Departamento de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla
Avda Reina Mercedes, s/n. 41012 SEVILLA
Fax : 95 455 71 39. **Tlf**: 95 455 71 39. **E-mail**: lsi@lsi.us.es

Una propuesta para el descubrimiento sistemático de servicios en fases tempranas de desarrollos ágiles

TESIS DOCTORAL

Autor: D. Jorge Sedeño López

Directores: Dr. D. Manuel Mejías Risoto

Dra. Dña. María José Escalona Cuaresma

Departamento de Lenguajes y Sistemas Informáticos

Universidad de Sevilla

Sevilla, Enero 2017

Luminosa e inmarcesible es la Sabiduría;

fácil es de contemplar para quienes la aman y

de descubrir por aquellos que la buscan.

(Sap. VI, 12).

Agradecimientos

Es este apartado de los agradecimientos uno de los más complejos, a mi juicio, en un proyecto de esta envergadura. Después de tanto trabajo intelectual y haciendo memoria desde el inicio, el autor comienza a darse cuenta de la multitud de personas e instituciones que han prestado su colaboración de una manera u otra a la elaboración de esta Tesis Doctoral. Es por ello que he de acudir a los lugares comunes, que no por ello dejan de ser verdaderos, y resumir en los siguiente párrafos aquellas ayudas fundamentales, no sólo desde los puntos de vista formales y materiales, sino también desde el punto de vista vital, ya que este apartado representa, lo digo con orgullo, un reconocimiento público a su labor.

Y como he de comenzar por lo primero, es a mis padres Gregorio y María José a quienes debo en primera instancia la oportunidad de haber realizado esta Tesis Doctoral. Todas las virtudes que aprendí en casa, los valores que me inculcaron y el cariño que me han dado, así como el esfuerzo que han realizado a lo largo de su vida para darnos una profunda formación cristiana, humana y profesional han cristalizado en este trabajo de Tesis Doctoral. También el apoyo y el buen humor de mis cinco hermanos Jaime, Jacobo, Sergio, Borja y Alejandra se materializan en cada línea de este trabajo.

Es ya en el terreno profesional donde he de agradecer a la Consejería de Cultura y a la Agencia Andaluza de Instituciones Culturales, no sólo el sustrato que representa el objeto de esta Tesis Doctoral, sino la calidad de los compañeros con los que me ha tocado trabajar codo con codo, durante muchos años: Javier, Ana María, Nacho, Ana Beatriz, Daniel, María Ángeles y tantos otros y de los responsables que he tenido a lo largo de esta trayectoria profesional, Manuel y Damián por parte de la Agencia y Carlos, Joaquín, Emilio y Antonio por parte de la Consejería, que desde el primer momento en el que les planteé la posibilidad de compaginar mi actividad profesional con la investigación pusieron a mi disposición todos los recursos necesarios.

Y como no, en la faceta investigadora, al grupo de Investigación de Ingeniería Web y Testing Temprano (IWT2) encabezado por los dos directores de esta Tesis Doctoral, Manuel y María José, por su dedicación y por toda la sabiduría formal y material que han volcado en este trabajo y en el propio autor y a los que me une, además, una larga relación profesional de varios años. Dentro de este grupo querría destacar especialmente a Carlos y a Eva, con los que comparto la línea de investigación en *"agile"* y con los que he tenido el honor de trabajar conjuntamente tanto a nivel profesional como académico. También a los jóvenes y no tan jóvenes compañeros del grupo de investigación de los que he aprendido mucho, Laura, Miguel, Julián o Francis entre otros, ya que aunque algo "más mayor" que el resto de investigadores del grupo, siempre he tenido que seguir su estela en el plano académico.

No puedo dejar de mencionar, dentro del Master de gestión de las Tecnologías de la Información y las Comunicaciones de la Universidad de Sevilla, al grupo de profesionales que formamos parte del alumnado y al profesorado de aquella edición del 2012 ya que fue el lugar donde encontré esta vocación investigadora, después de años de ejercicio de la profesión, y que ahora culmina, como su principio, en este trabajo de Tesis Doctoral.

Sé que me dejo en el tintero a tantos otros como Feliciano o Luis que han colaborado de una manera u otra con su apoyo, su sabiduría, su aliento y su sentido común y a otros muchos a los que, aunque no pueda nombrarlos uno a uno, nominalmente, cada gesto que han tenido ha sido grabado de manera profunda en la cabeza y en el corazón y que se plasma en este agradecimiento formal, en estas primeras líneas del trabajo.

Muchas gracias a todos.

Jorge.

Índice de Contenidos

Índice de Contenidos ..... **v**

Índice de Tablas ..... **x**

Índice de Figuras ..... **xii**

Índice de Expresiones ..... **xiv**

Resumen de la Tesis Doctoral ..... **xv**

Capítulo I Introducción **1**

1. Introducción 1

1.1. Los Servicios 2

1.2. Uso de las metodologías ágiles en la Ingeniería Web 4

1.3. Estructura del Contexto 6

2. Estructura de la Tesis Doctoral 8

3. Conclusiones 9

Capítulo II Estado del arte y trabajos relacionados **11**

1. Introducción 11

1.1. Metodología para desarrollar la revisión sistemática de la literatura 11

2. Estudio del estado del arte para el descubrimiento de Servicios. 13

2.1. Preguntas de Investigación 13

2.2. Estrategia de Búsqueda 13

2.3. Criterios de inclusión y exclusión 15

2.4. Análisis y presentación de resultados 16

3. Estudio del estado del arte para la ingeniería de requisitos ágil 17

3.1. Desarrollo de la revisión sistemática de la literatura 18

4. Estudio del estado del arte dentro del paradigma SOA 20

4.1. Preguntas de Investigación 20

4.2. Estrategia de Búsqueda 20

4.3. Criterios de inclusión y exclusión 22

4.4. Aseguramiento de la calidad 23

4.5. Análisis y presentación de resultados 24

4.6. Limitaciones de este estudio 28

5. Conclusiones 29

Capítulo III Planteamiento del Problema **31**

1. Planteamiento del problema 31
2. Objetivos 32
3. Influencias 33
3.1. El paradigma SOA en las organizaciones. 34
3.1.1 Análisis de Imperativos 36
3.1.2 El Modelo Objetivo SOA 38
3.1.3 Estrategia Iterativa 39
3.2. Requisitos en la Ingeniería Web 41
3.2.1 Metamodelo de Requisitos Web 42
3.3. Metodologías Ágiles 44
4. Conclusiones 48
Capítulo IV Metamodelo de Servicios 50
1. Introducción 50
2. Metamodelo de Servicios 52
2.1. Contexto y planteamiento previo 52
2.2. Definición del metamodelo 53
2.2.1 Metaclase «AccessPoint» 54
2.2.2 Metaclase «Domain» 55
2.2.3 Metaclase «Error» 56
2.2.4 Metaclase «Interface» 56
2.2.5 Metaclase «Message» 57
2.2.6 Metaclase «Normal» 58
2.2.7 Metaclase «Operation» 59
2.2.8 Metaclase «Policy» 59
2.2.9 Metaclase «Service» 60
2.2.10 Metaclase «ServiceLevelAgreement» 65
2.2.11 Metaclase «Stakeholder» 66
2.2.12 Metaclase «Tag» 67
2.3. Ejemplo de Servicio dentro del metamodelo 68
3. Sintaxis concreta de los metamodelos 69
3.1. Perfil UML para el metamodelo de Servicios 70
4. Conclusiones 72
Capítulo V Metamodelo Ágil de Requisitos 73

1. Introducción ... 73
2. Metamodelo Ágil de Requisitos ... 74
2.1. Contexto y planteamiento previo ... 74
2.2. Definición del metamodelo ... 75
2.2.1 Metaclase «AcceptanceCriteria» ... 76
2.2.2 Metaclase «DefinitionOfDone» ... 77
2.2.3 Metaclase «Epic» ... 77
2.2.4 Metaclase «Person» ... 78
2.2.5 Metaclase «ProjectTeamMember» ... 79
2.2.6 Metaclase «relationship» ... 80
2.2.7 Metaclase «Task» ... 80
2.2.8 Metaclase «UserStory» ... 81
2.3. Ejemplo de Requisito ágil basado en Valor dentro del metamodelo ... 83
3. Perfil UML para el metamodelo ágil de requisitos basado en Valor ... 85
4. Conclusiones ... 86
Capítulo VI Descubrimiento de Servicios Candidatos ... 87
1. Introducción ... 87
2. Relación entre metamodelos ... 88
2.1. Relación entre entidades del metamodelo ... 89
2.2. Relación entre atributos de las entidades del metamodelo ... 90
2.3. El concepto de semejanza y el concepto de puntuación en motores de búsqueda ... 91
2.4. Estructuración de las consultas ("Querys") ... 93
3. Proceso de descubrimiento de Servicios Candidatos ... 95
3.1. El descubrimiento de servicios como actividad dentro del desarrollo ágil ... 95
3.2. Fases del proceso de descubrimiento de Servicios Candidatos ... 96
4. Conclusiones ... 100
Capítulo VII Soporte tecnológico: Framework DS4aRE ... 101
1. Introducción ... 101
2. Elementos del framework DS4aRE ... 101
2.1. Enterprise Architect ... 102
2.2. Apache Solr ... 103
2.3. Algoritmo DS4aRE ... 105
2.3.1 Interface «Constantes» ... 106

2.3.2 Class «Configuration» 107
2.3.3 Class «Service» 108
2.3.4 Class «ServiceLocator» 108
2.3.5 Class «AgileRequirement» 109
2.3.6 Class «ServicePortfolio» 112
2.3.7 Class «CandidateService» 114
2.3.8 Interface «IDiscoveryServices» 115
2.3.9 Class «DiscoveryServices» 116
2.3.10 Ejemplo de Clase Principal 123
3. Visión unitaria del Framework DS4aRE 124
3.1. Grado de automatización del proceso de descubrimiento 127
4. Conclusiones 128
Capítulo VIII Validación en un entorno real 129
1. Introducción 129
2. Antecedentes 130
2.1. La transformación (2008-2012) 130
2.2. El uso de metodologías ágiles (2012-2015) 136
3. La Situación Actual 138
3.1. Implantación del marco teórico y del framework DS4aRE 138
3.2. Validación de la solución 141
4. Conclusiones 153
Capítulo IX Aportaciones, trabajos futuros y conclusiones 154
1. Aportaciones de este trabajo de Tesis 154
1.1. Estudio del Estado del Arte 155
1.2. Formalización del Catálogo de Servicios 155
1.3. Formalización de requisitos Web mediante procesos y técnicas ágiles 156
1.4. Proceso para el descubrimiento de los Servicios Candidatos 156
1.5. Framework DS4aRE 156
1.6. Validación en un entorno real 156
2. Trabajos futuros y nuevas líneas de investigación 157
3. Marco estratégico en el que se ha desarrollado este trabajo de investigación 158
3.1. Relación con el grupo de investigación IWT2 158
3.2. Trabajos realizados en la Consejería de Cultura 159

4. Conclusiones 160
Referencias Bibliográficas 162
Anexo I Servicios de Administración Electrónica 170
1. Información de los Servicios a indexar 170
Anexo II Actividad Profesional e Investigadora 176
1. Actividad Profesional 176
2. Eventos de interés científico 177
3. Publicaciones 178
3.1. Capítulos de libro 178
3.2. Revistas 178
3.3. Conferencias internacionales 178
3.4. Conferencias nacionales 179
4. Proyectos de Investigación 179
5. Redes de Transferencia e investigación 180

Índice de Tablas

Tabla I.1. Definición canónica de «SOA» [OASIS 2006]. 1
Tabla I.2. Definición canónica de «Servicio» [W3C 2004]. 3
Tabla I.3. Definición de «Servicio Web» [W3C 2004]. 3
Tabla I.4. Definición de «Servicio Gobernado» [Sedeño 2012]. 3
Tabla I.5. Definición de «Servicio Candidato». 4
Tabla II.1. Definición de la pregunta de investigación RQ1. 13
Tabla II.2. Términos de búsqueda para acotar los conceptos de la RQ1. 14
Tabla II.3. Búsqueda realizada en las principales librerías digitales para RQ1. 15
Tabla II.4. Criterios de inclusión y exclusión en la estrategia de búsqueda para RQ1. 15
Tabla II.5. Definición de la pregunta de investigación RQ2. 20
Tabla II.6. Términos de búsqueda para acotar los conceptos de la RQ2. 21
Tabla II.7. Búsqueda realizada en las principales librerías digitales para RQ2. 22
Tabla II.8. Criterios de inclusión y exclusión en la estrategia de búsqueda para RQ2. 22
Tabla II.9. Definición de la pregunta de calidad QA1. 24
Tabla II.10. Definición de la pregunta de calidad QA2. 24
Tabla II.11. Definición de la pregunta de calidad QA3. 24
Tabla II.12. Esquema de información para cada estudio incluido. 24
Tabla II.13. Resumen de la información para cada estudio incluido. 26
Tabla IV.1. Árbol de Tipificación de Servicios. 62
Tabla IV.2. Descripción de los tipos de SLA. 65
Tabla IV.3. Ejemplo de metamodelado de un Servicio concreto. 68
Tabla IV.4. Correspondencia de UML y metamodelo de Servicios. 70
Tabla V.1. Descripción de los valores del tipo de relación. 80
Tabla V.2. Ejemplo de metamodelado de un requisito ágil concreto. 83
Tabla V.3. Correspondencia de UML y metamodelo de Requisitos Ágiles basado en Valor. 85
Tabla VI.1. Definición de entidad principal. 88
Tabla VI.2. Definición de entidad adyacente. 89
Tabla VI.3. Correspondencia entre entidades principales en cada metamodelo. 89
Tabla VI.4. Correspondencia entre entidades adyacentes en cada metamodelo. 90
Tabla VI.5. Correspondencia entre atributos de entidades principales. 91

Tabla VI.6. Correspondencia entre atributos de entidades adyacentes. 91
Tabla VII.1. Estudio comparativo de herramientas de modelado. 102
Tabla VII.2. Grado de automatización del proceso de descubrimiento. 127
Tabla VIII.1. Ficha de un Servicio concreto. 132
Tabla VIII.2. Resultados con los principales sistemas construidos. 141
Tabla VIII.3. Instancia de un requisito ágil basado en Valor. 144
Tabla VIII.4. Requisito ágil de Solicitud del Carnet de Lector. 148
Tabla VIII.5. Requisito ágil de Solicitud del Carnet de Lector de un autorizado. 148
Tabla VIII.6. Requisito ágil de Consulta de Solicitudes. 149
Tabla VIII.7. Requisito ágil de Notificación Telemática del Carnet. 149
Tabla VIII.8. Resultados para la aplicación Tarjeta de Usuario. 151
Tabla VIII.9. Comparativa de costes económicos para un requisito. 152
Tabla AI.1. Servicio de Notificaciones Telemáticas. 170
Tabla AI.2. Servicio de Registro Telemático de entrada y salida. 171
Tabla AI.3. Servicio de Login con certificado en la Fachada de Ticket. 171
Tabla AI.4. Servicio de Firma electrónica con certificado digital. 172
Tabla AI.5. Servicio de Firma Electrónica de Servidor. 172
Tabla AI.6. Servicio de Envío a Portafirmas. 173
Tabla AI.7. Servicio de Sellado de documentos. 173
Tabla AI.8. Servicio de Generación de Documentos con Plantill@. 174
Tabla AI.9. Servicio de Generación de Formularios con Formul@. 174
Tabla AI.10. Servicio de Publicación en BOJA. 175

Índice de Figuras

Figura I.1. Planteamiento inicial para el descubrimiento de servicios en desarrollos ágiles. 6
Figura I.2. Propuesta para el proceso de descubrimiento de Servicios. 8
Figura II.1. Pasos definidos en el método SLR. 12
Figura II.2. Proceso para aplicar los criterios de inclusión y exclusión. 23
Figura II.3. Número de estudios incluidos y excluidos en las fases de búsqueda. 25
Figura II.4. Aseguramiento de la calidad en los estudios incluidos. 26
Figura III.1. Análisis de Imperativos. 37
Figura III.2. Modelo Objetivo SOA. 38
Figura III.3. Metodología Iterativa del Modelo SOA. 40
Figura III.4. Metamodelo propuesto para la Ingeniería de Requisitos Web. 43
Figura III.5. Framework Agile propuesto 46
Figura III.6. Actividades de la fase de lanzamiento. 47
Figura III.7. Estructura de la aproximación propuesta para un requisito ágil. 48
Figura IV.1. Propuesta para la descubrimiento de Servicios en entornos de ágiles 50
Figura IV.2. Metamodelo de Servicios. 54
Figura IV.3. Perfil UML del metamodelo de Servicios. 72
Figura V.1. Metamodelo de requisitos ágiles basado en valor. 75
Figura V.2. Perfil UML del metamodelo de requisitos ágiles basado en Valor. 86
Figura VI.1. Niveles de relación entre metamodelos. 87
Figura VI.2. Actividad para el descubrimiento de Servicios dentro de una iteración. 96
Figura VI.3. Diagrama conceptual del proceso de descubrimiento de Servicios Candidatos 96
Figura VI.4. Diagrama de secuencia funcional del proceso de descubrimiento de Servicios. 99
Figura VII.1. Interfaz de Solr para configuración y administración. 103
Figura VII.2. Diagrama de Clases del proyecto DS4aRE. 105
Figura VII.3. Estructura MAVEN del proyecto DS4aRE con su fichero de configuración. 106
Figura VII.4. Arquitectura tecnológica del framework DS4aRE. 125
Figura VII.5. Diagrama de Secuencia del framework DS4aRE. 126
Figura VIII.1. Dominios Funcionales de Negocio. 131
Figura VIII.2. Dominios Cross u Horizontales a la organización. 131
Figura VIII.3. Dominios de Soporte y Tecnológicos. 131

Figura VIII.4. Procedimiento para el uso de un Servicio. 133
Figura VIII.5. Procedimiento para la modificación / versionado de un Servicio. 134
Figura VIII.6. Repositorio de Servicios en conexión con los Sistemas de Información 136
Figura VIII.7. Secuencia del proceso de descubrimiento de Servicios Candidatos. 143

Índice de Expresiones

Expresión II.1. Expresión para la consulta sobre el objetivo de la Tesis Doctoral..... 14
Expresión II.2. Expresión para la consulta genérica de términos para la RQ2..... 21
Expresión IV.1. Restricción OCL para la metaclase «*Service*» (I)..... 64
Expresión IV.2. Restricción OCL para la metaclase «*Service*» (II)..... 65
Expresión IV.3. Restricción OCL para la metaclase «*Stakeholder*». 67
Expresión V.1. Restricción OCL para la metaclase ProjectTeamMember..... 79
Expresión VI.1. Función para score en Lucene [Apache Lucene 2012]..... 93
Expresión VI.2. Query para las entidades principales..... 94
Expresión VI.3. Query para las entidades adyacentes..... 94
Expresión VI.4. Algoritmo para la obtención de la lista de Servicios Candidatos..... 97
Expresión VII.1. Código fuente JAVA para la interfaz «Constantes»..... 107
Expresión VII.2. Código fuente JAVA para la clase «Configuration»..... 107
Expresión VII.3. Código fuente JAVA para la clase «Service»..... 108
Expresión VII.4. Código fuente JAVA para la clase «ServiceLocator»..... 108
Expresión VII.5. Código fuente JAVA para la clase «AgileRequirement»..... 109
Expresión VII.6. Código fuente JAVA para la clase «ServicePortfolio»..... 112
Expresión VII.7. Código fuente JAVA para la clase «CandidateService»..... 114
Expresión VII.8. Código fuente JAVA para la interfaz «IDiscoveryServices»..... 115
Expresión VII.9. Código fuente JAVA para la clase «DiscoveryServices»..... 116
Expresión VII.10. Código fuente JAVA para la clase «Test»..... 123
Expresión VIII.1. Configuración del tipo de dato text_es_ccul..... 139
Expresión VIII.2. Campos que forman parte del documento Servicio..... 139
Expresión VIII.3. Ejemplo de documento de JSON para la indexación de cada Servicio..... 140
Expresión VIII.4. Estructura de la respuesta en JSON que devuelve Solr al lanzar una Query..... 140
Expresión VIII.5. Consulta en JSON para las entidades principales..... 144
Expresión VIII.6. Respuesta en JSON para las entidades principales..... 144
Expresión VIII.7. Consulta en JSON para las entidades adyacentes..... 145
Expresión VIII.8. Respuesta en JSON para las entidades adyacentes..... 146
Expresión VIII.9. Servicios Candidatos para la Tarjeta de Usuario..... 149

Resumen de la Tesis Doctoral

Actualmente, para nuestra sociedad de la información, los Servicios representan una de las formas básicas de proporcionar valor en las relaciones [Mefteh & Benhassen, 2015]. La prestación de estos Servicios estará profundamente relacionada con la Web, medio telemático por excelencia en la relaciones actuales con las diferentes organizaciones que los prestan.

Por esta razón, la metodología de desarrollo de estas organizaciones, no solo tendrá que estar relacionada con los Servicios, sino con la Web, es decir, relacionado con el paradigma de la Ingeniería Web. Estas organizaciones están abocadas a incorporar metodologías de desarrollo de software cuyo paradigma pueda permitir la integración, de forma natural y en las fases más tempranas, de las aplicaciones Web a desarrollar con los Servicios de la organización [Sedeño et al. 2014c].

En estrecha relación con este desarrollo Web, se encuentran las metodologías ágiles, que se han ido consolidando y extendiendo en organizaciones que desarrollan software en entornos Web en los últimos años [Torrecilla et al. 2015].

Al final, gran parte de la funcionalidad que ofrece este tipo de organizaciones estará contenida en los servicios, es decir, los servicios dan solución a la gran mayoría de los requisitos funcionales que están en el contexto de la organización. Estos servicios (y por tanto la funcionalidad o requisitos) estarán descritos y publicados, por ende, en el Catálogo de Servicios de la organización.

