publicadas varias líneas de acción simultáneamente.
  5. Un teatro puede tener publicados varios escenarios simultáneamente.
  6. Un teatro puede tener publicadas varias matrices de evaluación simultáneamente.
  7. Sólo los actores que pertenezcan a un teatro concreto pueden intercambiar información entre ellos.
  8. Sólo puede destruir un teatro el sistema (actor) que lo creó.

- **Actor:**

  1. Un actor puede pertenecer a varios teatros de manera simultánea.
  2. Un actor puede tener un sólo rol de tipo simulador constructivo o sistema C2.
  3. Para publicar una línea de acción, una matriz de evaluación o un escenario en un teatro el actor debe pertenecer a él.
  4. Un actor que publica una línea de acción debe tener el rol de sistema C2.
  5. Un actor puede tener publicadas varias líneas de acción de forma simultánea en uno o varios teatros.
  6. Un actor que publica una matriz de evaluación (asociada a una línea de acción) debe tener el rol de simulador constructivo.
  7. Un actor puede publicar varias matrices de evaluación asociadas a una misma línea de acción.
  8. Un actor puede publicar una matriz de evaluación asociada a una misma línea de acción publicada en varios teatros.
  9. Un actor puede tener publicadas varias matrices de evaluación de forma simultánea en varios teatros.
  10. Un actor que publica un escenario debe tener el rol de sistema C2.
  11. Un actor puede tener publicados varios escenarios de forma simultánea en varios teatros.

5.3.2. Definición de las plantillas de datos IC2D

El IC2D proporciona a los procesos que componen el IC2F un conocimiento *a priori* de las estructuras de datos que pueden manejar. Teniendo en cuenta las necesidades de interoperabilidad identificadas es imprescindible que este componente de la IC2A se encargue de las siguientes funcionalidades, clasificadas en siete grupos:

1. **Definición de ficheros.** Define la estructura de los ficheros que pueden ser empleados por los elementos que componen la IC2A.
2. **Definición de protocolos.** Define la información necesaria para identificar y utilizar los protocolos empleados por los sistemas C2 y sistemas de simulación constructiva para la IC2A concreta.
3. **Definición de esquemas.** Define la estructura de la información que se va a intercambiar entre los elementos que componen la IC2A.
4. **Definición de reglas de negocio.** Define las reglas de negocio que afectan a los datos que gestionan cada uno de los elementos que componen la IC2A en los casos de interoperabilidad definidos.
5. **Definición de procesos.** Define la estructura de información que especifica los diferentes procesos del IC2F.
6. **Definición de estados.** Define la estructura de la información de estado que puede ser empleada en una IC2A concreta.
7. **Definición de cambios.** Define la estructura y sistema de registro de la versión de los diferentes elementos que componen un IC2F.

El contenido de cada uno de los grupos puede cambiar su composición durante la ejecución de una IC2A concreta. Por este motivo la composición inicial del IC2D no es fija y puede variar, aunque bien es cierto que es único para toda la IC2A en ejecución.

5.3.2.1. Definición de ficheros

Como se analizaba con anterioridad, la alternativa escogida para enviar información entre simuladores constructivos y sistemas C2 es el intercambio de ficheros. La estructura de estos ficheros debe ser reconocida tanto por los elementos que la envían como por los que la reciben.

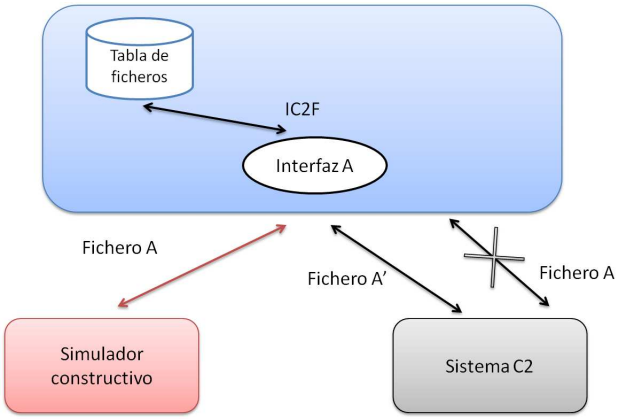

Figura 5.6: Esquema de envío de ficheros entre componentes de la IC2A

Para lograr este objetivo la estructura de los ficheros debe estar definida y difundida a los elementos que forman una IC2A. De esta manera cada sistema puede enviar y recibir ficheros que tienen una estructura aceptada por la arquitectura de interoperabilidad (figura 5.6).

Para ello se ha optado por que los ficheros estén definidos como una entrada en una tabla de definición de ficheros con el siguiente formato:

- Identificador único del fichero.
- Nombre del fichero.
- Tipo de sistema que lo envía/recibe (simulador constructivo/sistema C2/ambos).
- Identificador único del esquema de datos que sigue el fichero (en una tabla de definición de esquemas que se explicará un poco más adelante).

5.3.2.1.1. Ejemplo de definición de ficheros. En este ejemplo se define una IC2A que permite realizar el intercambio de la información de las líneas de acción empleando ficheros para ello.

Por lo tanto en la tabla de definición de ficheros del IC2D hay que crear la siguiente entrada:

- Identificador único: F1.
- Nombre fichero: Líneas de acción.

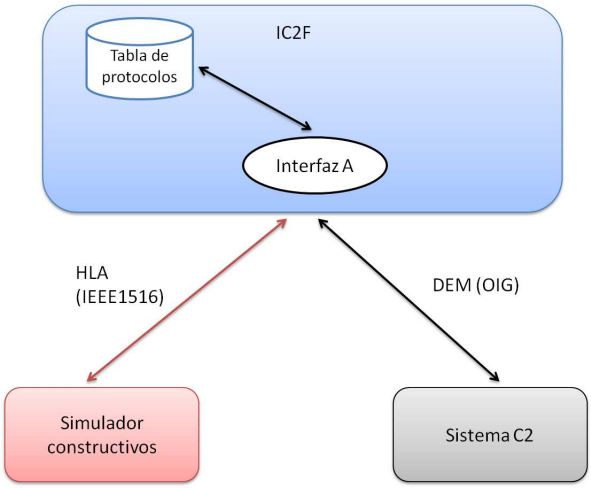

Figura 5.7: Esquema de la convergencia de protocolos de intercambio de información entre componentes de la IC2A

- Sistema que lo envía/recibe: Sistema C2/Simulador constructivo.
- Identificador único de esquema: E34.

Esta entrada significa que existe un fichero cuyo identificador es F1 y que tiene un esquema identificado como E34 en la tabla de definición de esquemas del IC2D.

Este fichero se llama Líneas de acción, lo envían los sistemas C2 y lo reciben los simuladores constructivos.

5.3.2.2. Definición de protocolos

Los diferentes elementos que componen una IC2A emplean una serie de protocolos específicos para el intercambio de información. Estos protocolos especifican cómo se debe enviar y recibir la información, qué estructura debe seguir, que normas debe cumplir, etc.

Pero para que los elementos de la arquitectura de interoperabilidad intercambien información entre ellos lo deben hacer del mismo modo, es decir, siguiendo el mismo protocolo. Por este motivo es necesario que se definan los protocolos que va a emplear cada uno de los elementos de tal manera que se gestionen de manera correcta las posibles diferencias (figura 5.7).

Se ha optado por utilizar una tabla de definición de protocolos en la que cada entrada almacena la siguiente información:

- Identificador único del protocolo.
- Nombre del protocolo.
- Tipo de sistema que lo utiliza para enviar/recibir (simulador constructivo/sistema C2/ambos).

5.3.2.2.1. Ejemplo de definición de protocolos. El protocolo definido por el JC3IEDM para el intercambio de información entre sistemas C2 es DEM. Y el empleado por los simuladores constructivos para el intercambio de información entre ellos es HLA.

Por lo tanto, la tabla de definición de protocolos del IC2D en un teatro con los dos tipos de actores incluye, como mínimo, estas dos entradas:

1. Entrada del protocolo DEM.
   - Identificador único: P1.
   - Nombre Protocolo: DEM.
   - Sistema que lo utiliza para enviar/recibir: Sistema C2/Sistema C2.
2. Entrada del protocolo HLA.
   - Identificador único: P2.
   - Nombre Protocolo: HLA.
   - Sistema que lo utiliza para enviar/recibir: Simulador constructivo/Simulador constructivo.

5.3.2.2.3. Definición de esquemas

Todos los esquemas utilizados en la IC2A se almacenan en una estructura centralizada, el repositorio o almacén de esquemas, y estos esquemas deben estar definidos previamente en la tabla de definición de esquemas en la que cada entrada almacena la siguiente información:

- Identificador único del esquema.
- Nombre del esquema.

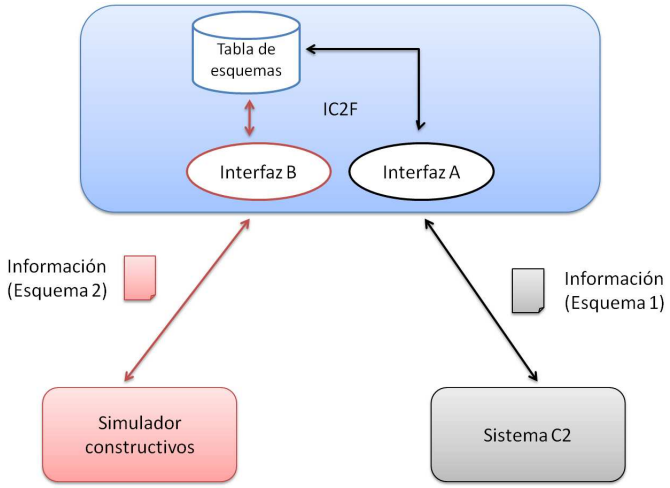

Figura 5.8: Esquema del intercambio de información basado en diferentes esquemas entre componentes del IC2A

- Tipo de sistema que lo envía/recibe (simulador constructivo / sistema C2 / ambos).
- Identificador de la regla de negocio que puede afectar a la estructura de datos (en el caso que se vea afectada por alguna).
- Estructura de información para ese esquema (definida en un formato estándar, por ejemplo, XSD).

La información intercambiada entre los diferentes elementos que componen una IC2A debe estar estructurada según uno de los esquemas registrado en esta tabla. Si alguna información no respeta esta estructura, no es aceptada por la IC2A.

Por lo tanto, cualquier componente de la IC2A debe poder consultar esta estructura. Para ello se deben utilizar protocolos de descubrimiento de esquemas almacenados, cada implementación concreta de la arquitectura seleccionará el más adecuado para sus necesidades o incluso implementará el suyo propio (figura 5.8).

Como recomendación, se especifica como estándar por defecto el NDMS (NATO Discovery Metadata Specification, [129]) desarrollado para el descubrimiento de metainformación (esquemas de información) a través de repositorios de esquemas (figura 5.9).

5.3.2.3.1. Ejemplo de definición de esquemas. En este ejemplo, continuando con el que se desarrolló para la definición de ficheros, se especifica el esquema

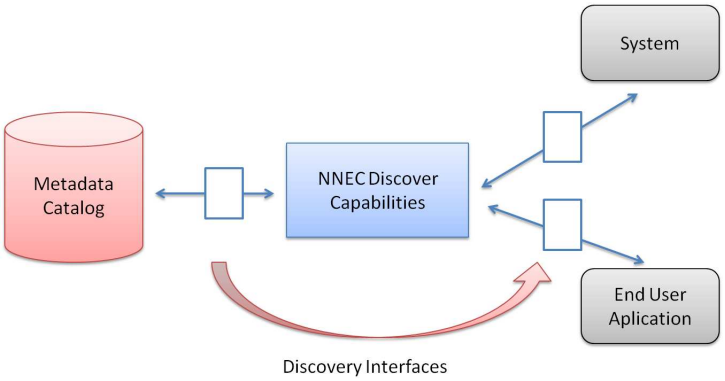

Figura 5.9: Esquema del estándar NDMS

que sigue la información que define una línea de acción.

Se crea la siguiente entrada en la tabla definición de esquemas del IC2D:

- Identificador único del esquema: E34
- Nombre del esquema: Esquema líneas de acción.
- Sistema que lo envía/recibe: Sistema C2/Simulador constructivo.
- Regla de negocio: R23.
- Esquema: Fichero *"Esquema E34. XSD"* cuyo contenido es el diagrama representado en la figura 5.10.

5.3.2.4. Definición de reglas de negocio

Los estándares que deben cumplir los elementos participantes en la arquitectura IC2A exigen que los datos cumplan unas restricciones o reglas de negocio predefinidas de manera determinista (no caben la subjetividad ni las interpretaciones). Por lo tanto, es necesario que la información intercambiada entre los componentes de la IC2A, además de cumplir un esquema definido, cumpla las reglas de negocio asociadas a ese esquema concreto (figura 5.11).

Desde el punto de vista de la IC2A es importante identificar qué tipo de componente de la misma puede verse afectado por una regla de negocio determinada. Las reglas de negocio pueden ser:

- Reglas comunes a los sistemas C2 y a los simuladores constructivos, y que afectan a la información que van a intercambiar.

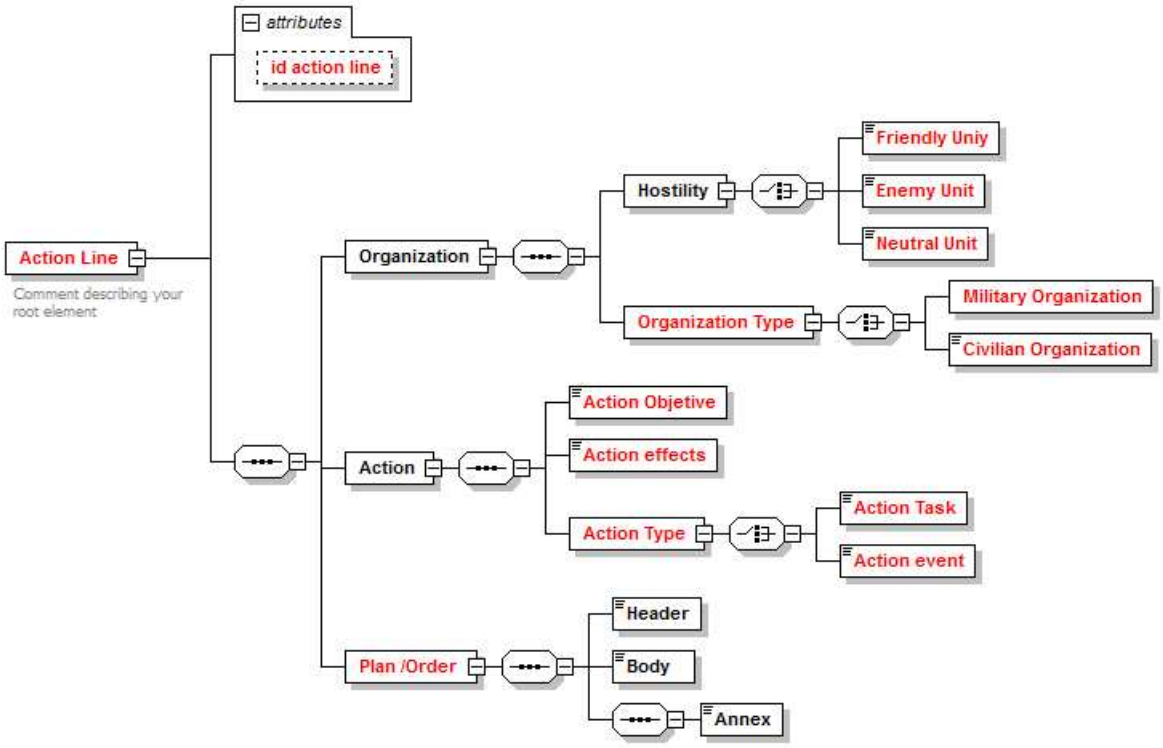

Figura 5.10: Ejemplo de esquema en formato XSD de la información de una línea de acción

- Reglas que sólo afectan a un tipo de sistema pero que tienen repercusiones en la información que se va a intercambiar.

- Reglas que no afectan directamente a la información que se va a intercambiar entre ambos tipos de sistemas, pero afectan a una tercera regla que sí afecta a la información que se va a intercambiar.

Sea cual sea el tipo de regla, es imprescindible que se defina de manera determinista para evitar fuentes de error, de ahí que la elección de un lenguaje de definición de reglas de negocio sea un factor crítico en la propuesta de la IC2A.

Se ha optado por recomendar por defecto el lenguaje OCL (Object Constraint Language)([130]), muy extendido en este tipo de contexto y empleado por el grupo de trabajo MDA (Model Driven Architecture) del programa MIP ([92]). Este lenguaje expresa las reglas de negocio con una filosofía orientada a objetos, en un formato que es de fácil validación y que se puede almacenar e intercambiar empleando XML.

Las reglas de negocio empleadas dentro de una IC2A concreta quedan reflejadas

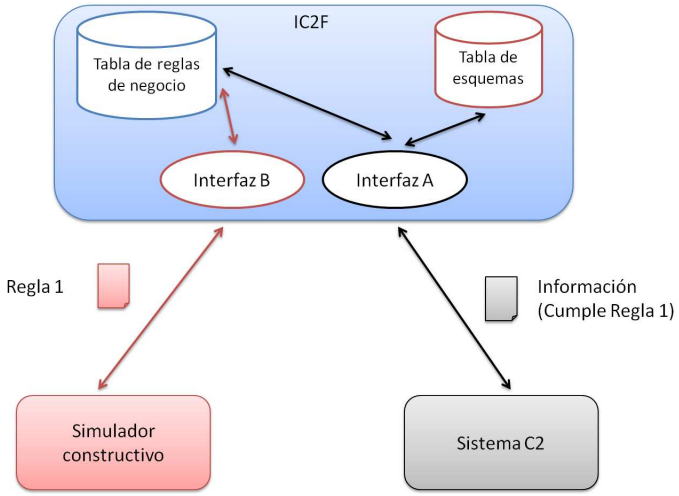

Figura 5.11: Esquema de la centralización de las reglas de negocio

en la tabla de definición de reglas de negocio con los siguientes campos:

- Identificador único de la regla de negocio.
- Nombre de la regla de negocio.
- Lenguaje de definición de la regla de negocio (debido a lo crítico que es este factor, se podría llegar a convenir una colección de lenguajes permitidos para una IC2A concreta).
- Entidad a la que afecta la regla de negocio (simulador constructivo/sistema C2/ambos sistemas/otra regla de negocio).
- (Opcional) En el caso que el valor de la columna anterior sea regla de negocio este campo permite identificar el identificador de la regla de negocio afectada.
- Definición de la regla de negocio en el lenguaje seleccionado.

5.3.2.4.1. Ejemplo de definición de reglas de negocio. Completando el ejemplo de la línea de acción de las secciones anteriores, el esquema definido para ella lleva asociada una regla de negocio que afecta a la información del mismo.

En este caso la regla de negocio afecta al elemento del esquema *actioneffect*. Los efectos de la acción llevan anexo un código que los describe, y su definición es obligatoria.

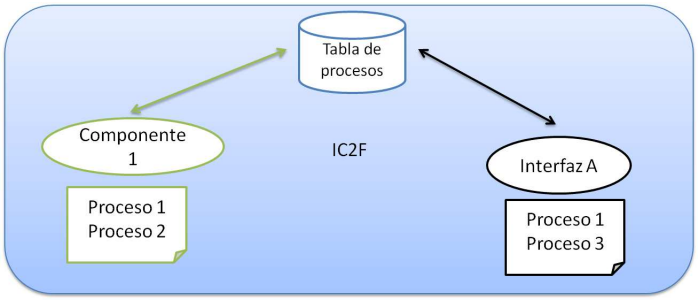

Figura 5.12: Esquema del almacenamiento de los procesos

Por lo tanto en la tabla definición de reglas de negocio del IC2D se debe crear la siguiente entrada:

- Identificador único de la regla de negocio: R23
- Nombre de la regla de negocio: Descripción del *actioneffect*.
- Lenguaje de definición de la regla: OCL.
- Entidad a la que afecta la regla de negocio: Ambos sistemas.
- En blanco (ya que no afecta a otra regla de negocio).
- Definición de la regla de negocio en el lenguaje seleccionado: "inv: not actioneffects.descriptionCode.oclIsUndefined()".

En este caso de define una regla en el lenguaje OCL que impide que el atributo *descriptionCode* de la entidad *actioneffects* sea nulo.

5.3.2.5. Definición de procesos

Los procesos que intervienen en el IC2F definido deben cumplir el principio de reutilización y evitar redundancias y duplicidades. Por este motivo es necesario un repositorio en el que se centralice la definición de todos los procesos que intervienen en un IC2F concreto (figura 5.12).

La tabla empleada para la definición de los procesos en una IC2A incluye los siguientes campos de información en cada entrada:

- Identificador único del proceso.

- Nombre del proceso.
- Descripción del proceso.
- Ficheros del proceso, expresados en uno o varios lenguajes de programación.

5.3.2.5.1. Ejemplo de definición de procesos. En este ejemplo se considera el proceso que se encarga de realizar la comprobación de que la información que envía uno de los sistemas de la IC2A respeta, al menos, uno de los esquemas definidos previamente para ella.

En este caso hay que crear en la tabla de definición de procesos del IC2D la siguiente entrada:

- Identificador único del proceso: P56.
- Nombre del proceso: Schema checking.
- Descripción del proceso: Este proceso comprueba si la información enviada por un sistema de la IC2A cumple, al menos, uno de los esquemas permitidos.
- Ficheros: Squema checking.cs.

5.3.2.6. Definición de estados

En ocasiones es necesario que algunos elementos de una IC2A tengan conocimiento de la información temporal que es generada por otros elementos de la arquitectura. Toda esta información temporal que se genera dentro de la IC2A, y que es necesario que sea compartida, debe estar gestionada de manera centralizada y su esquema de información registrado (figura 5.13).

La tabla de definición de estados está compuesta por entradas que contienen la siguiente información:

- Identificador único del estado.
- Nombre del estado.
- Descripción del estado.
- Identificador del esquema de la información de estado.

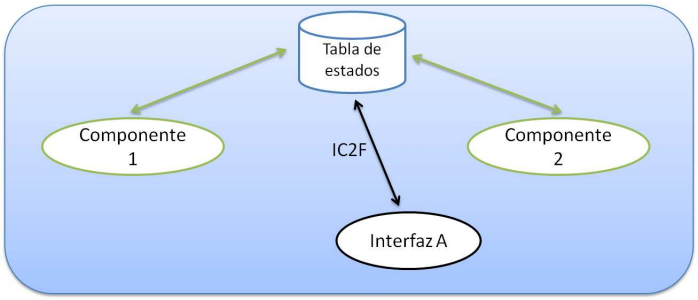

Figura 5.13: Esquema del almacenamiento de los estados de la información intercambiada

5.3.2.6.1. Ejemplo de definición de estados. Mientras un proceso transforma la información de una línea de acción de una estructura C2 a otra entendible por un simulador constructivo, la información del estado en el que se encuentra esa transformación puede ser necesaria para otro proceso.

Para definir la estructura de la información que define la situación en la que se encuentra la transformación de la línea de acción de una estructura a otra hay que crear la siguiente entrada en la tabla de definición de estados del IC2D:

- Identificador único del estado: ET12.
- Nombre del estado: Estado de transformación de la línea de acción.
- Descripción del estado: Información temporal acerca de la situación en la que se encuentra la transformación de una línea de acción.
- Identificador del esquema de la información de estado: E42 (representado para este ejemplo en la figura 5.14).

5.3.2.7. Definición de cambios

Los componentes que forman parte del IC2F pueden coexistir en diferentes versiones. Es conveniente registrar todas las versiones existentes de un mismo elemento en un repositorio centralizado cuya consulta facilite el mantenimiento y la gestión del IC2F (figura 5.15), además de permitir una compatibilidad hacia atrás de la IC2A definida.

Cada entrada de la tabla de definición de cambios almacena la siguiente información:

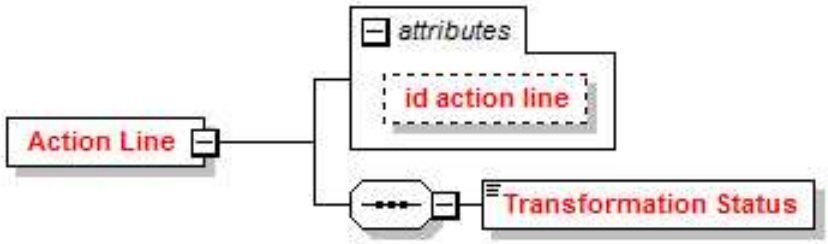

Figura 5.14: Ejemplo de esquema de la información de estado de transformación de una línea de acción

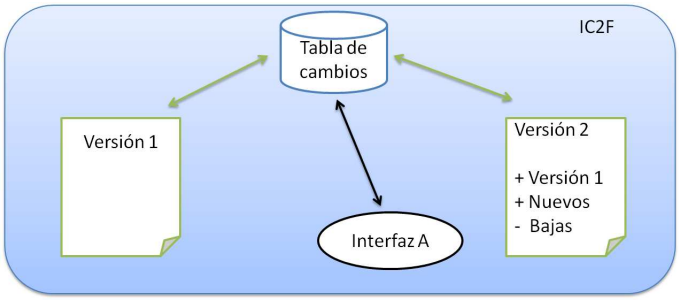

Figura 5.15: Esquema del almacenamiento de los cambios

- Identificador del cambio.
- Elemento afectado.
- Version del cambio.
- Fecha del cambio.
- Descripción del cambio.

5.3.2.7.1. Ejemplo de definición de cambios. En este ejemplo el esquema de la información de las matrices de evaluación es actualizado a una nueva versión que es necesario que coexista con la versión anterior por aspectos de compatibilidad.

Se introduce la siguiente entrada en la tabla de definición de cambios:

- Identificador del cambio: C1
- Elemento afectado: E22.

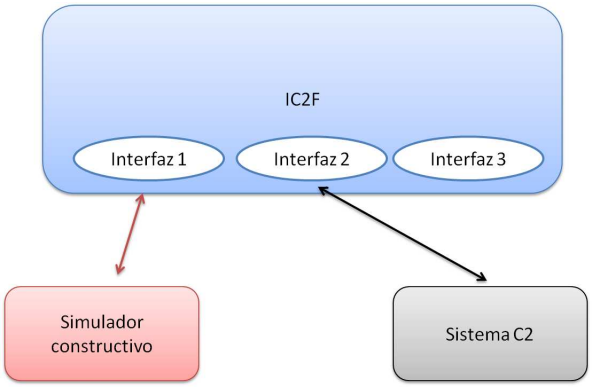

Figura 5.16: Esquema de las interfaces del IC2F

- Version del cambio: 2.
- Fecha del cambio: 12/03/2011.
- Descripción del cambio: Se añade a las matrices de evaluación un nuevo criterio denominado *Tiempo Requerido* que permita tener en cuenta en las decisiones una estimación del tiempo necesario para completar una línea de acción.

5.3.3. Interfaces de la IC2A

La arquitectura IC2A define el IC2F como el elemento intermedio a través del cual se comunican los sistemas de simulación constructiva y los sistemas C2. Los elementos pertenecientes al IC2F y que permiten la comunicación entre este y los sistemas C2 o simuladores constructivos son los interfaces (figura 5.16).

Los módulos en los que se divide el IC2F están compuestos por varias interfaces que le dan la funcionalidad definida para él. De esta manera el IC2F está compuesto por los siguientes módulos:

- **Módulo base del IC2F.**
- **Módulo de generación de escenarios.**
- **Módulo de generación de líneas de acción.**
- **Módulo de evaluación de líneas de acción.**

La arquitectura IC2A no impone que un IC2F concreto tenga que tener implementados todos los módulos de interfaces. De hecho, dependiendo de la finalidad de

la IC2A definida el IC2F puede tener sólo algunos módulos implementados. Teniendo en cuenta este criterio se pueden clasificar los IC2F como:

- **IC2F generales.** Implementan todos los módulos de interfaces definidos en la especificación.
- **IC2F específicos**. Implementan sólo alguno de estos módulos (los necesarios para resolver la interoperabilidad en el contexto específico para el que se utiliza), pero no el conjunto completo.