Es por ello que este proceso de búsqueda de funcionalidad ya implementada en estas organizaciones es a veces complejo, manual y su éxito responde a la buena voluntad de los participantes en el mismo. Esto representa un problema grave para la reutilización del software, de los servicios y del conocimiento, y por tanto incide directamente en la eficacia y en la eficiencia, tanto en la prestación de servicios como en el propio desarrollo de aplicaciones. El trabajo de Tesis Doctoral, presentado en este documento, se ve motivado por los problemas planteados anteriormente dentro de las organizaciones que prestan servicios para el descubrimiento de la funcionalidad que ya se encuentra en el contexto de la organización.

Por tanto, el objetivo principal de la presente Tesis Doctoral será proponer en un único proceso, la formalización de un requisito, fundamentándolo en técnicas ágiles (debido a su agilidad y completitud), que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios dentro del contexto, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito.

El cuerpo de esta Tesis Doctoral pues, se cimenta sobre la definición de una serie de metamodelos. Para ello, se define un metamodelo en el que se formalizarán los Servicios pertenecientes al Catálogo de Servicios de dicha organización, que contendrá la funcionalidad identificada y viva, en el contexto, de forma normalizada. A su vez, se define un metamodelo de requisitos que permita la formalización ágil, temprana y completa de los nuevos requisitos. Para este punto será necesario el uso de las nuevas técnicas y metodologías ágiles que se han usado con buen resultado, precisamente, para disponer de un conjunto homogéneo de requisitos funcionales y no funcionales, completo y ágil.

Como resultado de la presente Tesis Doctoral, obtenemos un proceso sistemático y coherente para el descubrimiento de los Servicios Candidatos, definiendo la relación entre los metamodelos que nos permita descubrir qué Servicios dentro del Catálogo de Servicios dan cobertura a un conjunto total o parcial de los requisitos, es decir, identificar los Servicios Candidatos para su análisis mediante la propuesta de un algoritmo que realice las consultas entre los diferentes campos de los metamodelos, basado en la puntuación de dichas búsquedas, a partir de esa correspondencia. Así mismo se plantea una arquitectura tecnológica capaz de soportar dicho proceso, el framework DS4aRE.

En conclusión, esta Tesis Doctoral plantea una solución a un problema específico: realizar el proceso sistemático y coherente para el descubrimiento de los Servicios Candidatos dentro del contexto de una organización que presta Servicios, a través de la formalización de requisitos usando técnicas y metodologías ágiles a fin de identificar, dentro del Catálogo de Servicios de la organización, qué funcionalidad de los nuevos requisitos está ya contenida.

Por último, el trabajo presenta la evaluación de los resultados obtenidos en un entorno de producción real, basado en la instanciación de dicha solución, cuya aplicación ha sido considerada como satisfactoria.

Uno de los beneficios fundamentales de esta gestión ágil de requisitos dentro del gobierno de los Servicios, desde las etapas más tempranas del desarrollo, desembocará en la eficacia y eficiencia de los recursos, del propio desarrollo software y de una mejor prestación de dichos servicios.

Desde las etapas más tempranas de elicitación de requisitos, se conocerá qué servicios dentro de la organización cubren parte de la funcionalidad, por lo que la reutilización del software se maximizará para estos desarrollos, con el consiguiente ahorro en tiempo y coste (humano y económico) y aumentando la calidad de las nuevas aplicaciones.

Así mismo, se mejora, dentro del Gobierno de TI, el Gobierno de los Servicios, debido a que la temprana identificación de uso, hace posible que se puedan ejecutar las políticas adecuadas a su ciclo de vida, redundando así en un mejor control de los servicios y por tanto mejorando la prestación de los mismos y minimizando el impacto de los sucesivos cambios de estado dentro de su ciclo de vida.

Capítulo I Introducción

Este capítulo describe cuál es el contexto del trabajo desarrollado en esta Tesis Doctoral. Para ello, en la primera sección se presenta una breve introducción para centrar el trabajo desarrollado. A continuación, las secciones segunda y tercera describen, respectivamente, cuál es la estructura de este documento y unas breves conclusiones del capítulo.

1. Introducción

Actualmente, para nuestra sociedad, los servicios representan una de las formas básicas de proporcionar valor en las relaciones. No en vano, se denomina sector servicios a aquél que se ha desarrollado exponencialmente en las sociedades occidentales y que agrupa a la gran mayoría de profesionales. Uno de los pilares fundamentales de esta nueva concepción son las tecnologías de la Información y las Comunicaciones, coloquialmente denominadas TIC (*sector TIC*), entre cuyas funciones principales están crear, posibilitar, desarrollar, mantener y prestar dichos servicios de forma telemática. [Mefteh & Benhassen 2015] [Barrett et al. 2015].

Existen diferentes tipos de prestadores de servicios en nuestra sociedad, ya sean tecnológicos, como las operadoras de telecomunicaciones, o de otra índole, como los servicios sociales, incluso existen organizaciones cuya naturaleza estriba precisamente en prestar servicios, nos referimos por ejemplo, en este último caso, a las administraciones públicas, en la medida en que su propia naturaleza y su razón de ser es precisamente la prestación de servicios a la ciudadanía, hasta el punto de haberse acuñado el término de "*Servicios Públicos*".

Estas organizaciones, por ejemplo, están obligadas a prestar los servicios que les han sido encomendados por los diferentes ejecutivos a la ciudadanía de forma telemática, a raíz de las leyes emanadas en la llamada "*Sociedad de la Información"*, tanto a nivel nacional [Ley 11/2007] [Ley 39/2015] como internacional [COM 743/2010], [COM 179/2016].

En todas estas organizaciones que proveen servicios (independientemente que la prestación de estos servicios pertenezca a la naturaleza de las mismas) sería deseable que se hubiese implantado una Arquitectura Orientada a Servicios (SOA), con un cierto grado de madurez, con objeto de poder gobernar mejor esos servicios que prestan, es decir, poder contar con la capacidad de operar bajo un paradigma orientado a servicios, por lo que sería de gran eficacia que estuviesen estructuradas bajo esta arquitectura [Peinado et al. 2015]. Este paradigma ha sido definido por OASIS como la capacidad para organizar y distribuir funcionalidad, como se observa en la definición de la Tabla I.1.

Tabla I.1. Definición canónica de «SOA» [OASIS 2006].

« Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains».

Será aconsejable por tanto que estas organizaciones puedan estar conformadas de manera que puedan operar, como se ha indicado anteriormente, bajo un paradigma orientado a servicios a fin de poder gobernar de forma más óptima, eficiente y eficaz dichos servicios.

Esta transformación de las organizaciones para poder operar bajo el paradigma de la arquitectura orientada a servicios fue desarrollada en profundidad por el autor de la presente Tesis Doctoral [Sedeño 2012] y se describirá con más detalle en los capítulos posteriores.

El resultado de esta transformación se condensará en el Modelo Objetivo SOA que adopte la organización y que podrá definirse como "*el conjunto de estructuras, metodologías, políticas y herramientas que posibilitan a una organización el poder operar bajo un paradigma SOA, pudiendo ejercer el gobierno de los servicios para su prestación de forma eficiente y eficaz*". [Sedeño et al. 2013]

Dentro de este Modelo Objetivo SOA, una de las piezas que conforman su núcleo y que permite la gestión del ciclo de vida de los Servicios a lo largo del tiempo, es la referente a la parte metodológica, especialmente en lo referente a la metodología de desarrollo software del Modelo SOA que se tenga implantado, que habrá de consistir, necesariamente, en una metodología que integre, en las etapas más tempranas del desarrollo, la capacidad de desarrollo de Sistemas de Información con el ciclo de vida de los Servicios [Sedeño et al. 2014a].

Al final, gran parte de la funcionalidad que ofrece este tipo de organizaciones estará contenida en los servicios, es decir, los servicios dan solución a la gran mayoría de los requisitos funcionales que están en el contexto de la organización, estos servicios (y por tanto la funcionalidad o requisitos) estarán descritos y publicados, por ende, en el Catálogo de Servicios de la organización.

Como se ha venido tratando hasta ahora en la introducción, se ha puesto el foco en los medios telemáticos, por lo tanto, la prestación de estos servicios a través de esos medios estará profundamente relacionado con la Web, medio telemático por excelencia en la relaciones actuales con las diferentes organizaciones que los prestan.

Por esta razón, la metodología de desarrollo de estas organizaciones, no solo tendrá que estar relacionada con los Servicios, sino con la Web, es decir, relacionado con el paradigma de la Ingeniería Web. Estas organizaciones están abocadas a incorporar metodologías de desarrollo de software cuyo paradigma pueda permitir la integración de forma natural, de las aplicaciones Web a desarrollar, en las fases más tempranas de su desarrollo, con los Servicios de la organización.

En estrecha relación con este desarrollo Web se encuentran las metodologías ágiles, que se han ido consolidando y extendiendo en organizaciones que desarrollan software en entornos Web en los últimos años.

Para contextualizar esta introducción, en las siguientes secciones nos vamos a centrar en el desarrollo de dos conceptos que se relacionarán a lo largo del presente trabajo, los Servicios y el uso de las metodologías ágiles en la Ingeniería Web, junto con la estructuración de los elementos del contexto que se muestran en esta introducción.

1.1. Los Servicios

En el punto anterior se ha hecho referencia al término servicio de diferentes maneras. A continuación, se definirá qué se entenderá por servicio a lo largo de la presente Tesis Doctoral, sin detrimento de los usos comunes de la palabra servicio que pudiesen haberse reflejado a lo largo de este capítulo.

Dado que estamos en un contexto relacionado con las Tecnologías de la Información y las Comunicaciones y que los servicios sobre los que se versará se prestarán a través de medios telemáticos, para la definición del dicho término se partirá de la definición canónica proporcionada por la *Word Wide Web Consortium* (W3C) descrita en la Tabla I.2.

Tabla I.2. Definición canónica de «Servicio» [W3C 2004].

«A service is an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realized by a concrete provider agent».

Esta definición viene acompañada, además, de una serie de relaciones con otros atributos o artefactos que nos serán de mucho interés, ya que nos indicarán que un servicio es prestado por una persona u organización, tiene un propietario y además de lo descrito en la Tabla I.2, puede tener asociado una serie de roles, políticas y semánticas [WC3 2004].

Aunque estemos en un contexto TIC y referido a servicios prestados a través de medios telemáticos, no se puede confundir con un Servicio Web, cuya definición está ligada a la tecnología (un sistema software) y no al negocio y que la W3C especifica con precisión, como se observa en la definición que aporta del mismo y que se muestra en la Tabla I.3.

Tabla I.3. Definición de «Servicio Web» [W3C 2004].

«A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards».

Dado que se está hablando de Servicios que desarrollan y prestan organizaciones concretas y debido a que nos dirigimos a una gestión tanto del desarrollo de las aplicaciones como de los servicios, dentro del contexto en que se enmarca la presente Tesis Doctoral vamos a elevar la definición de Servicio para hablar de "Servicio Gobernado", de manera que nos referiremos siempre a ese concepto cuando hablemos de Servicio, ya que lo hemos de entender dentro de un contexto en una organización concreta. En la Tabla I.4 se pude observar que el servicio no es por tanto algo abstracto, sino que está incardinado en un contexto concreto, con unas relaciones concretas y que permite ser gestionado por ese mismo contexto.

Tabla I.4. Definición de «Servicio Gobernado» [Sedeño 2012].

«Un servicio gobernado es un componente auto contenido que provee funcionalidad de negocio junto con un conjunto extensible de propiedades no funcionales gestionadas mediante políticas (como seguridad, monitorización o gestión del ciclo de vida) que responde a las peticiones a través de un interfaz público bien definido en base a estándares».

Al conjunto de Servicios Gobernados que se exponen dentro de una organización o contexto se le denominará Catálogo de Servicios, en cuanto que forma un conjunto estructurado de servicios que contiene la funcionalidad de una organización o contexto concreto.

De esta manera, dentro de las fases más tempranas de la Ingeniería Web se deberá realizar **la gestión sistemática de requisitos de una Aplicación Web dentro de un contexto de**

Servicios, es decir, se deberán proporcionar los mecanismos y procesos necesarios para realizar, de forma sistemática, el descubrimiento de aquellos Servicios que estén dentro del Catálogo de Servicios y que contengan parte de las necesidades elicitadas para el desarrollo de un Sistema de Información concreto.

Ante las nuevas necesidades que se le plantean a la organización será necesario descubrir qué servicios, dentro del Catálogo de Servicios de dicha organización, podrían cubrir estas necesidades, a través de la definición de una serie de mecanismos y procesos que soporten el descubrimiento de la funcionalidad ya implementada, a través de los servicios, en el contexto de la organización.

Será necesario por tanto proponer una formalización de ese conjunto inicial de requisitos, de manera que puedan gestionarse en un entorno de Servicios de manera sistemática, ayudando en el descubrimiento de los Servicios, dentro de la organización, que contengan ese conjunto de requisitos. A estos Servicios se les denominará Servicios Candidatos y quedarán definidos en la Tabla I.5 de la siguiente manera:

Tabla I.5. Definición de «Servicio Candidato».

«Un Servicio Candidato es aquel Servicio, dentro del Catálogo de Servicios, que cubre un aspecto funcional de un desarrollo concreto. Se le denomina candidato, ya que podrá ser utilizado dentro del desarrollo de ese aplicativo como encapsulamiento de uno o varios requisitos».

1.2. Uso de las metodologías ágiles en la Ingeniería Web

Como se ha indicado antes, los Servicios se prestan de forma telemática a través de Internet y esto ha motivado que se aumenten las necesidades de adaptación rápida a los requerimientos de las organizaciones que prestan servicios y de sus consumidores. Estos eventos han surgido paralelamente a la aceptación de la Ingeniería Web como una disciplina dentro de la Ingeniería de Software [Pressman 2000]. La Ingeniería Web puede definirse, por tanto, como un conjunto de métodos, técnicas y herramientas de la Ingeniería de Software que posibilitan el desarrollo de Sistemas de Información Web. Hay varias características de los proyectos web que los diferencian del resto de proyectos de desarrollo de software [Pressman 2000],[Dephande et al. 2002]:

- Entrega de valor tan pronto como sea posible ("*as soon as possible*") y reducción del tiempo que transcurre desde la idea a la venta ("*time-to-market*").
- Requisitos críticos en las interfaces persona-ordenador.
- Mutabilidad de los requisitos en cortos intervalos de tiempo.
- Necesidad de una alta velocidad de adaptación a los cambios de requerimientos.
- Estructura de navegación compleja y aspectos específicos de seguridad.

Es importante destacar que algunas de las características mencionadas no son exclusivas del desarrollo Web y pueden aparecer también en otros proyectos software. Sin embargo, la concurrencia de todas ellas, al mismo tiempo, puede ser identificada como una especificidad de un Sistema de Información Web [Torrecilla et al. 2015].

En estos entornos, las metodologías ágiles de desarrollo software, con una constante monitorización y medición y la frecuente intervención basada principalmente en el uso de procesos empíricos [Schwaber 1996], se han convertido en una sólida alternativa para las

organizaciones que desarrollan software en el ámbito de la Ingeniería Web [Ambler 2002]. Estas metodologías ofrecen un marco adecuado para las características de desarrollo Web anteriormente expuestos [Ran et al. 2008], como una respuesta rápida a los cambios, la adaptabilidad o la reducción del tiempo de desarrollo [Pikkarainen et al. 2008]. Además, en el caso de Sistemas de Información Web, donde los requisitos cambian más rápidamente, la forma incremental e iterativa del proceso de elicitación de requisitos de las metodologías ágiles se adapta mucho mejor [Cohn 2005], [Dybá & Dingsoyr 2008].

Por lo tanto y como otro pilar de la presente introducción y en estrecha relación con el desarrollo Web, se encuentran las metodologías ágiles, ya que debido a las peculiares características en el tratamiento de los requisitos en este tipo de aplicaciones, se ha demostrado que se adaptan de manera muy adecuada a los entornos Web [Torrecilla et al. 2015], al estar centradas en técnicas empíricas e iterativas sobre las necesidades de los usuarios y al tener la flexibilidad adecuada a la hora de adaptarse a los cambios de los mismos y del entorno. Entre estas técnicas empíricas que permitirán modelar de manera ágil un conjunto de requisitos en las etapas más tempranas del desarrollo, destacan las recomendaciones de las metodologías ágiles más habituales:

- Scrum [Sutherland & Schwaber 2011],
- eXtreme Programming (XP) [Beck & Andres 2004],
- Crystal [Cockburn 2004],
- Lean Software Development [Poppendieck 2003]
- Feature Driven Development (FDD) [Palmer 2002].

Estas metodologías ágiles, al utilizar el concepto de requerimiento (necesidad), generalmente no lo hacen como los paradigmas tradicionales. La diferencia fundamental estriba en que las metodologías ágiles, que están basadas en procesos empíricos e iterativos, tienen como principal valor la agilidad, en una toma no detallada y completa de requisitos y su posterior estimación y planificación [Han et al. 2008]. Por el contrario, los paradigmas basados en metodologías tradicionales (en cascada, iterativas o incrementales) pivotan sobre una fase muy detallada de toma de requisitos, que es necesaria para ejecutar las sucesivas transformaciones de sus modelos y pasar a las siguientes fases del desarrollo.

En cualquier caso, el punto de partida de ambos son los requisitos. El objetivo principal de la presente Tesis Doctoral será, por tanto, **proponer la formalización de un requisito Web fundamentándolo en técnicas ágiles (debido a su agilidad y completitud) y que pueda ser gestionado contra el Catálogo de Servicios, a fin de descubrir qué Servicios son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito, en las fases más tempranas del desarrollo.**

Uno de los beneficios fundamentales de esta gestión ágil de requisitos dentro el Gobierno de los Servicios, desde las etapas más tempranas del desarrollo, desembocará en la eficacia y eficiencia de los recursos, del propio desarrollo software y de una mejor prestación de dichos servicios.

Se adecuará, además, a contextos u organizaciones cuya naturaleza esté orientada a la prestación de servicios a través de medios telemáticos, en los que se han de realizar no sólo el desarrollo de las aplicaciones a través de las cuales se habrá de prestar esos servicios, sino además, gobernar dichos Servicios.

1.3. Estructura del Contexto

Una vez que hemos ido identificando los elementos en esta introducción, en este apartado se propone una estructura del contexto que pueda dar respuesta al objetivo principal propuesto para esta Tesis Doctoral. Esta propuesta, dentro del campo MDE, debería contar con en una serie de modelos y un algoritmo que los correlacione, para llevar a cabo, de una manera eficiente y eficaz, el proceso de gestión de los requisitos de aplicaciones, usando técnicas ágiles, contra un Catálogo de Servicios y posibilitando el gobierno de dichos Servicios dentro de la organización.

El conjunto de Requisitos de una aplicación a desarrollar (R) deberá ser contrastado con el Catálogo de Servicios (S), que contiene el conjunto estructurado de Servicios del contexto, de manera que se pueda obtener un subconjunto de Servicios, denominado Servicios Candidatos (S'), que cubran un subconjunto de requisitos de la aplicación (R') y que se podrán utilizar junto al resto de requisitos que no están cubiertos por ninguno en los servicios (R'') para el desarrollo de dicha aplicación.

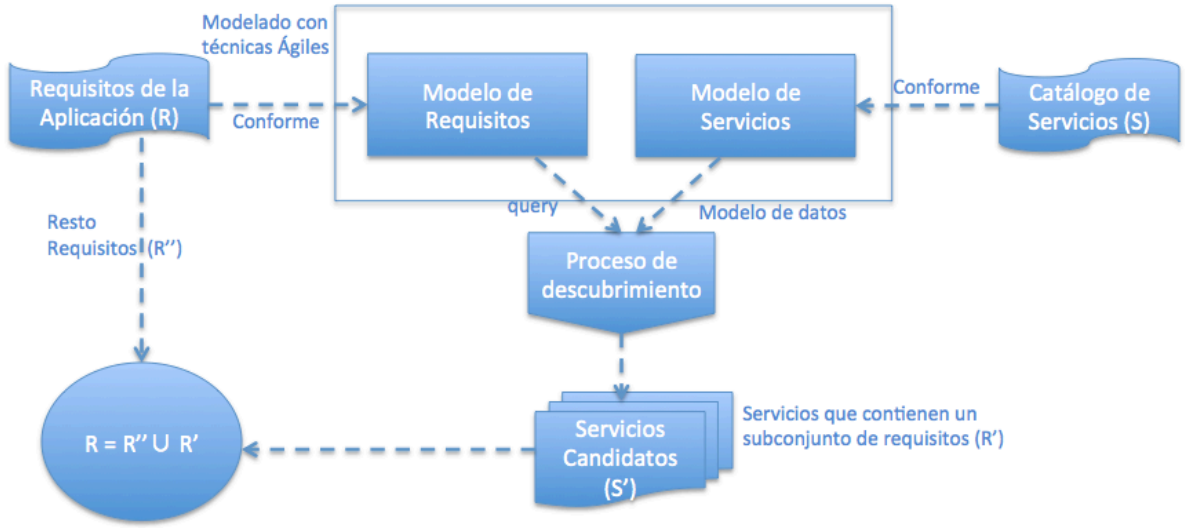

Figura I.1. Planteamiento inicial para el descubrimiento de servicios en desarrollos ágiles.

La Figura I.1 esboza de manera gráfica una propuesta para la gestión de requisitos bajo un paradigma ágil, dentro de un contexto orientado a Servicios, que se desplegará a lo largo de este trabajo de Tesis Doctoral y en la que se puede observar cómo se imbrican cada uno de los mundos expuestos en la introducción, dentro de una organización que presta servicios y cuya metodología de desarrollo Web está dentro del campo de la Ingeniería Web y que utilizan en cierta medida metodologías y técnicas ágiles.

Como se puede observar en dicha Figura I.1, una vez que se ha realizado el proceso de descubrimiento de Servicios, el conjunto de requisitos de la aplicación (R) deberá seguir siendo el mismo, es decir, deberá contar con la funcionalidad de aquellos requisitos cubiertos por los servicios (R') junto con los requisitos no son cubiertos por ningún servicio en la organización (R'').

Sería necesario realizar una serie de consideraciones para enmarcar exactamente el contexto de la presente Tesis Doctoral y que se pueden deducir de todo lo expuesto anteriormente:

- El Catálogo de Servicios es preexistente en la organización y por tanto la presente Tesis Doctoral no está enfocada a la creación y/o retroalimentación de dicho repositorio.
- La Tesis Doctoral no está enfocada al proceso de deducción de nuevos Servicios a partir de los requisitos, es decir, no se preocupa de la decisión de qué requisitos podrían llegar a convertirse en Servicios para esa organización.
- La integración de los Servicios Candidatos con el resto de requisitos de la aplicación no es objeto de la presente Tesis Doctoral ya que el proceso que se representa en la figura se encamina al descubrimiento de los Servicios Candidatos.

Tal y como es posible apreciar en la figura anterior, la propuesta expone una serie de elementos que actuarán de forma coherente y sistemática. Aunque los aspectos teóricos de estos elementos serán desarrollados con detalle en los siguientes capítulos, a continuación, se presenta una visión preliminar e introductoria de los mismos:

1. **El modelo de Servicios.** Dentro de una organización basada en Servicios, el Catálogo de Servicios es la piedra angular. Pero para que este Catálogo de Servicios pueda entrar a formar parte de un sistema automatizado, es necesario una formalización del mismo. Para ello se propondrá un metamodelo en el que esté descrito con detalle cada uno de los servicios, tanto en sí mismos, como en la relación con otros elementos del Modelo Objetivo SOA [Sedeño 2012].

2. **El modelo de requisitos**. Será necesario una propuesta de formalización del concepto de funcionalidad, utilizando técnicas ágiles. Será un requerimiento ordenado y priorizado, que pueda ser ejecutado en diversos momentos del tiempo y que al estar basado en procesos empíricos e iterativos tenga como principal valor la agilidad. La formalización y modelado de una serie de técnicas ágiles (como son las Historias de Usuario [Cohn 2005], Definiciones de Hecho, Card o Personas [Beck & Andres 2004], entre otras) nos permitirá diseñar el metamodelo para la descripción completa, temprana y ágil de los requisitos de una aplicación a desarrollar [Sedeño el al. 2014b]. Así mismo, el metamodelo propuesto contendrá las especificaciones necesarias del negocio, las relaciones entre los diferentes elementos y el alcance de la aplicación.

3. **La relación entre modelos**. Como parte nuclear de este trabajo de Tesis Doctoral, estará la identificación de la relación entre los metamodelos propuestos, con el objetivo de descubrir, de forma sistemática y automatizada, aquellos Servicios, dentro del Catálogo de Servicios, que cubren los requisitos presentados por la aplicación a desarrollar. Para ello se trabajará en la correlación entre los campos de cada uno de los metamodelos. De aquí se deducirá qué granularidad deberá tener cada metamodelo para poder relacionarse y qué grado de importancia tendrá cada campo en el descubrimiento del Servicio y se propondrá un algoritmo que utilice como consulta los requisitos formalizados y como modelo de datos el Catálogo de Servicios.

4. **Los Servicios candidatos**. Una vez realizada la relación entre metamodelos, se obtendrán unos Servicios Candidatos que constituirán el subconjunto base de los requisitos iniciales del desarrollo y que se habrán de analizar. Estos Servicios, junto con el resto de requisitos del nuevo desarrollo, constituirán el conjunto completo de requisitos de la aplicación. La integración de los Servicios Candidatos, como se ha indicado antes, con el resto de requisitos, quedará fuera del alcance de la presente Tesis Doctoral.

Esta propuesta presenta, además de los elementos mencionados, un proceso para el descubrimiento de los Servicios Candidatos, secuencialmente estructurado en el tiempo, cuyos principales pasos se describen de forma gráfica en la Figura I.2.

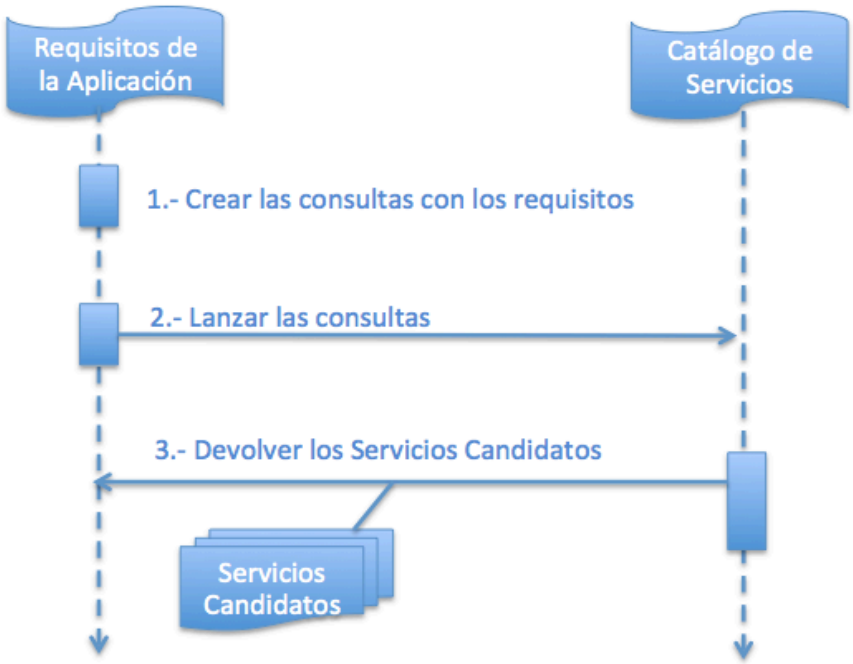

Figura I.2. Propuesta para el proceso de descubrimiento de Servicios.

El estudio y la descripción formal de cada paso dentro de esta Figura I.2 se describirá con detalle en capítulos posteriores, aunque ahora se haga una descripción somera y ordenada en el tiempo del proceso de descubrimiento que se habrá de aplicar a todos los requisitos:

1. Se modelarán los requisito usando técnicas ágiles. Una vez modelados, cada uno de los requisitos constituirá una serie de consultas contra el Catálogo de Servicios.
2. Se lanzarán dichas consultas contra el Catálogo de Servicios que actuará a modo de base de datos.
3. Como resultado de haber lanzado esas consultas se obtendrán los Servicios Candidatos, pertenecientes al Catálogo de Servicios, que den cobertura a ese requisito.

Nótese, como se indicó anteriormente, que el Catálogo de Servicios es necesario pero anterior en el tiempo y por tanto su creación no pertenece al proceso sistemático de descubrimiento de servicios, siendo un paso anterior y necesario y que está relacionado con el Gobierno de los Servicios.

2. Estructura de la Tesis Doctoral

Después de este capítulo introductorio, el Capítulo II ofrece un estudio del estado del arte sobre cuáles son los trabajos relacionados con esta Tesis Doctoral, en cuanto a otras propuestas para el proceso de descubrimiento de Servicios dentro de los paradigmas mencionados, reflejando los puntos de interacción actuales entre los mismos. Esta visión general permitirá analizar cómo

está definida, dentro de la literatura existente, el proceso de descubrimiento de los Servicios Candidatos a través de la formalización de requisitos utilizando técnicas ágiles en el campo de la Ingeniería Web, así como la formalización del Catálogo de Servicios de una organización y la relación existente entre los requisitos y los servicios, en orden al propio descubrimiento de los Servicio Candidatos.

Con la visión general que ofrece el estudio presentado en el Capítulo II sobre la situación actual del descubrimiento de Servicios Candidatos en el contexto expuesto, se sientan las bases para que durante el Capítulo III se pueda realizar una definición del problema a tratar. El Capítulo III proporciona, por tanto, una descripción detallada del planteamiento del problema del que trata esta tesis, del entorno de trabajo y define, además, de manera concreta, los retos y objetivos a satisfacer. En este capítulo además, serán definidas con más detalle las influencias que han impulsado la realización de este trabajo de Tesis Doctoral.

Planteado el problema y los objetivos a alcanzar, se comienza a trabajar en ellos en los siguientes capítulos. Por un lado, el Capítulo IV presenta la propuesta sobre cuáles deberían ser los elementos del metamodelo para realizar un tratamiento adecuado de los Servicios, así como definir un lenguaje específico de dominio adecuado que permita su implementación. Por otro lado, el Capítulo V presentará la propuesta para un metamodelo de requisitos a través de la formalización de una serie de metodologías y técnicas ágiles muy útiles en la elicitación rápida y temprana de los mismos, incluyendo también el lenguaje específico de dominio correspondiente. Será en el Capítulo VI donde se describa el proceso para el descubrimiento de los Servicios Candidatos desde la identificación del requisito hasta la obtención del Servicio Candidato, pasando por el análisis de las relaciones entre los metamodelos planteados y la implementación de un algoritmo de correlación. Así mismo, en el Capítulo VII se describirá el entorno tecnológico sobre el que se sustenta este proceso así como el framework que soporta dicho proceso.

A continuación, el Capítulo VIII ofrece la validación del modelo teórico propuesto en los anteriores capítulos mediante un caso de estudio en un entono real, realizado en la Consejería de Cultura de la Junta de Andalucía (España) donde se materializan los fundamentos teóricos de la presente Tesis Doctoral.

Más adelante, esta trabajo ofrece, ya en el Capítulo IX, las aportaciones de esta Tesis Doctoral, los trabajos futuros de investigación que se han abierto en base a los resultados obtenidos, las relaciones nacidas durante la elaboración de este trabajo y unas conclusiones finales.

Para finalizar la presentación de este trabajo de investigación, este documento culmina con dos apéndices, el Anexo I en el que se describe el Catálogo de Servicios sobre el que se ha trabajado en el Capítulo VIII y que forma parte del Catálogo de Servicios de la Consejería de Cultura y el Anexo II, que presenta la trayectoria profesional e investigadora del autor de este trabajo de Tesis Doctoral.

3. Conclusiones

La Tesis Doctoral que se presenta en este trabajo viene motivada por un problema que ha sido identificado dentro de las organizaciones que trabajan con servicios: la capacidad para gestionar, de forma temprana y ágil, un conjunto de requisitos de un nuevo desarrollo cuando

ya existe un Catálogo de Servicios y dicha funcionalidad podría encontrarse en el contexto, de cara al descubrimiento de los Servicios Candidatos.

El capítulo de introducción, aparte de enmarcar y contextualizar el trabajo de la presente Tesis Doctoral, ofrece una definición de servicio tal y como se ha de entender en todo este trabajo y muestra una propuesta inicial del proceso de descubrimiento de Servicios Candidatos.

Llegados a este punto resulta conveniente mencionar que este trabajo ha sido llevado a cabo en el seno del grupo de investigación Ingeniería Web y Testing Temprano (IWT2) de la Universidad de Sevilla [IWT2 2017], grupo referenciado en el Plan Andaluz de Investigación como PAIDI TIC021.

Además y de manera más concreta, la presente Tesis Doctoral se ha desarrollado dentro de un marco estratégico del grupo IWT2 en el que se pretende explorar e investigar cómo combinar de manera satisfactoria el paradigma de la Ingeniería Web y las metodologías ágiles, una de las líneas de investigación dentro del grupo. En este sentido, y como se describe en el Capítulo IX, son varias las Tesis Doctorales que se están llevando a cabo en este contexto y de forma paralela a la que se presenta en este documento.

Capítulo II Estado del arte y trabajos relacionados

Una vez esbozado el contexto de la presente Tesis Doctoral en el Capítulo I, nos centraremos en este capítulo en el estado del arte y en los trabajos relacionados con el objeto de esta Tesis Doctoral: proponer la formalización de un requisito Web fundamentándolo en técnicas ágiles (debido a su agilidad y completitud) y que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios, dentro del contexto de una organización, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito.

Se detallará en este capítulo la metodología que se ha aplicado para la realización de la búsqueda del estado del arte a través del estudio sistemático de la literatura y de los trabajos relacionados con el objeto descrito.

Para ello este capítulo presentará, una vez definida la metodología, los pasos que se han ido ejecutando para conducir esta búsqueda y los resultados que se han alcanzado, presentando los aspectos más relevantes que ha conllevado la realización de este estudio.

Una vez analizados los diferentes resultados, el capítulo finaliza con una serie de conclusiones que se desprenden de dicho análisis.

1. Introducción

El principal enfoque de esta sección se centrará en el estudio de los trabajos relacionados con el objeto de esta Tesis Doctoral. Para ello se realizará una búsqueda sistemática de la literatura actual, según el método que se expone a continuación, para recabar qué otros estudios han sido publicados acerca del objeto de la presente Tesis Doctoral.

Para ello aplicaremos al objeto de esta Tesis Doctoral, de forma ordenada, cada uno de los pasos metodológicos que se expondrán en las siguientes secciones.

1.1. Metodología para desarrollar la revisión sistemática de la literatura

Para llevar a cabo este estudio se va a utilizar la guía propuesta por Kitchenham y Charters conocida como revisión sistemática de la literatura ("*Sistematic Literature Review"*, SLR) dado que es la más ampliamente aceptada en el campo de la ingeniería del software. Este método permite identificar, evaluar e interpretar todos los datos existentes para una pregunta de investigación ("*Research Question*", RQ) en un área específica [Kitchenham & Charters 2007]. Sobre esta propuesta existen una serie de mejoras enfocadas a optimizar los procesos de búsqueda, ya que éstos condicionan en gran medida los resultados obtenidos:

- En 2011 Zhang et al. [Zhang et al. 2011] propusieron mejorar la búsqueda añadiendo aquellos estudios previamente conocidos o encontrados no directamente en la búsqueda manual (concepto que definen como "*quasi-gold standard*", QGS) así como una mejora en la evaluación del rendimiento de la búsqueda ("*quasi-sensitivity*"). Dos años después estos mismos autores [Zhang & Ali Babar 2013] defendieron la importancia de las revisiones sistemáticas de la literatura de forma empírica, si bien advirtieron del

necesario equilibrio entre seguir un método rigurosamente y el esfuerzo necesario para garantizar que todos los potenciales estudios han sido incluidos.

- El 2013 se realizó otra propuesta de mejora sobre el método de búsqueda. Wohlin y Prikladniki [Wohlin & Prikladniki 2013] propusieron seguir un enfoque denominado "*snowballing*" o "bola de nieve" de forma que, una vez que se ha seguido la propuesta de Kitchenham, aquellos estudios incluidos son revisados para analizar tanto los estudios que citan (ampliaría el conjunto con artículos anteriores), como aquellos por los que son citados (ampliación basada en artículos posteriores), pudiendo repetir este proceso recursivamente con el objetivo de encontrar estudios relacionados con la temática que con las palabras clave de búsqueda hubieran quedado fuera del listado inicial.

A la vista de todas las propuestas de mejora recibidas, en 2013 Kitchenham y Brenton [Kitchenham & Brereton 2013] realizaron una revisión de las guías y sugirieron nuevas mejoras para las mismas, entre las que están no utilizar cadenas de búsquedas estructuradas y en su lugar añadir estudios primarios obtenidos de otras búsquedas.

Debido a que el objeto del presente trabajo de Tesis Doctoral es muy específico y acotado se va a comenzar el estudio a partir de las guías propuestas por Kitchenham y en el caso de tener resultados relevantes se le aplicarán las mejoras referentes a la inclusión de artículos relacionados para obtener el efecto "bola de nieve" descrito en esta sección y poder así ampliar el campo de búsqueda.

Para conducir esta búsqueda conforme a lo expuesto en esta sección se seguirán las etapas siguientes a lo largo del documento:

- Planificación, en la que se establecerán las bases de lo que queremos obtener, cómo lo vamos a obtener y la forma en la que se va a representar.
- Ejecución, en la que siguiendo la metodología SLR se realizarán las búsquedas, la selección, el registro y el análisis de los estudios.
- Informes, en la que se documentará formalmente el estado del arte actual.

Figura II.1. Pasos definidos en el método SLR.

Como se puede observar en la Figura II.1, los pasos a realizar para conducir dicha revisión sistemática de la literatura serán los siguientes:

- La definición de las preguntas de investigación para el objeto de la presente Tesis Doctoral.
- La identificación de las fuentes y la definición de los términos para la realización de la búsqueda sistemática.
- La definición de los criterios de inclusión y exclusión para filtrar los resultados de la búsqueda anterior.
- La definición de los estándares de calidad necesarios para las búsquedas realizadas y su aplicación a dichas búsquedas.

- El análisis de los estudios obtenidos en las fases anteriores y la presentación de los resultados de investigación obtenidos.

A continuación se aplicará esta metodología al problema de investigación del presente trabajo.

2. Estudio del estado del arte para el descubrimiento de Servicios.

Como se ha comentado en la introducción de este capítulo, el objeto del presente trabajo de investigación es proponer la formalización de un requisito Web, fundamentándolo en técnicas ágiles y que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios, dentro del contexto de una organización, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito. Para ello se realizará a continuación la revisión sistemática de la literatura, según la metodología expuesta anteriormente.

2.1. Preguntas de Investigación

Debido a que el objeto del presente estudio responde a un objetivo muy específico, se enunciará una única pregunta de investigación que dé respuesta a este objeto. En la Tabla II.1 se enuncia dicha cuestión.

Tabla II.1. Definición de la pregunta de investigación RQ1.

RQ1. ¿Qué procesos de descubrimiento de Servicios Candidatos que cubran un requisito se han propuesto dentro del desarrollo con metodologías ágiles?.

Atendiendo al objeto expuesto anteriormente queremos analizar con esta cuestión:

- ¿Qué procesos de descubrimiento de servicios para requisitos ágiles se han propuesto?
- ¿Cuál es el grado de formalización de los mismos?
- ¿Están estos procesos integrados con alguna de las metodologías ágiles habituales en el desarrollo de aplicaciones?
- ¿Están estas actividades inscritas en las fases más tempranas del desarrollo?
- ¿Están estos procesos pensados para su implantación en un contexto concreto u organización?

2.2. Estrategia de Búsqueda

En este apartado se va a detallar el método que se ha utilizado para realizar una búsqueda exhaustiva en las principales librerías digitales a fin de localizar aquellos artículos en revistas, congresos, conferencias y talleres que puedan ayudar a establecer el estado del arte en la temática de la que se ocupa este trabajo.

Debido a que los términos de búsqueda condicionan en gran medida los resultados de la búsqueda, se han combinado un conjunto de términos y se han realizado búsquedas para analizar, a nivel empírico, los resultados obtenidos y, a la vista de los mismos, ir seleccionando las mejores palabras clave para optimizar la búsqueda manual y equilibrar el método riguroso y el esfuerzo necesario.

Como se ha indicado en el capítulo de introducción en esta Tesis Doctoral, existen dos paradigmas que se interrelacionan mutuamente; el paradigma de la arquitectura orientada a Servicios y el de las metodologías ágiles, estando además, dentro del marco de este trabajo, relacionados en las etapas más tempranas del desarrollo, es decir, en la elicitación de requisitos, que pertenece al campo de la Ingeniería de Requisitos (RE). Por esta razón y con el objeto de poder enfocar mejor la búsqueda en la intersección de ambos, se ha optado por una serie de conceptos que definan la sustancia de cada paradigma dentro de un proceso de descubrimiento de servicios.

En la Tabla II.2 podemos observar la relación de términos usados para afinar cada uno de los conceptos que entran a formar parte de la de la pregunta RQ1 realizada anteriormente, de manera que esta tabla reflejará en la primera columna el concepto perteneciente al dominio de la pregunta y en la segunda los términos usados para refinar el concepto.

Tabla II.2. Términos de búsqueda para acotar los conceptos de la RQ1.

Concepto	Términos
Servicio	SOA, service, web services, services oriented architecture.
Requisitos ágiles	Agile requirements, agileRE, agile elicitations, agile metamodel, agile requirement model, agile MDWE
Metodologías ágiles	Agile, agile methologies, agile technics, scrum, extreme programing, xp, ASD
Descubrimiento	discovery, search, discovery of services, discovering services, UDDI

Como se indica en la tabla anterior, después de este estudio preliminar de términos se han seleccionado los términos de búsqueda con los que quedará enmarcada la pregunta de estudio RQ1. Estos términos hacen referencia a procesos para descubrimiento de servicios usando técnicas ágiles. Los términos elegidos son (en inglés):

- "*service*", que será la referencia dentro el paradigma SOA a Servicio, siendo el término más amplio y que puede proporcionar el mayor espectro posible en la búsqueda.
- "*agile*", se ha despojado al termino ágil de las adjetivaciones de metodologías o técnicas, aun a riesgo de entenderse la agilidad fuera del contexto en el que se utiliza en este trabajo de Tesis Doctoral, pero permitiendo que cualquier referencia a este marco metodológico quede recabada.
- "*requirement",* de la misma manera y debido a que parte del núcleo de la Tesis Doctoral reside en el modelado de requisitos, se han relajado los términos de búsquedas a sólo los requisitos, sin adjetivarlo taxonómicamente como un requisito Web o ágil, dando un amplio margen al concepto de requisito en esta búsqueda.
- "*discovery*", se ha elegido el término descubrimiento al ser un término habitual dentro de las arquitecturas orientadas a servicios para referirse a la búsqueda de Servicios.