Lo que hay que aclarar es que la especificación de la arquitectura IC2A no concibe implementaciones parciales de los módulos que contengan sólo algunos de los interfaces. Si un IC2F tiene implementado un módulo es porque dispone de todos los interfaces definidos para ese módulo. De esta manera la especificación, comprensión, desarrollo y mantenimiento de un IC2F es mucho más sencilla.

5.3.3.1. Módulo base del IC2F

El módulo base es el único módulo que es obligatorio que exista en cualquier IC2F definido. Los interfaces pertenecientes a este módulo tienen como finalidad la gestión de la información de configuración de la IC2A.

La información de configuración de una IC2A concreta va desde la definición de los teatros y sus actores, hasta la información que contiene el IC2D.

Hay que especificar que los interfaces definidos en este módulo no pueden modificar ni eliminar la información contenida inicialmente en el IC2D (sólo pueden ampliarla). Esto es debido a que la información inicial del IC2D se considera la base de una IC2A definida y modificarla puede variar la finalidad de la misma completamente.

Este módulo incluye las siguientes funciones:

- **Creación/Destrucción de un teatro.** Este interfaz crea o destruye un teatro en el IC2F al cual se pueden unir los actores interesados en interoperar bajo una misma nomenclatura.

  El actor que ejecute este interfaz queda automáticamente unido al teatro que crea. Esto se debe a que si un actor crea un teatro es porque tiene una necesidad de interoperabilidad y por lo tanto tiene que pertenecer a él.

  Las únicas restricciones que existen es que no pueden existir dos teatros con el mismo nombre y un teatro sólo puede ser destruido por el actor que lo creó.

- **Descubrimiento de teatros.** Este interfaz devuelve un listado con el nombre de todos los teatros que hay creados actualmente en el IC2F.

- **Alta/Baja de actores en un teatro.** Este interfaz permite a un actor unirse o darse de baja en un teatro previamente creado en un IC2F.

  Es necesario para que dos o más actores interoperen que estén unidos a un mismo teatro, en caso contrario no podrían comunicarse entre ellos.

  Todo teatro tiene asociado como mínimo un actor, el que lo ha creado, y que no puede darse de baja de él ya que implicaría su destrucción. Por lo tanto el actor que crea un teatro no ejecuta este interfaz, ya que no hace falta que se dé de alta y no puede darse de baja.

- **Alta de nuevos ficheros.** El IC2D contiene la definición de las estructuras que pueden tener los ficheros intercambiados en la IC2A.

  Este conjunto inicial de tipos de ficheros contenidos en el IC2D puede ser ampliado. Por este motivo este interfaz permite a un actor añadir un nuevo tipo de fichero al IC2D.

  Al emplear este interfaz hay que tener en cuenta que a partir de ese momento la IC2A acepta ese tipo de fichero para todos sus teatros.

- **Alta de nuevos protocolos.** De igual manera, este interfaz permite añadir un nuevo protocolo al conjunto inicial definido en el IC2D. Cualquier protocolo añadido a la arquitectura se podrá utilizar en cualquiera de los teatros existentes en la IC2A.

- **Alta de nuevos esquemas.** El IC2D contiene la definición de los esquemas en los que puede ser intercambiada la información entre los elementos de la IC2A.

  Este conjunto inicial de esquemas contenidos en el IC2D puede ser ampliado. Por este motivo este interfaz permite a un actor añadir un nuevo esquema al IC2D.

  Al emplear este interfaz hay que tener en cuenta que a partir de ese momento la IC2A acepta ese tipo de esquema para todos sus teatros.

- **Alta de reglas de negocio.** Este interfaz permite a un actor dar de alta nuevas reglas de negocio en el IC2D.

Al igual que pasa con los ficheros, protocolos y esquemas, a partir del momento en el que se añade una nueva regla de negocio a la arquitectura, ésta es aceptada por la IC2A en todos sus teatros.

- **Alta de procesos.** Mediante este interfaz un actor puede añadir un nuevo proceso al IC2D. Estos procesos nuevos pueden ser versiones mejoradas de los que había anteriormente o ser creados desde cero para incorporar una nueva función a la arquitectura.

- **Gestión de estados.** Un actor que ejecute este interfaz puede dar de alta o de baja, o leer, la información de estado publicada por otro actor que pertenece a su mismo teatro.

  Mediante este interfaz los actores pueden gestionar e intercambiar la información temporal que sea necesario compartir en cada momento.

- **Alta de versiones.** Este interfaz permite a un actor notificar un cambio, es decir, añadir una nueva versión de un elemento al IC2D.

- **Lectura de la información del IC2D.** Este interfaz permite la lectura de la información que está contenida actualmente en el IC2D.

5.3.3.2. Módulo de generación de escenarios

El intercambio de escenarios entre los actores de un mismo teatro es un requerimiento fundamental para que la arquitectura IC2A permita resolver los dos casos de interoperabilidad identificados al principio de este capítulo.

Este módulo incluye todos los interfaces que son necesarios para realizar una correcta gestión de los escenarios por parte de los actores que pertenecen a un mismo teatro.

Un escenario puede ser compartido en varios teatros pero debe ser declarado de forma explícita en todos ellos. Es decir, el actor que quiera compartir un escenario en varios teatros lo debe publicar en todos para que los actores que pertenecen a cada uno de ellos puedan ejecutar los interfaces de este módulo sobre ese escenario.

Los interfaces que componen este módulo son los siguientes:

- **Publicación de escenario.** Mediante este interfaz un actor comparte la información de un escenario concreto con todos los actores que pertenecen al teatro donde se ha publicado el escenario.

Es evidente que el escenario publicado debe cumplir la estructura y reglas de negocio definidas en el IC2D, por lo tanto, será necesario validarlo siempre antes de publicarlo.

- **Lectura de un escenario.** Mediante este interfaz un actor puede obtener la información publicada en su teatro de un escenario concreto.

- **Despublicación de escenario.** El actor que ejecuta este interfaz sobre un escenario debe ser el actor que lo publicó. De esta manera el actor deja de compartir el escenario con el resto de actores de ese teatro.

- **Modificación de un escenario.** El actor que publicó un escenario puede querer, en un momento determinado, hacer modificaciones sobre el mismo de tal manera que refleje los cambios que se hayan producido respecto de la situación inicial.

  Este interfaz permite la publicación de escenarios que son una versión posterior al inicialmente publicado. De esta manera se pueden tener varias versiones de un mismo escenario.

  En ningún caso este interfaz borra un escenario anterior ya que pueden existir diferentes actores trabajando con versiones distintas de un mismo escenario.

- **Validación de un escenario.** Antes de publicar un escenario es necesario comprobar que cumple las restricciones de la IC2A. Para validar el escenario basta con comprobar que:

  - El esquema de la información del escenario sigue un esquema definido en la IC2A (tabla definición de esquemas IC2D).
  - El tipo de fichero del escenario, en el caso en el que sea necesario, está admitido en la IC2A (tabla de definición de ficheros IC2D).
  - Cumple las reglas de negocio incluidas en la tabla de definición de reglas de negocio del IC2D.
  - Los protocolos empleados para su envío y recepción están especificados en la tabla de definición de protocolos del IC2D.

- **Descubrimiento escenarios.** Mediante este interfaz se obtiene un listado de los escenarios publicados en un teatro específico.

5.3.3.3. Módulo de generación de líneas de acción

Para resolver el segundo caso de interoperabilidad es necesario que los actores pertenecientes a un teatro tengan la capacidad de gestionar las líneas de acción relacionadas con él. Este módulo incluye todos los interfaces necesarios para llevar a cabo estas tareas, entendiendo como línea de acción el planeamiento de una maniobra futura en una situación de combate. Los interfaces asociados a la evaluación de la línea se dejan para el siguiente módulo.

Las líneas de acción deben estar desarrolladas sobre un escenario publicado, al menos, en un teatro. Y cualquiera de los actores del teatro pueden publicar una línea de acción sobre un escenario previamente publicado.

Los interfaces definidos para este módulo son:

- **Publicación de una línea de acción.** Este interfaz permite a un actor compartir una línea de acción (basada en un escenario ya publicado) con el resto de actores de ese teatro.

- **Despublicación de una línea de acción.** Mediante este interfaz un actor deja de de compartir la línea de acción con el resto de actores de un teatro. La línea de acción sólo puede ser despublicada por el actor que la publicó.

- **Lectura de una línea de acción.** Este interfaz permite a un actor perteneciente a un teatro obtener la información sobre una línea de acción que ha sido publicada por otro actor en ese teatro (esta información incluye la especificación de la matriz de evaluación asignada a esa línea de acción y los escenarios asociados a ella).

- **Suscripción a una línea de acción.** Un actor que se suscribe a una línea de acción comunica al teatro su intención de evaluar la línea de acción. De esta manera el resto de actores del teatro conocen los actores que van a evaluarla.

- **Especificación de la matriz de evaluación.** Mediante el empleo de este interfaz el actor que publica la línea de acción especifica el esquema o esquemas de la matrices de evaluación que quiere que le sean devueltas.

  Hay que tener en cuenta que si el actor que pretende suscribirse a una línea de acción no puede devolver una matriz de evaluación con el esquema definido entonces, no podrá suscribirse a esa línea, ya que no podrá evaluarla según las necesidades del actor que la publica.

- **Asignación de una línea de acción a un escenario.** Mediante este interfaz un actor asigna un escenario publicado (por él mismo u otro actor en ese teatro) a una línea de acción, ya que ésta carece de significado sin esta asociación. En ningún caso se puede evaluar una línea de acción que no está asignada a ningún escenario.

  Este interfaz facilita la reutilización de líneas, ya que permite realizar de manea sencilla diferentes evaluaciones de la misma línea pero sobre diferentes escenarios.

- **Validación de una línea de acción.** Mediante este interfaz el actor que publica una línea de acción valida que lo hace conforme a lo definido en el IC2D. En este caso se debe comprobar que:

  - El esquema de datos de la línea de acción esta definido en la tabla de definición de esquemas del IC2D.
  - El esquema de datos de la matriz de evaluación solicitada está definido también en esa misma tabla.
  - El tipo de fichero de la línea de acción, en el caso en el que sea necesario, está definido en la tabla de definición de ficheros del IC2D.
  - Se cumplen las reglas de negocio definidas para los esquemas antes mencionados y que se incluyen en la tabla de definición de reglas de negocio del IC2D.
  - Los protocolos empleados para el envío y recepción de la línea están soportados por la arquitectura, ya que se incluyen en la tabla de definición de protocolos del IC2D.

- **Descubrimiento de líneas de acción.** Este interfaz devuelve un listado de las líneas de acción que hay publicadas en un teatro o de las que están asignadas a un escenario concreto.

5.3.3.4. Módulo de evaluación de líneas de acción

En este módulo se incluyen los interfaces que faltan para resolver el segundo caso de interoperabilidad, todos aquellos que están relacionados con la evaluación de las diferentes líneas de acción.

Los interfaces de este módulo se encargan por tanto de la gestión de las matrices de evaluación que se obtienen como resultado de evaluar las diferentes líneas de acción sobre los escenarios que se les asocian:

- **Publicación de una matriz de evaluación.** Este interfaz permite a un actor compartir una matriz de evaluación, resultado de evaluar una línea de acción sobre un escenario, con el resto de actores del teatro.

  Para poder publicar la matriz de evaluación de una línea de acción, el actor se ha debido suscribir a ella previamente (empleando el interfaz de suscripción del módulo anterior), indicando así que se dispone a realizar la evaluación requerida.

- **Despublicación de la matriz de evaluación.** Este interfaz permite a un actor que deje de compartir una matriz de evaluación publicada anteriormente. Únicamente puede despublicar una matriz de evaluación el actor que la publicó.

- **Lectura de la matriz de evaluación.** Este interfaz permite a un actor obtener la información de una matriz de evaluación que ha sido previamente publicada por otro actor del mismo teatro.

- **Asignación de una matriz de evaluación a una línea de acción.** Este interfaz permite al actor que publica una matriz de evaluación que especifique a qué línea de acción (sobre un escenario concreto al que está asociada a su vez) corresponde.

- **Validación de la matriz de evaluación.** Este interfaz comprueba que la matriz de evaluación publicada por un actor está conforme a los especificado en el IC2D y al esquema de las matrices de evaluación que especificó el actor que publicó la línea de acción.

- **Descubrimiento de matrices de evaluación.** Este interfaz devuelve un listado de las matrices de evaluación que hay publicadas en un teatro o de las matrices de evaluación que tiene asignada una línea de acción concreta.

5.4. Definición de los modelos de interoperabilidad de referencia: IRM tipo 1 e IRM tipo 2

Estableciendo un paralelismo con la interoperabilidad entre simuladores constructivos tratada hasta el capítulo anterior en esta tesis doctoral, en este punto se dispone de una arquitectura que, de igual manera que HLA, permite esta interoperabilidad de manera prácticamente automática. Pero IC2A se diferencia de HLA en dos puntos importantes. El primero, está orientada a la integración de sistemas de distinto tipo y por lo tanto con arquitecturas diferentes. El segundo, que se deriva del primero, esto añade un nivel más de dificultad a la interoperabilidad, por lo que se ha optado por definir IC2A como una arquitectura que permita la interoperabilidad parcial entre ambos tipos de sistemas (no general o completa como HLA), en concreto, que resuelva los dos casos de interoperabilidad más comunes identificados. Si se fueran identificando nuevos casos en el futuro y se ampliara IC2A para darles cobertura, podría terminar siendo el equivalente a HLA pero para el contexto de simulación y mando y control, por lo tanto, la especificación sería mucho más compleja.

Si se continúa con la analogía, de la misma forma que HLA presentaba cierta ambigüedad al permitir que los mismos problemas de interoperabilidad se resolvieran de diferente manera, IC2A no garantiza que la solución a los dos casos de interés sea siempre la misma. Por eso en esta sección se completa la definición de la arquitectura con la definición de dos modelos de interoperabilidad de referencia, el 1 y el 2, que estandaricen la manera de resolver los casos 1 y 2 mediante la utilización de IC2A. Esto es lo que garantiza realmente que en una situación en la que sea necesario que un sistema C2 y uno o varios simuladores constructivos interoperen, lo puedan hacer de manera automática y transparente al usuario.

En este caso los IRMs están basados en actividades que se definen para dos tipos de sistemas diferentes y es la propia arquitectura IC2A la que asegura que una actividad no se pueda ejecutar hasta que se haya ejecutado la actividad anterior.

5.4.1. IRM tipo 1: Generación de un escenario real para un simulador constructivo

Este IRM consiste en dos tipos de actores (roles) donde el actor A (sistema C2) envía un STS (Scenario Transfer Specification) al actor B (simulador constructivo).

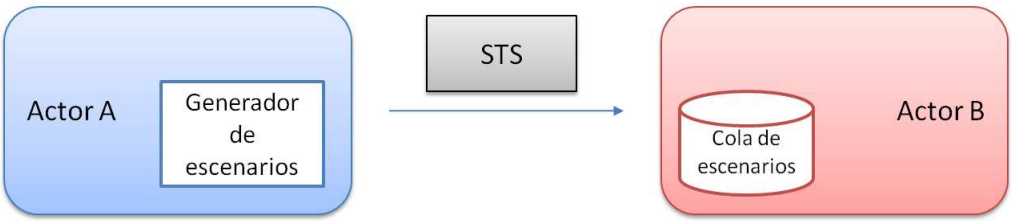

Figura 5.17: Esquema del IRM tipo 1

Este IRM (figura 5.17) modela la transferencia de un escenario entre dos actores. El actor A tiene un módulo final de generación de escenarios que crea el escenario y lo envía al actor B, que lo recibe y lo almacena en una cola de escenarios.

El concepto de transferencia de escenario en este IRM indica que una vez transferido el escenario, las actualizaciones del actor A sobre el mismo no tienen reflejo en el escenario que tiene el actor B. Es decir, se trata del mismo concepto de transferencia empleado en el capítulo 3.

5.4.1.1. Scenario Transfer Specification

La especificación del escenario que se transfiere (STS) consiste en un nombre que lo define y un fichero que contiene la estructura del mismo.

Un ejemplo de un escenario que se transfiere empleando el estándar MSDL sería:

- Nombre: Escenario Operación 1.
- Fichero: S_OP1.XSD (esquema XML definido en la figura 5.18).

5.4.1.2. Actividades definidas para el IRM tipo 1

Siguiendo la especificación de la arquitectura IC2A, un sistema C2 (que en este IRM es el actor A) debe realizar las siguientes actividades para ejecutar este modelo:

1. Crea un teatro nuevo. Para ello emplea el interfaz de IC2F *Creación de un teatro* (al crear el teatro el actor A queda dado de alta en él).
2. Valida que el STS que va a trasferir tiene una estructura permitida y cumple las reglas de negocio especificadas en el IC2D. Para ello emplea el interfaz

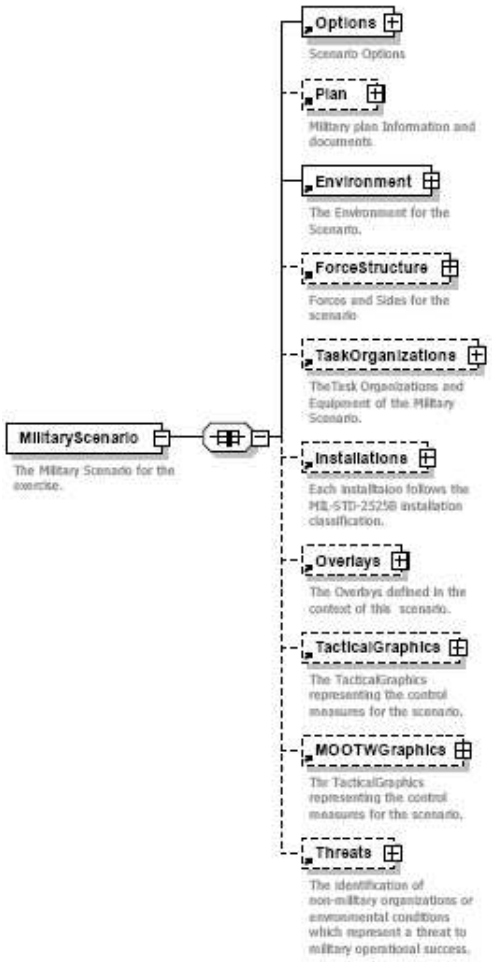

Figura 5.18: Ejemplo de esquema de un escenario definido en MSDL ([19])

Validación de un escenario, ya que un escenario que no se haya validado con anterioridad no puede ser publicado.

3. Publica el STS en el teatro que creó en la actividad 1. Para realizar esta operación emplea el interfaz *Publicación de escenario*.

4. Cuando ha trasferido el STS lo despublica del teatro. Para realizar esta operación emplea el interfaz *Despublicación de escenario*.

5. Una vez trasferida la STS destruye el teatro que creó en la actividad 1. Para ello emplea el interfaz *Destrucción de un teatro*.

Por otro lado las actividades que realiza el actor B (simulador constructivo) para recibir la información del ETS trasferido son las siguientes:

1. Lee los teatros que están creados actualmente en la IC2A. Para ello emplea el interfaz *Descubrimiento de teatros*.

2. Se une al teatro que tiene publicado el STS del cual desea obtener la información. Para ello emplea el interfaz *Alta de actores en un teatro*.

3. Obtiene una lista de los escenarios que están publicados en ese teatro. Para ello emplea el interfaz *Descubrimiento de escenarios*

4. De la lista anteriormente obtenida selecciona el STS del cual quiere obtener la información. Para ello emplea el interfaz *Lectura de un escenario*.

5. Cuando recibe la información del STS, se da de baja del teatro al que se unió en la actividad 2. Para ello emplea el interfaz *Baja de actores en un teatro*.

5.4.2. IRM tipo 2: Evaluación de las líneas de acción planeadas en un sistema C2

En este IRM se definen dos roles diferentes en los que pueden participar los actores (rol A y rol B). El rol A esta asociado a los sistemas C2 y el rol B a los simuladores constructivos.

En este caso el actor A necesita evaluar una línea de acción definida en la fase de planeamiento de una operación militar. Por lo tanto la transfiere al actor B para que la evalúe (figura 5.19).

El actor B que recibe la línea de acción la evalúa y devuelve una matriz de evaluación con el resultado.

En este IRM, al igual que en el anterior, se emplea el concepto de transferencia porque una vez que se envía la línea de acción en un sentido, o la matriz de evaluación en el contrario, las actualizaciones sólo tienen efecto en el sistema origen (y no en el destino, que una vez recibida la información, se desvincula del origen) .

5.4.2.1. Action Line Specification y Evaluation Matrix Specification

Son las estructuras de información que se transfieren para la correcta ejecución de este modelo.

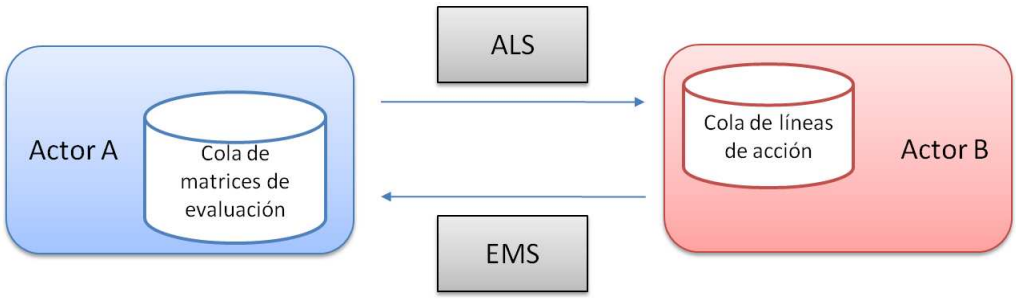

Figura 5.19: Esquema del IRM tipo 2

El ALS es la especificación de la línea de acción, por lo que incluye el nombre de la línea de acción y el esquema que sigue la información de la misma.

Un ejemplo de una ALS sería el siguiente:

- Nombre: Línea de acción A.
- Esquema: AL_A.XSD (figura 5.10).

El EMS es la especificación de la matriz de evaluación que el actor B devuelve al actor A. Esta especificación se compone del nombre de la misma y un esquema de la información que debe seguir dicha matriz.

Un ejemplo de una EMS sería el siguiente:

- Nombre: Matriz de evaluación B.
- Esquema: EM_B.XSD (figura 5.20).

5.4.2.2. Actividades definidas para el IRM tipo 2

Las actividades realizadas por el actor A (sistema C2) en este IRM son las siguientes:

1. Crea un teatro nuevo en la IC2A. Para ello emplea el interfaz de IC2F *Creación de un teatro* (el actor A al crear el teatro queda dado de alta en él).
2. Valida la información de la ALS que va a publicar en el teatro. Para ello emplea el interfaz *Validación de una línea de acción.*

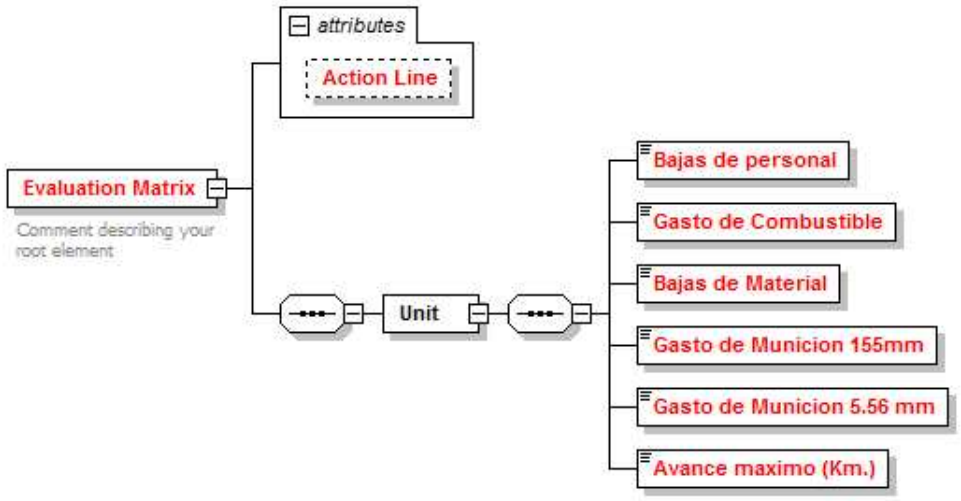

Figura 5.20: Ejemplo de esquema de una matriz de evaluación

3. Publica en el teatro creado en la actividad 1 la ALS que quiere trasferir. Para ello emplea el interfaz *Publicación de una línea de acción.*

4. Publica la estructura de la matriz de evaluación que quiere obtener para esa línea de acción. De esta manera el actor B sabe la estructura en la que debe publicar los resultados de la evaluación. Para realizar esta función se emplea el interfaz *Especificación de la matriz de evaluación.*

5. Publica el escenario que sirve como base para evaluar la línea de acción. Para ello emplea la interfaz *Publicación de escenario.*

6. Relaciona la línea de acción al escenario publicado en la actividad anterior. Para ello emplea el interfaz *Asignación de una línea de acción a un escenario.*

7. Obtiene un listado de las EMS que han sido publicadas como resultado de evaluar la línea de acción anterior. Para ello emplea el interfaz *Descubrimiento de matrices de evaluación.*

8. Lee la información de una o varias EMS (dependiendo de los objetivos del análisis). Para ello emplea el interfaz *Lectura de la matriz de evaluación.*

9. Cuando recibe las EMS despublica la línea de acción del teatro. Para ello emplea el interfaz *Despublicación de una línea de acción.*

10. Destruye el teatro que creo en la actividad 1. Para ello emplea el interfaz *Destrucción de un teatro*.

Las actividades realizadas por el actor B (simulador constructivo) en este IRM son las siguientes:

1. Solicita un listado de los teatros que están creados actualmente en la IC2A. Para ello emplea el interfaz *Descubrimiento de teatros*.

2. Se une al teatro que tiene publicado el escenario del cual desea obtener la información para evaluarlo. Para ello emplea el interfaz *Alta de actores en un teatro*.

3. Obtiene una lista de las líneas de acción que están publicadas en ese teatro. Para ello emplea el interfaz *Descubrimiento de líneas de acción*.

4. Comunica a la IC2A su intención de evaluar una línea de acción.Para ello emplea el interfaz *Suscripción a una línea de acción*.

5. Obtiene la información de la línea de acción, matrices de evaluación y escenarios asignados a la misma. Para ello emplea el interfaz *Lectura de una línea de acción*.

6. Cuando termina de evaluar la línea de acción obtenida en la actividad anterior, elabora la matriz de evaluación siguiendo la estructura especificada por el actor A y la valida. Para validar que el formato de la matriz es correcto emplea el interfaz *Validación de la matriz de evaluación*.

7. Publica en el teatro la EMS validada en la actividad anterior. Para ello emplea el interfaz *Publicación de una matriz de evaluación*.

8. Asigna la EMS a la línea de acción que ha evaluado. Para ello emplea el interfaz *Asignación de una matriz de evaluación a una línea de acción*.

9. Abandona el teatro al que esta unido para ejecutar este IRM. Para ello emplea el interfaz *Baja de actores en un teatro*.

Capítulo 6

Aplicación práctica de la arquitectura IC2A

La arquitectura IC2A propuesta en el capítulo anterior junto con los nuevos modelos de interoperabilidad de referencia definidos para la estandarización de los dos casos de interoperabilidad de interés, permite la interoperabilidad parcial entre simuladores constructivos y sistemas C2.

Pero como ocurría en el caso de HLA y la definición de los IRMs para simulación constructiva, esta definición no está completa si no se demuestra que es posible utilizar la arquitectura y los modelos en un escenario real de aplicación militar.

En este capítulo se verifica la definición de la arquitectura propuesta utilizando una arquitectura software real para realizar su implementación y modelando con ella las necesidades de interoperabilidad analizadas en el capítulo anterior. De la misma forma, se validan los modelos de interoperabilidad de referencia definidos (IRM tipo 1 e IRM tipo 2) mediante un caso de aplicación real.