Con estos términos se ha generado la siguiente consulta genérica de términos sobre las diferentes fuentes. Esta consulta se muestra en la Expresión II.1:

Expresión II.1. Expresión para la consulta sobre el objetivo de la Tesis Doctoral.

```
ST= ((service*) AND (agile) AND (requirement*) AND (discovery*))
```

Dado que los criterios para realizar la búsqueda, es decir, los campos sobre los que esos términos deben ser encontrados, también condicionan los resultados y, añadiendo el hecho de que los diversos buscadores integrados en las librerías digitales no ofrecen los mismos elementos sobre los que llevar a cabo el filtrado, se han particularizado estos términos para cada buscador con el objetivo de minimizar el riesgo de excluir a priori estudios que pudiesen ser de interés. Como norma general, se ha lanzado la consulta para encontrar todos los términos en todo el contenido del artículo, añadiendo la restricción del resumen o la unión de título, resumen y palabras clave cuando los resultados fuesen demasiados extensos. En el caso contrario, si no se obtienen resultados, se eliminará alguno de los términos menos relevantes para realizar la búsqueda.

En la Tabla II.3 se muestran los resultados del uso de la expresión anterior en cada una de las fuentes.

Tabla II.3. Búsqueda realizada en las principales librerías digitales para RQ1.

Fuente	Términos	Resultados
Google Scholar	{service and requirement and discovery} en el título	6 resultados
Science Direct	{service and agile and requirement and discovery} en todo el documento y {discovery} en el título	11 resultados
Springerlink	{service and discovery} en el título y {agile and requirement } en todo el texto	5 resultados
Web of Science	{service and agile and requirement and discovery} en el título, en el abstract y en las palabras claves	0 resultados
IEEEExplore	{service and requirement and discovery and agile} en todo el documento y {discovery and service} en el título	21 resultados
ACM Digital Library	{service and requirement and discovery and agile} en todo el documento y {discovery and service} en el título	0 resultados

Con este volumen de estudios seguiremos el tercer paso dentro de la metodología propuesta.

2.3. Criterios de inclusión y exclusión

Uno de los pasos establecidos en las guías es el de establecer unos criterios objetivos para seleccionar, de los estudios primarios candidatos, aquellos que son incluidos para realizar el análisis del estado del arte. Dado que el método de búsqueda es empírico, será necesario cribar de forma manual para profundizar y detectar si cada estudio puede contribuir en el trabajo de revisión sistemática en marcha. Para ello en la Tabla II.4 se exponen los criterios de inclusión y exclusión.

Tabla II.4. Criterios de inclusión y exclusión en la estrategia de búsqueda para RQ1.

Criterios de inclusión	Criterios de exclusión
Describen un proceso de descubrimiento de Servicios.	Estudio Repetido.

Pertenece al ámbito SOA y la ámbito de las metodologías ágiles de forma conjunta.	Estudio no publicado.
Pertenece al campo de la Ingeniería de Software.	Idioma diferente del inglés.
	Editorial, resumen de taller o panel.
	No se consigue el texto completo.

Una vez estudiados los resultados, esta búsqueda no ha arrojado ninguno que resultara concluyente, obteniendo incluso combinaciones que no pertenecen al campo de la Ingeniería de Software.

Una vez realizado este paso y con los resultados obtenidos, no es posible seguir utilizando el resto de pasos propuestos por Kitchenham, ni realizar el proceso de "bola de nieve" propuesto por Wohlin y Prikladniki, ni el del aseguramiento de la calidad, debido a la inexistencia de estudios sobre los que trabajar, por lo que pasaremos directamente a analizar y presentar estos resultados.

2.4. Análisis y presentación de resultados

Cabe destacar que el problema para el estudio sistemático en este caso y una vez realizada dicha búsqueda, radica en la intersección de ambos paradigmas a través de un proceso que produzca el descubrimiento de los Servicios Candidatos en las fases tempranas del desarrollo con metodologías ágiles.

A raíz de las búsquedas anteriores y después de realizar una criba manual de estos estudios, se constata que sólo en dos de ellos existe una relación directa entre ambos paradigmas, enlazando de forma coherente y sistemática los requisitos y los servicios [Krogdahl et al. 2005] [Chehili et al. 2013], pero realizando el enfoque diametralmente opuesto al objeto de esta Tesis Doctoral, es decir, usar técnicas ágiles, pero no para el descubrimiento de Servicios Candidatos en la construcción de aplicaciones en la fase de requisitos, sino para la deducción de Servicios en el diseño de las arquitecturas orientadas a servicios a partir de los requisitos, que es precisamente uno de los aspectos que quedan fuera del alcance de partida de este trabajo de Tesis Doctoral, ya que como se comentó en el Capítulo I, no forma parte del contexto de esta Tesis Doctoral y es anterior en el tiempo al proceso de descubrimiento propuesto, siendo preexistente en el tiempo. Por tanto **concluimos que no se ha encontrado en la literatura actual ningún estudio o trabajo relacionado directamente con el objeto específico de la presente Tesis Doctoral**, de manera que actualmente se puede determinar que no se han encontrado trabajos relacionados con el descubrimiento o búsqueda de servicios dentro de un contexto usando metodologías ágiles, en orden a dar soporte a un conjunto de requisitos dado.

Por esta razón, en los siguientes apartados de este capítulo vamos a centrarnos en realizar un estudio no exhaustivo de aquellos elementos constitutivos de esta Tesis Doctoral, con el objeto de revisar la literatura existente:

- Dentro de la ingeniería de requisitos ágil, la formalización mediante un metamodelo de los elementos que conforman el requisito ágil y que son resultado de aplicar las metodologías y técnicas ágiles más habituales.
- Dentro del paradigma SOA, la formalización del Catálogo de Servicios, a través de un metamodelo de una organización que preste Servicios. Dentro de ese modelado no sólo

deberán estar los elementos básicos de un servicio y de su interfaz sino que deberán quedar recogidos los elementos mínimos del contexto de la organización, necesarios para incardinar ese servicio en un entorno concreto.

3. Estudio del estado del arte para la ingeniería de requisitos ágil

En este apartado se realizará una revisión del estado del arte de la ingeniería de requisitos ágil que se centrará en las características necesarias para abarcar el objeto de esta Tesis Doctoral, es decir, la formalización mediante un metamodelo de un requisito ágil.

Para ello comenzaremos con una breve introducción a la Ingeniería de Requisitos Ágil ("*agile RE*") para partir, a continuación, hacia la formalización mediante un metamodelo de dichos requisitos debido a que es uno de los elementos que forman parte del núcleo de este trabajo de Tesis Doctoral.

El objetivo de la fase de la Ingeniería de Requisitos ha sido siempre obtener un conjunto estable y completo de requisitos, que sirvan como base para las siguientes fases del proceso de desarrollo. Las actividades que se utilizan para lograr este objetivo son: elicitación, especificación y validación de requisitos [Lowe & Hall 1999].

La elicitación de requisitos es la actividad mediante la cual las funcionalidades del sistema que se construirá son obtenidas de cualquier fuente disponible. Los fundamentos generales de la elicitación de requisitos para la ingeniería de software no se han modificado cuando se aplican a los Sistemas Web. Sin embargo, existen objetivos específicos para los Sistemas Web que han de ser tratados de forma especial y que no se encuentran en los fundamentos generales y son esenciales para la realización de esa elicitación. Se concretan en:

- La identificación de los requisitos de contenido.
- La identificación de los requisitos funcionales en términos de necesidades de navegación y procesos de negocios.
- La definición de escenarios de interacción para diferente grupos de usuarios de la Web.

Además, una vez identificados, la especificación de los requisitos consiste en realizar una descripción de los mismos. Para esta especificación se pueden usar diferentes técnicas que van desde la descripción textual informal hasta la especificación formal.

Es importante escribir especificaciones o construir modelos que sean comprensibles para los responsables, proporcionando información suficiente para los desarrolladores, y permitiendo la validación de los modelos por los usuarios finales.

En la ingeniería de requisitos, todos los requisitos se consolidan en un documento de especificación. En base a esto se estiman calendario y presupuesto. Los requisitos se consideran como verdaderos y los cambios se realizan mediante un proceso de solicitud de cambio que puede llegar a ser muy pesado.

En los años 80 ya se había indicado [Takeuchi et al. 1986] que los desarrollos basados en fases secuenciales no eran muy adecuados debido a la falta de flexibilidad. A partir de entonces se han ido desarrollando nuevos modelos de procesos. Por un lado, existen modelos de procesos iterativos como *Rational Unified Process* [Kruchten 2004]. Por otro lado, existen metodologías ágiles como *Scrum* [Schwaber 1997], [Schwaber 2004], *Extreme Programming* (XP) [Beck & Andres 2004], *Feature-Driven Development* [Palmer 2002] o *Kanban* [Anderson 2010] .

En 2001, los líderes de estas diferentes corrientes se unieron y crearon el Manifiesto para el Desarrollo de Software Ágil (ASD) [Beck et al. 2001]. El manifiesto ágil incluye los valores y principios que contribuyeron a optimizar el proceso ASD y en los que tienen también una fuerte influencia la colaboración del equipo [Schön et al. 2015]. El manifiesto ágil proporciona los cuatro valores fundamentales que figuran a continuación:

- Los individuos e interacciones sobre procesos y herramientas.
- El software sobre la documentación exhaustiva.
- La colaboración con el cliente sobre la negociación del contrato.
- La respuesta a los cambios sobre el seguimiento de un plan.

Por lo tanto en la ingeniería de requisitos ágil se ha cambiado el foco, puesto que el movimiento ágil de desarrollo de software se ha alejado de la planificación, girando hacia modelos de procesos guiados por el valor.

3.1. Desarrollo de la revisión sistemática de la literatura

Dentro del grupo de investigación IWT2 y como se indicará en el Capítulo IX la investigación de requisitos ágiles está muy avanzada, por lo que para la realización de la revisión sistemática de la literatura en este aspecto parcial del objeto de esta Tesis Doctoral nos centraremos en el SLR realizado dentro del grupo de investigación IWT2 [Schön et al. 2017], en el que se presenta un estudio del estado del arte pormenorizado de la ingeniería de requisitos ágil, poniendo el foco en la implicación e interacción de los interesados y los usuarios en la toma de requisitos, dentro del campo del diseño guidado por el usuario, con valiosas aportaciones sobre la participación de los usuarios, las técnicas y artefactos utilizados, la documentación y los requisitos no formales y que incluye a su vez la revisión de los SLR de:

- [Inayat et al. 2015] cuyo objetivo principal es revisar la literatura existente sobre los desafíos y prácticas ágiles en la ingeniería de requisitos, incluyendo una buena discusión sobre los trabajos relacionados. Su objetivo era entender cómo se resuelven los problemas tradicionales en la ingeniería de requisitos ágil. Como conclusión proporcionaron 17 prácticas comúnmente utilizadas y también desafíos prácticos a los que los equipos ágiles tuvieron que hacer frente. Entre las prácticas más usadas y aceptadas están las historias de usuario, la priorización del requisito, la gestión del cambio, el modelado de requisitos, la gestión de requisitos, las reuniones de revisión y las pruebas de aceptación, el emparejamiento para el análisis de requisitos, las retrospectivas y la planificación continua.
- [Soares et al. 2015] que combina una revisión de la literatura con un estudio exploratorio. Se analizaron las dificultades al trabajar con los requisitos en un entorno ágil, sobre todo las causas que pueden conducir a la falta de documentación (por ejemplo, los requisitos insuficientes o incompletos). Contribuyeron con su trabajo a un importante tema de investigación, la documentación en el ASD se trata a menudo de manera inadecuada. Los autores definieron 10 dificultades que se presentan cuando se da la identificación y gestión de requisitos ágiles.

A través de estos estudios podemos obtener una serie de factores claves que se repiten y que de cara a la formalización de requisitos ágiles en un metamodelo deberían estar presentes:

- La Historia de Usuario es una de las técnicas principales avaladas por todos los estudios y que se está convirtiendo en un estándar de facto dentro de las metodologías ágiles, será por tanto necesario que el requisito formalizado tenga como fundamento la Historia de Usuario, debido a que es el más utilizado dentro del campo de la Ingeniería de requisitos actualmente [Schön et al. 2017].
- El conjunto de requisitos ha de ser lo más completo posible, es decir, ha de abarcar el máximo número de requisitos posibles y tener por tanto el mayor alcance en el tiempo posible de ese conjunto de requisitos. A través de los estudios anteriores, podemos observar que el artefacto ágil denominado "*Product Backlog*" (pila de producto), es el utilizado habitualmente para representar las Historias de Usuarios y realizar su gestión.
- El conjunto de requisitos ha de estar priorizado. Dentro del estudio destaca "*Scrum"*, como metodología que permite, debido a su flexibilidad, integrar dentro de sus procesos tanto las Historias de Usuario, como la pila de producto, como otras técnicas ágiles para la estimación, basadas en la comparación y en métodos "*Wideband-Delphi"*. Una de las premisas fundamentales de "*Scrum*" es la priorización de los elementos de dicha pila. A través de los estudios comentados, la manera más habitual de priorización es la basada en el valor, siendo esta forma coherente con el enfoque que ha tomado la Ingeniería de Requisitos Ágil, permitiendo que la granularidad de los requisitos esté orientada a la entrega de valor, característica fundamental para la posterior comparación con los Servicios.

A raíz del estado del arte mostrado por los anteriores estudios, no existe ninguna aproximación que integre en una misma propuesta todos estos elementos característicos, de manera que a la hora de formalizarse un requisito ágil, en orden a poder ser utilizado en el descubrimiento de Servicios, se deberá integrar en una misma concepción los siguiente elementos:

- Deberá contar con las Historias de Usuario, como técnica ágil fundamental a la hora de representar la definición de los requisitos, debido a su mayor adopción. [Näkki et al. 2011],[Rivero et al. 2014].
- Deberá contar con "*Scrum*" como proceso de organización de las Historias de Usuario, al ser la técnica más flexible y con mayor penetración en el agilismo. [Maguire 2013].
- Deberá contar con técnicas "*Wideband-Delphi*", basadas en el juicio de los expertos y estimación por comparación, como "*Planning Pocker*", en la priorización de requisitos, para hallar el valor ganado, debido a que el coste para la estimación del valor con estas técnicas está minimizado, haciéndolas más eficientes. [Nawrocki et al. 2014], [Torrecilla et al. 2015].
- Deberá realizar una formalización de toda esta funcionalidad en un metamodelo en lenguaje UML, debido a su universalidad y conocimiento por la casi la totalidad de los miembros de un equipo de desarrollo software. Se habrán de modelar en UML todos los artefactos que intervienen en la descripción del requisito y no solo algunos [Dragicevic et al. 2014], [Bourimi et al. 2010].
- Deberá integrar la interacción de los interesados ("*stakeholders*") y del equipos de desarrollo ("*agile team*") [Schön et al. 2017].

A raíz de estas conclusiones y con la descripción en profundidad ya en la sección 3 del Capítulo III de las influencias en este campo, se formulará en el Capítulo V de esta Tesis Doctoral una propuesta que reúna en una sola definición todos los elementos mencionados anteriormente a través de la formalización de los requisitos ágiles en un metamodelo.

4. Estudio del estado del arte dentro del paradigma SOA

Como se ha indicado anteriormente en la división que se hizo del objeto de estudio de esta Tesis Doctoral, la revisión sistemática de la literatura se centrará en este apartado, dentro de la arquitectura SOA, en la formalización del Catálogo de Servicios a través de un metamodelo dentro de una organización que preste Servicios. Dentro de este modelado no sólo deberán definirse los elementos básicos de un servicio y de su interfaz, sino que deberán quedar recogidos los elementos mínimos del contexto de la organización necesarios para incardinar ese servicio en un entorno concreto, es decir, modelar los servicios desde el punto de vista de la funcionalidad (con lo que se dejaría fuera los metamodelos de índole tecnológico o ligados a una implementación concreta) y que estuviese orientado a presentar dichos servicios como un Catálogo de Servicios desde el punto de vista funcional, de cara al descubrimiento de los mismos.

A continuación y como se explicitó en el primer apartado de este capítulo, para la realización de este segundo estudio parcial en orden al objeto principal de esta investigación, se seguirán los pasos propuestos en la metodología de Kitchenham a fin de realizar un trabajo de investigación riguroso para esta cuestión.

4.1. Preguntas de Investigación

El objeto del presente apartado es realizar la investigación sobre el descubrimiento de Servicios que pueda ser gestionado contra un Catálogo de Servicios dentro del contexto de una organización. Es éste un objetivo parcial del objeto de la presente tesis Doctoral y por tanto se enunciará una pregunta de investigación que dé respuesta a dicha consideración. En la Tabla II.5 se enuncia dicha cuestión.

Tabla II.5. Definición de la pregunta de investigación RQ2.

RQ2. ¿Qué metamodelos que modelen el Catálogo de Servicios se han propuesto dentro de una organización en orden al descubrimiento de Servicios?

Atendiendo al objetivo expuesto anteriormente queremos analizar con esta cuestión:

- ¿Se proponen metamodelos de Servicios?
- ¿Incluyen los metamodelos la descripción funcional de los Servicios?
- ¿Incluyen los metamodelos el contexto de la organización?
- ¿Están orientados los metamodelos al descubrimiento de Servicios?
- ¿Reflejan los metamodelos el Catálogo de Servicios de la organización?

4.2. Estrategia de Búsqueda

En esta sección se va a detallar el método que ha permitido realizar una búsqueda exhaustiva en las principales librerías digitales para localizar aquellos artículos en revistas, congresos, conferencias y talleres que puedan ayudar a establecer el estado del arte en la temática que nos ocupa.

Como se comentó en el anterior estudio, debido a que los términos de búsqueda condicionan en gran medida los resultados de búsqueda, se han combinado un conjunto de términos y se han realizado búsquedas para analizar, a nivel empírico, los resultados obtenidos y, a la vista de los

mismos, ir seleccionando las mejores palabras clave para optimizar la búsqueda manual y equilibrar el método riguroso y el esfuerzo necesario.

En la Tabla II.6 podemos observar la relación de términos usados para afinar cada uno de los conceptos que entran a formar parte de la pregunta de investigación RQ2. Esta tabla reflejará en la primera columna el concepto perteneciente al dominio de la pregunta y en la segunda columna los términos usados para referenciar dicho concepto.

Tabla II.6. Términos de búsqueda para acotar los conceptos de la RQ2.

Concepto	Términos
Servicio	SOA, service, web services, services oriented architecture.
Catálogo de Servicios	catalogue, entreprise, portfolio, services portfolio, services catalogue, enterprise services.
Metamodelo	model, meta-model, meta-meta-model, metamodel, MDE, model based, model-based, model driven, model-driven, MDA
Descubrimiento	discovery, search, discovery of services, discovering services, UDDI

Después del este estudio preliminar de términos presentado en la Tabla II.5 se han seleccionado los términos de búsqueda con los que quedaría enmarcada la pregunta de investigación RQ2. Estos términos hacen referencia a un metamodelo de Servicios dentro de una organización que expone su Catálogo de Servicios desde el punto de vista técnico y funcional. Los términos elegidos son (en inglés):

- *"metamodel"*, principalmente y dado que estamos buscando un metamodelo, será necesario que este término esté reflejado.
- "*portfolio*", debido a que buscamos que el metamodelo formalice también el Catálogo de Servicios dentro de una organización.
- "*service*", debido a que el término servicio, dentro del paradigma SOA, abarca en la búsqueda el mayor espectro posible.
- "*discovery*", se ha elegido este término al ser un término habitual, dentro de las arquitecturas orientadas a servicios, para referirse a la búsqueda de Servicios y que proporcionará por tanto un mayor rango en el espectro de la búsqueda.

Con esto términos se ha generado una consulta genérica de términos sobre las diferentes fuentes. En la Expresión II.2 se muestra dicha consulta.

Expresión II.2. Expresión para la consulta genérica de términos para la RQ2.

```
ST=((metamodel*) AND (service*) AND (portfolio) AND (dicover*))
```

Como se ha indicado en el apartado anterior, para las búsquedas en las principales librerías digitales, los campos sobre los que esos términos deben ser encontrados también condicionan los resultados y añadiendo el hecho de que los diversos buscadores integrados en las librerías digitales no ofrecen los mismos elementos sobre los que llevar a cabo el filtrado, se han particularizado estos términos para cada buscador con el objeto de minimizar el riesgo de excluir a priori estudios que pudieran ser de interés. Como norma general, se ha lanzado la

consulta para encontrar todos los términos en todo el contenido del artículo, añadiendo la restricción del resumen o la unión de título, resumen y palabras clave cuando los resultados fuesen demasiados extensos. En caso contrario, ante la ausencia de resultados se ha optado por eliminar algunos términos en orden de importancia para ampliar el rango de búsqueda en dichas librerías digitales.

En la Tabla II.7 se muestran los resultados del uso de la expresión anterior en cada una de las fuentes.

Tabla II.7. Búsqueda realizada en las principales librerías digitales para RQ2.

Fuente	Términos	Resultados
Google Scholar	{service and metamodel } en el título	30 resultados
Science Direct	{service and metamodel and portfolio and discovery} en todo el texto	55 resultados
Springerlink	{service and metamodel and portfolio and discovery} en todo el texto	48 resultados
Web of Science	{service and metamodel } en el título	17 resultados
IEEEExplore	{service and metamodel and portfolio and discovery} en todo el texto	43 resultados
ACM Digital Library	{service and metamodel and portfolio and discovery} en todo el texto	1 resultados

Con este volumen de estudios seguiremos el tercer paso dentro de la metodología propuesta.

4.3. Criterios de inclusión y exclusión

El tercero de los pasos establecidos en las guías [Kitchenham & Charters 2007] es el de establecer unos criterios objetivos para seleccionar, de los estudios primarios candidatos, aquellos que son incluidos para realizar el análisis del estado del arte. Dado que el método de búsqueda es empírico, será necesario cribar de forma manual para profundizar y detectar si cada estudio puede contribuir en el trabajo de revisión sistemática en marcha. Para ello en la Tabla II.8 se exponen los criterios de inclusión y exclusión.

Tabla II.8. Criterios de inclusión y exclusión en la estrategia de búsqueda para RQ2.

Criterios de inclusión	Criterios de exclusión
Describen un metamodelo.	Estudio repetido.
Pertenece al ámbito SOA.	Estudio no publicado.
Describe aspectos funcionales del Servicio.	Idioma diferente del inglés.
Pertenece al campo de la Ingeniería de Software.	Editorial, resumen de taller o panel.
	No se consigue el texto completo.

Con estos criterios de inclusión y exclusión y partiendo de los estudios encontrados, se describe a continuación el proceso que se realizará para la revisión de dichos estudios:

1. Se realizará una criba sobre el título, el resumen y las palabras claves calificando los artículos como:
    a. Sí, se incluye ese estudio en la revisión sistemática en marcha.
    b. No, se excluye ese estudio de la revisión sistemática en marcha.
    c. Parcial, se tienen dudas sobre su inclusión.
2. Sobre los estudios con resultado "Parcial" del paso anterior, se vuelve a realizar la criba manual pero esta vez sobre el texto completo de manera que en este caso se valorarán definitivamente como:
    a. Sí, se incluye ese estudio en la revisión sistemática en marcha.
    b. No, se excluye ese estudio de la revisión sistemática en marcha.

En la Figura II.2 se describe gráficamente este proceso.

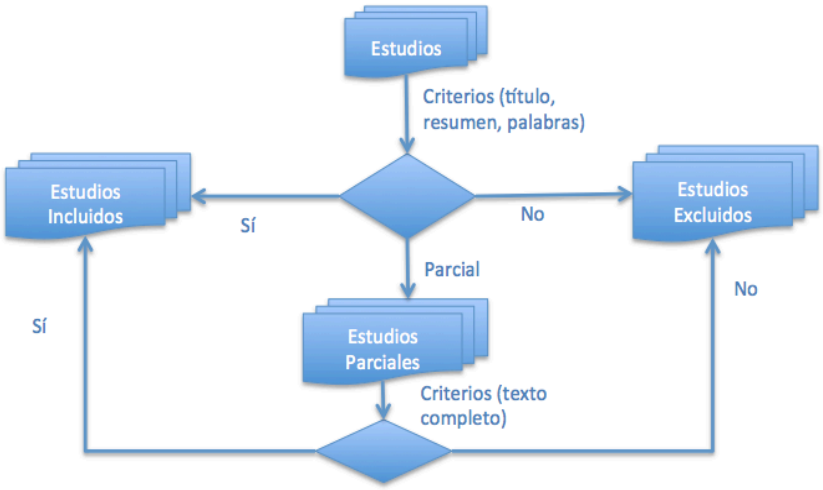

Figura II.2. Proceso para aplicar los criterios de inclusión y exclusión.

Una vez que el proceso ha finalizado, sobre el conjunto de estudios que han sido analizados considerando su texto completo, se realiza la nueva búsqueda para encontrar las referencias de los mismos, es decir, aquellos estudios posteriores que citaron a los anteriores y posibles nuevos trabajos sobre las mismas propuestas o de los mismos autores. El proceso de selección de la Figura II.2 se lanza de nuevo con este conjunto de estudios atendiendo a los mismos criterios de filtrado.

4.4. Aseguramiento de la calidad

Para evaluar la calidad de los estudios seleccionados en razón a cumplir el objetivo planteado en este estado del arte, se ha establecido un cuestionario que debe ser rellenado en cada uno. Para evaluar la RQ2 se necesita valorar por un lado, si cada propuesta analiza el trabajo realizado hasta la fecha, de forma que justifique hasta dónde había llegado la investigación actual y qué hueco pretendía cubrir, y por otro la propia descripción del metamodelo de Servicios, que se asociarán a las preguntas de calidad QA1 y QA2 respectivamente y que se presentan en las Tablas II.9 y II.10 respectivamente.

Tabla II.9. Definición de la pregunta de calidad QA1.

QA1. ¿El estudio realiza una revisión de la literatura actual u otros trabajos relacionados? • *Sí, en el caso de que exponga los trabajos relacionados y enmarque su enfoque o propuesta en ese contexto.* • *Parcialmente, si sólo menciona algunos trabajos pero no establece el marco contextual en el que se desarrolla el estudio.* • *No, cuando no mencione trabajos relacionados.*

Tabla II.10. Definición de la pregunta de calidad QA2.

QA2. ¿El estudio propone un metamodelo descrito a nivel textual y formal? • *Sí, en el caso que el estudio describa el metamodelo de forma textual y formal.* • *Parcialmente, cuando sólo se describa de forma textual.* • *No, cuando no describe ningún metamodelo de forma específica.*

Así mismo será necesario conocer si ese metamodelo está implantado en alguna organización o si es puramente teórico sin que se haya instanciado en una organización concreta. Este aspecto de la calidad se verá reflejado en la pregunta de calidad QA3 que se muestra en la Tabla II.11 y que recoge el siguiente rango de valores.

Tabla II.11. Definición de la pregunta de calidad QA3.

QA3. ¿El estudio propone la instanciación del metamodelo en una organización concreta? • *Sí, en el caso que el estudio describa la implantación y proceso de instanciación.* • *Parcialmente, cuando se instancie el metamodelo como parte de un problema real pero de forma teórica.* • *No, cuando no se ha instanciado el metamodelo en ninguna organización, ni para un problema real pero de forma teórica.*

4.5. Análisis y presentación de resultados

Para la presentación de los resultados obtenidos en esta última fase del estudio se define un esquema de información a modo de ficha en la que se han recogido los datos de cada uno de los estudios incluidos en esta revisión sistemática de la literatura, teniendo como objetivo facilitar el proceso de análisis de la información recapitulada. La Tabla II.12 recoge los datos de la ficha para cada estudio.

Tabla II.12. Esquema de información para cada estudio incluido.

Tipo de Información	Datos Almacenados
Información básica.	Título, autor y año de publicación.
Datos de la Publicación.	Libro, revista, conferencia en el que fue publicado, palabras clave, resumen y bibtex para su cita.
Propuesta del metamodelo.	Propuesta del metamodelo para la formalización del Catálogo de Servicios de una organización.
Validación.	Si se presenta la validación mediante la implantación de dicho

	metamodelo en un entorno real o al menos se realiza la validación de modo teórico sobre un problema real.
Aseguramiento de la calidad.	Evaluación de las preguntas QA1, QA2 y QA3.
Referencias.	Aquellas otras posibles citas del artículo que puedan ser incluidas en otra iteración, así como otros posibles estudios que lo citan o nuevas publicaciones de los autores en esta línea. Cualquier otro estudio que pueda considerarse para su inclusión en este estudio del arte.
Otros.	Cualquier otra anotación que sea de utilidad para analizar las contribuciones del estudio en el estado del arte.

Para facilitar la gestión de las referencias se ha utilizado la herramienta JabRef [Alver 2012], que ha sido seleccionada frente a otras alternativas por su capacidad de importar desde múltiples formatos [Basak 2015].

A continuación se presentan los resultados cuantitativos realizados en este estudio del arte. En la Figura II.3 se muestran los estudios incluidos y excluidos, tanto para la fase de búsqueda como para la fase posterior, mediante la técnica de refinamiento, añadiendo estudios antecedentes y posteriores, que han supuesto finalmente la inclusión de un estudio más.

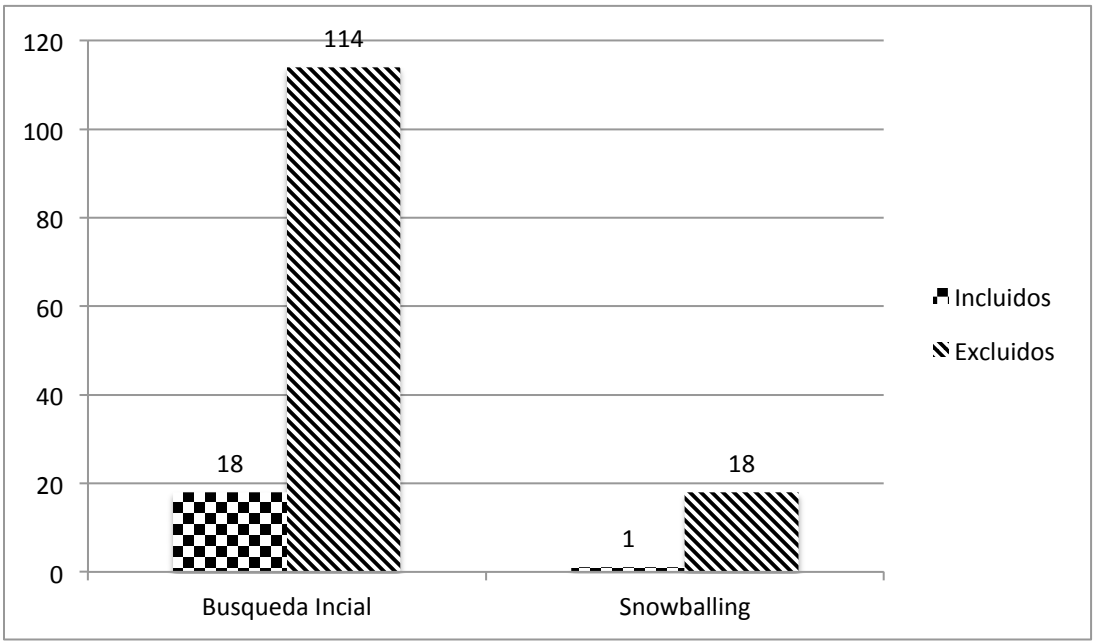

Figura II.3. Número de estudios incluidos y excluidos en las fases de búsqueda.

Así mismo y una vez se ha realizado el aseguramiento de la calidad en todos los estudios incluidos, se muestran a continuación, en la Figura II.4, los resultados obtenidos en relación a dicha evaluación de calidad.

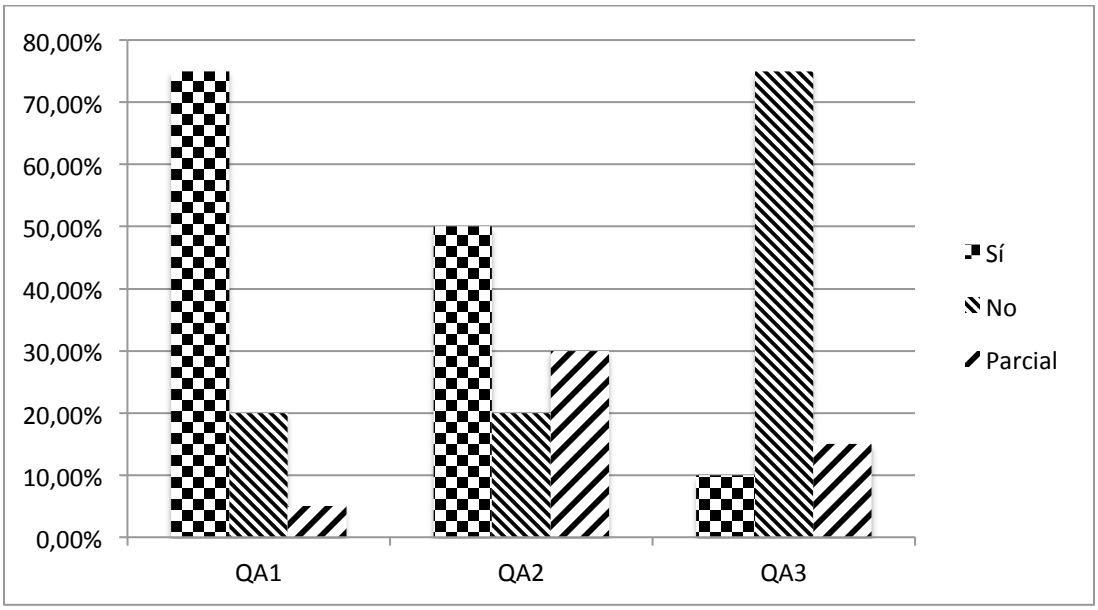

Figura II.4. Aseguramiento de la calidad en los estudios incluidos.

Como principal conclusión de la evaluación de la calidad realizada en este estudio, se observa que si bien los estudios están generalmente bien motivados y están enmarcados los trabajos previos, la definición de los metamodelos no es completa en todos los casos. Así mismo destaca especialmente en lo referente a la validación, que la gran mayoría de estudios no aporta ninguna implantación y los que la aportan son generalmente validaciones teóricas sobre problemas reales salvo algún caso en que se ha implantado dicho metamodelo en un entorno real.

Es por tanto la dificultad para implantar estos metamodelos en entornos reales lo que lastra la calidad de los estudios presentados, si bien desde el punto de vista teórico son correctos y completos.

A continuación se presentarán en la Tabla II.13 el resumen de los estudios utilizados en el análisis cuantitativo expuesto en párrafos anteriores. En este tabla la primera columna es la referencia, la segunda columna el título del trabajo y las tres columnas siguientes la evaluación de la calidad asignada a cada estudio, siendo el valor P la representación de la escala de evaluación "Parcialmente".

Tabla II.13. Resumen de la información para cada estudio incluido.

Referencia	Título	QA1	QA2	QA3
[Ameller et al. 2015]	"Development of service-oriented architectures using model-driven development: A mapping study"	Sí	P	No
[Arni-Bloch & Ralyte 2009]	"MISS: A Metamodel of Information System Service"	P	Sí	No
[Arsanjani et al. 2007]	"S3: A Service-Oriented Reference Architecture"	Sí	No	No
[Benguria et al. 2007]	"A platform independent model for service oriented architectures"	Sí	Sí	P
[Braytee et al. 2015]	"A Review and Comparison of Service E-Contract Architecture Metamodels"	Sí	Sí	No

[Braga et al. 2016]	"AIRES: An Architecture to Improve Software Reuse"	Sí	No	No
[Cauvet 2010]	"Method Engineering: A Service-Oriented Approach"	Sí	No	No
[Comerio et al. 2015]	"Service portfolio management: A repository-based framework"	Sí	P	Sí
[Duddy et al. 2010]	"Design of a Model-generated Repository As a Service for USDL"	P	P	No
[Emig et al. 2007]	"Model-driven development of SOA services"	P	Sí	P
[Gill et al. 2016]	"Scaling for agility: A reference model for hybrid traditional-agile software development methodologies"	Sí	P	No
[Hahn & Slomic 2008]	"Agent-based Extensions for the UML Profile and Metamodel for Service-oriented Architectures"	Sí	Sí	No
[Hock-Koon & Oussalah 2010]	"Composite service metamodel and auto composition"	Sí	Sí	No
[Lethrech et al. 2007]	"A Generic Metamodel for Adaptable Service Oriented Systems Modeling using DSM Approach"	Sí	Sí	P
[Ott et al. 2011]	"Foundations of a Reference Model for SOA Governance"	No	P	No
[Peinado et al. 2015]	"A metamodel and taxonomy to facilitate context-aware service adaptation"	Sí	Sí	No
[Piprani et al. 2008]	"A Metamodel for Enabling a Service Oriented Architecture"	Sí	Sí	No
[Postina et al. 2010]	"An EA-approach to Develop SOA Viewpoints"	Sí	Sí	Sí
[Ramadour et al. 2010]	"Towards services paradigm: principles and models"	Sí	P	No
[Silva-Lepe et al 2009]	"A Simplified Approach to Describing, Discovering and Composing Situational Enterprise Services"	Sí	P	No

Una vez concluido al análisis cuantitativo se ha realizado un análisis cualitativo sobre dichos estudios. Los estudios más relevantes que intentan proporcionar información para dar respuesta a la pregunta de investigación RQ2 se han seleccionado a continuación, enumerando aquellas características consideradas como importantes en relación al estudio en curso. Estos estudios seleccionados son:

- [Silva-Lepe et al. 2008] debido a que muestra una propuesta para describir, descubrir y componer servicios dentro de un contexto empresarial, tratando los servicios como incardinados dentro de una organización y con un contexto particular. Sin embargo aunque ofrece de forma descriptiva un modelo, no formaliza ningún metamodelo en UML para ser utilizado y tampoco presenta perfiles UML que puedan ser utilizados en un entorno de producción real.
- [Emig et al. 2007] y [Peinado et al. 2015] presentan un metamodelo y una taxonomía para adaptar los servicios que son sensibles al contexto. Según los autores, las empresas están adoptando arquitecturas orientadas a servicios para responder a los rápidos cambios en el mercado. A pesar de que hay excelentes herramientas y marcos para la adopción de arquitecturas orientadas a servicios y para el desarrollo de servicios, la

última adaptación al contexto no ha sido tratada adecuadamente todavía. Los enfoques actuales se centran principalmente en la resolución de cuestiones sensibles al contexto para las aplicaciones web, centrándose principalmente en la adaptación del lado del cliente pero existe una clara falta de taxonomía orientada a la propia organización. Estas propuestas presentan un metamodelo que aunque está orientado a múltiples contextos y tiene una serie de elementos interesantes, no está orientado desde un punto de vista funcional sino tecnológico.

- [Braytee et al. 2015] realiza una interesante comparación entre diversos metamodelos de servicios que tienen el contexto como parte del mismo, pero enfocándolos desde el punto de vista estructural, de comportamiento y tecnológico, haciendo demasiado hincapié en la parte tecnológica. Concluye que es necesario que en el metamodelo se aúnen la parte estructural y funcional con un cierto grado de abstracción, es decir, sin estar demasiado condicionado por la tecnología. El estudio no finaliza con la propuesta formal, en ningún lenguaje, de este metamodelo.

El resto de estudios quedan fuera del objeto directo de esta Tesis Doctoral ya que son:

- Estudios sobre metamodelos de arquitecturas SOA, que ofrecen metamodelos que intentan metamodelar la arquitectura SOA [Hahn et al. 2008] o los Sistemas de Información como Servicios [Arni-Bloch et al. 2009], saliéndose del objeto de estudio de este apartado.
- Estudios sobre metamodelos de servicios, pero enfocados en un grado de profundidad tecnológica más cercanos a los Web Services [Balazs et al. 2013], [Elyacoubi et. al 2009] o [Postina et al. 2010]. Como ocurría con el punto anterior, quedan fuera del alcance de este capítulo al estar centrados en soluciones tecnológicas particulares para metamodelar los servicios desde el punto de vista tecnológico y no funcional.

Podemos concluir por tanto, a raíz de este estudio, que para el descubrimiento de los Servicios Candidatos será necesario definir un metamodelo de servicios que reúna las características que siguen:

- Ha de ser abstracto, independiente de la tecnología, pero lo suficientemente flexible para describir las características estructurales del mismo desde el punto de vista de la construcción del servicio.
- Es necesario que ese metamodelo integre las características funcionales de los servicios mediante diversas taxonomías.
- Es necesario que este metamodelo integre las características del contexto de la organización en la que esté incardinado el servicio, de manera que la estructura de la organización ayude a definir también dicho servicio, debido a que éste va a ser usado en una organización concreta y será por tanto un servicio gobernado, por lo que los elementos que indicaba el OMG (interesados, políticas, acuerdos de nivel de servicio, etc..) deberán estar presentes en dicho metamodelo.

4.6. Limitaciones de este estudio

Aunque se ha intentado seguir las mejores guías y prácticas propuestas para llevar a cabo una revisión sistemática de la literatura, el estudio contiene una serie de limitaciones que se exponen a continuación:

- En primer lugar, el número de motores de búsqueda incluidos limita los resultados obtenidos. Dado que el campo de aplicación podría ser muy diverso en el ámbito de las arquitecturas orientadas a servicios, en lugar de identificar conferencias y revistas específicas se ha decidido realizar búsquedas en las principales bases de datos, en particular: Google Scholar, Science Direct, Springerlink, Web of Science, IEEEXplore y ACM Digital Library. Aun pudiendo suponer que son una muestra muy representativa de las bases de datos de publicaciones existentes, es evidente que ampliar la búsqueda con otras bases de datos podría ampliar los estudios candidatos, siendo por tanto una limitación del presente estudio.
- Otra actividad que ha podido limitar los resultados obtenidos es la relacionada con los criterios establecidos para incluir o no los potenciales trabajos en la revisión. Por un lado, se ha decidido incluir sólo trabajos publicados en inglés, por lo que cualquier propuesta en otro idioma ha quedado excluida, aunque tras el análisis desarrollado, se piensa que esta limitación es menor y que ningún trabajo ha quedado fuera por este motivo. Por otro lado, la falta de acceso al texto completo ha sido un motivo para excluir algún trabajo, algo que ha ocurrido con un 2% de las potenciales iniciativas.
- Para evaluar la calidad de los estudios incluidos se han propuesto una serie de preguntas que nos permitiesen analizar si con los mismos se tenía suficiente información como para responder a las preguntas de investigación que nos planteábamos. Para cada pregunta, se ha evaluado con un "Sí", "Parcialmente" o "No", existiendo un cierto grado de subjetividad que podría limitar este estudio.
- Por último, el propio proceso seguido para evaluar los diversos estudios supone en sí una limitación, debido a que la primera fase de inclusión se realiza sobre el título, resumen y palabras clave. Extender esa primera actividad de filtrado sobre el texto completo podría generar la inclusión de algún nuevo estudio en el que su contenido no hubiera sido bien resumido y expresado en los campos anteriores. Sin embargo y aunque presente una limitación la relación entre rigurosidad y esfuerzo se adecua de este modo.

5. Conclusiones

De los apartados precedentes, una vez que se ha realizado la revisión sistemática de la literatura sobre el objeto de esta Tesis Doctoral, se concluye que no se ha encontrado ningún estudio que resuelva el proceso de descubrimiento de Servicio Candidatos que cubran los requisitos de una nueva aplicación, dentro del desarrollo con metodologías ágiles y cuyos servicios estén gobernados dentro de una organización y especificados en su Catálogo de Servicios, por lo que será objeto de este trabajo de Tesis Doctoral proponer dicho proceso de descubrimiento.