6.1. Aspectos clave en la implementación de la arquitectura IC2A

El objetivo de la arquitectura IC2A es permitir la interoperabilidad parcial entre dos familias de sistemas de apoyo a la conducción de operaciones militares (simuladores constructivos y sistemas C2) con arquitecturas muy diferentes. Es decir, la arquitectura software que se escoja para su implementación debe permitir, como mínimo, la convergencia entre sistemas débilmente acoplados.

Además de tener en cuenta este primer criterio para escoger la arquitectura sobre la que construir IC2A, se deben tener en cuenta otros criterios dadas las propuestas del capítulo anterior:

1. **Arquitectura cliente-servidor.** El IC2F es el elemento de la IC2A a través del cual se conectan los sistemas con necesidades de interoperabilidad. Por lo tanto la arquitectura cliente-servidor es idónea para implementar IC2A, siendo los sistemas que necesitan interoperar los clientes, y el IC2F el servidor que atiende sus peticiones.

2. **Flexibilidad para integrar diferentes protocolos de intercambio de datos.** Cada tipo de sistema se va a comunicar con el IC2F empleando su propio mecanismo de intercambio de datos. Por este motivo la arquitectura seleccionada debe soportar intercambios de información mediante protocolos diferentes.

3. **Adaptable a los diferentes cambios de versión de los sistemas.** Como se explicó en el capítulo anterior, sistemas del mismo tipo pero bajo la denominación de diferentes versiones pueden participar en un entorno IC2A. Por lo tanto la arquitectura seleccionada debe ser capaz de integrar diferentes versiones de sistemas que participen en un mismo escenario de interoperabilidad.

4. **Arquitectura de empleo difundido en la actualidad.** Para que IC2A se convierta en un estándar de facto es necesario que se implemente sobre una arquitectura software ampliamente conocida y difundida, evitando así un exceso de complejidad en los procesos de aprendizaje, desarrollo, mantenimiento, documentación, etc.

Estos criterios han llevado a la selección como plataforma para la implementación de la IC2 de una arquitectura orientada a servicios (Services Oriented Architecture, SOA, [54]). De esta manera las interfaces de la IC2A son servicios que deben proveer las funcionalidades del IC2F a los clientes. En concreto se ha optado por una arquitectura de tipo ESB (Enterprise Service Bus, [131], [132]), lo que se justificará en las siguientes secciones.

Un ESB es un componente software que actúa como middleware entre diferentes aplicaciones heterogéneas permitiendo su interoperabilidad. La denominación de bus, aunque no se trate de un elemento hardware, denota la posibilidad del ESB de

evitar las comunicaciones punto a punto entre aplicaciones. Se trata de un bus basado en eventos e implementado mediante estándares SOA, hoy en día, Web Services ([133]).

La diferencia principal entre un ESB ([131], [134]) y una arquitectura SOA clásica está en que el ESB debe proveer a los servicios que se conectan a él ciertas capacidades. De esta manera se aligera la programación de los servicios, que ya no son responsables de proporcionar todas estas funcionalidades sino que confían en el ESB para que las resuelva. Algunos ejemplos son la capacidad de proceso de eventos, la validación, enriquecimiento, transformación y operación de mensajes, la gestión, monitorización y auditoría o la garantía de un cierto nivel de calidad de servicio y seguridad. En las siguientes secciones se profundiza algo más en estos conceptos para justificar al idoneidad de este tipo de arquitectura para la implementación de la arquitectura IC2A.

6.1.1. Conceptos básicos de ESB

Un ESB es una arquitectura definida para integrar sistemas que emplean diferentes estándares y están diseñados para diferentes entornos ([131], [132]). Cuanto más diferentes sean los sistemas que se intentan integrar, más efectivo es el empleo del ESB ya que su implantación puede llegar a ser muy costosa y, por tanto, poco rentable en el caso de sistemas que puedan converger empleando otras arquitecturas software de integración.

La arquitectura ESB se basa en el envío de mensajes a un servidor que son enrutados de manera inteligente hacia aquellos servicios que deben ejecutar las funciones solicitadas por el cliente ([54]). El concepto de bus aparece porque el envío de mensajes entre los componentes conectados por el ESB se realiza de forma flexible y horizontal.

La flexibilidad proviene de la definición de las diferentes rutas que pueden seguir los mensajes dependiendo de su contenido: mensajes con la misma estructura, y el mismo origen y destino, pueden seguir rutas diferentes según sus contenidos.

En cuanto a la horizontalidad, esta propiedad aparece porque el envío de los mensajes no sigue un camino ascendente (llamadas jerarquizadas) entre componentes. Los mensajes son enviados al componente que, dependiendo de su contenido, sea el más idóneo.

La estructura de un ESB consta de los siguientes componentes (figura 6.1):

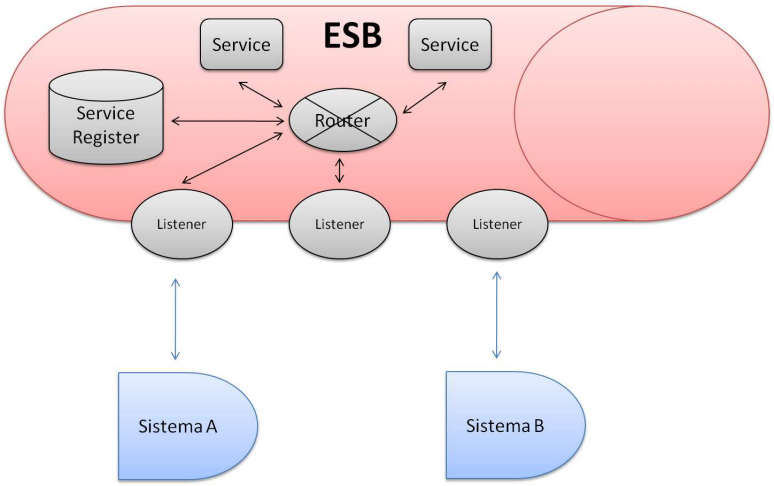

Figura 6.1: Esquema de un ESB estándar

- **Listener o end points.** Éste es el componente con el que se comunica un cliente del ESB. Por lo tanto este componente debe emplear los mismos protocolos y estándares que emplea el cliente.
- **Mensajes.** Los mensajes son los elementos que se envían los componentes ESB entre sí y con los clientes (sólo en el caso de los listeners).
- **Router.** Este elemento recibe los mensajes de un componente y los encamina a otros componentes dependiendo de su contenido.
- **Servicio.** Realiza las funciones necesarias para satisfacer de manera total o parcial lo que ha solicitado el cliente a través del listener. La función que realiza un servicio está condicionada por el mensaje que recibe, y el servicio siempre envía un mensaje para informar de los resultados de la citada función cuando se finaliza su ejecución al cliente que la solicitó.
- **Service register.** Es un repositorio donde se encuentra toda la información de los componentes que están publicados en el ESB.

6.1.2. Justificación de la utilización del ESB como arquitectura para la implementación de la IC2A

El ESB, es por definición, una arquitectura cliente-servidor idónea para la integración de sistemas débilmente acoplados empleando para ello una arquitectura

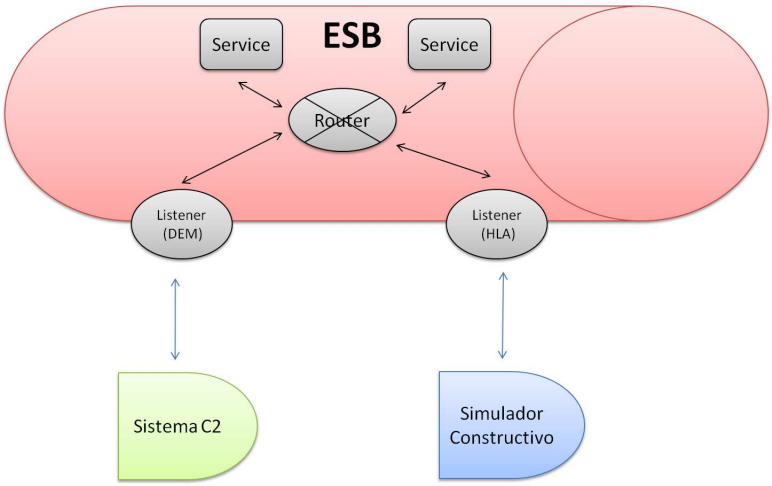

Figura 6.2: Esquema conceptual de la integración de sistemas C2 con simuladores constructivos mediante la utilización de un ESB

orientada a servicios.

Además es una arquitectura muy flexible que permite que convivan en el mismo entorno varios protocolos de intercambio de información con varias versiones del sistema, todo ello de manera bastante sencilla ([135]) gracias a la configuración de diferentes listeners y a la perfecta configuración del enrutamiento de los mensajes (se utilizan listeners adaptados a los diferentes mecanismos de intercambio de información y se configuran diferentes rutas para las diferentes versiones de los sistemas).

Por lo tanto se puede concluir que el ESB es una arquitectura que cumple los criterios propuestos para la implementación de IC2A con éxito y que resulta idónea para su verificación (figura 6.2).

6.1.3. Retos que plantea la utilización de un ESB y utilidad de los patrones SOA para superarlos con éxito

La búsqueda de la convergencia entre los diferentes sistemas de información utilizados en una organización y la poca integración inicial entre ellos fue uno de los principales motivos que hizo surgir arquitecturas como ESB para resolver los problemas de interoperabilidad.

En un principio ESB fue considerado como una nueva generación de EAI (Enterprise Application Integration). Pero un ESB no se basa en una lógica de negocio

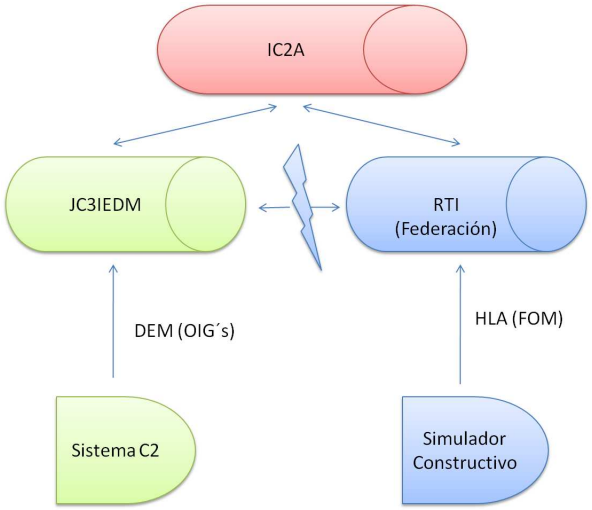

Figura 6.3: Esquema general de la IC2A

como EAI sino que es una infraestructura SOA ([131], [136]) y bajo esta premisa principal se han desarrollado las diferentes implementaciones de ESB que se emplean en la actualidad.

La aplicación de este concepto de ESB a un ámbito táctico-militar plantea diferentes retos derivados de las diferencias entre este entorno y el empresarial.

La primera de ellas es que los sistemas C2 militares son muy específicos del ámbito para el cual fueron diseñados y que emplean estándares específicos de este entorno en la mayoría de los casos (figura 6.3).

La segunda es que la diferencia existente entre la arquitectura de los sistemas C2 y la de los simuladores constructivos es mayor que la existente entre la de los sistemas de información que se suelen integrar en entornos empresariales. Es decir, en esta tesis doctoral el entorno de aplicación presenta una dificultad mayor de integración dado el alto nivel de heterogeneidad presente.

En esta investigación se pretende aprovechar la orientación a servicios del ESB para superar estas dos posibles barreras mediante la utilización de patrones SOA ([137]). Estos patrones permiten la reutilización de conocimiento y la propuesta de mejores prácticas en la realización de desarrollos e integraciones orientados a servicios.

El objetivo es estandarizar las técnicas de diseño más frecuentes y las soluciones para los problemas típicos que aparecen durante la realización de un proyecto SOA.

Los noventa patrones SOA que se han documentado hasta el momento se pueden

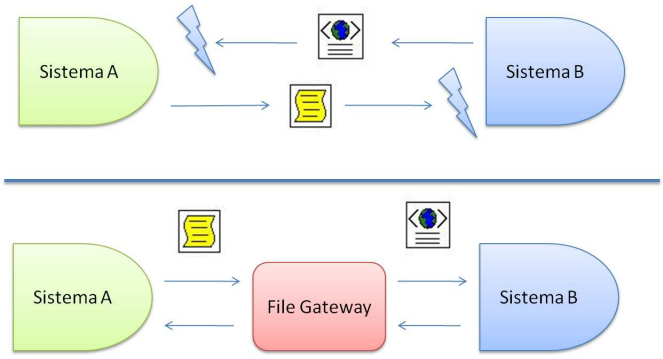

Figura 6.4: Esquema del patrón SOA File Gateway

dividir en tres categorías dependiendo del nivel de diseño en el que se centren: Arquitectura del Inventario de Servicios, Arquitectura de Composición de Servicios y Arquitectura de Servicio. Y la intención es que estos patrones evolucionen con el tiempo gracias a las contribuciones de la comunidad de desarrolladores.

En el caso de la arquitectura IC2A se definen los siguientes patrones de utilidad para la resolución de las necesidades de interoperabilidad listadas en el capitulo anterior:

6.1.3.0.1. Intercambio de ficheros. Esta necesidad de interoperabilidad se puede diseñar tomando como referencia el patrón *SOA File gateway* ([137]).

En este caso los sistemas que se desea integrar intercambian ficheros con estructuras diferentes y que por lo tanto, si se envían sin un proceso de transformación intermedio no son comprensibles para ninguno de ellos (figura 6.4).

Para resolver este problema el patrón recomienda modelar un servicio intermedio que recibe un fichero en un formato determinado y lo transforma según la estructura aceptada por el sistema receptor (figura 6.4).

6.1.3.0.2. Transformación de datos. Esta necesidad de interoperabilidad se puede resolver tomando como referencia el patrón *SOA Data Format Transformation* ([137]).

En este caso la estructura de la información intercambiada entre los dos sistemas es diferente. Es decir, el sistema receptor no entiende la información que recibe del sistema emisor (figura 6.5).

Para resolver este problema el patrón recomienda modelar un servicio intermedio

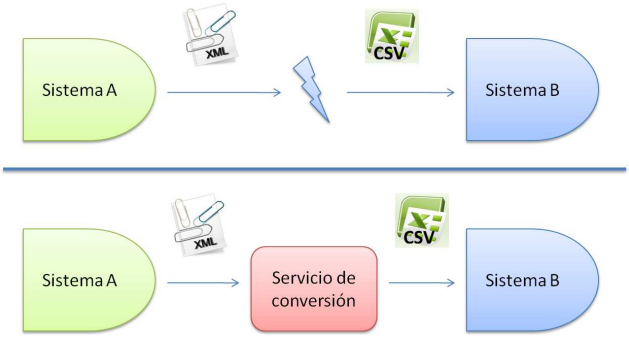

Figura 6.5: Esquema del patrón SOA Data Format Transformation

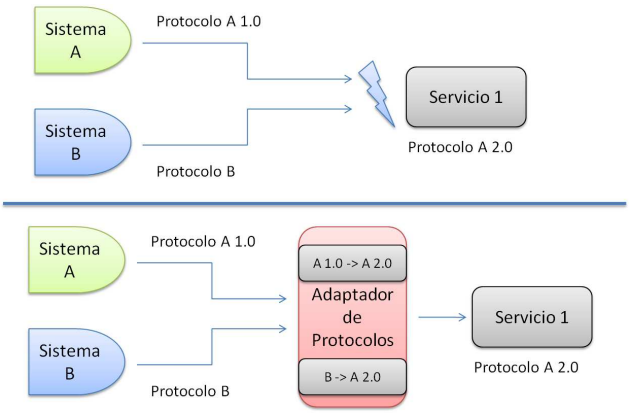

Figura 6.6: Esquema del patrón SOA Protocol Bridging

que reciba la información y la transforme a una estructura entendible por el sistema que la va a recibir (figura 6.5). El estándar de transformación XSLT ([138]) puede ser un estándar de interés para asociarlo con este patrón SOA y resolver el problema que se plantea ya que permite la transformación de datos con estructura XML.

6.1.3.0.3. Convergencia de protocolos. Esta necesidad de interoperabilidad se puede solucionar empleando el patrón *SOA Protocol Bridging* ([137]).

Este patrón modela la situación en la que dos sistemas que intentan acceder a un servicio no pueden hacerlo porque el protocolo a través del cual intentan acceder no es compatible con el que emplea el servicio (figura 6.6).

Para resolver este problema el patrón recomienda modelar un elemento intermedio entre los sistemas y el servicio que entienda el protocolo de los sistemas y lo adapte al protocolo que entiende el servicio (figura 6.6).

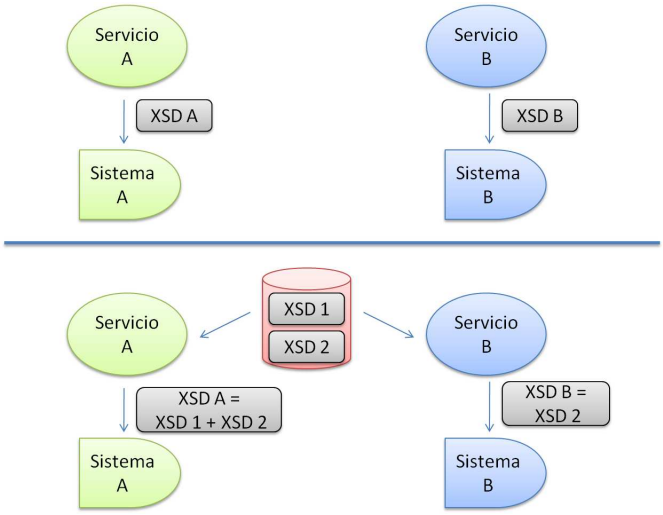

Figura 6.7: Esquema del patrón SOA Schema Centralization

6.1.3.0.4. Centralización de esquemas. Esta necesidad de interoperabilidad se puede a diseñar tomando como referencia el patrón *SOA Schema Centralization* ([137]).

Cada definición de servicio modela la información que emplea según un esquema de información. Esto supone, cuando el número de servicios es elevado, un problema de control sobre las estructuras de información que se están empleando en una organización (figura 6.7).

Para solucionar este problema este patrón define estructuras de información básicas cuya composición forma los diferentes esquemas de información que emplean los servicios (estructuras más complejas). Las estructuras simples se almacenan de manera centralizada consiguiendo un mayor control de los esquemas que se están empleando en una organización (figura 6.7).

6.1.3.0.5. Centralización de reglas. Esta necesidad de interoperabilidad se puede solucionar tomando como referencia el patrón *SOA Rules Centralization* ([137]).

Los modelos de negocio de las organizaciones imponen que las funciones de los servicios lleven anexas, en muchos casos, restricciones que deben ser aplicadas.

El modelo de negocio de una organización no es fijo y algunos aspectos pueden evolucionar o cambiar afectando a las reglas definidas en los servicios. El impacto de este cambio de las reglas de negocio es difícilmente calculable cuando el número de servicios es elevado (figura 6.8).

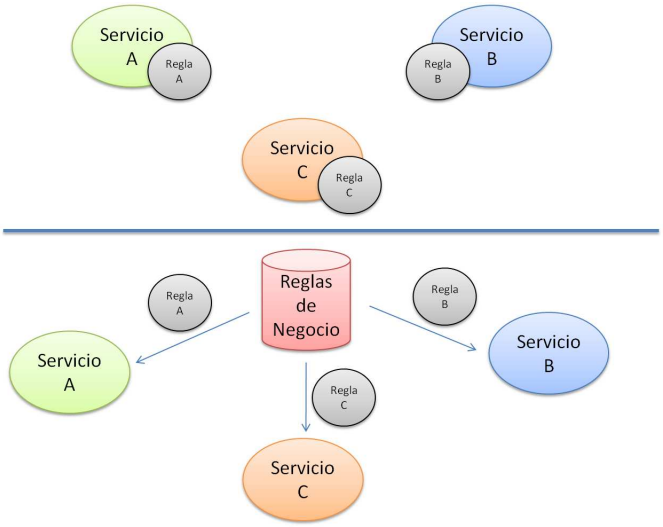

Figura 6.8: Esquema del patrón SOA Rules Centralization

Para solucionar este problema este patrón lo que propone es que todas las reglas de negocio estén centralizadas en un repositorio de tal manera que los servicios que las necesiten consulten este repositorio de reglas (figura 6.8).

También propone que la definición de las reglas de negocio sea lo más sencilla posible. De esta manera si un servicio necesita una regla compleja pueda ser compuesta a partir de varias reglas simples. Mediante la aplicación de este patrón se consigue una mejor definición de las reglas de negocio y se reduce la redundancia y la complejidad de su gestión.

6.1.3.0.6. Centralización de procesos. Esta necesidad de interoperabilidad se puede resolver tomando como referencia el patron *SOA Process Centralization* ([137]).

Un sistema ejecuta procesos que contienen toda su funcionalidad. Si se consideran los procesos de manera independiente puede existir una falta de coherencia entre ellos, o redundancia de dos o varios procesos que soporten funcionalidades idénticas del sistema.

Este patrón SOA propone de nuevo una centralización de los procesos de manera que se mantenga la coherencia entre ellos y se reduzca su redundancia (figura 6.9). El repositorio de procesos soporta toda la funcionalidad del sistema. Por lo tanto cada una de las funciones del sistema se configura con la suma de varios procesos del repositorio central de procesos.

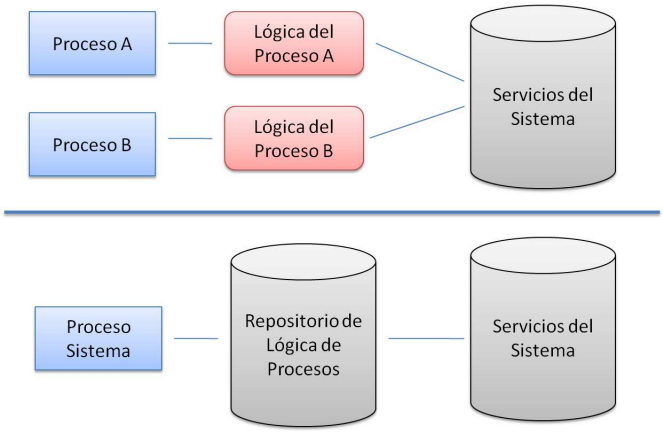

Figura 6.9: Esquema del patrón SOA Process Centralization

6.1.3.0.7. Compatibilidad de procesos. Esta necesidad de interoperabilidad se puede diseñar tomando como referencia el patrón *SOA Compatible Change* ([137]).

Dentro de un sistema pueden coexistir varias versiones diferentes de un mismo servicio. Esto puede generar problemas de funcionamiento cuando un cliente invoca a un servicio con diferente versión para la que fue diseñado.

Este patrón propone la existencia de una única versión de un servicio que sea diseñado para guardar la compatibilidad hacia atrás. Es decir, las variaciones introducidas en un servicio no conllevan la creación de un nuevo servicio con diferente versión sino que es el mismo servicio con funcionalidad ampliada para soportar la nueva versión (figura 6.10).

6.1.3.0.8. Repositorio de estado. Esta necesidad de interoperabilidad se puede solucionar tomando como base el patrón *SOA State Repository* ([137]).

Los servicios suelen emplear datos temporales para ejecutar sus funciones (en muchos casos mediante acceso a repositorios de datos). Estos datos son almacenados en las memorias de los sistemas mientras se están procesando, donde pueden pasar mucho tiempo consumiendo recursos hardware críticos para el rendimiento de los sistemas.

Por este motivo este patrón SOA propone la creación de repositorios de información temporal donde los servicios puedan almacenar estos datos temporales (estados) y utilizarlos cuando se necesite (figura 6.11). De esta manera se liberan recursos críticos de los sistemas (como es la memoria) y se aumenta el rendimiento de los mismos.

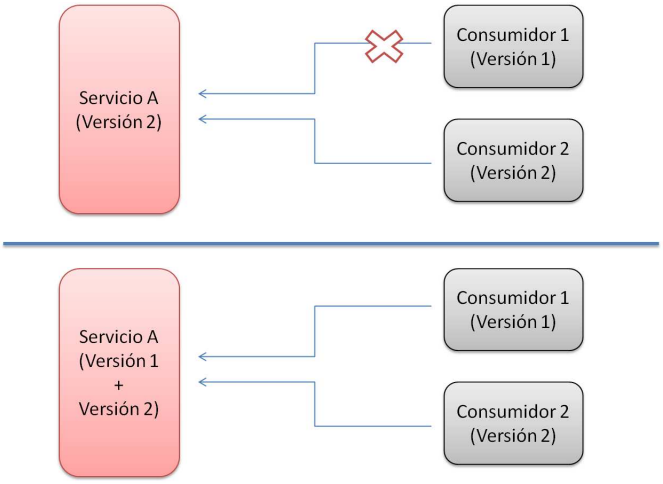

Figura 6.10: Esquema del patrón SOA Compatible Change

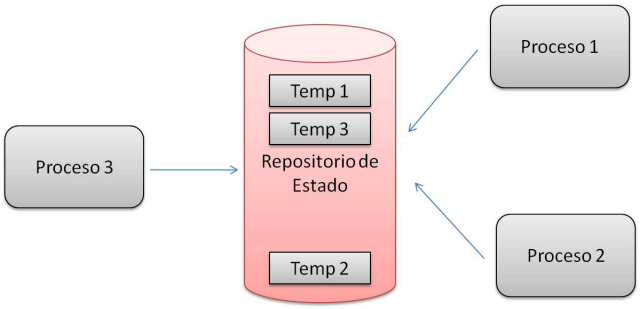

Figura 6.11: Esquema del patrón SOA State Repository

6.1.3.0.9. Núcleo básico. Esta necesidad de interoperabilidad se puede diseñar tomando como referencia el patrón *SOA Service Facade* ([137]).

Un núcleo básico de servicios que da soporte a un sistema debe poder aceptar contratos o acuerdos (agreements, [139]) con servicios que amplíen su funcionalidad. Pero puede darse el caso de que un servicio del núcleo básico sólo acepte contratos con ciertos servicios, siendo necesario emplear otras versiones del servicio del núcleo para poder realizar contratos con el resto de servicios (figura 6.12).

Para solucionar este problema el patrón SOA propone la configuración, dentro de un mismo servicio, de varios contratos. De esta manera con una única versión del servicio del núcleo se pueden realizar enlaces con todo tipo de servicios que amplíen la funcionalidad del sistema (figura 6.12). Y se maneja el concepto del núcleo como una composición de servicios donde sólo existe una única versión de los mismos.

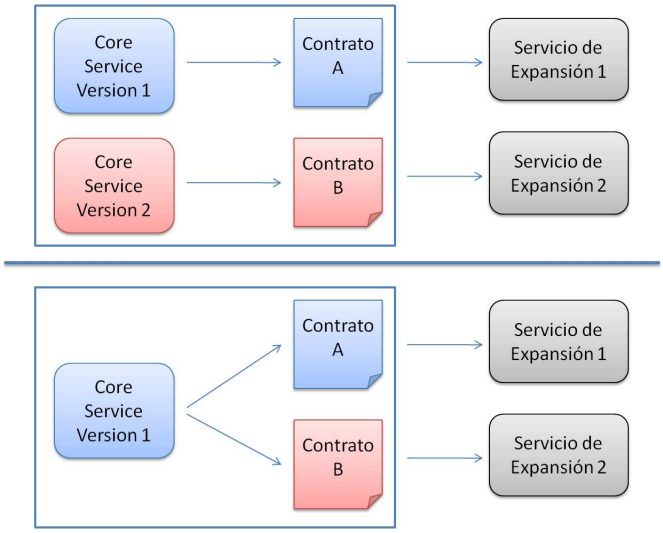

Figura 6.12: Esquema del patrón SOA Service Facade

6.1.4. Implementación de la arquitectura IC2A sobre un ESB genérico utilizando los patrones SOA

Para concluir esta sección se revisa la definición de la IC2A mapeando los componentes propuestos en el capítulo anterior a la arquitectura software escogida para su implementación, el ESB, y teniendo en cuenta la posibilidad de utilizar los patrones SOA para resolver los problemas planteados por las necesidades de interoperabilidad. Este es el último paso necesario antes de implementar realmente la arquitectura sobre un ESB comercial y pasar a la validación de los IRMs de tipo 1 y 2.

Se parte de la siguiente situación inicial (figura 6.13) para la definición de las necesidades de interoperabilidad:

- Existe un teatro creado donde se modela la necesidad de interoperabilidad.
- Cada uno de los sistemas (bien sea un sistema C2 o un simulador constructivo) es un actor que está unido al teatro citado en el punto anterior.
- En el ESB existe, como mínimo, un listener que recibe y envía los mensajes de los simuladores constructivos empleando HLA.
- En el ESB existe, como mínimo, un listener que recibe y envía los mensajes de los sistemas C2 empleando el protocolo DEM.

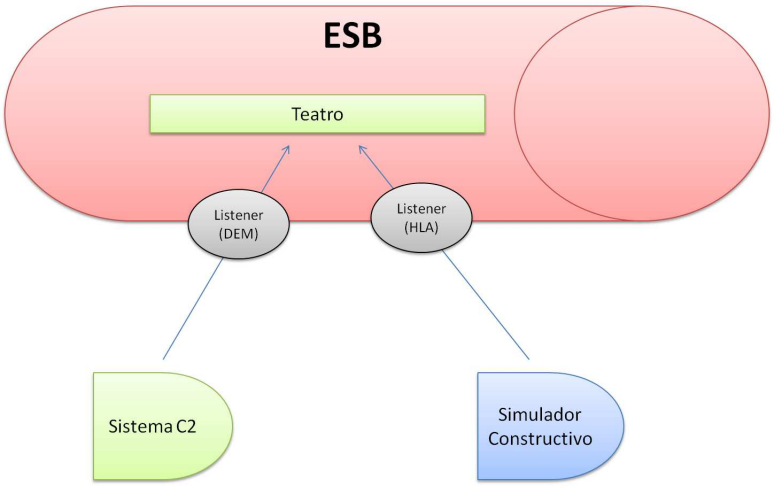

Figura 6.13: Esquema del punto de partida

- Existe como mínimo un actor con el rol sistema C2 y otro con el rol simulador constructivo para el desarrollo de cada necesidad de interoperabilidad.

Tomando esta situación como punto de partida, se propone modelar las necesidades de interoperabilidad de la siguiente manera.

6.1.4.1. Intercambio de ficheros

Siguiendo la arquitectura IC2A definida en el capítulo anterior, un actor publica un fichero en el teatro y el otro actor solicita el fichero publicado que se encuentra en un formato no comprensible para él.

Para ejecutar estas actividades propuestas en la IC2A con un ESB se definen los siguientes componentes (figura 6.14):

- **Router de intercambio de ficheros:** Este elemento encamina los mensajes que recibe a un servicio u otro dependiendo del contenido de los mismos.
- **Servicio de gestión del teatro:** Este servicio gestiona la publicación o lectura (solicitada por otros servicios) de un fichero en el teatro.
- **Servicio de intercambio de ficheros:** Este servicio transforma los ficheros de un formato a otro. Por lo tanto los mensajes que recibe indican qué tipo de transformación se requiere e incluyen el fichero que se debe transformar.

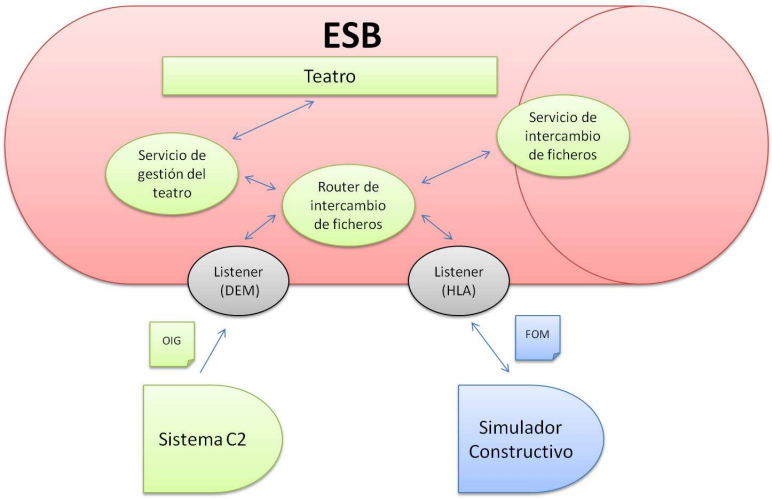

Figura 6.14: Esquema de intercambio de ficheros en un ESB

En primer lugar un actor con el rol sistema C2 publica en el teatro un fichero. Para ello lo envía al listener, que a través del router, lo envía a su vez al servicio de gestión de teatro. Este servicio registra que ese fichero concreto está publicado en el teatro.

Otro actor con el rol simulador constructivo quiere obtener el fichero publicado anteriormente con lo que envía a un listener (HLA) un mensaje solicitando la lectura del mismo.

El listener (HLA) a través del servicio de routing solicita al servicio de gestión de teatro esa información, que le es enviada al servicio de intercambio que transforma el fichero a un formato entendible por el simulador constructivo. Y este fichero es enviado al simulador constructivo a través del listener (HLA).

6.1.4.2. Transformación de datos

En esta necesidad de interoperabilidad un actor con el rol sistema C2 publica en el teatro una información siguiendo una estructura determinada. El otro actor con el rol simulador constructivo solicita la información que ha publicado el sistema anterior en el teatro pero que se encuentra en un formato que no es comprensible para él.

Se definen los siguientes elementos (figura 6.15):

- **Router de transformación de datos:** Este elemento encamina los mensajes

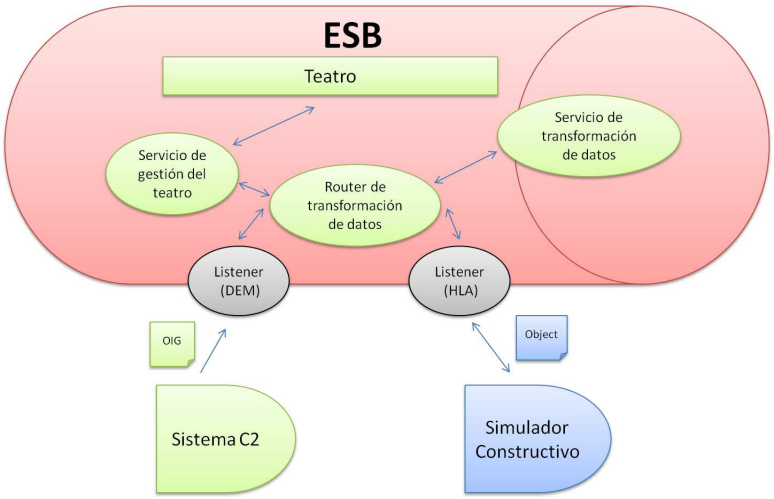

Figura 6.15: Esquema de transformación de datos en un ESB

que recibe a un servicio u otro dependiendo del contenido de los mismos.

- **Servicio de gestión del teatro:** Este servicio gestiona la publicación o lectura (solicitada por otros servicios) de unos datos estructurados en el teatro.

- **Servicio de transformación de datos:** Este servicio transforma los datos de un formato a otro. Por lo tanto los mensajes que recibe indican qué tipo de transformación se requiere e incluyen los datos que se deben transformar.

En este caso el actor sistema C2 envía unos datos a través del listener(DEM) para que sean publicados en el teatro. Estos datos llegan al router, que al detectar que es una solicitud para el teatro, la envía al servicio de gestión del teatro para su publicación.

El actor simulador constructivo solicita la información publicada anteriormente a través del listener (HLA). Este mensaje de solicitud es enviado al router, que a continuación lleva a cabo dos tareas. La primera es solicitar esa información al servicio de gestión del teatro, la segunda, solicitar su transformación (al formato requerido por el simulador constructivo) al servicio de transformación de datos.

Una vez realizadas estas acciones la información es enviada al simulador constructivo a través del listener (HLA).

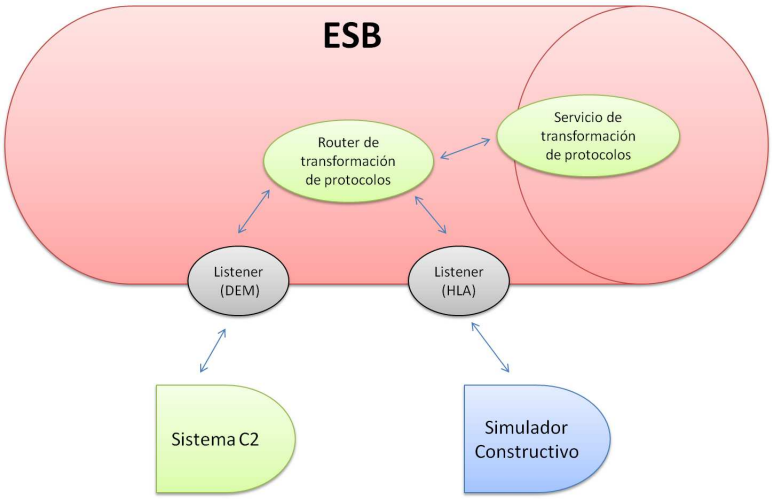

Figura 6.16: Esquema de transformación de protocolos en un ESB

6.1.4.3. Convergencia de protocolos

Los protocolos que emplean los actores que pertenecen a un teatro con diferente rol en la mayoría de los casos no son los mismos. Por lo tanto es necesaria una convergencia entre ellos. Para definir esta necesidad de interoperabilidad en un ESB se definen los siguientes componentes (figura 6.16):

- **Router de transformación de protocolos**: La función principal de este elemento es recibir los mensajes de transformación de protocolos requeridos y encaminarla al servicio que gestiona esta transformación.

- **Servicio de transformación de protocolos**: Este servicio es un adaptador de protocolos, por tanto recibe un mensaje en un protocolo determinado y lo transforma a otro protocolo.

Un actor con rol sistema C2 o simulador constructivo a través del listener(DEM) o el listener(HLA) envía información al ESB según su protocolo definido. En este caso esa información llega al router, que la envía al servicio que gestiona la transformación de protocolos requerida. El resultado es enviado al sistema C2 o simulador constructivo (según corresponda) en un protocolo que sea entendible por él.

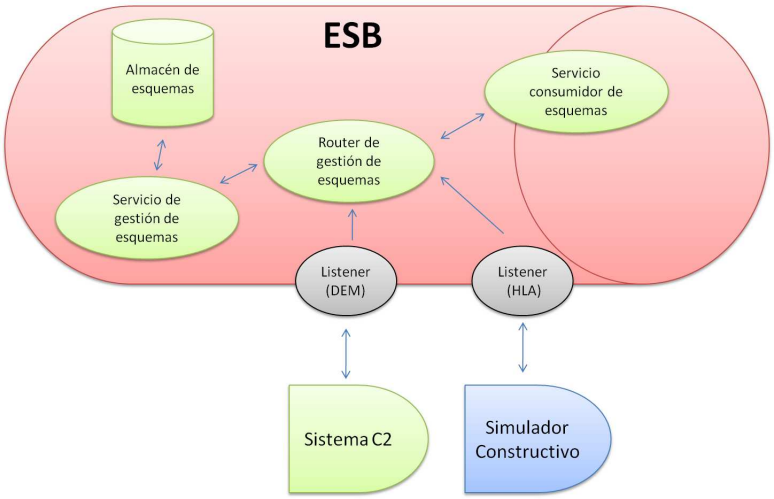

Figura 6.17: Esquema de centralización de esquemas en un ESB

6.1.4.4. Centralización de esquemas

La información que se maneja dentro del ESB debe estar conforme a unas estructuras definidas para una IC2A concreta. Estas estructuras no forman un conjunto cerrado ya que se pueden dar de alta nuevos esquemas.

Para definir esta necesidad de interoperabilidad dentro de un ESB se definen los siguientes elementos (figura 6.17):

- **Router de gestión de esquemas:** Este elemento recibe mensajes referentes a los esquemas y los encamina al servicio del ESB adecuado para procesar las peticiones o respuestas.
- **Almacén de esquemas:** Es un repositorio donde se almacenan físicamente los esquemas de información.
- **Servicio de gestión de esquemas:** Este servicio se encarga de la gestión del almacén de esquemas (inserción, búsqueda, borrado, etc).
- **Servicio consumidor de esquemas:** Se denomina de esta manera al servicio que necesita acceder al repositorio de esquemas.

Al aplicar esta necesidad de interoperabilidad en un ESB se pueden producir dos situaciones diferentes(figura 6.17):

1. Si un actor es el que solicita una operación sobre un esquema. Para ello, envía la solicitud a través del listener y éste se la envía al router. El router detecta que es una solicitud de gestión de esquemas y se la envía al servicio de gestión de esquemas que es propietario del almacén.

2. Si un servicio es el que solicita una operación sobre un esquema. En este caso el servicio envía esta solicitud al router de esquemas que a través del servicio de gestión de esquemas realiza la operación solicitada en el almacén de esquemas.

6.1.4.5. Centralización de reglas

En esta necesidad de interoperabilidad las reglas de negocio deben estar centralizadas en un único repositorio de tal manera que los sistemas puedan dar de alta nuevas reglas o acceder a ellas para consultarlas.

Para adaptar este concepto de la IC2A a un ESB es necesario definir los siguientes elementos (figura 6.18):

- **Router de gestión de reglas:** Este elemento encamina los mensajes relacionados con reglas de negocio a un servicio u otro según su contenido.
- **Almacén de reglas:** Es un repositorio donde se almacenan físicamente las reglas.
- **Servicio de gestión de reglas:** La función principal de este servicio es gestionar el almacén de reglas, del que es propietario.
- **Servicio consumidor de reglas:** Es un servicio que requiere acceder al almacén de reglas para insertar una nueva o para validar la que está manejando.

Al aplicar esta necesidad de interoperabilidad en un ESB se pueden dar dos situaciones diferentes (figura 6.18):

1. Si un actor es el que solicita una operación sobre una regla. Para ello, envía la solicitud a través del listener y éste se la envía al router. El router detecta que es una solicitud de gestión de regla y se la envía al servicio de gestión de reglas que es propietario del almacén.

2. Si un servicio es el que solicita una operación sobre una regla. En este caso el servicio envía esta solicitud al router de reglas que a través del servicio de gestión de reglas realiza la operación solicitada en el almacén de esquemas.

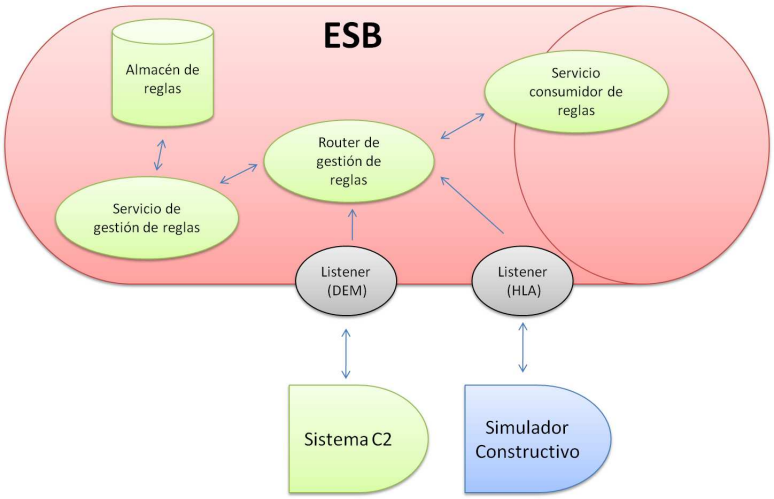

Figura 6.18: Esquema de centralización de reglas en un ESB

6.1.4.6. Centralización de procesos

Esta necesidad de interoperabilidad se refiere a que los procesos que dan cobertura a las interfaces de la IC2A deben estar centralizados en un repositorio. Para definir en un ESB este concepto definido por la IC2A son necesarios los siguientes componentes (figura 6.19):

- **Router de gestión de procesos:** Este elemento encamina los mensajes referentes a la gestión de procesos dentro del ESB.
- **Almacén de procesos:** Es un repositorio donde se almacenan físicamente los procesos.
- **Servicio de gestión de procesos:** Este servicio es el encargado de la gestión del almacén de procesos y de todas las tareas necesarias para su actualización y mantenimiento.
- **Servicio consumidor de procesos:** Se denomina de esta manera al servicio que solicita información para realizar una función concreta a través de uno de los procesos almacenados en el repositorio de procesos.

Cuando un servicio del ESB necesita realizar una función envía un mensaje al router de procesos solicitando información sobre el proceso adecuado para ejecutarla. El router de procesos envía un mensaje al servicio de gestión de procesos

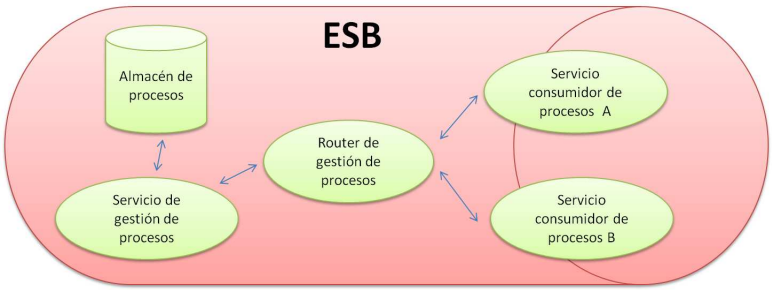

Figura 6.19: Esquema de centralización de procesos en un ESB

solicitando su búsqueda en el almacén de procesos y envía esta información al router. Por último el router devuelve esta información al proceso consumidor que la solicitó inicialmente.

6.1.4.7. Compatibilidad de procesos

Para que una IC2A pueda integrar varios sistemas C2 o simuladores constructivos es necesario que la IC2A pueda trabajar con diferentes versiones de los sistemas. Para adaptar esta necesidad de la IC2A a un ESB, se requiere la participación de los siguientes elementos (figura 6.20):

- **Router de compatibilidad de procesos:** Este elemento recibe un mensaje del servicio consumidor solicitando una actividad de otro servicio. Dependiendo de la versión que requiera el servicio consumidor, el router solicitará una función u otra (dependiendo de la versión) del servicio.
- **Servicio consumidor:** Este servicio requiere la función de otro servicio para completar su actividad.

El servicio consumidor envía un mensaje al router de compatibilidad de procesos solicitando la actividad que desea realizar. El router invoca una función u otra al servicio encargado de realizarla dependiendo de la versión del proceso con la que es compatible el servicio consumidor (figura 6.20). Este servicio realiza la función solicitada y devuelve el resultado al router, y a través del él, al servicio consumidor.

6.1.4.8. Repositorio de estado

Esta necesidad de interoperabilidad define el tratamiento de la información temporal necesaria para que los servicios del ESB realicen sus actividades. Para adaptar

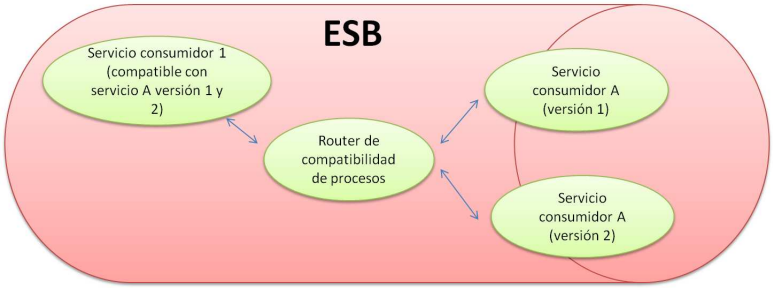

Figura 6.20: Esquema de compatibilidad de procesos en un ESB

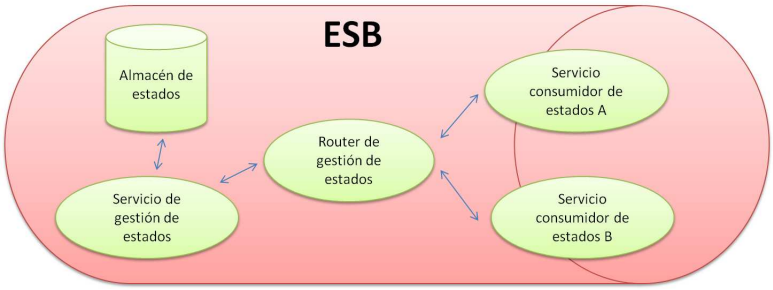

Figura 6.21: Esquema de repositorio de estados en un ESB

al ESB esta necesidad de interoperabilidad de la IC2A son necesarios los siguientes elementos (figura 6.21):

- **Router de gestión de estados:** Este elemento encamina en el ESB los mensajes relacionados con la información temporal que manejan los servicios.
- **Almacén de estados:** Es el elemento en que está almacenada físicamente la información temporal (estados).
- **Servicio de gestión de estados:** Este servicio es el encargado de la gestión del almacén de estados.
- **Servicio consumidor de estados:** Se denomina de esta manera al servicio que solicita el acceso al almacén de estados.

Cuando un servicio necesita almacenar información temporal que puede necesitar en un futuro, o requiere información de estado almacenada con anterioridad, solicita al router de estados esa información. El router envía este mensaje al servicio de

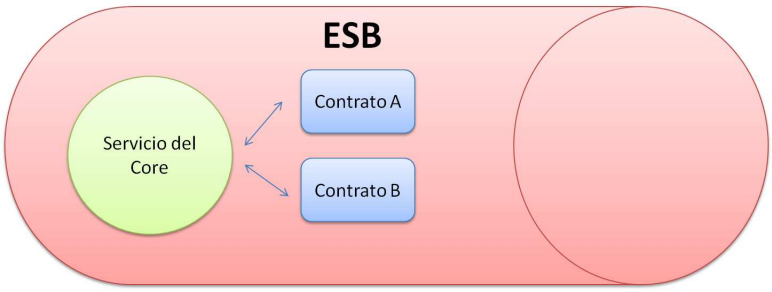

Figura 6.22: Esquema del núcleo básico en un ESB

gestión de estados, que realiza la actividad solicitada como propietario del almacén de estados.

En el caso en que la solicitud implicara una lectura de información, este último servicio la envía al router de estados que a su vez la envía al servicio consumidor de estados que la solicitó.

6.1.4.9. Núcleo básico

La adaptación de esta necesidad de interoperabilidad a un ESB se realiza mediante la aplicación directa del patrón *SOA Service Facade*.

Por lo tanto, si se quieren expandir los servicios definidos en el ESB como *core* bastaría con definir los nuevos contratos y agregárselos a los mismos (figura 6.22).

6.2. Implementación completa de la IC2A sobre JBoss ESB y validación de los modelos de interoperabilidad de referencia

En esta sección se muestra cómo trasladar las propuestas realizadas en la sección anterior a un ESB concreto, demostrando la posibilidad de implementar la arquitectura IC2A en un escenario real. Esta implementación se utilizará también para validar la definición de los dos IRMs para sistemas C2 y simuladores constructivos propuestos en el capítulo anterior.

6.2.1. Justificación de la elección de JBoss ESB

El ESB escogido para la implementación de la arquitectura IC2A es JBoss ESB. Los criterios que se han seguido en esta investigación para escoger dicho ESB son los siguientes:

- **Basado por completo en el concepto de arquitectura ESB.** La implementación del ESB seleccionado para la validación de IC2A debe diferir lo menos posible del concepto teórico de ESB empleado para realizar todas las propuestas realizadas en este capítulo.

  En muchas ocasiones las implementaciones de los productos tipo ESB incorporan módulos que intentan hacer transparentes para los usuarios y desarrolladores ciertas características del ESB, con el objeto de facilitar su utilización. Aunque este tipo de implementaciones de ESB pueden ser útiles desde el punto de vista empresarial, no lo son para los propósitos de esta tesis doctoral, ya que en los escenarios objeto de investigación son necesarias todas las funcionalidades y características de una arquitectura ESB pura, sin simplificaciones ni capas de abstracción extra.

- **Código abierto.** Es imprescindible tener acceso al código de la arquitectura ESB para analizar el funcionamiento de todos sus componentes en detalle y para modificar, si fuera necesario, aquellos que tengan relación con la arquitectura IC2A.

- **Ampliamente extendido.** La validación de la arquitectura IC2A debe realizarse mediante la implementación sobre un ESB conocido por la comunidad, utilizado en diferentes sectores y sin necesidad de requerimientos específicos (es decir, de uso general).

- **Basado en los estándares habituales.** Por motivos similares, la validación de la arquitectura IC2A debe realizarse sobre la pila de estándares habitual. Los ESB suelen estar basados en los estándares asociados a los web services, aunque en algunos casos se desarrollan protocolos específicos para ciertos sectores y/o fabricantes, son este tipo de buses los que quedan descartados para las necesidades de esta investigación.

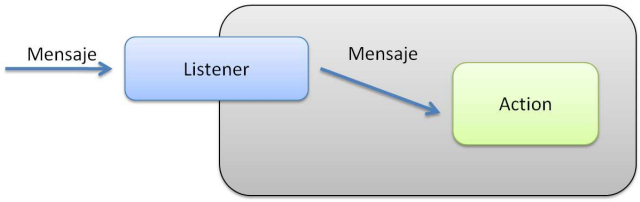

Figura 6.23: Esquema de un servicio genérico en JBoss ESB ([143])

6.2.2. Descripción de la arquitectura de JBoss ESB

JBoss ESB se concibe como un conjunto de servicios que intercambian mensajes entre ellos de manera horizontal, es decir, con una arquitectura de tipo bus ([140], [141], [142]).

Un servicio en JBoss ESB está compuesto por los siguientes elementos ([143]) mostrados en la figura 6.23:

- **Listener.** Es el componente a través del cual un servicio puede ser llamado o invocado por otro servicio o por un cliente (*service consumer*). Este componente es denominado habitualmente *end point* al ser la frontera del servicio con el resto de componentes.

  JBoss ESB incorpora en su núcleo los siguientes listeners ([144]): Http(s), FTP, File, JMS, Email, SQL, Hibernate, Socket, SOAP y WEB.

- **Action.** Este componente es el encargado de realizar las acciones propias para las que está diseñado un servicio ([145]).

  Por ejemplo, puede encargarse de realizar funciones como enviar un mensaje a otro servicio o acceder a una base de datos.

- **Mensaje.** Es la información que se intercambian dos servicios de manera horizontal a través del ESB.

  Cuando un mensaje llega a un servicio lo hace a través de un listener que entiende el formato en el que se envió el mensaje (por ejemplo, si el mensaje sigue el formato de JMS lo recibe el listener JMS). Una vez recibido por el servicio adecuado, el mensaje fluye a través de las diferentes *Actions* que pueden modificarlo o reenviarlo a otros servicios.

La arquitectura de JBoss ESB es muy flexible y está formada por un núcleo (JBoss ESB core) que puede ser ampliado según las necesidades de los usuarios. Los

estándares de mayor interés para esta tesis doctoral que están soportados por JBoss ESB son:

- **JMS (Java Message Service).** Es un middleware orientado a mensajes (MOM) desarrollado por Sun Microsystems. JBoss ESB emplea este estándar para el envío de mensajes entre los servicios conectados al bus ([146]).
- **SOAP (Simple Object Access Protocol).** Es un estándar definido por la organización W3C que especifica la manera en la que dos objetos ejecutándose en procesos diferentes pueden comunicarse mediante el intercambio de ficheros estándar XML ([147]).
- **XSLT (eXtensible Stylesheet Language Transformations).** Es un estándar desarrollado por la organización W3C. Este lenguaje tiene como finalidad la de definir un vocabulario XML para especificar cómo un fichero XML se puede transformar en otro fichero XML ([138]).
- **XPATH (XML Path Language).** Es un estándar definido por la organización W3C. Permite generar expresiones que pueden recorrer de manera diferente un fichero XML. De esta manera se pueden definir diferentes modos de transformación de un fichero XML ([148]).
- **JUDDI (Java Universal Description, Discovery and Integration).** Esta implementación Java del estándar UDDI de la organización OASIS permite mantener un catálogo de servicios de manera que se publican en forma de registros y pueden ser descubiertos por los diferentes sistemas que necesitan interactuar de alguna manera con ellos ([149]).
- **JBPM (Java Business Process Management).** Es un sistema flexible y extensible de administración de flujo de procesos de negocio basado en Java ([150], [151]). Permite expresar de manera sencilla los procesos de negocio en términos de tareas, acciones, tiempos de espera, temporizadores, etc.