Así mismo y una vez que se ha realizado la revisión sistemática de la literatura sobre la relación de los dos paradigmas que interviene en dicho proceso de descubrimiento y a raíz de los resultados obtenidos, será objeto también de este trabajo proponer dos metamodelos que se relacionen en aras de realizar dicho proceso de descubrimiento de Servicios Candidatos.

En cuanto al metamodelo de requisitos ágiles:

- Deberá contar con las Historias de Usuario, como técnica ágil fundamental a la hora de representar la definición de los requisitos, debido a su mayor adopción.
- Deberá contar con "*Scrum*" como proceso de organización de las Historias de Usuario, al ser la técnica más extendida y con mayor penetración en el agilismo.
- Deberá contar con técnicas "*Wideband-Delphi*", basadas en el juicio de los expertos y en estimación por comparación, en la priorización de requisitos, para hallar el valor ganado, debido a que la inversión para la estimación del valor con estas técnicas está minimizado, haciéndolas más eficientes.
- Deberá realizar una formalización de toda esta funcionalidad en un metamodelo en lenguaje UML, debido a su universalidad y conocimiento por la casi totalidad de los miembros de un equipo de desarrollo software. Se habrán de modelar en UML todos los artefactos que intervienen en la descripción del requisito.
- Deberá integrar la interacción entre los interesados ("*stakeholders*") y los equipos de desarrollo ("*agile team*").

En cuanto al metamodelo de Servicios:

- Deberá integrar tanto las características funcionales de los servicios mediante diversas taxonomías como las características del contexto de la organización en la que esté incardinado el servicio.
- Deberá ser abstracto, es decir, independiente de la tecnología, pero lo suficientemente flexible para incluir las características estructurales básicas desde el punto de vista de la construcción del servicio.
- Deberá realizarse dicha formalización en un metamodelo en lenguaje UML, debido a su universalidad y conocimiento por la casi totalidad de los miembros de un equipo de desarrollo software.

Una vez detectadas estas carencias al realizar el estudio de la literatura actual y no haber encontrado ninguna solución que integre todas la necesidades para la realización de estas Tesis Doctoral, el trabajo continua, ya en el Capítulo III, definiendo los objetivos de esta Tesis Doctoral y describiendo de forma detallada el problema a resolver.

Capítulo III Planteamiento del Problema

La presentación del estudio del estado del arte del capítulo anterior ha servido para realizar una revisión de la literatura actual y para constatar que no existe actualmente un proceso para el descubrimiento de Servicios Candidatos, usando requisitos formalizados con técnicas ágiles e integrado de forma sistemática con un Catálogo de Servicios preexistente en la organización. Además, el estudio de la literatura actual nos ha permitido definir las carencias que existen sobre diversas propuestas de modelización de Servicios en orden a la formalización del Catálogo de Servicios dentro de una organización, así como sobre la formalización y modelado de un requisito usando metodologías ágiles, de cara a definir mejor tanto el problema concreto como los objetivos a cubrir en esta Tesis Doctoral.

Una vez conocido, por el capítulo anterior, el estado del arte, el capítulo continúa planteando los objetivos de la tesis. Este capítulo comienza, por tanto, con una descripción detallada del planteamiento del problema y precisa cuáles son los objetivos a conseguir.

Con los objetivos enmarcados y el problema definido, el capítulo continúa describiendo cuáles han sido las influencias que han desembocado en la realización de la propuesta presentada en esta Tesis Doctoral y concluye con una serie de consideraciones finales que cierran la definición del alcance del problema.

1. Planteamiento del problema

Dentro de una organización es habitual tener, tanto para la parte de Gobierno de los Servicios como para la parte de desarrollo de nuevas aplicaciones, proveedores diferentes, fuentes de financiación diferentes, equipos internos diferentes, diferentes responsables e incluso diferentes competencias [Hynes 2015]. Es por ello que este proceso de búsqueda de funcionalidad ya implementada, en estas organizaciones, es a veces complejo, manual y su éxito responde a la buena voluntad de los participantes en el mismo. Esto representa un problema grave para la reutilización del software, de los servicios y del conocimiento e incide por tanto directamente en la eficacia y en la eficiencia, tanto en la prestación de Servicios, como en el propio desarrollo de aplicaciones, al utilizar paradigmas que se caracterizan por necesitar una especificación completa y detallada de la fase de toma de requisitos, para posteriormente ejecutar las siguientes fases del desarrollo.

No existe por tanto, en esas organizaciones, un proceso formal y automatizado para el descubrimiento de funcionalidad dentro de un contexto específico en una organización, siendo ese contexto precisamente los servicios.

Cuando es necesario desarrollar aplicaciones con una metodología del campo de la Ingeniería Web, dentro de un contexto de servicios, es necesario, de una manera sistemática y automatizada, detectar qué requisitos del nuevo desarrollo se encuentran ya en el contexto de la organización. Estos Servicios, que contienen parte de la funcionalidad del nuevo desarrollo y pertenecen al Catálogo de Servicios, se denominan Servicios Candidatos.

Para esta formalización y automatización sistemática y a raíz de las conclusiones obtenidas durante el capítulo anterior, se plantean los problemas que se quieren solventar en este trabajo de investigación.

Esto implicará necesariamente la formalización completa y temprana de los requisitos con un grado de coherencia (granularidad) adecuado para poder descubrir qué servicios cubren la funcionalidad a ejecutar, siendo el diseño de esta formalización uno de los problemas clave a resolver, ya que deberá ser completa y ágil.

Así mismo, el Catálogo de Servicios ha de estar formalizado, por lo que se realizará en esta Tesis Doctoral una propuesta para la formalización de ese Catálogo de Servicios a través de un metamodelo de Servicios, ya que la falta de formalización de los servicios hace inviable que el proceso se pueda automatizar.

La búsqueda de la relación entre la formalización de los requisitos y el Catálogo de Servicios, será otro de los problemas sobre los que tratará esta Tesis Doctoral, ya que habrá de definirse un proceso capaz de correlacionar ambos modelos y poder obtener así los Servicios Candidatos.

Como se ha podido observar, se trata de realizar un paso previo antes de comenzar la descripción formal y exhaustiva de la funcionalidad, para ver cuál está ya contenida dentro del contexto de la organización, en este caso, qué requisitos están ya identificados dentro de los servicios de los que consta el Catálogo de Servicios de la organización y son susceptibles, por tanto, de poder ser especificados y utilizados en el nuevo desarrollo.

Una de las mayores dificultadas, como se enunciaba anteriormente, estará en la definición de un método para formalizar dichos requisitos con una granularidad adecuada o semejante a la de los servicios dentro del Catálogo de Servicios, de manera que pueda hallarse un proceso automático que realice la correlación con la mayor precisión posible. Debido a que se busca la eficiencia y la eficacia, dicho proceso deberá tener un coste razonablemente pequeño en su definición.

Todos estos retos nos llevan a considerar la realización de una propuesta metodológica para el descubrimiento de Servicios Candidatos, dentro de un contexto, que contengan la funcionalidad de un subconjunto de los requisitos de un nuevo desarrollo y por tanto, sean candidatos a ser utilizados en el mismo, debido a que la carencia de este proceso aumenta el tiempo y el coste del desarrollo, no favorece la reutilización del software y disminuye la eficiencia y eficacia tanto en el propio desarrollo de aplicaciones como el gobierno de los servicios.

2. Objetivos

Después de concretar cuál es el alcance del problema, llega el momento de describir de manera concreta los objetivos que se pretenden alcanzar con esta Tesis Doctoral. En este sentido, se establecen los siguientes objetivos a alcanzar:

1. **Análisis de la Situación.** El primer objetivo de la presente Tesis Doctoral será el realizar un análisis del estado de la formalización de requisitos con metodologías ágiles, así como del modelado de Servicios, en orden a ser utilizados en un proceso de

descubrimiento de Servicios Candidatos dentro de un contexto de servicios. Así mismo, el propio proceso será también objeto del estudio del estado del arte.

2. **Definir un Metamodelo de Servicios** para reflejar el Catálogo de Servicios de la organización en el que se definirán y formalizarán los Servicios de dicho catálogo. Gracias a este metamodelo la funcionalidad identificada y viva, en el contexto, estará recogida y formalizada. En este metamodelo se recogerán los aspectos funcionales necesarios para la organización, taxonomía y definición de los Servicios de la misma, así como los elementos estructurales de la organización, necesarios para la incardinación del Servicio dentro de la misma.
3. **Definir un Metamodelo de Requisitos** que permita la formalización ágil, temprana y completa de los requisitos. Para este punto será necesario el uso de las nuevas técnicas ágiles que se han usado con buen resultado, precisamente, para disponer de un conjunto homogéneo de requisitos funcionales y no funcionales. Entre las técnicas a utilizar destacarán: (i) las Historias de Usuario para la formalización del conjunto total de requisitos; y (ii) la utilización de las técnicas enfocadas en el Valor de Negocio (*EVM*, "*Earned Value Management Techniques*") para dar la homogeneidad adecuada al nivel de los Servicios y que se describirán más adelante.
4. **Definir la relación entre los Metamodelos y el proceso de descubrimiento de Servicios Candidatos**, que permita la definición de un proceso para descubrir, de forma sistemática y coherente, qué Servicios dentro del Catálogo de Servicios dan cobertura a un conjunto total o parcial de los requisitos, es decir, identificar los Servicios Candidatos para su análisis.
5. **Desarrollar las herramientas necesarias para la ejecución del proceso,** mediante una versión aplicable y generalista de los metamodelos, definiendo lenguajes específicos de dominio adecuados para que sea posible instanciar ambos metamodelos, así como las herramientas software necesarias y la arquitectura tecnológica para dar soporte al proceso de descubrimiento de Servicios Candidatos.
6. **Evaluar los resultados obtenidos en un entorno de producción** validando el modelo teórico propuesto en un entorno real cuya aplicación haya sido considerada como satisfactoria.

3. Influencias

Los resultados de investigación expuestos en los siguientes capítulos están influenciados por diferentes trabajos, que están relacionados con los paradigmas expuestos en la introducción de esta Tesis Doctoral.

Referente a los Servicios, existe una influencia directa de los trabajados realizados en la Consejería de Cultura de la Junta de Andalucía (España) y materializados en el Trabajo Fin de Master del autor [Sedeño 2012] y en otras propuestas que dan lugar al Modelo Objetivo SOA para una administración pública [Sedeño et al. 2013, 2014a, 2014c]. Estas propuestas han estado encaminadas a transformar una administración pública en una organización capaz de operar bajo un paradigma de Servicios, definiendo un Modelo Objetivo SOA y proponiendo una metodología para realizar, primero la transformación de la organización hasta la implantación

de dicho modelo y después, el gobierno del mismo, ya en la fase operacional. De estas propuestas y de las conclusiones obtenidas en la revisión sistemática de la literatura del capítulo anterior, se deducirá el metamodelo de Servicios presentado en esta Tesis Doctoral.

Con respecto a las metodologías ágiles, destacar los recientes trabajos dentro del Grupo de investigación IWT2 relacionado con la planificación, estimación y gestión de proyectos de desarrollo Web mediante metodologías ágiles, que han abarcado desde la creación de una metodología propia [Torrecilla et al. 2015] para la gestión de proyecto Web, implantada en la Consejería de Cultura, hasta la integración con paradigmas basados en la Ingeniería Web guiada por Modelos [Sedeño et al. 2014b], haciendo especial hincapié en la toma de requisitos de forma ágil, pasando por la realización de diferentes casos prácticos con proyectos reales en varias administraciones [Torrecilla et al 2014a, 2013b y 2013a], orientados tanto a la prestación de Servicios de Gobierno Electrónico con metodologías ágiles, como a la gestión de la infraestructura para dicha prestación. También se ha estudiado cómo la adopción de estas técnicas por parte de las organizaciones posibilitan la obtención de los estándares de madurez CMMI-Dev niveles 2, 3 y 4 [Torrecilla et. al 2012, 2014b y 2016].

Por último y en relación con la Ingería de requisitos, el lenguaje de especificación WebRE para requisitos [Escalona & Knoch 2012] como principal exponente de una agilización en la toma de requisitos, ya que sólo unas pocas metodologías Web proporcionan un enfoque para la obtención de los requisitos y las técnicas para su especificación. En este caso, WebRE contiene los conceptos claves necesarios para la especificación de los requisitos de los Sistemas Web. Los beneficios que aporta esta especificación se pueden resumir en:

- Unificación de elementos de representación de requisitos, así como su correspondencia con los principales artefactos de las principales metodologías del campo de la Ingeniería Web guiada por Modelos, como son NDT [Escalona 2004][Escalona et al. 2008], OOHDM [Rossi & Schwabe 1998], UWE [Koch & Kraus 2002] o W2000 [Baresi et al. 2003].
- Los elementos del metamodelo se asignan a las construcciones de modelado de las diferentes metodologías Web. De esta manera se da el requisito previo para, modelo a modelo, realizar transformaciones, lo que permite la construcción de diferentes puntos de vista de los requisitos de un sistema Web que utilizan diferentes metodologías Web.
- Los conceptos claves que se utilizan para la definición de un lenguaje de modelado común se presentan en un perfil UML para requisitos Web, lo que facilita su posterior modelado y estandarización en el uso.

Una vez enunciadas estas influencias, esta sección describe con más detalle cada una de las propuestas que han servido como base para obtener los resultados que se presentan a lo largo de capítulos posteriores.

3.1. El paradigma SOA en las organizaciones.

Como se ha comentado en el apartado anterior, el metamodelo de Servicios que se propone es el resultado de diferentes trabajos del autor, que se han ido desarrollando a lo largo del tiempo, y si bien el desarrollo se ha centrado en la administración pública, en el contexto de esta Tesis Doctoral este tipo de organizaciones suponen un ejemplo concreto de organizaciones en cuya naturaleza está la prestación de servicios y que al ser dueños de esos servicios, han de realizar

un gobierno eficiente y eficaz de los mismos. Así mismo, las administraciones públicas son las encargadas, en última instancia, del desarrollo de las aplicaciones a través de las cuales prestan esos Servicios a la ciudadanía, por lo que todas las consideraciones sobre el Modelo Objetivo SOA son aplicables a esta Tesis Doctoral.

Actualmente, las Arquitecturas Orientadas a Servicios están muy difundidas entre las organizaciones de carácter privado, tanto en la teoría como en la práctica. Sin embargo, cuando nos centramos en las Administraciones Públicas y por ende en aquellas organizaciones que compartan sus características, nos encontramos que las implantaciones de estas arquitecturas han sido reducidas a meros desarrollos de Servicios Web, orquestación de procesos o gobiernos de servicios basados solamente en políticas de carácter técnico, impidiendo así una prestación de los servicios realmente eficaz. [Gichoya 2005]

A raíz de la experiencia en la Consejería de Cultura de la Junta de Andalucía (descrita en el Capítulo VIII) y de otros trabajos del autor, se realizó una propuesta metodológica para la transformación de una Administración Pública en una organización con capacidad para operar bajo un paradigma SOA, de cara a poder implantar el Modelo Objetivo SOA, apoyándose en todos los aspectos del contexto de negocio de una administración y siendo los disparadores del proceso la propia naturaleza de la Administración Pública.

La adopción del paradigma SOA es una tarea compleja que involucra aspectos tecnológicos, organizacionales y de gobierno y no solo requiere un conocimiento teórico de los procesos, sino un conocimiento profundo de dichos procesos en el contexto de la organización.

Hay diferencias fundamentales entre organizaciones privadas y públicas, y habitualmente las soluciones aplicadas con éxito en entornos privados no se han podido aplicar con un resultado satisfactorio en entornos públicos [Haynes 2015].

En necesario por tanto, para poder aplicar el Modelo Objetivo SOA, entender el contexto de una organización pública para extender la práctica a todas las organizaciones.

A diferencia de las organizaciones privadas, enfocadas en los beneficios y en dar valor a sus "*stackeholders*", está en el corazón de la administración el interés público. Su éxito está asociado al cumplimiento de objetivos sociales. Además y de manera simultánea, estas organizaciones han de dar respuesta a demandas sociales y políticas que no tiene reflejo en las privadas, y para completar el problema, las administraciones crean políticas cuyo éxito es más difícil de evaluar que los resultados financieros o productivos del sector privado [Tregear & Jenkins 2007].

Los procesos en las administraciones públicas son más complejos que en las privadas [Repa 2006], debido a que los procesos de toma de decisiones son lentos y complejos y tienen una estructura organizativa más rígida y politizada. Estos procesos de decisión dependen además de departamentos jurídicamente independientes, muchas veces con intereses encontrados, que han de articularse junto con otras normativas legales, así como con niveles competenciales diferentes. Esto hace que mejoras o modificaciones de los procesos se conviertan en una ardua tarea, que a menudo requiere cambios legales. Por lo tanto, una metodología para la transformación de organizaciones deberá contar con salvar los siguientes impedimentos [Hall 2007]:

- Trabajar con una burocracia compleja y muy amplia.
- Cambios contantes en los procedimientos administrativos.
- Competencias por los recursos financieros (pocas fuentes de financiación).
- Ir más allá de la implantación de Servicios Web [Goudos et al. 2007], la orquestación de procesos [Popescu et al. 2013] o el cumplimiento de reglas de carácter técnico [Castellano et al. 2005].

Es necesario por tanto que sea una metodología basada en el entendimiento de los elementos organizacionales y sus relaciones. Existen metodologías basadas en la comprensión de los elementos de la organización y sus relaciones y, también, del análisis de los objetivos [Delgado et al. 2013]. De la misma manera hay varios enfoques sobre el gobierno de TI como ITIL, COBIT o TOGAF con una profunda relación con la prestación del Servicio. Sin embargo, estos métodos no especifican cómo transformar las Administraciones Públicas, atendiendo a su naturaleza jurídica, en organizaciones capaces de operar bajo el paradigma SOA.

En definitiva, la propia administración, y por ende toda organización, debe ser transformada para funcionar en entornos SOA incorporando sus propias barreras legales y humanas como parte de la metodología. Hay bastante literatura sobre frameworks de Arquitectura Empresarial (EA) para las organizaciones privadas sobre los modelos de referencia basados en SOA [Ruas de Oliveira et al, 2010] [Alwadain et al. 2013], mientras que hay muy poca información acerca de los factores críticos de éxito (CFS) en Administraciones Públicas [Abdul-Manan & Hyland, 2013] [Koumaditis et al. 2013]. A pesar de tener implantaciones de arquitecturas empresariales relacionadas con los productos y servicios, no se encuentra ningún framework genérico para cualquier tipo de Administración Pública.

Todos los conceptos expuestos hasta ahora como el marco regulatorio, la estructura organizativa y los procesos de gobierno, se incluyen en el "contexto de negocio", como parte del análisis de los imperativos que se describirá en el apartado siguiente.

De cara a esta Tesis Doctoral, se enumerarán brevemente los aspectos fundamentales de esta aproximación metodológica, de cara a entender la proveniencia de los diferentes elementos del Metamodelo de Servicios. Se irá haciendo hincapié especialmente en aquellos aspectos del proceso relevantes para esta Tesis Doctoral.

3.1.1 Análisis de Imperativos

El método del Análisis de Imperativos (Figura III.1) representa el paso previo hacia la transformación de una administración en una organización basada en SOA. Partimos del contexto de negocio de una Administración Pública para llevar a cabo este proceso de análisis. Este contexto muestra el negocio de la organización, sus áreas funcionales y su estructura orgánica, a fin de determinar su situación con respecto al futuro modelo de gobierno. En lo que respecta a esta Tesis Doctoral, el final de este proceso cristalizará en el Modelo Objetivo SOA, en el que se encuentra el Catálogo de Servicios, del cual se propone la formalización en el Capítulo IV de este trabajo.

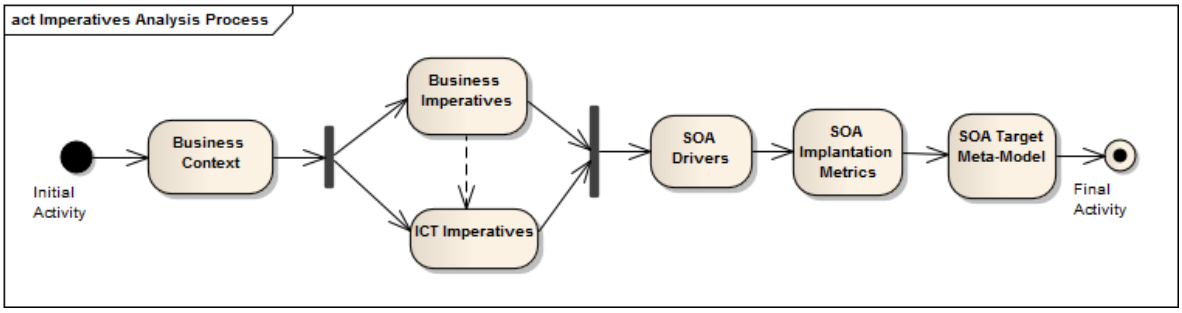

Figura III.1. Análisis de Imperativos.

El punto de partida de este método son los imperativos -entendidos como deber o requisito inexcusable-, que se pueden definir para una Administración Pública y que una vez conseguidos pueden permitir alcanzar la excelencia en la prestación de los servicios a los ciudadanos, así como en la gestión de los procesos internos en los que se apoyan.

Los servicios suministrados por la Administración Pública a los ciudadanos constituyen una parte esencial de este contexto de negocios. De la misma manera, también podemos observar la determinación de los factores que condicionan la organización desde ambos puntos de vista, el negocio y las TIC (política corporativa definida a nivel de TI).