6.2.3. Implementación y validación de los IRMs

6.2.3.1. Consideraciones previas

JBoss ESB, por definición, está formado únicamente por servicios. Por lo tanto, para aplicar los modelos de interoperabilidad de referencia 1 y 2 definidos en el

capítulo anterior es necesario clasificar los diferentes servicios que lo componen en diferentes grupos:

- **Servicios que se comunican con los simuladores constructivos.** Estos servicios son los que intercambian mensajes directamente con los simuladores constructivos. En este grupo de servicios además de los listeners ya definidos en JBoss ESB se debe definir un listener de HLA (este listener es necesario para una conexión nativa con los simuladores constructivos que implementan el estándar IEE1516).

- **Servicios que se comunican con los sistemas C2.** Estos servicios son los que se comunican directamente con los sistemas C2. En este grupo de servicios además de los listeners ya definidos en JBoss ESB se debe definir un listener de DEM (este listener es necesario para una conexión nativa con los sistemas C2 que emplean el estándar de comunicación definido en MIP, DEM).

- **Servicios que se comunican con otros servicios.** En este grupo se engloban aquellos servicios que se comunican con otros servicios del ESB y por lo tanto, los listeners definidos de manera nativa en JBoss ESB son suficientes para ellos.

6.2.3.2. Especificación de las actividades comunes en JBoss ESB

En una arquitectura IC2A se definen como actividades comunes a los sistemas C2 y los simuladores constructivos aquellas que son realizadas para gestionar los teatros (actividades incluidas dentro del módulo básico del IC2F), ya que son imprescindibles en los dos modelos de interoperabilidad de referencia.

Desde este punto de vista, un teatro se puede considerar como un repositorio de información de estado o información temporal. Por lo tanto su modelo se puede asemejar al diseñado para la necesidad de interoperabilidad de repositorio de estado.

Para diseñar la gestión de teatros dentro de JBoss ESB se definen dos servicios con la siguiente composición y funciones (figura 6.24):

- **Servicio módulo base.** Este servicio es el responsable de recibir todas las llamadas de los sistemas C2 y simuladores constructivos a las interfaces definidas en el módulo base del IC2F. Está compuesto de dos listeners (uno HLA y otro DEM) para recibir solicitudes de sistemas C2 y de simuladores constructivos.

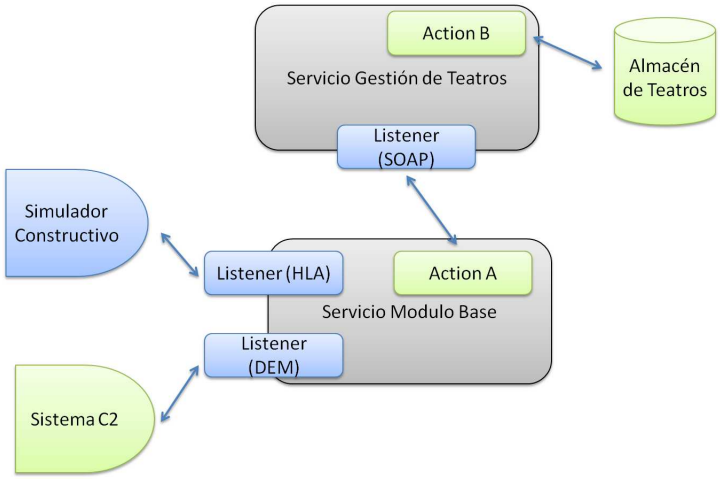

Figura 6.24: Esquema de la gestión de teatros dentro de un ESB

Además incluye *Actions* para gestionar las diferentes interfaces del IC2F en relación a la gestión de teatros.

- **Servicio de gestión de teatros.** Este servicio se encarga de la gestión de la información de los teatros en ejecución, que se almacena en un repositorio de estados llamado almacén o repositorio de teatros. Por lo tanto sus funciones son, por ejemplo, insertar un nuevo teatro, devolver la información de un teatro, devolver una lista con los teatros existentes o almacenar que un actor pertenece a un teatro.

  Para ello dispone un listener que recibe el mensaje que proviene del servicio módulo base y lo envía a una *Action* que realiza la operación solicitada en el almacén de teatros (figura 6.24).

Cabe destacar una diferencia respecto a la especificación de la necesidad de interoperabilidad repositorio de estado para un ESB teórico: el elemento de enrutado (router de gestión de estados) no existe ya que su función la asume el elemento *Action* del servicio de módulo base.

6.2.3.3. Especificación de las actividades para el IRM tipo 1

Las actividades específicas realizadas en este modelo de interoperabilidad se pueden agrupar según correspondan a las realizadas por los simuladores constructivos o a las realizadas por los sistemas C2.

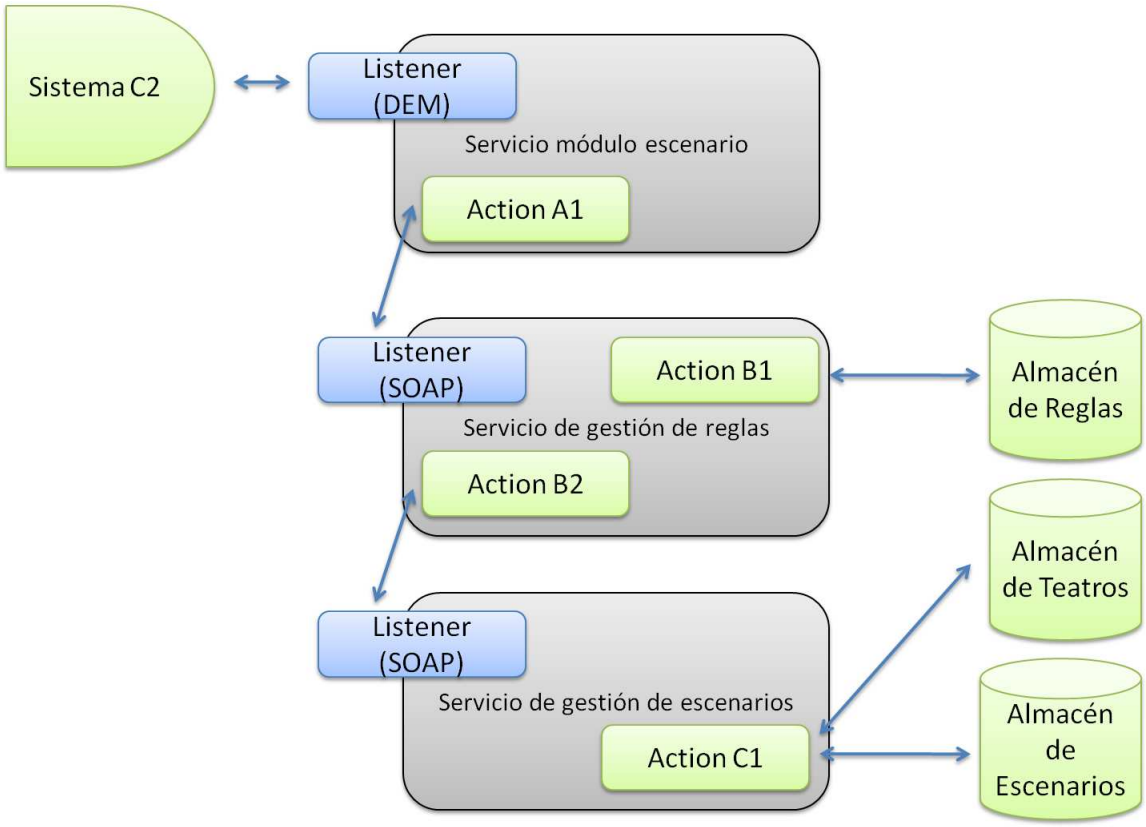

Figura 6.25: Esquema de la publicación de escenarios en un teatro IC2A

6.2.3.3.1. Actividades de los sistemas C2. En este grupo se engloban las actividades que realizan los sistemas C2 para publicar un escenario en un teatro.

Para gestionar la publicación de escenarios de un sistema C2 en un teatro se requieren los siguientes tipos de servicios definidos para JBoss ESB (figura 6.25):

- **Servicio módulo escenario.** Este servicio definido para JBoss ESB es el encargado de recibir los mensajes de los sistemas C2 para publicar un escenario a través de un listener DEM. Y después de procesarlos, se envían al servicio de gestión de reglas.

- **Servicio de gestión de reglas.** Este servicio comprueba que la información del escenario que se va a publicar cumple las reglas de negocio definidas para él.

  Este servicio se compone de un listener SOAP que recibe los mensajes y los envía a un servicio que lee las reglas de aplicación al escenario en el almacén de

reglas de negocio. Luego comprueba que la información del escenario cumple las reglas que le envía servicio módulo escenario.

Si el escenario cumple las reglas de negocio, es enviado al servicio de gestión de escenarios para su publicación en el teatro.

- **Servicio de gestión de escenarios.** Este servicio almacena el escenario y el teatro en el que está publicado en el repositorio de escenarios y de teatros.

  Este servicio se compone de un listener SOAP que recibe un mensaje que envía información de un escenario. La *Action* es la que se encarga de almacenar la información enviada en el repositorio de escenarios y en el de teatros respectivamente.

6.2.3.3.2. Actividades de los simuladores constructivos. En este grupo se engloban todas las actividades que realiza un simulador constructivo para leer un escenario publicado en un teatro.

El simulador constructivo lee el escenario con un protocolo diferente (HLA) del que se utilizó para su publicación. La convergencia entre protocolos se consigue con JBoss ESB, empleando servicios con listeners diferentes para cada tipo de sistema.

Para gestionar la lectura de un escenario por parte de un simulador constructivo se diseñan en JBoss ESB los siguientes tipos de servicios (figura 6.26):

- **Servicio módulo escenario.** Este módulo recibe un mensaje solicitando información sobre un escenario específico por parte de un simulador constructivo.

  Un listener HLA recibe este mensaje y lo envía a una *Action* que lo procesa y lo reenvía al servicio de gestión de escenarios.

- **Servicio de gestión de escenarios.** Este servicio recibe a través del listener SOAP un mensaje en el que se solicita información sobre un escenario concreto.

  Este mensaje se envía a una *Action* que comprueba que el simulador constructivo que la solicita es miembro del teatro en el que este escenario está publicado. Si es así, se envía un mensaje al servicio de gestión de esquemas.

- **Servicio de gestión de esquemas.** El objetivo de este servicio es comprobar que el esquema en el que se ha solicitado la información del escenario existe en el almacén de esquemas (aquí se encuentran todos los esquemas datos de alta en la IC2A) y transformarlo, en el caso en el que sea necesario, según ese esquema.

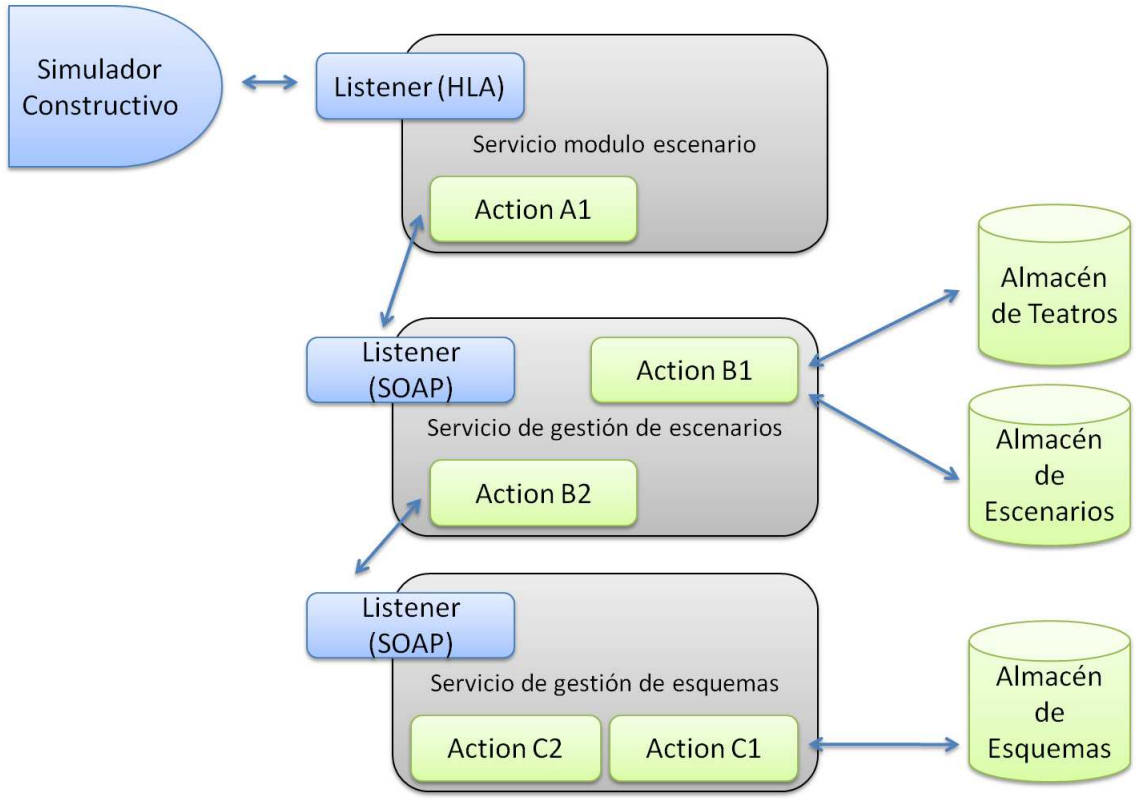

Figura 6.26: Esquema de la lectura de escenarios en un teatro IC2A

En este caso el listener SOAP recibe el mensaje donde se envía la información del escenario y del esquema en el que se quiere recibir. Una *Action* comprueba que el esquema existe en el almacén de esquemas. Y la otra *Action* transforma la información del escenario según el esquema solicitado.

6.2.3.4. Especificación de las actividades para el IRM tipo 2

Las actividades realizadas para la ejecución de este modelo de interoperabilidad se pueden dividir en tres grupos: las realizadas por los sistemas C2 para publicar una o varias líneas de acción, las realizadas por los sistemas C2 para leer la información de las matrices de evaluación de esas líneas de acción y las realizadas por los simuladores constructivos para publicar las matrices de evaluación una vez simulada la línea de acción.

6.2.3.4.1. Actividades de los sistemas C2 para publicar líneas de acción. En este grupo se engloban las actividades que realizan los sistemas C2 para pu-

blicar las líneas de acción que es necesario que sean evaluadas por los simuladores constructivos.

Para gestionar la publicación de líneas de acción de un sistema C2 en un teatro se va a definir los siguientes tipos de servicios en JBoss ESB (figura 6.27):

- **Servicio módulo de líneas de acción.** Este servicio recibe los mensajes enviados por los sistemas C2 solicitando la publicación de una línea de acción en un teatro.

  Este servicio se compone del listener DEM que recibe el mensaje con la línea de acción y el teatro en el se va a publicar. Este elemento envía a la *Action* la línea de acción, que es procesada y enviada al servicio de gestión de reglas.

- **Servicio gestión reglas.** Este servicio comprueba que la línea de acción respeta las reglas almacenadas en el almacén de reglas que están definidas para ella.

  Por lo tanto el listener SOAP recibe el mensaje con la línea de acción. A continuación una *Action* lee en el almacén de reglas las reglas relacionadas con esta línea de acción y otra *Action* comprueba que las cumple.

  En el caso en que cumpla las reglas de negocio, se envía un mensaje al servicio de gestión de líneas de acción.

- **Servicio gestión líneas de acción.** Este servicio tiene dos objetivos. El primero de ellos es almacenar en el almacén de líneas de acción y en el de teatros la línea de acción publicada en un teatro.

  Y el segundo objetivo es comprobar que los esquemas de las matrices de evaluación que están asociadas a la línea de acción coinciden con alguno de los almacenados en el almacén de esquemas.

  Para ello el servicio recibe un mensaje a través del listener SOAP y una *Action* comprueba que los esquemas de las matrices de evaluación asociadas coinciden con las almacenadas en el almacén de esquemas. A continuación otra *Action* almacena la línea de acción y el teatro asociado en los almacenes de líneas de acción y de teatros respectivamente.

6.2.3.4.2. Actividades de los simuladores constructivos para publicar las matrices de evaluación. En este grupo se engloban las actividades que realizan

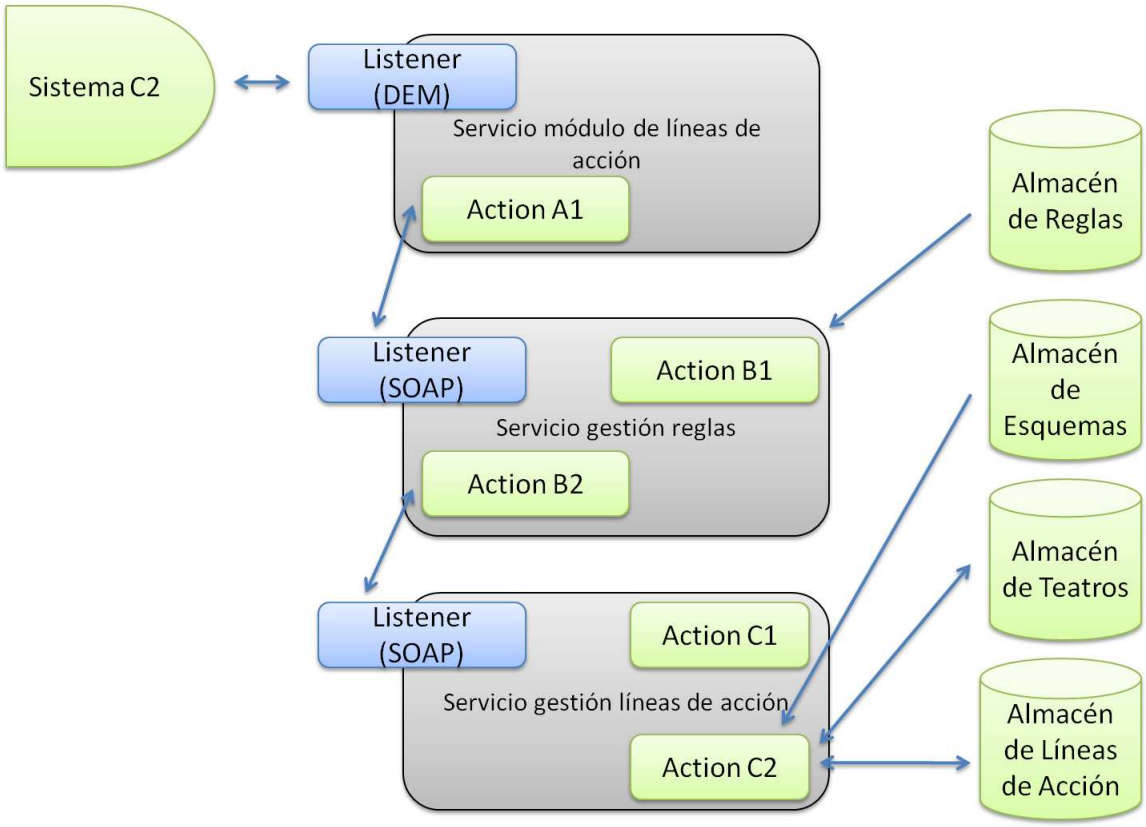

Figura 6.27: Esquema de la publicación de líneas de acción en un teatro IC2A

los simuladores constructivos para publicar las matrices de evaluación resultado de evaluar las líneas de acción.

Para gestionar la publicación de las matrices de evaluación de un simulador constructivo en un teatro se requieren los siguientes tipos de servicios definidos para JBoss ESB (figura 6.28):

- **Servicio módulo de evaluación líneas de acción.** Este servicio recibe los mensajes enviados por los simuladores constructivos a través de un listener HLA. Una *Action* procesa este mensaje y lo envía al servicio de gestión de reglas.

- **Servicio de gestión de reglas.** Este servicio comprueba que la matriz de evaluación cumple las reglas de negocio definidas para ella y está conforme a un esquema de información definido para las matrices de evaluación de una línea de acción.

  Para ello contiene un listener SOAP que recibe el mensaje. Una *Action* lee

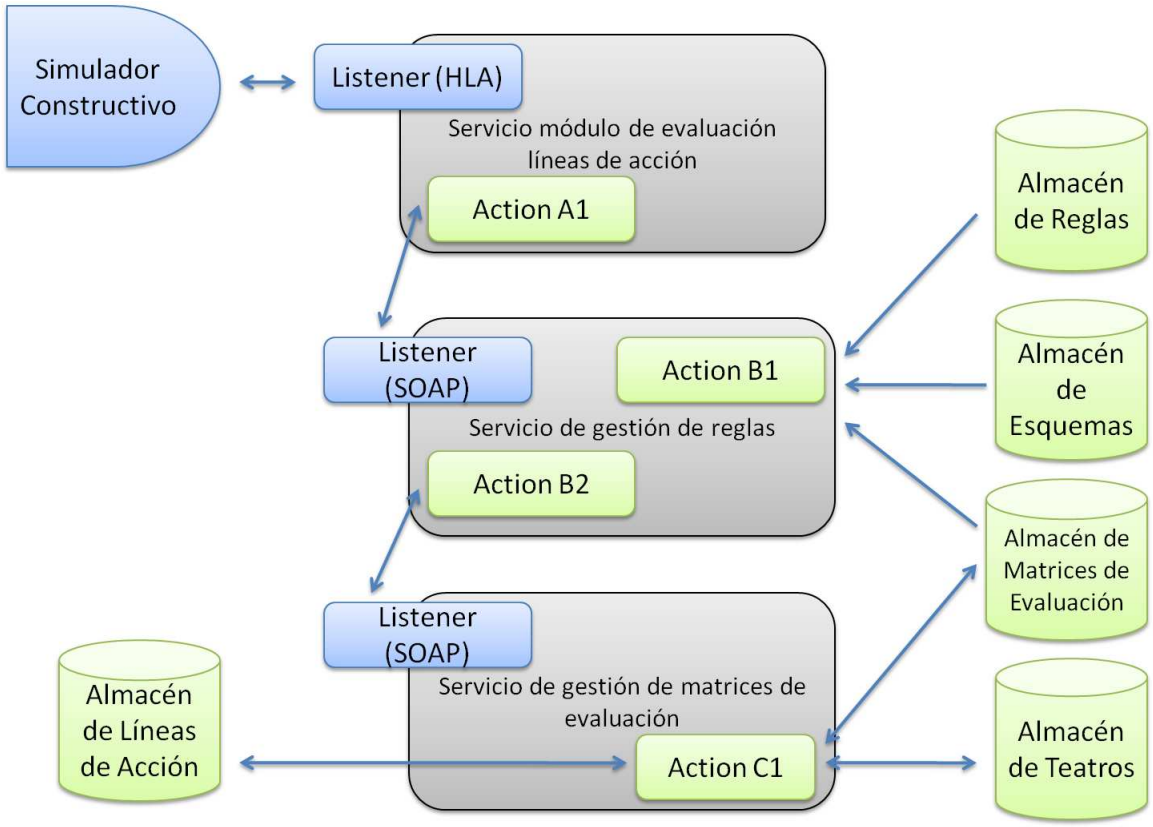

Figura 6.28: Esquema de la publicación de matrices de evaluación en un teatro IC2A

los esquemas y reglas de negocio asociados a esa matriz de evaluación (en el almacén de reglas de negocio y en el de esquemas).

La otra *Action* comprueba que la matriz cumple las reglas de negocio y está estructurada según un esquema definido para esa línea de acción.

- **Servicio de gestión de matrices de evaluación.** Este servicio almacena la matriz de evaluación que está asociada a un teatro y a una línea de acción.

  Está compuesto de un listener SOAP que recibe el mensaje y una *Action* que guarda la matriz de evaluación asociada a una línea de acción.

6.2.3.4.3. Actividades de los sistemas C2 para leer las matrices de evaluación asociadas a una línea de acción. En este grupo se engloban las actividades que realizan los sistemas C2 para leer las matrices de evaluación resultado de evaluar las líneas de acción publicadas.

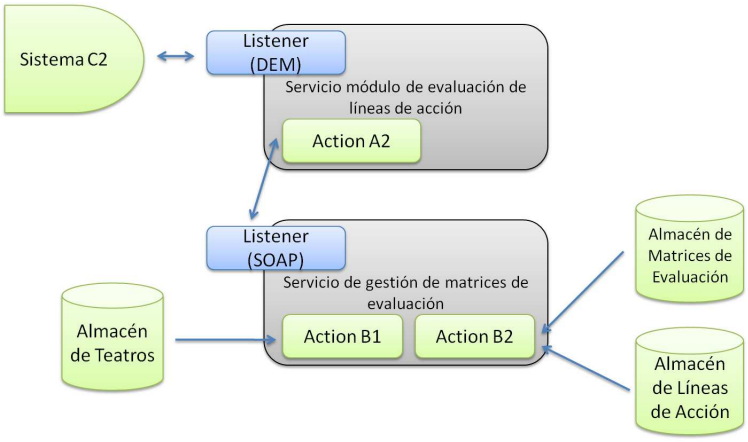

Figura 6.29: Esquema de la lectura de matrices de evaluación en un teatro IC2A

Para gestionar la lectura de las matrices de evaluación de un sistema C2 en un teatro se requieren los siguientes tipos de servicio definidos para JBoss ESB (figura 6.29):

- **Servicio módulo de evaluación de líneas de acción.** Este servicio se compone de un listener DEM que recibe el mensaje de solicitud de lectura de una matriz de evaluación asignada a una línea de acción.

  Posteriormente existe una *Action* que procesa este mensaje y lo envía al servicio de gestión de matrices de evaluación.

- **Servicio de gestión de matrices de evaluación.** Este servicio tiene como finalidad realizar una lectura de una matriz de evaluación en el almacén de matrices de evaluación.

  Para ello se compone de un listener SOAP que recibe un mensaje de solicitud de información sobre una matriz de evaluación asignada a una línea de acción.

  Luego existe una *Action* que comprueba que el simulador constructivo solicitante pertenece al teatro donde la matriz está publicada. Otra *Action* accede al almacén de matrices de evaluación para leer la matriz de evaluación solicitada (asignada a una línea de acción).

Capítulo 7

Conclusiones y líneas de investigación futuras

Para terminar, se presentan las conclusiones más relevantes que se pueden extraer de la presente tesis doctoral. Existen unas conclusiones generales que provienen del estudio teórico realizado y del análisis de todos los resultados experimentales obtenidos. El resto de las conclusiones son particulares de cada uno de los capítulos incluidos en el presente documento, por tanto, de cada una de las fases o etapas de la investigación realizada.

7.1. Conclusiones generales

La conclusión más importante que se puede extraer de esta tesis doctoral es la confirmación de la hipótesis de partida presentada en la sección 1.2. Es decir, se ha demostrado que es posible proponer modelos, metodologías y herramientas que resuelvan los problemas de interoperabilidad entre simuladores constructivos de aplicación militar entre sí, y entre estos y los sistemas de mando y control.

Para llegar a esta conclusión se ha realizado un análisis exhaustivo del problema de la interoperabilidad, primero entre simuladores constructivos entre sí, y después entre este tipo de sistemas y los sistemas C2. Y se ha verificado que la metodología propuesta en el capítulo 1 es adecuada para confirmar la hipótesis propuesta.

En el caso de ambos tipos de sistemas de apoyo a la conducción de operaciones militares se ha comenzado por realizar un estudio de los posibles escenarios de aplicación en los que la interoperabilidad entre estos sistemas puede ser necesaria. A continuación se ha realizado un análisis exhaustivo de la solución que se ha dado

a estos problemas, tanto en entornos de aplicación militar como en otro tipo de entornos (especialmente, industriales y empresariales, por las similitudes entre los sistemas de información utilizados en estos contextos y el militar).

Tomando como punto de partida estas soluciones previas y teniendo en cuenta las características y necesidades específicas de los entornos de conducción de operaciones militares, ha sido posible definir y proponer modelos, metodologías y herramientas específicas para estos entornos. Todas las propuestas realizadas en esta tesis doctoral cumplen con un requisito imprescindible, son compatibles con el estándar IEEE1516 (HLA), la especificación JC3IEDM, los estándares CBML y MSDL, y con otros estándares de facto en integración de sistemas como XML o las arquitecturas SOA.

Además se ha verificado que las soluciones propuestas son aplicables en escenarios reales y que permiten que los sistemas interoperen mediante el concepto de federación de los sistemas entre sí, no mediante la utilización de arquitecturas de tipo middleware que actúen como meras pasarelas de intercambio de datos.

7.2. Interoperabilidad entre simuladores constructivos de aplicación militar

Como se analizó en el capítulo 2 la arquitectura HLA permite, en teoría, la interoperabilidad entre los simuladores que se desarrollen conforme a su especificación. Esto incluye a los simuladores constructivos de aplicación militar, y sin embargo, es muy difícil conseguir en ejercicios reales el grado de interoperabilidad necesario en las operaciones de las fuerzas armadas actuales.

Por este motivo se ha analizado en el capítulo 3 la posibilidad de adaptar los modelos de interoperabilidad de referencia ó IRMs, definidos por SISO para estandarizar la manera de utilizar el estándar IEEE1516 en entornos industriales, al contexto de la simulación constructiva.

Para realizar esta adaptación se han tomado como punto de partida las necesidades específicas que impone el entorno táctico-militar a este tipo de simuladores y el grado de cobertura que tendrían estos IRMs sobre el modelo de referencia JC3IEDM.

Se ha llegado a la conclusión de que es viable adaptar los IRMs de aplicación industrial a simuladores constructivos militares con un alto grado de cobertura inicial sobre el JC3IEDM, por lo que se han definido modelos militares tipo A, B, C y D equivalentes a los industriales. Pero se ha tenido que definir un nuevo tipo de IRM, el

tipo E, para incluir el escenario de interoperabilidad en el que se comparte/transfiere un Plan/Orden.

Además se ha incorporado a la definición de los modelos de referencia el concepto de rol de un federado, de manera que se puedan gestionar escenarios en los que intervengan más de dos federados. También se ha analizado el módulo *Time Management* del estándar IEEE1516, llegando a la conclusión de que la división de los IRMs de aplicación militar en eventos es una solución idónea para los simuladores constructivos, y que es posible realizarla gracias a la utilización de las funcionalidades de este módulo. Se ha comprobado que su utilización no introduce ningún problema de latencia en la ejecución de las simulaciones ya que el concepto de tiempo real en simulación constructiva no exige cumplir con restricciones del tiempo por debajo del orden del minuto.

Por ello se han definido los IRMs para simuladores constructivos militares especificando los eventos que los componen y las actividades que realiza cada rol (cada tipo de federado) en cada uno de ellos. Dadas las especiales características del entorno militar, en la definición de los modelos se han incorporado mecanismos de tolerancia a fallos que permitan comprobar la integridad de los datos compartidos y/o transferidos en los diferentes modelos.

Las conclusiones de este capítulo se han publicado de manera parcial en los siguientes trabajos: [152], [153], [154], [155].

7.3. Modificación de una RTI real para la ejecución automática de los IRMs militares

En el capítulo 4 se han validado los modelos definidos en el capítulo anterior realizando una propuesta de implementación concreta sobre una RTI real que cumple el estándar IEEE1516.

Para ello se han definido unos criterios de selección de esta RTI (teniendo en cuenta el contexto de aplicación) y siguiendo estos criterios, se ha escogido Pórtico como RTI para esta prueba de concepto.

A continuación, para cada uno de los cinco tipos de IRMs definidos, se han propuesto las modificaciones que se deben introducir en los componentes de Pórtico para asegurar que los federados ejecutan las actividades definidas en cada evento según su rol. De esta manera se garantiza que los federados participantes en un

IRM realizan las actividades esperadas de forma automática y que por lo tanto, no existen ambigüedades en su forma de interoperar cuando se suman a una simulación distribuida.

Gracias a este estudio, se ha validado que es posible modificar una RTI real para aplicar los modelos de interoperabilidad de referencia definidos para el contexto militar. Además esta modificación no es compleja, basta con crear y/o modificar ciertas clases y con añadir nuevas funcionalidades a los componentes de Pórtico ya existentes. Estas funcionalidades son en la mayoría de los casos comprobaciones que permiten controlar la secuencia de eventos de cada IRM y, en el caso de encontrar errores, interrumpir esta secuencia y no dejar que avance la ejecución del modelo.

Las conclusiones de este capítulo han permitido codirigir el Trabajo Fin de Máster [156].

7.4. Interoperabilidad entre simuladores constructivos y sistemas C2

En el capítulo 5 se han identificado los casos habituales en los que es necesaria la interoperabilidad entre sistemas C2 y simuladores constructivos militares. Estos casos de interoperabilidad se han resumido en dos: la generación por parte de un sistema C2 de un escenario real para un simulador constructivo y la evaluación por parte de un simulador constructivo (mediante la creación de una matriz de evaluación) de una línea de acción planeada por un sistema C2.

No existe ninguna arquitectura de interoperabilidad (de la misma manera que HLA para los simuladores constructivos) que permita resolver estos casos de interoperabilidad de la manera deseada. Por este motivo se han concretado a bajo nivel las necesidades de interoperabilidad que estos casos introducen. Los nueve aspectos imprescindibles para conseguir la interoperabilidad deseada entre simuladores constructivos y sistemas C2, han sido el punto de partida para la propuesta de la arquitectura IC2A (Interoperability C2 Architecture).

Esta arquitectura se describe con detalle en el capítulo 5, comenzando por unas reglas básicas para su definición y continuando por las plantillas de datos e interfaces que dan cobertura a las nueve necesidades de interoperabilidad especificadas.

Pero esta arquitectura plantea los mismos retos que HLA: los dos casos de interoperabilidad posibles se pueden resolver de diferente manera utilizando para ello

la IC2A. Por eso se han definido dos modelos de interoperabilidad de referencia o IRMs para los dos casos identificados. De esta forma, el IRM de tipo 1, especifica cómo resolver la generación por parte de un sistema C2 de un escenario real para un simulador constructivo utilizando para ello la arquitectura IC2A. Y el IRM de tipo 2, define cómo llevar a cabo evaluación por parte de un simulador constructivo de una línea de acción planeada por un sistema C2 utilizando la arquitectura IC2A y sus componentes para ello.

Con las herramientas, modelos y metodologías definidos en este capítulo, se garantiza la interoperabilidad parcial (para los casos 1 y 2) entre simuladores constructivos y sistemas C2. Y se podría conseguir un mayor grado de interoperabilidad que resolviera nuevos casos en el futuro de manera sencilla gracias a la escalabilidad funcional de la solución propuesta.

7.5. Aplicación práctica de la arquitectura IC2A

En el capítulo anterior se define la arquitectura IC2A, pero no se especifica una arquitectura software real adecuada para su implementación. En este capítulo se ha comenzado por analizar unos criterios de selección válidos para el entorno de la conducción de operaciones militares en la actualidad.

Siguiendo estos criterios de selección y teniendo en cuenta el tipo de arquitectura definida en el capítulo anterior, se ha llegado a la conclusión de que una arquitectura de tipo ESB (Enterprise Service Bus) es la mejor opción para implementar la arquitectura IC2A.

Como esta implementación plantea bastantes retos, se ha decidido además recurrir a los patrones SOA (ya que es posible utilizarlos al ser un ESB una arquitectura orientada a servicios) de diseño para reutilizar y aprovechar el conocimiento previo de otros diseñadores en la resolución de las necesidades de interoperabilidad. De esta manera se acude además, a un estándar de facto en el diseño con arquitecturas orientadas a servicios (los patrones son universalmente conocidos y eliminan ambigüedades), que puede ayudar a extender la arquitectura IC2A entre las fuerzas armadas de diferentes países.

Una vez establecido el marco de trabajo teórico más adecuado, se ha seleccionado una implementación comercial basada en ESB, en concreto, JBoss ESB, para realizar la prueba de concepto que permita validar la arquitectura IC2A definida en el capítulo anterior y los IRMs para simuladores constructivos y sistemas C2. Y se ha

demostrado que es posible resolver los casos de interoperabilidad 1 y 2 cumpliendo con los requisitos impuestos en el capítulo 1 de esta tesis doctoral.

7.6. Líneas de trabajo futuro

Teniendo en cuenta todas las conclusiones presentadas, se han cumplido los objetivos que se propusieron en el capítulo 1.

Sin embargo, a partir de la tesis doctoral realizada y de estas conclusiones, surgen nuevas líneas de investigación muy interesantes para completar y ampliar el trabajo realizado. Algunas de estas líneas son:

- Estudiar si los modelos de interoperabilidad de referencia definidos para simuladores constructivos son de aplicación a otros niveles de simulación militar como la virtual o en vivo.
- Estudiar si es viable definir modelos de interoperabilidad de referencia para la convergencia entre simuladores de diferentes niveles.
- Estudiar si es posible que la arquitectura IC2A integre simuladores de diferentes niveles (virtual o en vivo) con los sistemas C2. Y si es posible, definir los modelos de interoperabilidad de referencia adecuados.
- Investigar si la arquitectura IC2A puede adaptarse a otros campos tácticomilitares donde la interoperabilidad sea una necesidad (es decir, incluir en la arquitectura otro tipo de sistemas además de los C2 y los simuladores constructivos).
- Trasladar las conclusiones de esta tesis doctoral, y adaptarlas si fuera necesario, a entornos civiles (emergencias, hospitalario, industrial, empresarial, etc). En esta dirección se han dado ya primeros pasos como el del trabajo [157].
- Analizar el impacto que puede tener la seguridad de la información en los IRMs definidos para simuladores constructivos y en la arquitectura IC2A.
- Proponer mecanismos de depuración de errores y auditoría dentro de los propios modelos de interoperabilidad de referencia.
- Analizar la posibilidad de ofrecer un simulador constructivo o una federación de ellos con el concepto de *Simulation as a Service* utilizando para ello el paradigma de computación *Cloud*.

Apéndice A

Tabla de Acrónimos

Acrónimo	Significado
ALS	Action Line Specification
APP	Allied Procedural Publications
ATCCIS	Army Tactical Command and Control Information System
BML	Battle Management Language
BOM	Based Object Model
C2	Command and Control
C2IEDM	C2 InterchangE Data Model
C3	Communication, Command and Control
CAX	Computer Assisted eXercises
CBML	Coalition Battle Management Language
CC	Conference Committee
CDDL	Common Developer and Distribution Licence
COI	Community Of Interest
COP	Common Operational Picture
COTS	Commercial Off-The-Shelf
CSPI	COTS Simulation Packages Interoperability
DARPA	Defense Advanced Research Projects Agency
DEM	Data Exchange Mechanism
DIS	Distributed Interactive Simulation
DMWG	Data Modelling Working Group
DoD	Department of Defense of United States
DSS	Data structure Shared Specification
EAI	Enterprise Application Integration

EMS	Evaluation Matrix Specification
ESB	Enterprise Service Bus
ESS	Event Shared Specification
ETS	Entity Transfer Specification
FEDEP	Federation Development and Execution Process
FIFO	First In First Out
FOM	Federation Object Model
FSE	Fire Support Element
FTP	File Transfer Protocol
GPU	Graphics Processing Unit
HLA	High Level Architecture
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IC2A	Interoperability C2 Architecture
IC2D	Interoperability C2 Data
IC2F	Interoperability C2 Framework
IC2I	Interoperability C2 Interfaces
IED	Improvised Explosive Device
IEEE	Institute of Electrical and Electronics Engineer
IEEE1516	High Level Architecture Standards
IRM	Interoperability Reference Model
JBML	Joint Battle Management Language
JBPM	Java Business Process Management
JC3IEDM	Joint C3 InterchangE Data Model
JMS	Java Message Service
JUDDI	Java Universal Description, Discovery and Integration
LC2IEDM	Land C2 InterchangE Data Model
LIFO	Last In First Out
LLD	Load List Data
LRC	Local Runtime Component
MDA	Model Driven Architecture
MEM	Messages Exchange Mechanism
MIP	Multilateral Interoperability Programme
MOLT	MIP Operational Level Test

MOM	Management Object Model o Messages Oriented Middleware
MOP	MIP Operating Procedures
MSDL	Military Scenario Definition Language
MSG	Modelling and Simulation Group
MSLT	MIP System Level Test
NATO	Noth Atlantic Treaty Organization
NDMS	NATO Discovery Metadata Specification
NMSG	NATO Modelling and Simulation Group
NRF	NATO Response Force
OCL	Object Constraint Language
OIG	Operational Information Group
OMT	Object Model Template
ORBAT	ORden de BATalla
OTAN	Organización Tratado Atlántico Norte
OWG	Operational Working Group
PDG	Product Development Group
PDU	Protocol Data Unit
PMG	Program Manager Group
PR4G	Modelo de radioteléfono de uso militar
pRTI	Portable Runtime Infrastructure
RGP	Recognize Ground Picture
RSS	Resource Shared Specification
RTI	Runtime Infrastructure
RTO	NATO Research and Technology Organization
SAC	Standards Activity Committee
SEAWG	Systems Engineering and Architecture Working Group
SIMACA	SIMulador de Artillería de CAmpaña
SISO	Simulation Interoperability Standards Organization
SISO-STD	SISO Standard
SLX	Simulation Language with eXtensibility
SOA	Services Oriented Architecture
SOAP	Simple Object Access Protocol
SOM	Simulation Object Model
SQL	Structured Query Language

STANAG	STANdardization AGreement
STD	STandarD
STR	Sistema en Tiempo Real
STS	Scenario Transfer Specification
TEWG	Test Evaluation Working Group
TM	Time Management
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URJC	Universidad Rey Juan Carlos
VBS2	Virtual BattleSpace 2
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XPATH	XML PATH Language
XSD	XML Schema Definition
XSLT	eXtensible Stylesheet Language Transformations