Una vez que el contexto de negocio ("*Bussines Context*") ha quedado fijado, se procede a la identificación de los imperativos de negocio ("*Bussines Imperatives*"), los cuales, hacen referencia a los retos del negocio y de los servicios hacia a los ciudadanos a los que la Administración Pública ha de hacer frente. Obtenemos después los imperativos TIC ("*ICT Imperatives*") que son la traducción de esos objetivos de negocio a un lenguaje técnico. Una vez identificados los imperativos de negocio, a través de un proceso deductivo que relaciona los retos del negocio, con posibles iniciativas que se pueden llevar a cabo desde el área de TIC con el fin de acometerlos con éxito.

Estos imperativos TIC han de mantener un cierto nivel de abstracción, ya que definen lo que se debe hacer, pero no cómo hacerlo. En un siguiente paso y basándonos en los imperativos TIC, se establecerán las implementaciones concretas bajo el paradigma SOA que puedan dar respuesta a esos imperativos TIC. Estas implementaciones SOA se denominan "drivers" ("*SOA Drivers*"). De esta manera podemos entender el valor que una implementación SOA puede proporcionar a nivel corporativo, en relación con la necesidad de hacer frente a los retos establecidos por los imperativos TIC y por extensión, a los imperativos de negocio.

Por lo tanto, una vez que la relación entre los Drivers SOA y los imperativos de negocio se ha definido (mostrándose a través de las diferentes matrices de trazabilidad)[1], se observa claramente que los Drivers SOA deben actuar como motor en la implantación de SOA. Se definirán métricas de implementación de SOA ("*SOA Implantation Metrics*"), después de identificar los Drivers SOA. La definición y puesta en práctica de estas métricas se llevan a cabo con el objetivo de evaluar si la organización se transforma para operar bajo un paradigma SOA.

[1] Una matriz de trazabilidad es un documento, por lo general en forma de tabla, que correlaciona los documentos de referencia que requieren una relación de muchos a muchos para determinar la integridad de la relación.

Cada métrica SOA representa cómo se mide la eficiencia de cada Driver SOA. Finalmente, llegaremos a una instancia del Modelo Objetivo SOA ("*SOA Target Metamodel*"), una vez que el conjunto de Drivers SOA hayan sido ejecutados.

3.1.2 El Modelo Objetivo SOA

En los trabajos del autor, como se ha mencionado anteriormente, se propone un Modelo Objetivo SOA para ser adoptado en la organización, que representa la estructura necesaria de componentes para comenzar con la fase de operación bajo un paradigma SOA.

Este modelo debe ser entendido por tanto, no sólo como parte de la definición de la metodología SOA, propuesta en el apartado anterior, sino como su propio resultado. Esto significa que es necesario definir primero un conjunto de componentes que, después de ejecutar el análisis de imperativos, muestre el estado final objetivo de la organización, que podrá operar bajo un paradigma SOA. Este diagrama de componentes comprende cuatro elementos fundamentales que se refieren a la organización (gobierno), al negocio (funcional), a la metodología y la tecnología y que se muestran en la Figura III.2.

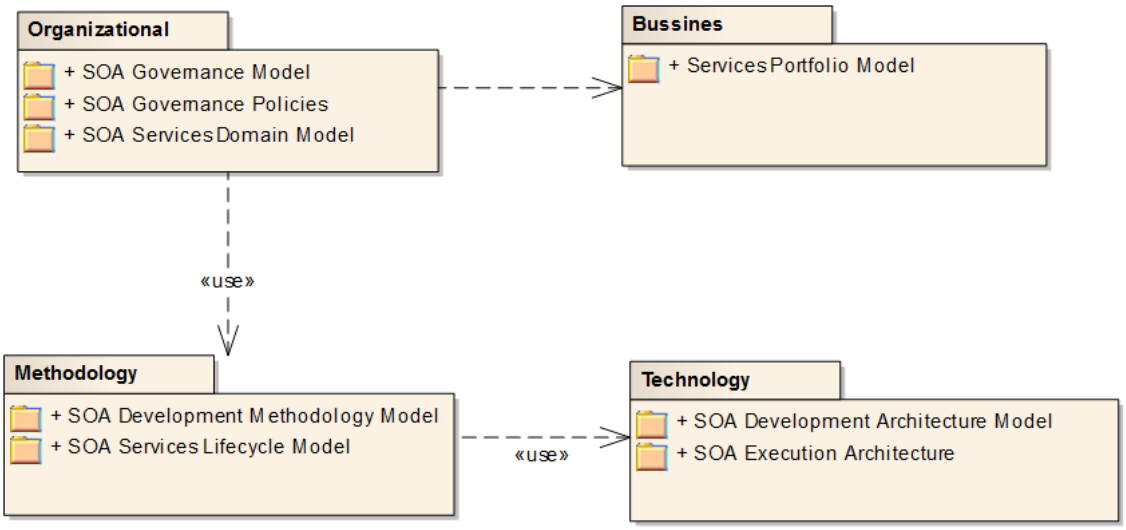

Figura III.2. Modelo Objetivo SOA.

A continuación se describirán brevemente cada uno de los componentes, haciendo especial mención en aquellos que participan en el Catálogo de Servicios propuesto en el Capítulo IV de esta Tesis Doctoral y que se centra especialmente en el Negocio y la Metodología; y por ende en la tecnología.

Componentes organizativos. En este nivel, se definirán las políticas y procedimientos que deberá estructurar el gobierno SOA. Ellos constituyen el conjunto de reglas, restricciones y requisitos para monitorizar el comportamiento y las acciones dentro de los Servicios SOA, para que puedan estar alineados con las necesidades del negocio obteniendo un resultado exitoso después de la adopción del paradigma SOA. Este modelo de gobierno se ocupará de la definición del Modelo de Dominios de los Servicios ("*SOA Services Domain Model*"), en el caso de una administración basado en la estructura orgánica y jurídica, y que representa una de las formas

de incardinar los Servicios dentro del Catálogo. El Modelo de Gobierno SOA ("*SOA Governance Model*") y sus políticas, no entran en el alcance de esta Tesis Doctoral, pero sería el siguiente paso a ejecutar sobre los Servicios Candidatos, ya que nos indicarían razones de gobierno, además de las expuestas en este trabajo como contenedores de funcionalidad del contexto y que hacen referencia la negocio, para su posible uso, desencadenado así todas las políticas ("*SOA Governance Policies*") del Ciclo de Vida de los Servicios y de Gobierno de los Servicios aplicados a esos Servicios Candidatos.

Componentes de negocios (funcional). Este componente se basa en el análisis de los procesos de negocio de la organización, obteniendo como entregable el Catálogo de Servicios (*"Services Porfolio Model")* que será la pieza angular formalizada con todo detalle en el Capítulo IV de este trabajo. Este Catálogo de Servicios recibe también el nombre, en la literatura, de Portfolio o Cartera de Servicios.

Componentes metodológicos. Esta parte del modelo de componentes se refiere a la necesidad de establecer las bases metodológicas que permitan proporcionar un marco de desarrollo común en dos vías:

- La definición del ciclo de vida de los Servicios SOA ("*SOA Services Lifecycle Model*").
- La definición de la metodología de desarrollo para la adopción de SOA ("*SOA Development Methodology Model*"). Es en este contexto donde se encuadra el problema propuesto en esta Tesis Doctoral ya que consistirá en una metodología de desarrollo de software que integre, dentro de las etapas más tempranas del desarrollo, el desarrollo de aplicaciones con el Gobierno de los Servicios, de la que una actividad fundamental y sobre la que versa este trabajo, consiste en la identificación de nuevos requisitos que ya estén en el "know-how" de la organización y que por tanto deberán ser integrados a través de los Servicios que los contienen y no desarrolladas de nuevo.

Componentes Tecnológicos. En este nivel estarán los componentes tecnológicos adecuados para el desarrollo orientado a servicios, dando prioridad a los que maximicen la reutilización del software y la plataforma de interoperabilidad. Entre ellos destacarán la Definición de la Arquitectura Desarrollo SOA ("*SOA Development Architecture Model")* y la definición e implantación de la Arquitectura de Ejecución SOA ("*SOA Execution Architecture*").

3.1.3 Estrategia Iterativa

Como se ha comentado en la introducción de esta sección, la implementación del paradigma SOA en una organización debe ser considerada como un proceso incremental, que no puede ser enfocado solamente desde las tecnologías de la información, sino desde sus propios procesos de negocio. La estrategia de una organización para tener éxito con esta iniciativa debe consistir en el establecimiento de una línea temporal y una progresión, donde se producen diferentes iteraciones, asegurando que la tecnología está alineada con los aspectos culturales, de organización y de gobierno de dicha organización.

Hay cuatro tipos de iteraciones que deben darse (Figura III.3) para alcanzar el objetivo de esta transformación SOA en la organización. Estas iteraciones serán parte de un ciclo de vida que influirá en la organización de una manera multidimensional, que podrá tener una periodicidad y/o actuar ante disparadores y que volverá a lanzar el análisis de imperativos ante los diferentes cambios de contexto que puedan ir dándose.

Esto dependerá de los disparadores del contexto de negocio, que volverá a poner en marcha el ciclo de vida de acuerdo con el nivel en que el evento se llevará a cabo. Cada una de las iteración estará a su vez formada por un conjunto de iteraciones en el siguiente nivel de abstracción. Estas iteraciones son:

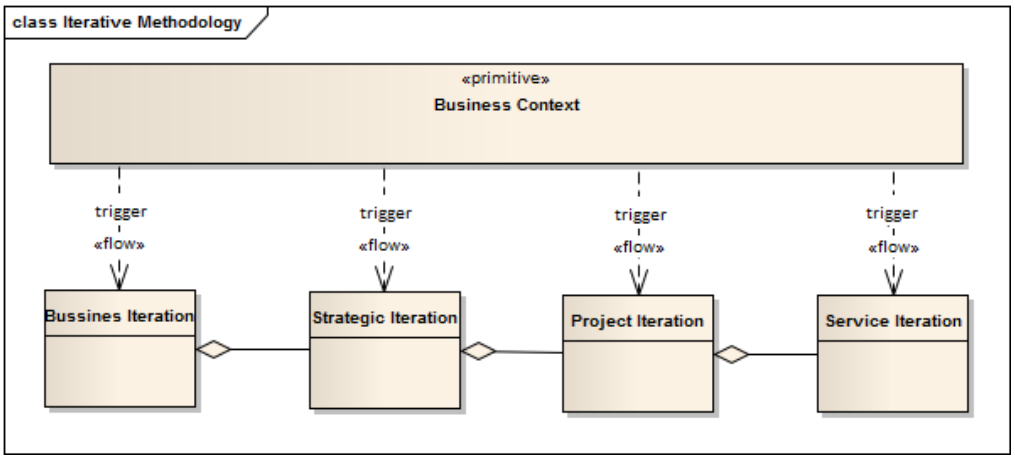

Figura III.3. Metodología Iterativa del Modelo SOA.

- **Iteraciones de negocio ("*Bussines Iteration*")**. Constituyen un ciclo de análisis de negocio que debe permitir la detección de cualquier cambio de importancia dentro de la organización.
- **Iteraciones Estratégicas ("*Strategic Iteration*")**. Para cada iteración de negocio, estas iteraciones estratégicas se centran en el concepto de transformación SOA, que se define entre el análisis de los imperativos de negocio y de las TIC, expuesto en la Figura III.1. Este tipo de iteración dará como resultado llevar a cabo múltiples iniciativas o proyectos de SOA. Como consecuencia de este proceso, la tecnología en sí actuará como elemento de transformación SOA.
- **Iteraciones del proyecto ("*Project Iteration*")**. El grado de ejecución alcanzado en los proyectos SOA propuestos y, por tanto, el grado de transformación de la organización, aumentará en estas iteraciones.
- **Iteraciones de Servicio ("*Service Iteration*")**. Estas iteraciones de Servicios SOA son implementaciones de los requerimientos de negocio o tecnológicos llevadas a cabo por la organización, que finalmente conforman el paradigma SOA.

El problema en que se enmarca esta Tesis Doctoral se encuentra en las iteraciones de Proyectos y Servicios, de manera que la utilización de la solución del presente trabajo pueda proporcionar a la organización que los diferentes proyectos a acometer estén integrados con los Servicios, es por tanto la integración entre los disparadores del proyecto y la utilización de los Servicios que ya existen en la organización.

Se puede realizar un pequeño esbozo con un predicado real obtenido del Gobierno Electrónico en el que uno de los principales asertos del contexto de negocio es la Ley 11/2007, la Ley de Acceso Electrónico de los Ciudadanos a los Servicios Públicos que generaría un imperativo de negocio del tipo "*aumentar el nivel tramitación de procedimientos a través de medio telemáticos*". El imperativo TIC asociado sería *"la implementación de un motor de tramitación de procedimientos administrativos que permita a la organización desplegar nuevos procedimientos*

de manera telemática" y el Driver SOA asociado se definiría como "*la definición de ese motor de tramitación como un conjunto de Servicios SOA de Administración Electrónica que puedan ser orquestados para componer los procedimientos administrativos complejos que se han de proporcionar a los ciudadanos*" y lógicamente se identificarían esos servicios, que una vez formalizados, pasarían a formar parte del Catálogo de Servicios (y por tanto entrarían a formar parte del Modelo Objetivo SOA).

Y es en este punto en el que la propuesta de gestión ágil de requisitos de esta Tesis Doctoral resolvería, siguiendo con el ejemplo, "*la identificación sistemática y automatizada de estos Servicios de Administración Electrónica, ante nuevos procedimientos (proyectos Web) a implementar para los ciudadanos*", es decir, se obtendrían los Servicios Candidatos para ese nuevo procedimiento a implementar pudiendo reutilizar el software y posibilitando el uso de estos Servicios a través de las reglas de Gobierno establecidas en el Modelo Objetivo SOA.

3.2. Requisitos en la Ingeniería Web

El presente trabajo de Tesis Doctoral, como se ha ido reflejando a lo largo de estos capítulos, está muy relacionado con la Ingeniería Web, en tanto que los Servicios son prestados en las organizaciones a través de medios telemáticos, es decir, será a través de la Web, el medio por excelencia, en el que serán prestados esos Servicios (y por tanto serán aplicaciones Web lo hay como sustrato tecnológico). Por esta razón los trabajos relacionados para mejorar la Ingeniería Web en la fase de requisitos, han resultado claves para el presente trabajo.

El objetivo de la fase de la Ingeniería de Requisitos ha sido siempre obtener un conjunto estable y completo de requisitos, que sirvan como base para las siguientes fases del proceso de desarrollo. Las actividades que se utilizan para lograr este objetivo son: elicitación, especificación y validación de requisitos [Lowe & Hall 1999].

La elicitación de requisitos es la actividad mediante la cual las funcionalidades del sistema que se construirá son obtenidas de cualquier fuente disponible. Los fundamentos generales de la elicitación de requisitos para la ingeniería de software no se han modificado cuando se aplican a los Sistemas Web. Sin embargo, existen objetivos específicos para los Sistemas Web que han de ser tratados de forma especial y que no se encuentran en los fundamentos generales y son esenciales para la realización de esa elicitación:

- La identificación de los requisitos de contenido.
- La identificación de los requisitos funcionales en términos de necesidades de navegación y procesos de negocios.
- La definición de escenarios de interacción para diferentes grupos de usuarios de la Web.

Además, una vez identificados, la especificación de los requisitos consiste en realizar una descripción de los mismos. Para esta especificación se pueden usar diferentes técnicas que van desde la descripción textual informal hasta la especificación formal en lenguajes como por ejemplo "Z" [Kappel et al. 2003][Escalona & Knoch 2007].

Desde un punto de vista general, esta especificación de requisitos se puede centrar en la descripción de los problemas o de las soluciones [Wieringa 2005]. La descripción del problema está orientado a los objetivo mientras que la descripción de la solución está orientada a patrones. En ambos casos, es importante escribir especificaciones o construir modelos que sean comprensibles para los responsables, proporcionando información suficiente para los

desarrolladores, y permitiendo la validación de los modelos por los usuarios finales. El desarrollo en el dominio web está influenciado por una mayor importancia de la interfaz de usuario, la volatilidad de las necesidades de los mismos y del modelo de negocio y una dificultad mayor en predecir las evoluciones tanto del sistema como del entorno de publicación.

Además, cada especificación de requisitos necesita ser descrita en el grado de detalle y formalidad que es apropiado para cada proyecto correspondiente. La conveniencia de la técnica de especificación se establece principalmente por el riesgo del proyecto y la complejidad de la aplicación Web que se construirá. Las técnicas que se pueden utilizar para la descripción de dichos requisitos abarcan desde el lenguaje natural, plantillas, casos de uso, los lenguajes formales o los prototipos. Las descripciones informales, como las Historias de Usuario o las descripciones semi-formales, como las plantillas y los casos de uso, son particularmente adecuados para describir cómo los usuarios perciben su interacción con un sistema Web.

Los casos de uso se refinan aún más utilizando para ello especificaciones formales o workflows. Ambas representaciones suelen incluir actores, pre y post condiciones, diagramas de flujo, excepciones y situaciones de error, fuentes de información adicionales, resultados a producir, referencias a otros documentos e interdependencias con otros modelos. En particular, en el desarrollo de sistemas Web los objetivos de información, de navegación y de proceso tienen que ser recogidos y especificados.

3.2.1 Metamodelo de Requisitos Web

Al revisar las diferentes propuestas de las principales metodología del campo de la Ingeniería Web guiada por Modelos como son NDT, OOHDM, UWE o W2000 se ha concluido que abordan muchos conceptos similares, sin embargo, no siempre con la misma terminología. Cada metodología tiene también sus puntos fuertes y débiles. Por ejemplo NDT propone una especificación detallada de requisitos desde el principio de un proyecto, pero las plantillas no son fáciles de completar, ya que requieren entrevistas intensivas. Por el contrario, las representaciones visuales como las propuestas por UWE, W2000 u OOHDM son más intuitivas para un primer plano. Por el contrario las notaciones gráficas suelen ser demasiado abstractas para las siguientes fases [Insfrán et al. 2002].

Los conceptos de modelado que presentaron para la especificación de requisitos Web se definen en base a las similitudes de los métodos que se analizaron. Están representados como metaclases UML y constituyen un metamodelo para los requisitos de Ingeniería Web (WebRE) que se representa en la Figura III.4 Se agrupan en dos paquetes, siguiendo la estructura del metamodelo UML: la parte estructural y la parte de comportamiento.

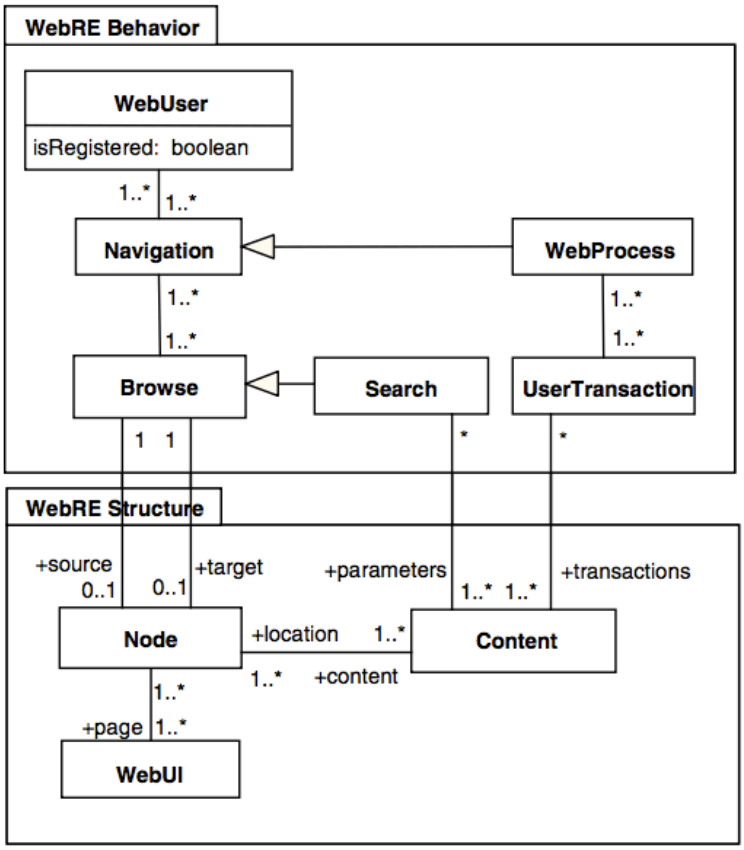

Figura III.4. Metamodelo propuesto para la Ingeniería de Requisitos Web.

(Referencia: [Escalona & Knoch 2012])

El primer paquete modela el comportamiento del usuario dentro de un sistema Web, para ello presenta las metaclases "*Navigation*", "*WebProcess*", "*WebUser*", "*Browse*", "*Search*" y "*UserTransaction*". La funcionalidad de un sistema Web está paralelizada por las acciones de navegación, tanto las del sistema ("*Navigation*") como las del usuario ("*Browse*") que a su vez se particularizan en procesos web ("*WebProcess*") y en búsquedas ("*Search*") respectivamente. Estos procesos Web se corresponden además con las transacciones ("*UserTransaction*") por parte de los usuarios. La navegabilidad de un sistemas Web comprende un conjunto de actividades que el "*WebUser*" realizará para llegar a un nodo de destino. Una actividad de exploración es la acción de seguir un enlace y está representada por la metaclase "*Browse*". Una actividad de exploración puede ser enriquecida por las acciones de búsqueda, que se representa por la metaclase "*Search*" y que constituye un subtipo de "*Browse*". Una búsqueda tendrá un conjunto de parámetros, que permiten definir las consultas sobre el contenido. Los resultados se muestran en el nodo de destino. Un "*WebUser*" es cualquier usuario que interactúa con un sistema Web.