Apéndice B

Diagramas de tiempos de los IRMs de aplicación militar

B.1. Diagramas de tiempos de los IRMs Tipo A

B.1.1. Diagramas de tiempos del IRM subtipo A1

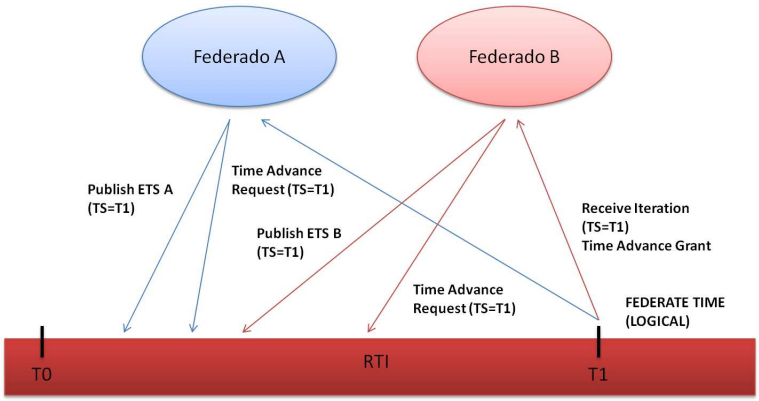

Figura B.1: Diagrama de tiempos del evento 1 en el IRM subtipo A1

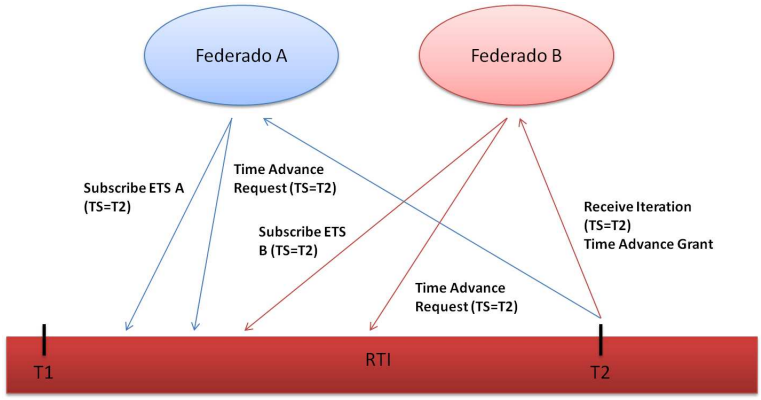

Figura B.2: Diagrama de tiempos del evento 2 en el IRM subtipo A1

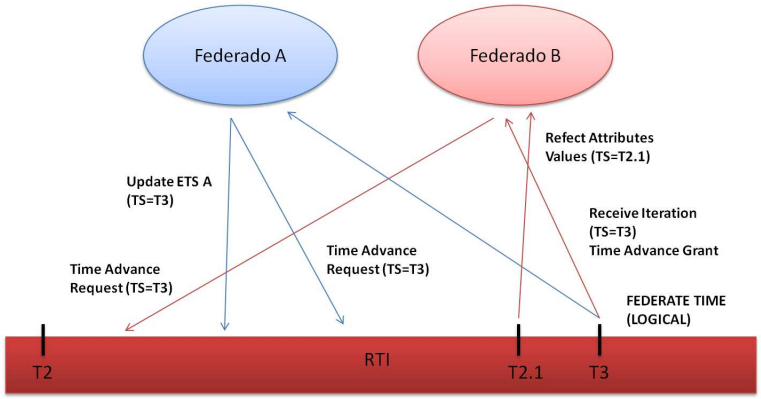

Figura B.3: Diagrama de tiempos del evento 3 en el IRM subtipo A1

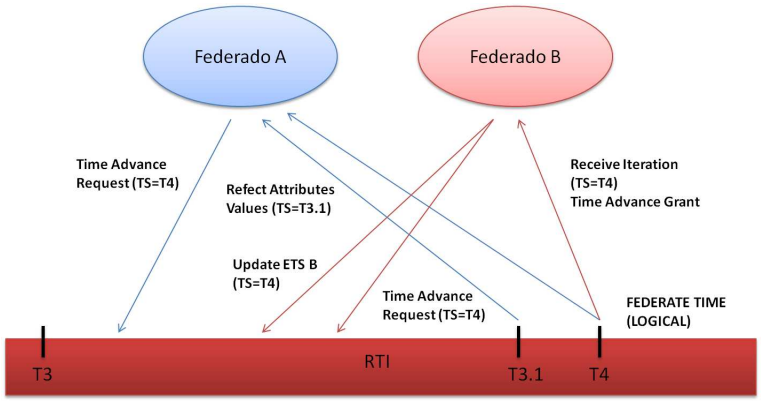

Figura B.4: Diagrama de tiempos del evento 4 en el IRM subtipo A1

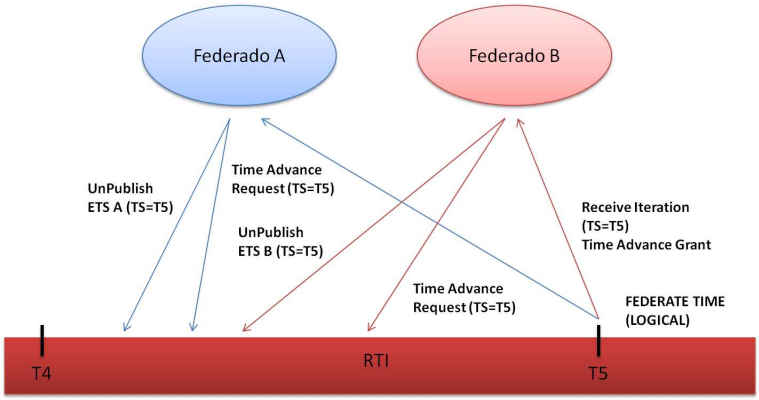

Figura B.5: Diagrama de tiempos del evento 5 en el IRM subtipo A1

B.1.2. Diagramas de tiempos del IRM subtipo A2

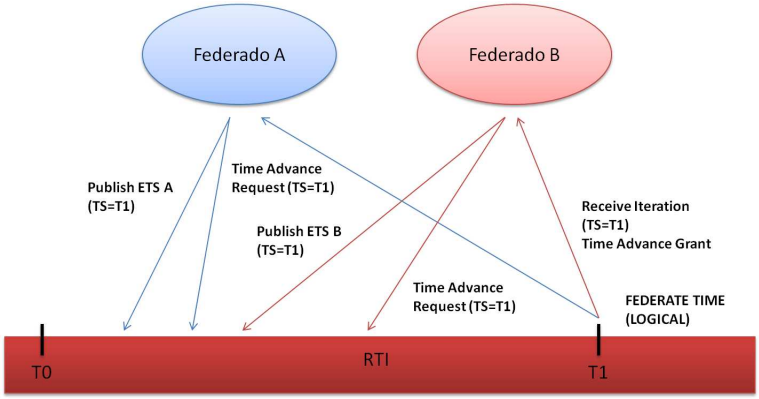

Figura B.6: Diagrama de tiempos del evento 1 en el IRM subtipo A2

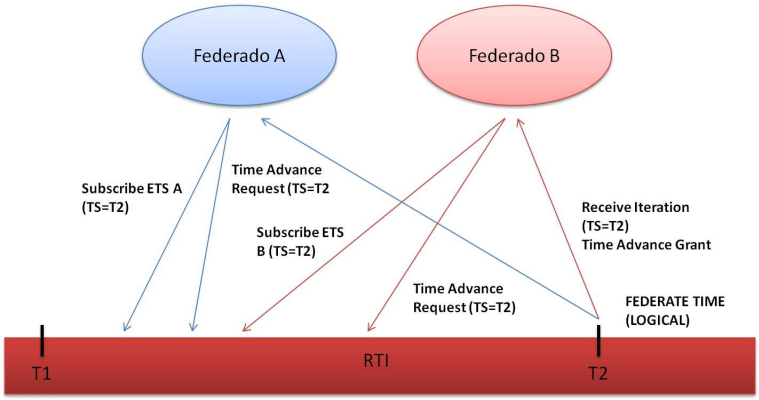

Figura B.7: Diagrama de tiempos del evento 2 en el IRM subtipo A2

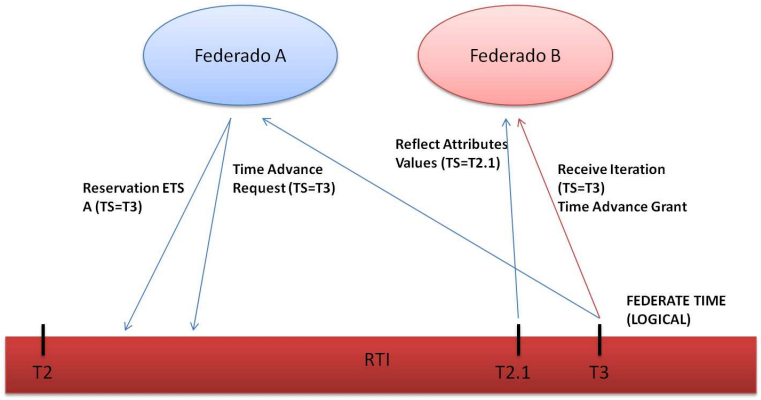

Figura B.8: Diagrama de tiempos del evento 3 en el IRM subtipo A2

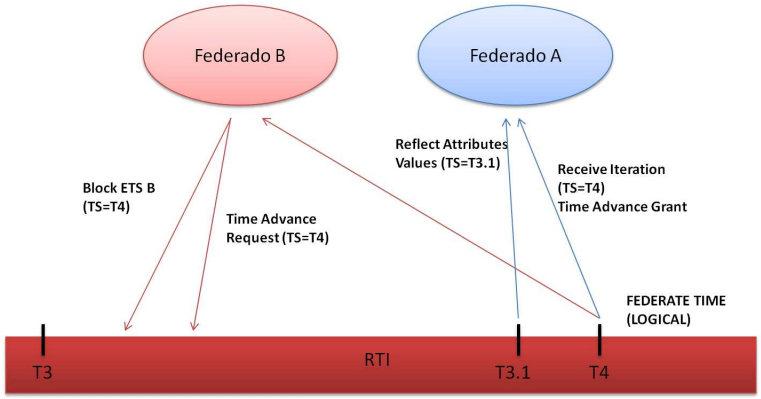

Figura B.9: Diagrama de tiempos del evento 4 en el IRM subtipo A2

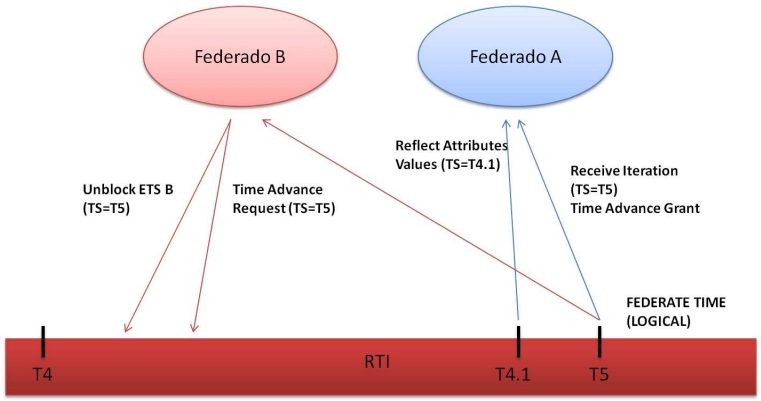

Figura B.10: Diagrama de tiempos del evento 5 en el IRM subtipo A2

B.1.3. Diagramas de tiempos del IRM subtipo A3

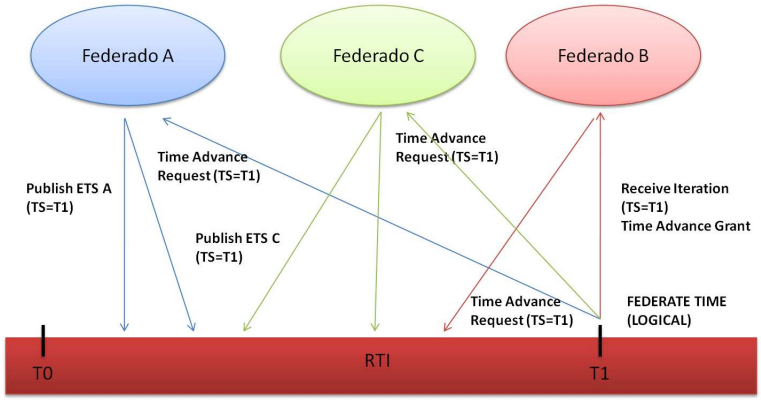

Figura B.11: Diagrama de tiempos del evento 1 en el IRM subtipo A3

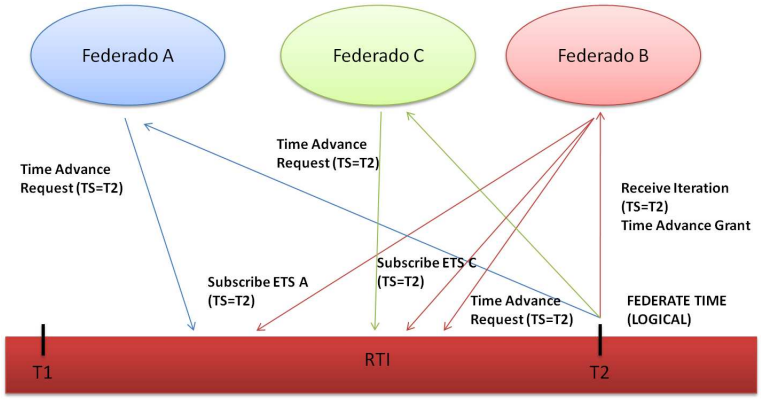

Figura B.12: Diagrama de tiempos del evento 2 en el IRM subtipo A3

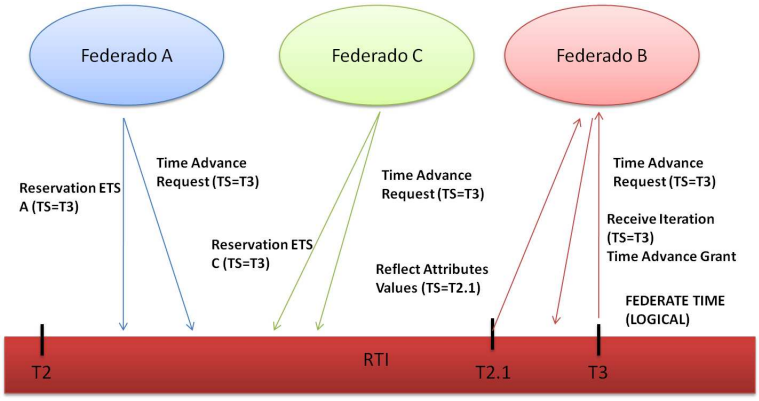

Figura B.13: Diagrama de tiempos del evento 3 en el IRM subtipo A3

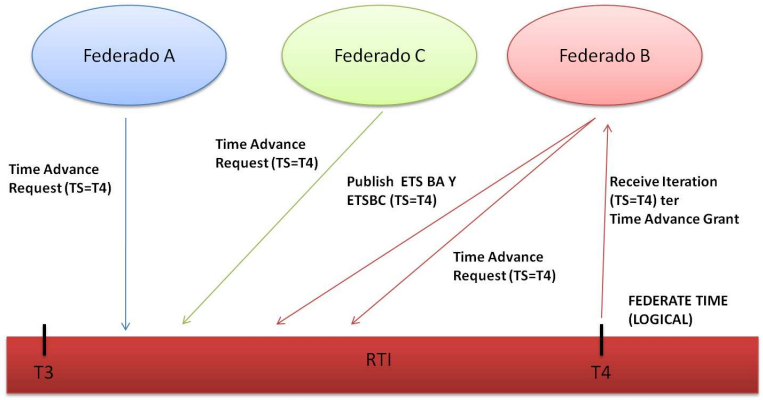

Figura B.14: Diagrama de tiempos del evento 4 en el IRM subtipo A3

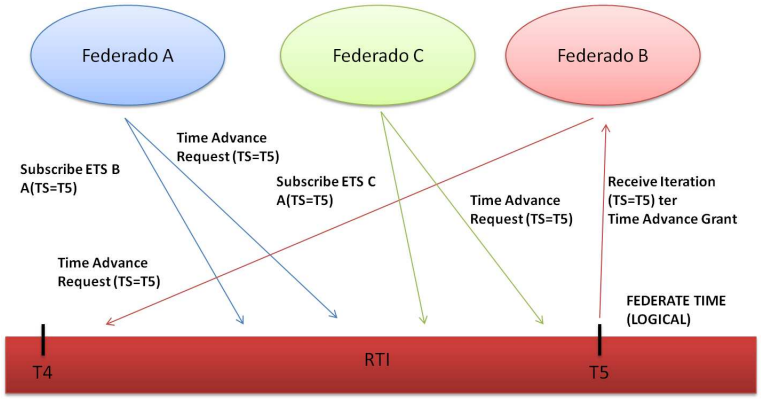

Figura B.15: Diagrama de tiempos del evento 5 en el IRM subtipo A3

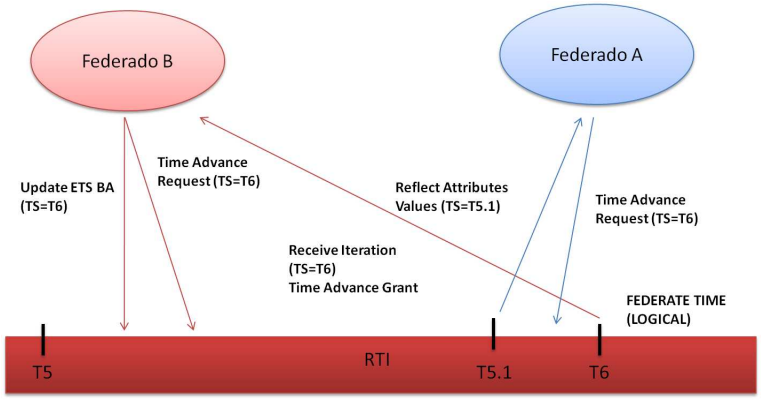

Figura B.16: Diagrama de tiempos del evento 6 en el IRM subtipo A3

B.2. Diagramas de tiempos del IRM Tipo B

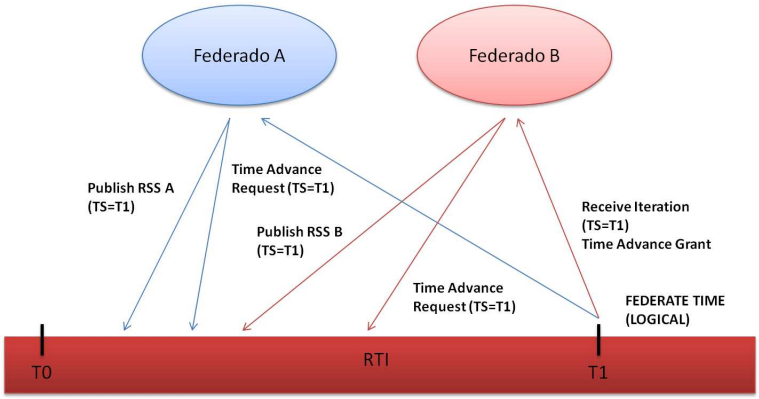

Figura B.17: Diagrama de tiempos del evento 1 en el IRM tipo B

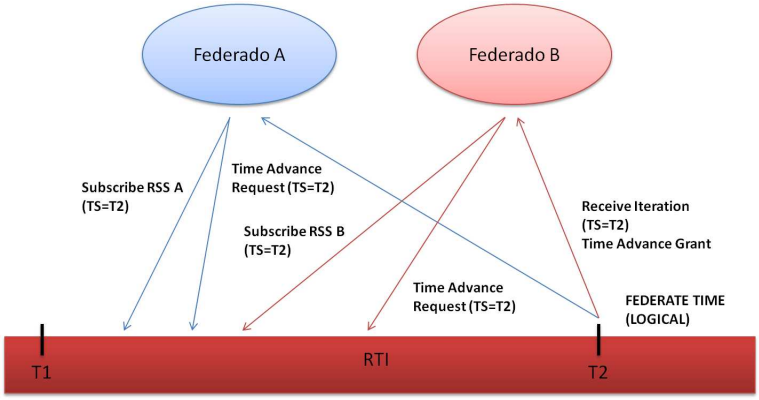

Figura B.18: Diagrama de tiempos del evento 2 en el IRM tipo B

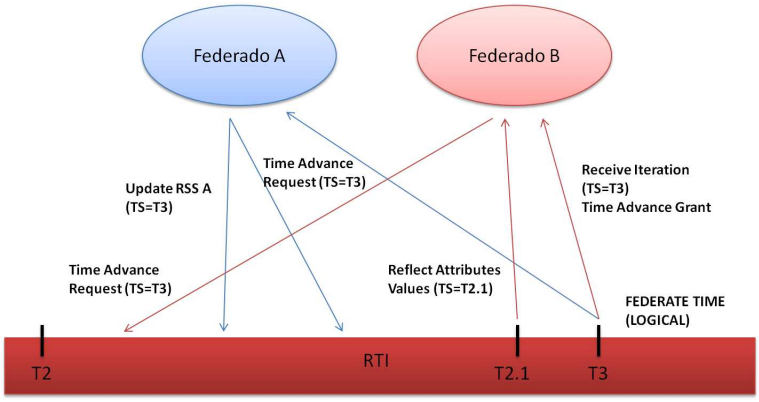

Figura B.19: Diagrama de tiempos del evento 3 en el IRM tipo B

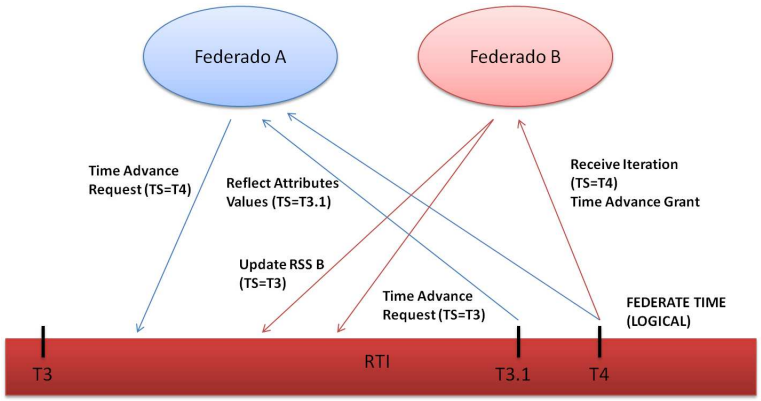

Figura B.20: Diagrama de tiempos del evento 4 en el IRM tipo B

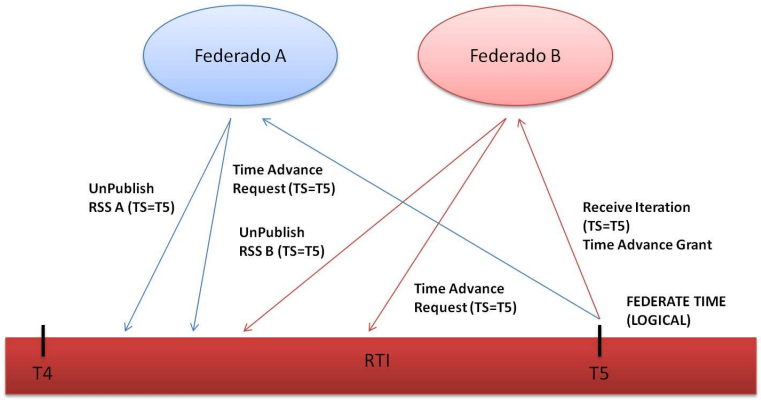

Figura B.21: Diagrama de tiempos del evento 5 en el IRM tipo B

B.3. Diagramas de tiempos del IRM Tipo C

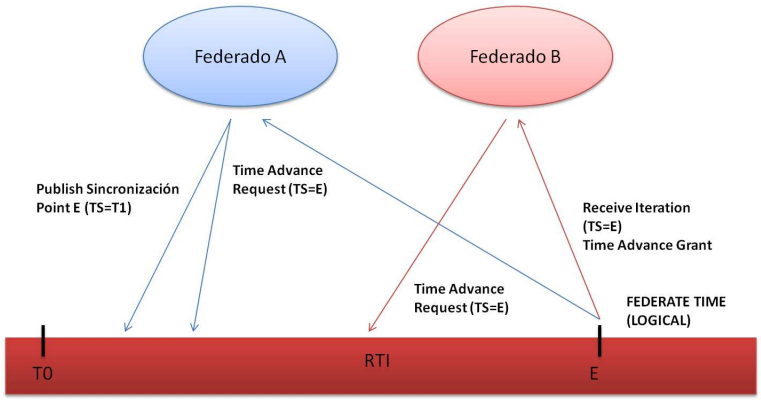

Figura B.22: Diagrama de tiempos del evento 1 en el IRM tipo C

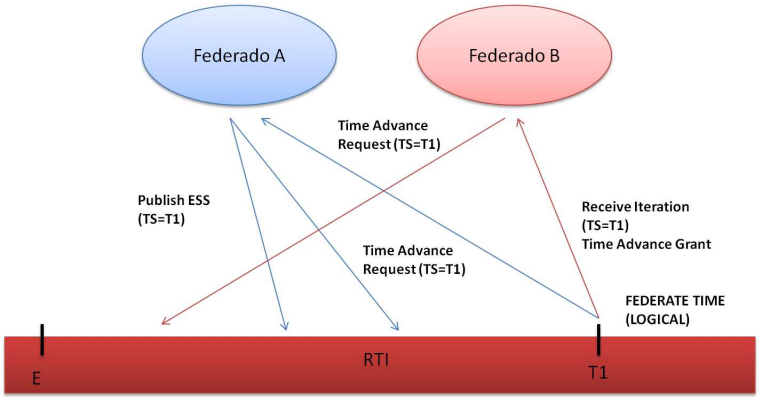

Figura B.23: Diagrama de tiempos del evento 2 en el IRM tipo C

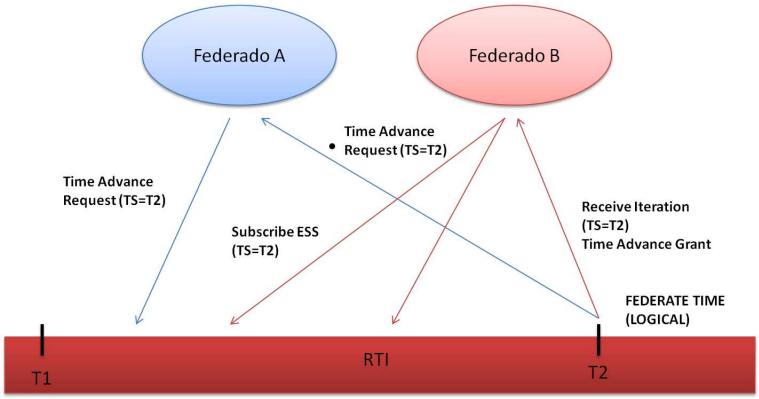

Figura B.24: Diagrama de tiempos del evento 3 en el IRM tipo C

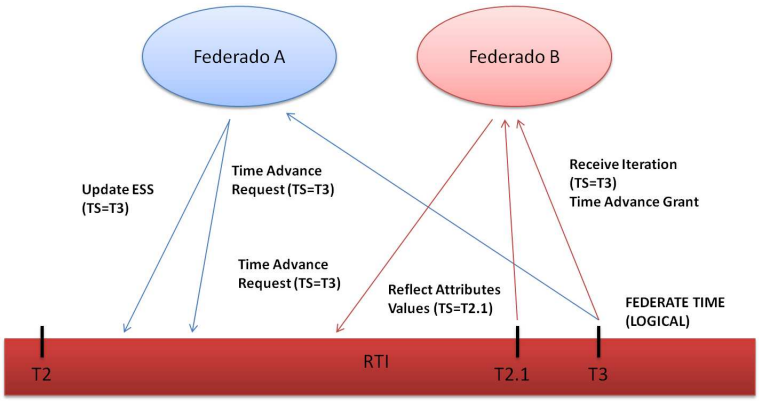

Figura B.25: Diagrama de tiempos del evento 4 en el IRM tipo C

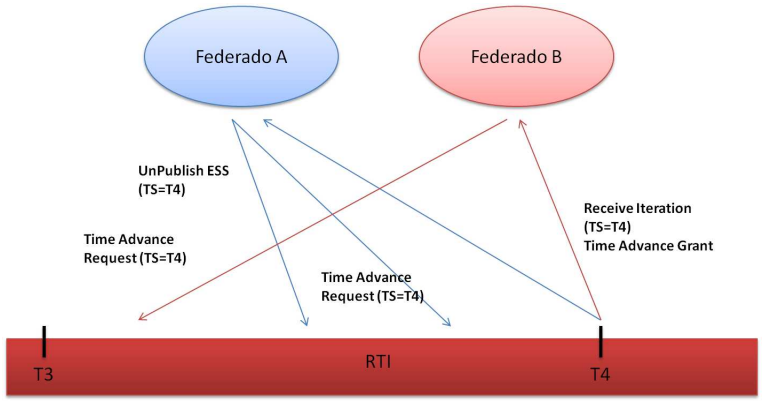

Figura B.26: Diagrama de tiempos del evento 5 en el IRM tipo C

B.4. Diagramas de tiempos del IRM Tipo D

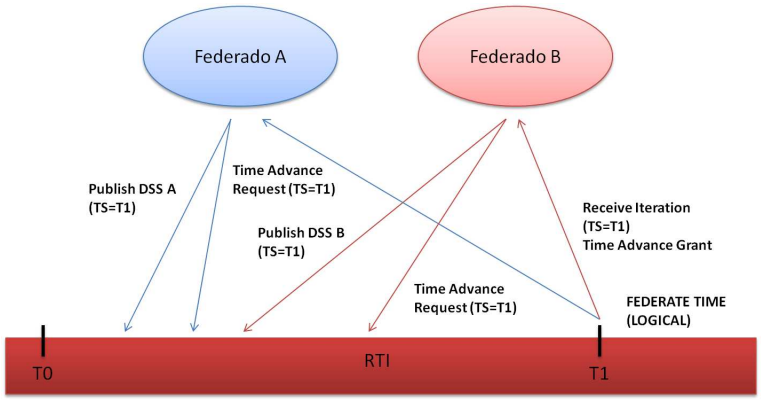

Figura B.27: Diagrama de tiempos del evento 1 en el IRM tipo D

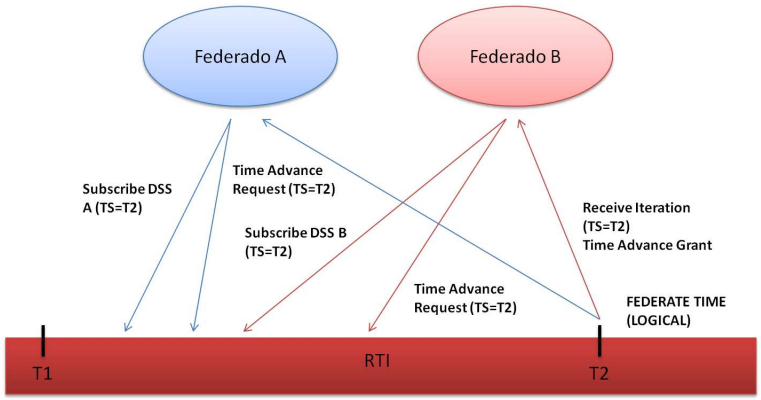

Figura B.28: Diagrama de tiempos del evento 2 en el IRM tipo D

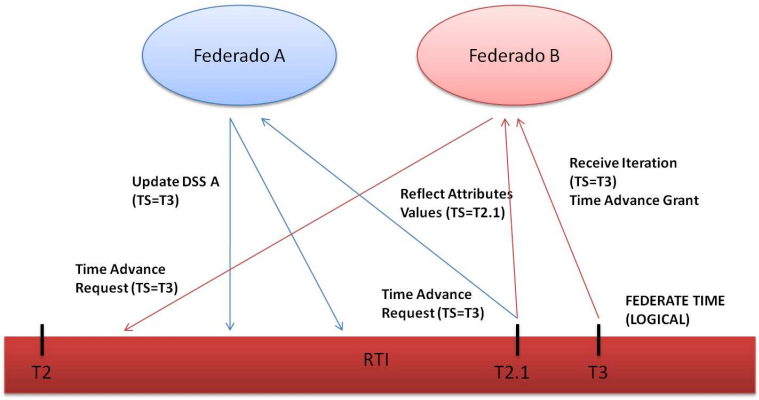

Figura B.29: Diagrama de tiempos del evento 3 en el IRM tipo D

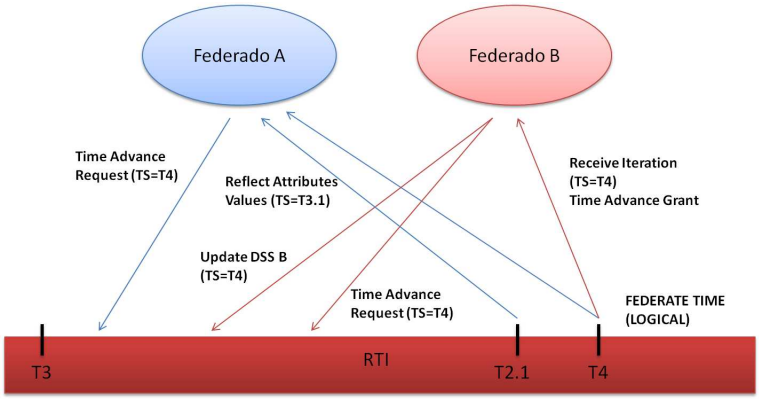

Figura B.30: Diagrama de tiempos del evento 4 en el IRM tipo D

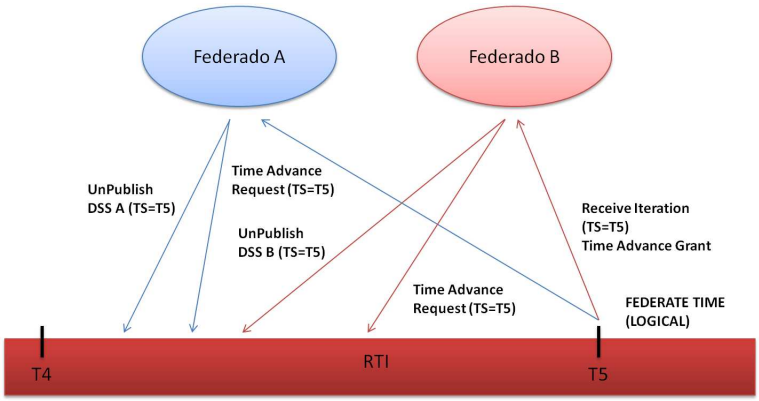

Figura B.31: Diagrama de tiempos del evento 5 en el IRM tipo D

B.5. Diagramas de tiempos del IRM Tipo E

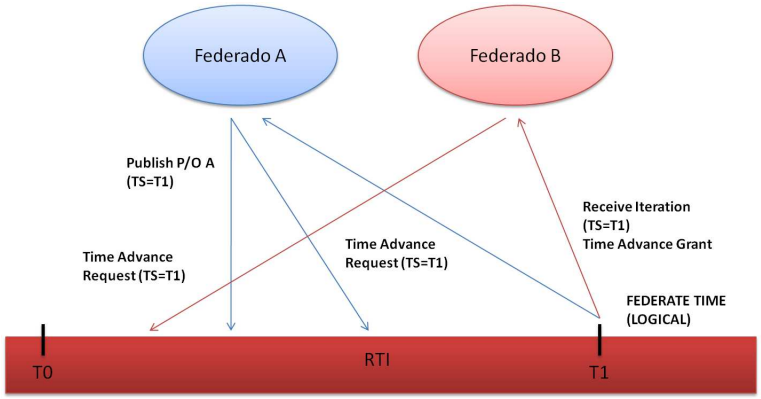

Figura B.32: Diagrama de tiempos de la fase 1: evento 1 en el IRM tipo E

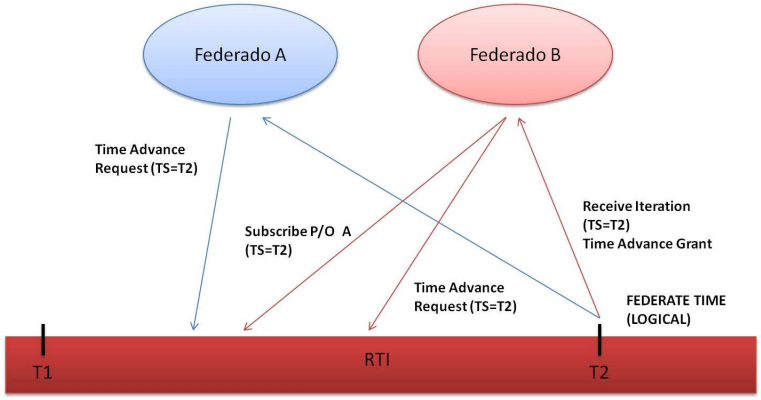

Figura B.33: Diagrama de tiempos de la fase 1: evento 2 en el IRM tipo E

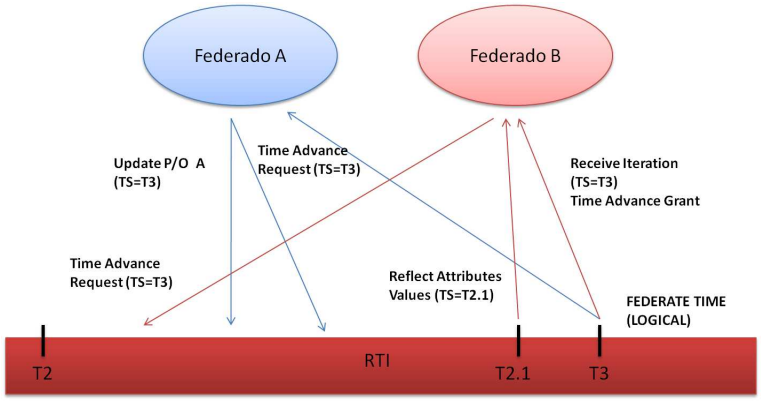

Figura B.34: Diagrama de tiempos de la fase 1: evento 3 en el IRM tipo E

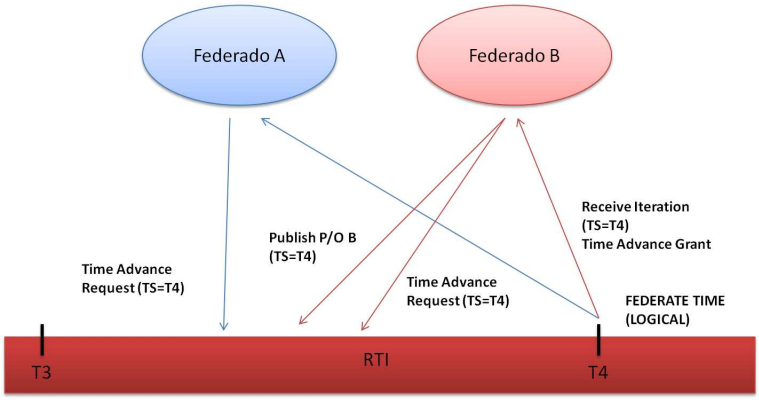

Figura B.35: Diagrama de tiempos de la fase 2: evento 4 en el IRM tipo E

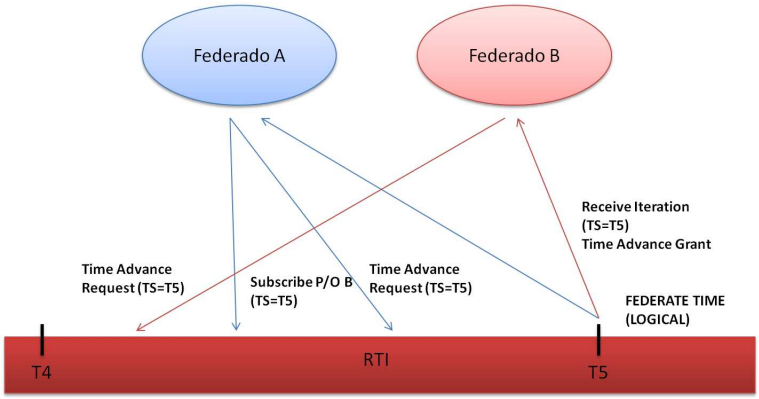

Figura B.36: Diagrama de tiempos de la fase 2: evento 5 en el IRM tipo E

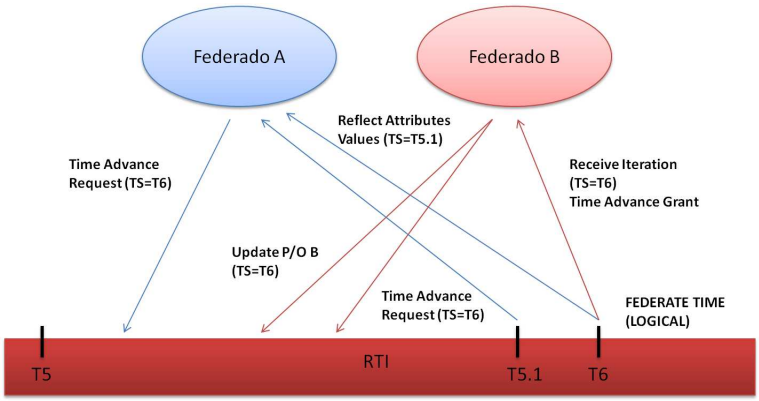

Figura B.37: Diagrama de tiempos de la fase 2: evento 6 en el IRM tipo E

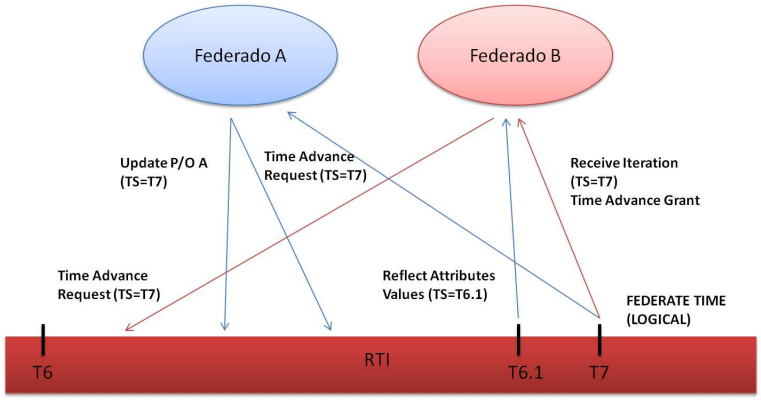

Figura B.38: Diagrama de tiempos de la fase 2: evento 7 en el IRM tipo E

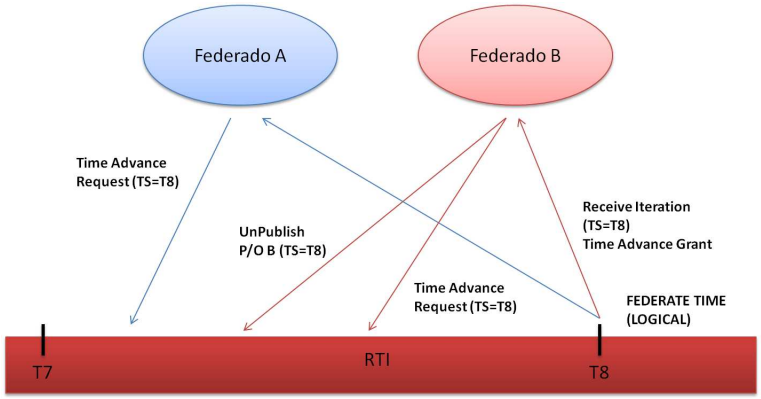

Figura B.39: Diagrama de tiempos de la fase 2: evento 8 en el IRM tipo E

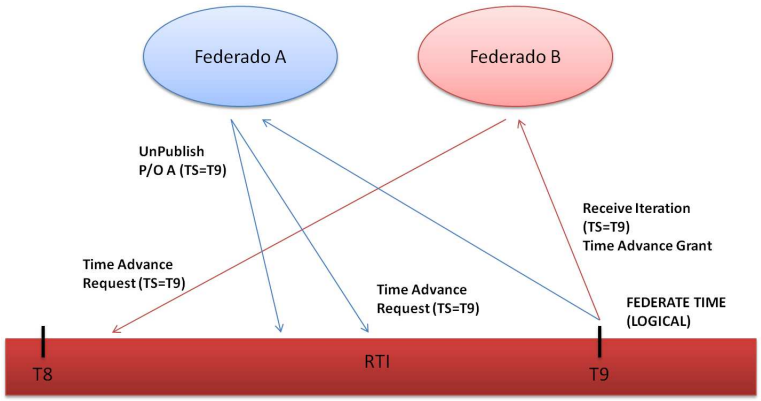

Figura B.40: Diagrama de tiempos del evento 9 en el IRM tipo E

Bibliografía

[1] Captain Ian Rodrigues. Simulations in human systems integration. In *Proceedings of the Simulation Maximising Organisational Benefits Conference*, pages 188–196, 2008.

[2] Margrethe Kobes; Nancy Oberijé; Karin Groenewegen. Serious gaming for behavioral assessment and research in case of emergency. An evaluation of experiments in virtual reality. In *Proceedings of the Simulation Concepts, Capability and Technology Conference*, pages 17–21, 2009.

[3] Miquel Angel Piera. *Cómo mejorar la logística de su empresa mediante la simulación*. Ediciones Díaz de Santos, 2006.

[4] Multilateral Interoperability Programme. MIP operating procedures (MOP), 2009. http://www.mip-site.org.

[5] North Atlantic Treaty Organization. Allied joint doctrine AJP-01(D), 2010. http://aerodefense.ihs.com.

[6] Shane D Arnott; John Winskill; Anthony Lines; Andrew McMahon. Creation of a warfighting in complex terrain system of systems analysis environment. In *Proceedings of the Simulation Maximising Organisational Benefits Conference*, pages 86–92, 2008.

[7] John Welsh. Serious games: The future in the palm of our hands. In *Proceedings of the Simulation Concepts, Capability and Technology Conference*, pages 28–36, 2009.

[8] Susannah J. Whitney; Philip Temby; Ashley Stephens. Evaluating game-based training: criteria, skills, and future directions. In *Proceedings of the Simulation Concepts, Capability and Technology Conference*, pages 177–181, 2009.

[9] Peter Ryan; Peter Ross; Will Oliver. Distributed interactive simulation revisited: Capabilities of the revised IEEE standard. In *Proceedings of the Simulation Concepts, Capability and Technology Conference*, pages 41–52, 2009.

[10] Paul C. Parks; Craig Rosenberg. Interactive distributed simulation environment for collaborative technology experiments and analysis. In *Proceedings of the Simulation Maximising Organisational Benefits Conference*, pages 56–63, 2008.

[11] Ministerio de Defensa. Interoperabilidad en simuladores constructivos: SIACOM-SIMBAD, 2007. http://www.mde.es.

[12] Philip Sammons; John Page. System in the loop control of multiple autonomous UAVs. In *Proceedings of the Simulation Maximising Organisational Benefits Conference*, pages 102–108, 2008.

[13] José María Guerrero. Evolución de los sistemas de mando y control: Interoperabilidad e integración. *Seguridad y Defensa: Tecnologías Avanzadas*, pages 51–55, 2006.

[14] The IEEE Standards Association. IEEE standard for modelling and simulation (MYS) High Level Architecture (HLA) -Framework and rules, 2000. http://www.ieee.org/; IEEE Std 1516-2000.

[15] The IEEE Standards Association. IEEE standard for modelling and simulation (MYS) High Level Architecture (HLA) -Federate interface specification, 2000. http://www.ieee.org/; IEEE Std 1516.1-2000.

[16] The IEEE Standards Association. IEEE standard for modelling and simulation (MYS) High Level Architecture (HLA) -Object model template (OMT) specification, 2000. http://www.ieee.org/; IEEE Std 1516.2-2000.

[17] Simulation Interoperability Standards Organization. Phase 1: Specification for Coalition Battle Management Language (CBML), 2008. http://www.sisostds.org.

[18] MIP PMG Working Group. MIP operational handbook, 2009. http://www.mip-site.org.

[19] Simulation Interoperability Standards Organization. Standard for military scenario definition language (MSDL), 2008. http://www.sisostds.org; SISO STD-007-2008.

[20] Simulation Interoperability Standards Organization. Coalition-Battle Management Language study group final report, 2005. http://www.sisostds.org.

[21] Multilateral Interoperability Programme. MIP system requirements specification (MSRS), 2009. http://www.mip-site.org.

[22] Multilateral Interoperability Programme. MIP tactical C2IS interoperability requirement (MTIR), 2009. http://www.mip-site.org.

[23] Multilateral Interoperability Programme. The Joint C3 Information Exchange Data Model JC3IEDM, 2009. http://www.mip-site.org.

[24] Frederic W.J.; Hamby M.B.; Mathison J.S.;Ñugen R.D. Simulation and C4I data collection in support of Force XXI training. In *Proceedings of the Simulation Winter Conference*, pages 887–893, 1998.

[25] Delaney P.J. Simulation modeling and analysis in support of brigade assault bridging operations planning. In *Proceedings of the Simulation Winter Conference*, pages 1094–1101, 2003.

[26] NATO research and technology organization. Recommendations on the establishment of a NATO simulation resource library, 2003. http://ftp.rta.nato.int.

[27] Thomas D.A.; Kwinn B.T.; McGinnis M.; Bowman B.A.; Entner M.D. The US Army enlisted personnel system: a system dynamics approach. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 1263–1267, 1997.

[28] Hill C.M.; Malone L.C. Caveats for simulation modeling in support of decision making. In *Proceedings of the Simulation Winter Conference*, pages 1102–1109, 2003.

[29] North Atlantic Treaty Organization. AMSP-01(A): NATO modelling and simulation standards profile, 2009. http://www.nato.int/docu/stanag.

[30] Tecnobit. Artillery simulator - SIMACA, 2002. http://www.tecnobit.es/.

[31] Bohemia interactive australia Pty Ldt. *White paper: Virtual battlespace 2 (VBS2) 1.3*. Bohemia Interactive Australia Pty Ldt, 2009.

[32] Steffen Straburger; Thomas Schulze; Marco Lemessi. Applying CSPI reference models for factory planning. In *Proceedings of the Simulation Winter Conference*, pages 603–609, 2007.

[33] B. G. Lawson; D. M. Nicol; R. M. Fujimoto L. F. Perrone; F. P. Wieland; J. Liu. Engineering AB into dynamic interoperability and composability via agent-mediated introspective simulation. In *Proceedings of the Simulation Winter Conference*, pages 1075–1082, 2006.

[34] North Atlantic Treaty Organization. Military operational requirements for computer assisted exercises (CAX) in NATO. http://www.nato.int.

[35] Ministerio de Defensa. Pasarela HLA-C2IS, 2007. http://www.mde.es.

[36] North Atlantic Treaty Organization. NATO modelling and simulation master plan version 1.0, 1998. http://www.rta.nato.int/.

[37] The IEEE Standards Association. IEEE standard for information technology - Protocols for distributed interactive simulations applications. Entity information and interaction, 1993. http://www.ieee.org/.

[38] Simulation Interoperability Standards Organization. The complete DIS PDU guide, 2005. http://www.sisostds.org.

[39] G.J. Jense; N.H.L. Kuijpers; A.C.M. Dumay. DIS and HLA: Connecting people, simulations and simulators in the military, space and civil domains. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 1–10, 1997.

[40] James W. Hollenbach. Inconsistency, neglect, and confusion; A historical review of DoD distributed simulation architecture policies. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 90–98, 2010.

[41] Katherine L. Morse; Mike Lightner; Reed Little; Bob Lutz; Roy Scrudder. Enabling simulation interoperability. *IEEE Computer*, volume 39(1):115–117, 2006.

[42] Jianxing Gong; Jian Huang; Jianguo Hao; Kedi Huang. Research on flexibly extensible simulation system framework. In *Proceedings of the System Simulation and Scientific Computing*, pages 316–321, 2008.

[43] Simulation Interoperability Standards Organization. Policies and procedures, 2008. http://www.sisostds.org.

[44] Simon J. E. Taylor; Navonil Mustafee; Steffen Strassburger; John Ladbrook; Stephen J. Turner; Malcolm Y. H. Low. The SISO CSPI PDG standard for commercial off-the-shelf simulation package interoperability reference models. In *Proceedings of the Simulation Winter Conference*, pages 594–602, 2007.

[45] Simulation Interoperability Standards Organization. BOM methodology strawman (BMS) specification, 2001. http://www.sisostds.org.

[46] North Atlantic Treaty Organization. Modelling and simulation architecture standards for technical interoperability: High level architecture (HLA), 2008. http://www.nato.int.

[47] NATO research and technology organization. Federation development and execution process (FEDEP) tools in support of NATO modelling and simulation programmes, 2004. http://ftp.rta.nato.int.

[48] The IEEE Standards Association. IEEE recommended practice for High Level Architecture (HLA) federation development and execution process (FEDEP), 2003. http://www.ieee.org/; IEEE Std 1516.3-2003.

[49] The IEEE Standards Association. IEEE recommended practice for verification, validation, and accreditation of a federation -An overlay to the high level architecture federation development and execution process, 2007. http://www.ieee.org/; IEEE Std 1516.4-2007.

[50] Zebin Wu; Huizhong Wu; Weiqing Li; Xu Zhang. Extending distributed simulations runtime infrastructure with web services. In *Proceedings of the International Conference on Automation and Logistics*, pages 1528–1532, 2007.

[51] Wei Zhang; Lei Feng; Jiwen Hu; Yabing Zha. An approach to service provisioning of HLA RTI as web services. In *Proceedings of the System Simulation and Scientific Computing*, pages 1–6, 2008.

[52] Björn Möller; Clarence Dahlin; Mikael Karlsson. Developing web centric federates and federations using HLA evolved web services API. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 1–7, 2007.

[53] Simulation Interoperability Standards Organization. Base object model (BOM) template specification, 2006. http://www.sisostds.org; SISO-STD-003-2006.

[54] Gary Thomas; Paul Gustavson; Barbara Sorensen. Making steps in achieving collaborative simulation development. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 71–79, 2008.

[55] Per M. Gustavsson; Joakim Wemmergard. Lessons learned from the implementation of a battle management language in a general scenario editor. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 32–41, 2009.

[56] Marc St-Onge; Fawzi Hassaine. Automated force management language. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 45–53, 2010.

[57] Lionel Khimeche; Patrick de Champs; Xavier Cuneo. APPLETs course of action modelling and C4I interfaces evolution. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 65–71, 2010.

[58] Andreas Tolk; Saikou Diallo Anil Ustun. Applying model-based data engineering to evaluate the alignment of information modeled within JC3IEDM, MSDL, and MATRIX-FOM. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 56–62, 2010.

[59] Simulation Interoperability Standards Organization. Standard for Commercial-Off-The-Shelf simulation package interoperability reference models, 2010. http://www.sisostds.org; SISO-STD-006-2010.

[60] Steffen Strassburger. The road to COTS-interoperability: From generic HLA-interfaces towards plug-and-play capabilities. In *Proceedings of the Simulation Challenges and Opportunities for a complex and networked world*, pages 1111–1118, 2006.

[61] Simon J. E. Taylor; Steffen Strassburger; Stephen J. Turner; John Ladbrook; Stephen J. Turne; Malcolm Y.H. Low; Xiaoguang Wang. Developing interoperability standards for distributed simulation and COTS simulation packages with the CSPI PDG. In *Proceedings of the Simulation Winter Conference*, pages 1101–1109, 2006.

[62] Min Wook Yoo; Tag Gon Kim. Design and implementations of surrogates for interoperation of HLA federations. In *Proceedings of Fall Simulation Interoperability Workshop*, pages 23–32, 2009.

[63] North Atlantic Treaty Organization. STANAG 4603: Modelling and simulation architecture standards for technical interoperability: High level architecture (HLA), 2008. http://www.nato.int.

[64] Peter Lendermann; Leon F. McGinnis; Steffen Straßurger; Matthias U. Heinicke; Charles McLean; Simon J.E. Taylor. Panel: Distributed simulation in industry -A real-world necessity or ivory tower fancy? In *Proceedings of the Simulation Winter Conference*, pages 1052–1062, 2007.

[65] J. Mark Pullen; Lori Topor; Ted Troccola; Stan Levine. A practical example of the integration of simulations, battle command, and modern technology. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 80–89, 2010.

[66] J. Mark Pullen; Michael R. Hieb; Ulrich Schade; Kellyn Kruger; Miloslaw Frey. Enabling the MSG-048 multinational demonstration 2007 with the command and control lexical grammar and JBML web services. In *Proceedings of the conference How is modelling and simulation meeting the defence Challenges out to 2015?*, pages 7.1–7.14, 2008.

[67] North Atlantic Treaty Organization. Minimum scale of connectivity for communications and information systems for NATO land forces, 2000. http://aero-defense.ihs.com.

[68] North Atlantic Treaty Organization. Allied joint operations AJP- 3, 2002. http://aero-defense.ihs.com.

[69] Ejército de Tierra Español. *Empleo de las Fuerzas Terrestres*. Mando de Adiestramiento y Doctrina, 2003.

[70] Multilateral Interoperability Programme. MIP MEM specification, 2009. http://www.mip-site.org.

[71] North Atlantic Treaty Organization. Formats for orders and designation of timings, locations and boundaries, 2000. http://www.nato.int.

[72] Multilateral Interoperability Programme. MIP DEM specification, 2009. http://www.mip-site.org.

[73] North Atlantic Treaty Organization. APP-6A military symbols for land based systems, 1999. http://www.nato.int.

[74] Department Of Defense. Interface standard common warfighting symbology (MIL-STD-2525B W). http://www.mapsymbs.com.

[75] North Atlantic Treaty Organization. Tactical data exchange - Link 16, 2006. http://www.nato.int.

[76] North Atlantic Treaty Organization. Tactical data exchange - Link 11/11B, 2004. http://www.nato.int.

[77] North Atlantic Treaty Organization. NATO improved link eleven (NILE) - Link 22, 2006. http://www.nato.int.

[78] Multilateral Interoperability Programme. MIP technical interface design plan (MTIDP), 2009. http://www.mip-site.org.

[79] ATCCIS MIP. ATCCIS Operational Requirements, 2002. http://www.mip-site.org.

[80] ATCCIS MIP. Data naming procedures for LC2IEDM data model, 2002. http://www.mip-site.org.

[81] ATCCIS MIP. The land C2 information exchange data model, 2002. http://www.mip-site.org.

[82] ATCCIS MIP. ATCCIS 2000 phase V work for the record, 2002. http://www.mip-site.org.

[83] Multilateral Interoperability Programme. The C2 information exchange data model (C2IEDM main), 2005. http://www.mip-site.org.

[84] Multilateral Interoperability Programme. MIP system requirement specification (SRS) 2.1, 2004. http://www.mip-site.org.

[85] Multilateral Interoperability Programme. JC3IEDM metamodel specification, 2009. http://www.mip-site.org.

[86] Multilateral Interoperability Programme. JC3IEDM extensible markup language (XML) references schemas and implementation guidance, 2009. http://www.mip-site.org.

[87] Multilateral Interoperability Programme. JC3IEDM compendium of text business rules, 2009. http://www.mip-site.org.

[88] Multilateral Interoperability Programme. JC3IEDM glossary, 2009. http://www.mip-site.org.

[89] IEEE computer society. Service-Oriented Architecture and web 2.0, 2007. http://www.computer.org/portal/web/buildyourcareer/fa007.

[90] Robert Beardsworth; Stuart Whitehead; Leslie Winters. Semantic standards for common C2 data in warfighting and modeling and simulation systems. In *Proceedings of the conference How is modelling and simulation meeting the defence Challenges out to 2015?*, pages 6.1–6.22, 2008.

[91] Oscar Pastor; Juan Carlos Molina. *Model-Driven Architecture in Practice*. Springer, 2007.

[92] MIP MDA working party. JC3IEDM platform independent model (PIM), 2009. http://mda.cloudexp.com/.

[93] Robert L. Wittman Jr.; Stephen Lopez-Couto. Using OneSAF to explore simulation and JC3IEDM alignment. In *Proceedings of the Simulation Improving Capability and Competitiveness Conference*, pages 56–54, 2007.

[94] Jr. Lunceford, W.H. The nexus of simulation with command and control: what each community can offer the other. In *Proceedings of the Simulation Winter Conference*, pages 1197–1204, 1999.

[95] Scott A. Carey; Martin S. Kleiner; Michael R. Hieb; Richard Brown. Standardizing Battle Management Language - facilitating coalition interoperability. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 10–22, 2006.

[96] Andreas Tolk; Saikou Diallo; Chuck Turnitsa. Merging protocols, grammar, representation, and ontological approaches in support of CBML. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 1–11, 2006.

[97] Ulrich Schade; Michael R. Hieb. Formalizing battle management language: A grammar for specifying orders. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 1–18, 2006.

[98] Simulation Interoperability Standards Organization. C2IEDM-BML data model and data dictionary, 2006. http://www.sisostds.org.

[99] Gary J. Farmer; Bruce W. Carlton. A methodology for COI-specific extensions to the C2IEDM. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 1–9, 2006.

[100] Andreas Tolk; Charles D. Turnitsa; Saikou Y. Diallo; Leslie S. Winters. Composable MYS web services for net-centric applications. *Journal of Defense Modeling & Simulation*, volume 3(1):11–38, 2006.

[101] Jeff Abbott; Curtis Blais; Saikou Diallo; Stan Levine; Per M. Gustavsson; Eva Nero; Ole Martin Mevassvik; Chuck Turnitsa. Coalition Battle Management Language (CBML) phase 1 specification development progress: An update to the MYS community. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 1–14, 2006.

[102] Fredrik Ullner; Adam Lundgren. *Lessons learned from implementing a MSDL scenario editor*. PhD thesis, LTH School of Engineering at Campus Helsingborg Computer Engineering, Helsingborg, 2008.

[103] Erdal Cayirci; Chunming Rong; Wim Huiskamp; Cor Verkoelen. *Snow leopard cloud: A multinational education training and experimentation cloud and its security challenges*. Springer, 2009.

[104] NATO research and technology organization. Multi-resolution exercise control structure for NATO education and training network, 2007. http://www.rta.nato.int/.

[105] Fernando Sevillano; Marta Beltrán. Arquitecturas HW y SW para sistemas de tiempo real. In *Proceedings Primer Workshop en Informática Industrial*, 2009.

[106] Facultad de Ingenería de Buenos Aires. Sistema electrónico de seguimiento y posicionamiento de una flota de vehículos en tiempo real, 2009. http://web.fi.uba.ar.

[107] University of CHICO. HLA Time Management: Design document, 1996. http://www.ecst.csuchico.edu.

[108] Bill Burke; Richard Monson-Haefel. *Entreprise javabeans 3.0*. O'Reilly, 2006.

[109] North Atlantic Treaty Organization. *STANAG 2014: Formats for orders and designation of timings; locations and boundaries*. North Atlantic Treaty Organization, 2000.

[110] Steffen Strassburger. Anatomy of an open source RTI. In *Proceedings of the Australia Simulation Improving Capability and Competitiveness Conference*, pages 65–72, 2007.

[111] PITCH. PITCH connect and interoperate, 2010. http://www.pitch.se/.

[112] General Dynamics. S2Focus, 2005. http://www.gdc4s.com.

[113] Raytheon. Network centric systems command & simulation solutions RTI NG Pro, 2010. http://support.raytheonvtc.com.

[114] Portico Open Source project. Portico user documentation, 2009. http://porticoproject.org/.

[115] Portico Open Source project. Portico developer documentation, 2009. http://porticoproject.org/.

[116] Portico Open Source project. An introduction to writing code for portico, 2009. http://porticoproject.org/.

[117] Portico Open Source project. Portico LRC architectural overview, 2009. http://porticoproject.org/.

[118] JBoss. *Jgroups tutorial*. JGroups project, 2009.

[119] JBoss. *Reliable multicasting with the Jgroups toolkit*. JGroups project, 2009.

[120] Portico Open Source project. Portico object model representation, 2009. http://porticoproject.org/.

[121] J. Mark Pullen; Douglas Corner; Samuel Singapogu. Scripted battle management language web service version 2. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 150–162, 2010.

[122] Stan Levine; Lori Topor; Ted Troccola; J. Mark Pullen. Net-enabled technology facilitated M&S/BC interoperability. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 130–141, 2010.

[123] Sixto Ríos. *Fundamentos de los sistemas de ayuda a la decisión*. Ra-Ma, 2002.

[124] Alana Zdinak. *Análisis y diseño de sistemas*. Prentice Hall, 2002.

[125] Ann Marriner. *Gestión y dirección de enfermería*. Elsevier, 2009.

[126] North Atlantic Treaty Organization. NATO interoperability standards and profiles version 3.0, 2009. http://www.nato.int.

[127] Lionel Khimeche; Patrick de Champs. APLET´s course of action modeling: A contribution to CBML. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 180–191, 2006.

[128] Lionel Khimeche; Patrick de Champs; Xavier Cuneo. APLET´s course of action modelling and C4I interfaces evolution. In *Proceedings of the Fall Simulation Interoperability Workshop*, pages 345–353, 2010.

[129] North Atlantic Treaty Organization. Guiance on the use of metadata element descriptions for use in the NATO discovery metadata specification (NDMS), 2006. http://www.nato.int.

[130] Scott W. Ambler. *The object primer: Agile model-driven development with UML 2.0.* University Press, 2004.

[131] David Besemer; Paul Butterworth; Luc Clément; Jim Green; Hemant Ramachandra; Jeff Schneider; Hub Vandervoort. *Service oriented architecture: Getting it right.* Jim Green, 2008.

[132] Jeff Davies. *The definitive guide to SOA.* Apress, 2007.

[133] Patrick Caulwell; Rajesh Chawla; Vivek Chopra. *Servicios web XML.* Anaya, 2002.

[134] Nicolai M. Josuttis. *SOA in Practice.* O'Reilly, 2007.

[135] David Chapell. *Enterprise service bus.* O'Reilly, 2004.

[136] Jiang Ji chen; Gao Ming. Enterprise service bus and an open source implementation. In *Proceedings of the International Conference in Management Science and Engineering*, 2006.

[137] Thomas Erl. *SOA design patterns.* Prentice Hall, 2009.

[138] World Wide Web Consortium (W3C). XSL transformations (XSLT) version 1.0, 1999. http://www.w3.org/.

[139] Twittie Senivongse; Rui Oliveira. *Distributed applications and interoperable systems.* Springer, 2009.

[140] JBOSS home an professional open source. JBOSS services guide, 2010. http://www.jboss.org/jbossesb/.

[141] JBOSS home an professional open source. JBOSS ESB requirements and architecture, 2006. http://www.jboss.org/jbossesb/.

[142] Javid Jamae; Peter Johnson. *JBOSS in action.* Manning Publications, 2009.

[143] JBOSS home an professional open source. JBOSS administration guide, 2010. http://www.jboss.org/jbossesb/.

[144] JBOSS home an professional open source. Overview of JBOSS ESB, 2010. http://www.jboss.org/jbossesb/.

[145] JBOSS home an professional open source. JBOSS ESB 4.2 deep dive, 2007. http://www.jboss.org/jbossesb/.

[146] Oracle. Java message service, 2002. http://java.sun.com/.

[147] World Wide Web Consortium (W3C). Simple object access protocol (SOAP) 1.1, 2000. http://www.w3.org/.

[148] World Wide Web Consortium (W3C). Xquery and Xpath full text 1.0, 2010. http://www.w3.org/.

[149] OASIS. UDDI version 3.0.2, 2004. http://www.oasis-open.org/.

[150] JBOSS home an professional open source. JBPM user guide, 2010. http://jboss.org/jbpm/.

[151] JBOSS home an professional open source. JBPM developers guide, 2010. http://jboss.org/jbpm/.

[152] Miguel Serna; José Miguel Castillo; Luis Pastor. Finding real interoperability among simulators by applying communication standards. In *Proceedings of the conference How is modelling and simulation meeting the defence Challenges out to 2015?*, pages 463–474, 2008.

[153] Miguel Serna; Marta Beltrán; Antonio Guzmán. Adaptación del IRM tipo A de SISO CSPI PDG para entornos militares. In *Proceedings Primer Workshop en Informática Industrial*, pages 118–125, 2009.

[154] Miguel Serna; Fernando Sevillano; Marta Beltrán y Antonio Guzmán. Defining the entity transfer interoperability reference model for military applications. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 234–241, 2010.

[155] Miguel Serna; Marta Beltrán; Antonio Guzmán. Interoperability reference models in military applications. In *Proceedings of the International Conference on Computer Modeling and Simulation*, pages 354–261, 2011.

[156] Guillermo Vivar. Implementación de un modelo de interoperabilidad para un entorno de simulación constructiva. Master's thesis, Universidad Rey Juan Carlos, Móstoles (Madrid), 2010.

[157] Fernando Sevillano; Miguel Serna; Marta Beltrán; Antonio Guzmán. A simulation framework to help in lean manufacturing initiatives. In *Proceedings of the European Conference On Modelling and Simulation*, pages 165–172, 2011.