El segundo paquete del metamodelo es el paquete de estructura, que contiene las metaclases utilizadas para describir los elementos estructurales de un sistema Web: "*Content*", "*Node*" y "*WebUI*". Un nodo es un punto de la navegación en el que el usuario encuentra la información. Cada actividad de exploración se inicia en un nodo fuente ("*source*") y termina en otro nodo objetivo ("*target*"). Los nodos se presentan al usuario en forma de páginas ("*WebUI*"). Hay que tener en cuenta que un nodo puede estar asociado a una o más páginas, y una página puede

estar asociado a uno o más nodos. El concepto de la página está representado por la clase metaclase "*WebUI*". Además, cada nodo puede mostrar diferentes piezas de información. Cada pieza de información de un sistema Web se representa como un contenido ("*Content*").

En lo que respecta a esta Tesis Doctoral, cuyo capítulo V representa un metamodelo de Requisitos, es el esfuerzo de WebRE por identificar y fijar unos elementos universales y definir la correlación de esos elementos con otras metodologías (a través de transformaciones) lo que delimitará y acotará los elementos mínimos con los que ha de contar el metamodelo de requisitos propuesto.

En el caso de la presente Tesis Doctoral se ha querido dar un paso más allá para conciliar las nuevas técnicas ágiles en la toma de requisitos con la Ingeniería Web, pero elevando todavía más la abstracción en la elicitación y especificación de requisitos al que lo lleva WebRE, centrándose en las Historias de Usuario, técnica que se desgranará en el siguiente apartado, pero atendiendo a la información que WebRE estima como necesaria para definir un metamodelo de estas características.

3.3. Metodologías Ágiles

En el entorno de esta Tesis Doctoral nos hemos centrado en las organizaciones que prestan Servicios a través de medios telemáticos y por tanto a través de la Web y relacionado, como se expone en el apartado anterior, con la toma de requisitos en la Ingeniería Web. El auge de Internet y la necesidad de una rápida adaptación a las cambiantes necesidades de los usuarios han coexistido en paralelo con la aceptación de la Ingeniería Web como una disciplina de la Ingeniería de Software [Dephande et al. 2002].

La Ingeniería Web se puede definir como un conjunto de métodos, técnicas y herramientas en Ingeniería de Software que ayuda a un equipo de desarrollo crear sistemas de información Web.

Hay varios enfoques para proyectos de desarrollo de software en lo que a la estimación y planificación se refiere y que provienen de diferentes campos (gestión clásica de proyectos, marco Ágil o basados en algoritmos), pero ninguno de ellos considera individualmente todas las necesidades especiales que los proyectos de desarrollo Web implicarían junto con una perspectiva basada en Valor [Dephande et al. 2002], [Derby et al. 2006]:

- Una estructura de navegación compleja.
- Unos requisitos críticos relacionados con la interfaz.
- Aspectos de seguridad.
- Aumento de la eficiencia del mantenimiento, evitando tiempos de parada.
- Entrega de valor lo antes posible ("as soon as posible").
- Reducción del tiempo desde la ideas hasta la venta ("time-to-market").
- Adaptación a los continuos cambios de requerimientos.

Es importante destacar que algunas de las características antes mencionadas no son exclusivas de los proyectos de desarrollo web y también pueden aparecer en los proyectos no web. Sin embargo, la concurrencia de todas ellas juntas al mismo tiempo puede ser identificado como una especificidad de un proyecto Web.

Se ha analizado en diversos trabajos, dentro del Grupos IWT2, que los métodos de planificación ágiles son apropiados para reducir *"el tiempo de lanzamiento al mercado"* (*"time-to-market")* y adaptarse a las necesidades mutables, así como las técnicas de la Ingeniería Web que abordan, entre otros temas, la complejidad de la navegación, la seguridad y unos requisitos críticos relacionados con la interfaz, pero ninguno de ellos por sí mismos puede cubrir todas las especificadas de los proyectos Web.

En estos entornos, las metodologías de desarrollo ágil de software, con el seguimiento y la medición constante, y la frecuente intervención basada principalmente en el uso de procesos empíricos [Schwaber 1995], se están convirtiendo en una sólida alternativa para aquellas organizaciones que han de planificar, estimar y desarrollar software orientado a la Web [Torrecilla et al. 2015]. Estas metodologías ofrecen un marco adecuado tanto para las características de desarrollo Web expuestos [Ambler 2002], como para la respuesta rápida a los cambios, la adaptabilidad y la reducción del tiempo de desarrollo [Glazer et al. 2008], [Pikkarainen et al. 2008].

Además, como se ha mencionado en los capítulos precedentes, en el enfoque clásico es necesaria una toma detallada de requisitos sobre un entrono estable, no siendo el caso de proyectos Web, donde los requerimientos cambian más rápido. La manera gradual e iterativa de procesar requisitos de las métodos ágiles [Cohn 2004], [Dybá & Dingsoyr 2008] se adaptarán mejor a este caso concreto.

Tanto en la dedicación profesional como investigadora del autor, las metodologías ágiles ocupan un enfoque importante debido a que dan solución a los problemas antes mencionados. Para introducir en esta Tesis Doctoral el enfoque ágil, se partirá de un trabajo en el que ha participado el autor [Torrecilla et al. 2015] sobre la estimación, planificación y gestión de proyectos Web con metodologías ágiles basadas en el Valor y que contextualizarán, junto con las conclusiones obtenidas en el Capítulo II, el metamodelo de requisitos del Capítulo V.

A raíz de la propuesta aportada en dicho artículo con el objeto de planificar, gestionar y estimar proyectos ágiles guiados por valor, se ofrecerá una visión general de estos elementos que compondrán el framework ágil propuesto y se destacarán aquellos que sean relevantes en la conexión con el Capítulo V del presente trabajo:

- El marco propuesto se basa en el ciclo de vida de "*Scrum*", incluyendo las modificaciones ad-hoc para adaptarse a las necesidades especiales de los proyectos Web.
- El marco propuesto se basa en un modelo de estimación no algorítmica que utiliza técnicas "*Wideband-Delphi*" para estimar inicialmente el proyecto, según marcan las metodologías de estimación y planificación ágiles, ya que estas técnicas de estimación responden mejor a las características especiales de los proyectos Web.
- El marco propuesto se basa en un enfoque ágil, por lo que la estimación y planificación es un proceso iterativo e incremental durante todo el proyecto, no sólo en la fase de lanzamiento. Este enfoque se adapta muy bien al corto ciclo de retroalimentación por el que se caracterizan los proyectos de desarrollo Web.
- El marco propuesto no sólo tiene en cuenta los temas de gestión de proyectos clásicos (alcance, coste, tiempo) sino también elementos como la calidad y el Valor de negocio, según lo propuesto por el "triángulo ágil". Esto permite una mejor identificación de lo

que debe ser implementado primero, un elemento crucial en los proyectos de desarrollo Web.

- El marco propuesto utiliza una variante de la técnica "*Earned Value Management*" (EVM) para ayudar a los miembros del equipo a gestionar el proyecto durante el ciclo de vida y controlar el alcance. En definitiva nos ayuda a dar valor, y por tanto prioridad, a la funcionalidad.
- El marco propuesto también mide la productividad del equipo con el objetivo de mejorar a lo largo del proyecto.

En la Figura III.5 se muestra el framework Ágil propuesto y cuyo ciclo de vida está diferenciado en dos fases, la fase de lanzamiento del proyecto y la fase de desarrollo del proyecto:

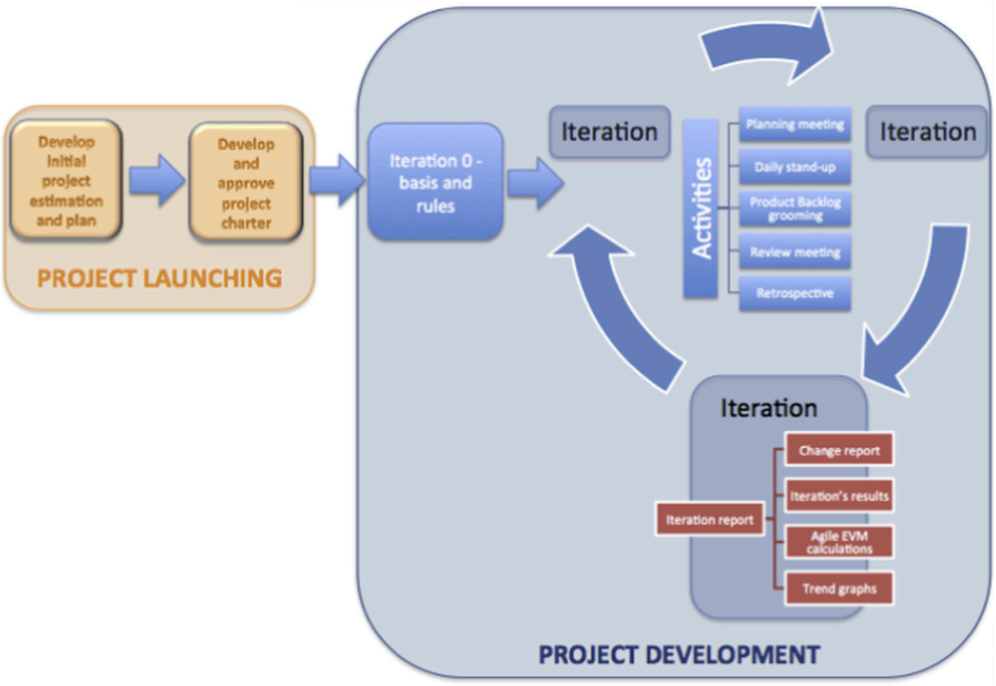

Figura III.5. Framework Agile propuesto.

(Referencia: [Torrecilla et al. 2015)

De cara al presente trabajo de Tesis Doctoral, nos centraremos en la fase de desarrollo del proyecto, en la medida en que propone una toma de requisitos con una serie de características que hace que esta elicitación sea rápida, completa y priorizada. El framework propuesto se basa sobre todo en el trabajo de "*Scrum*", metodología ampliamente conocida, incluyendo algunas otras técnicas para cubrir los procesos de estimación, planificación y de gestión llevadas a cabo en los proyectos de desarrollo. A continuación se presentará el proceso recomendado para estimar proyectos Web en la fase de lanzamiento del framework propuesto (Figura III.6).

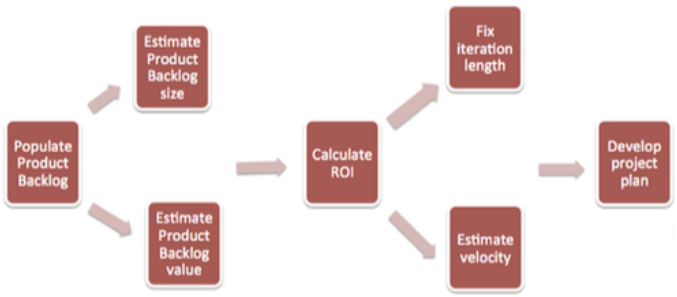

Figura III.6. Actividades de la fase de lanzamiento.

(Referencia: [Torrecilla et al. 2015)

Por otro lado, el ciclo de vida de los proyectos ágiles, en especial "*Scrum*", comienza con la definición de una pila de producto ("*Product Backlog*") delimitado en un contexto, que dará forma al proyecto Web y como indicaba la figura anterior, contiene las siguientes actividades:

- Rellenar "*Product Backlog*".
- Estimar el tamaño del "*Product Backlog*" y su valor.
- Calcular el retorno de la inversión ("*ROI*") y priorizar el "*Product Backlog*".
- Ajustar la longitud de la iteración y la estimación de la velocidad del equipo.
- Desarrollar el Plan inicial del proyecto.

Como tipo de elemento para rellenar la pila de producto, se propone la Historia de Usuario, por su consolidación en el paradigma Ágil.

Para esta gestión ágil de requisitos modelaremos la técnica denominada "Historias de Usuario", término popularizado por Mike Cohn [Cohn 2005], que se define como un conjunto acotado de funcionalidad que puede ser entregado a un proveedor, usuario o sistema de información, de manera que genera un valor que antes no existía, una vez que se ha concluido ese quantum de tiempo. Puede ser definido como una necesidad que genera valor por sí misma.

Los elementos fundamentales del requerimiento ágil [Sedeño et al. 2014b] que presentamos como una estructura de la aproximación al metamodelo del Capítulo V (Figura III.7), que expondremos a continuación, están tomados de este marco ágil y son:

- La Historia de Usuario, conjunto de funcionalidad formada por un identificador unívoco, una descripción que responderá a dicha funcionalidad, una fecha de alta, una valoración de negocio (unidades ficticias de negocio), una valoración de esfuerzo (unidades de esfuerzo ficticias, llamadas puntos de historia) y un elemento de ordenación y priorización, que en nuestra propuesta, será el retorno de la inversión (ROI), calculado como la división entre el valor de negocio y el esfuerzo y que formará el "*core*" del requerimiento ágil. La gran mayoría de técnicas ágiles incluyen estos elementos a excepción del ROI, que será el elemento clave en la elección de las historias de usuario a utilizar, ya que nos proporcionará el marco espacio-temporal para su elección y que se ha demostrado muy eficaz [Torrecilla et al. 2013a] [Torrecilla et al. 2013b]. Las Historias de Usuario han de cumplir además los aspectos INVEST [Beck & Andres 2004]: Independiente, Negociable, Valorable, Estimable, Acotado ("*Small*"), y Testeable.

- La Definición de Hecho, que en las metodologías ágiles se entiende como un aserto que describe cuándo una Historia de Usuario está terminada. Por tanto, una serie de definiciones de hecho constituirán los elementos que se transformarán en las pruebas, pudiendo utilizarse técnicas como "*Test Driven Development*".
- Las *Constraint*, que ayudarán a delimitar la historia de usuario en un contexto concreto y que serán equivalentes a los requerimientos no funcionales.
- Las Personas, entidad análoga a los Actores en otros paradigmas, que tienen una relación con la Historia de Usuario. De estas relaciones entre la entidad Persona y la Historia de Usuario se deducirá el esquema de navegación Web. Se ha denominado Persona y no Actor al haber elegido el nombre de la técnica ágil Personas [Beyer 2010] dedicada al modelado de estas entidades y para no crear confusión con Actores cuya relación está basada en la causa-efecto, como por ejemplo, se da en un caso de uso.

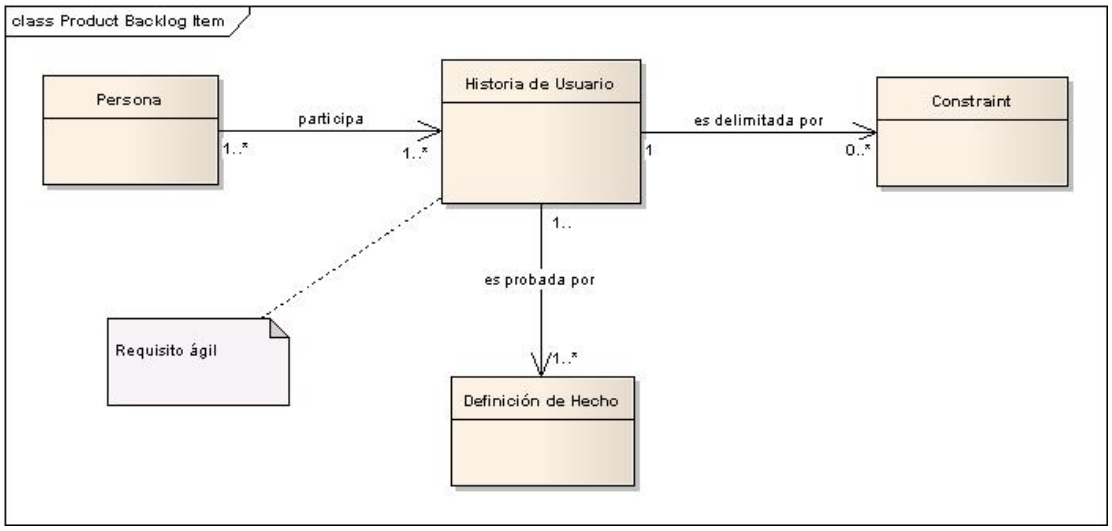

Figura III.7. Estructura de la aproximación propuesta para un requisito ágil.

Esta propuesta inicial de metamodelo conformaría un requisito ágil, formado por estas cuatro clases. Cada uno de estos requisitos ágiles formarán los elementos de la pila de producto. Realmente representa un modelo de requerimiento que incorpora su propia priorización, de manera que podríamos hablar de entrega de valor priorizada o maximizada y por tanto ágil.

En este sentido y en lo que respecta a esta Tesis Doctoral la convergencia entre **entrega de valor y prestación de servicio**, está muy cercana en la mente del usuario, de manera que nos proporcionará una granularidad semejante en la definición de ambos, asimismo, el proceso de descubrimiento de Servicios objeto de esta Tesis Doctoral será invocado dentro de la fase de desarrollo de esta metodología ágil y será su definición parte de este trabajo de investigación.

4. Conclusiones

A lo largo de este capítulo se ha descrito el problema concreto que se desea resolver. Para ello, el capítulo ha comenzado centrando el alcance del problema a abordar una vez analizadas las conclusiones obtenidas tras el estudio del estado del arte desarrollado en el Capítulo II.

Planteado el alcance del problema, el capítulo define específicamente cuáles son los objetivos a alcanzar y las premisas que hay que seguir para alcanzarlos.

A partir de dichos objetivos, se han presentado las influencias que han resultado ser fuente de inspiración para la elaboración de este trabajo.

Con todo esto, se han cerrado los marcos de introducción de la Tesis Doctoral y se han sentado las bases adecuadas para comenzar a presentar, en los siguientes capítulos, nuestra propuesta de solución.

Capítulo IV Metamodelo de Servicios

Como se indica en las influencias sobre la Arquitectura Orienta a Servicios expuesta en el Capítulo III, uno de los componentes fundamentales del Modelo Objetivo SOA es el Catálogo de Servicios, que representa una abstracción formalizada del negocio.

En este apartado se define la formalización de una serie de artefactos que conforman la unidad mínima de representación de un Servicio y por tanto y en última instancia de la funcionalidad del negocio identificada y clasificada dentro de una organización. Es decir, se formalizarán en definitiva requisitos que la organización ya ha incorporado a su contexto con objeto de ser reutilizados y/o usados por otros agentes de dicha organización.

Este metamodelo está definido y representado formalmente por medio de diagramas de clases UML y basado en el estándar de meta-metamodelado MOF («*Meta-Object Facility*») [OMG 2011b] propuesto por la OMG. A su vez en este capítulo se definirá un perfil UML para este metamodelo con objeto de poder ser utilizado en las diferentes herramientas UML y poder por tanto instanciar este metamodelo.

1. Introducción

Como se indicó en el Capítulo I, una vez realizado el estudio del estado del arte en el Capítulo II y haber definido los objetivos en el Capítulo III, presentamos de nuevo el esquema global de este trabajo, descrito en la Figura IV.1, con objeto de encuadrar los siguientes capítulos.

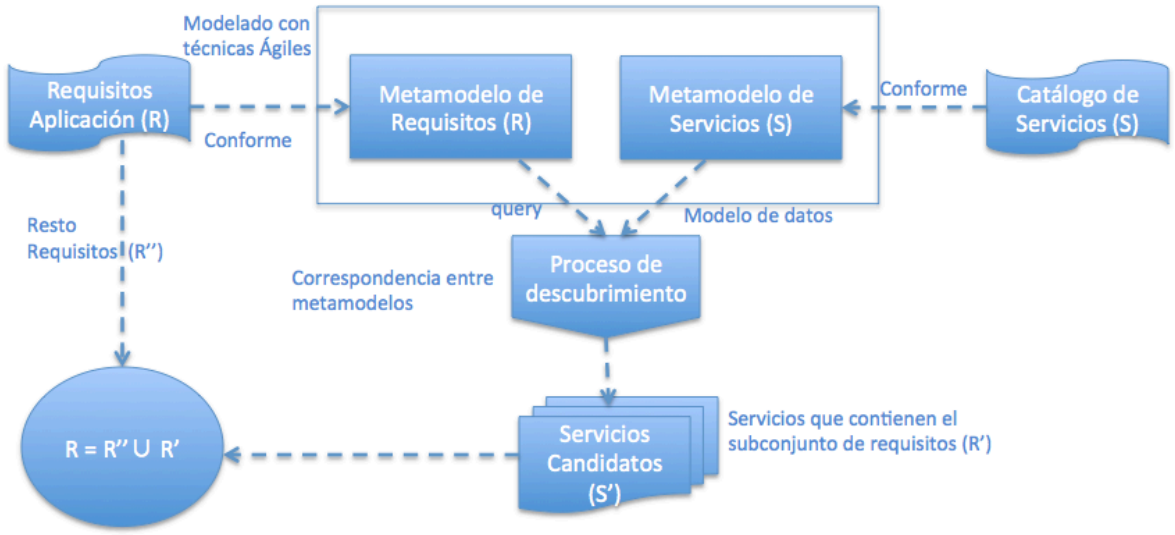

Figura IV.1. Propuesta para la descubrimiento de Servicios en entornos de ágiles.

El objetivo principal de la presente Tesis Doctoral versa en la propuesta de formalización de un requisito Web fundamentándolo en técnicas ágiles (debido a su agilidad y completitud) y que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios, dentro del contexto, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito.

La solución propuesta estará basada en una serie de metamodelos y un proceso que los correlacione, para llevar a cabo, de una manera eficiente y eficaz, el proceso de gestión de los requisitos de aplicaciones contra un Catálogo de Servicios usando técnicas ágiles y posibilitando el gobierno de dichos Servicios dentro de la organización.

En este capítulo nos centraremos en un metamodelo en el que se ha recogido la información básica que ha de tener un Servicio (Metamodelo de Servicios, S) para poder informar de su funcionalidad y ser usado en el contexto de la organización dentro de la fase de Ingeniería de Requisitos.

Dentro de este metamodelo se han asociado dos descripciones de niveles diferentes de un Servicio:

- La parte estrictamente funcional y que pertenece al negocio, en la que se describe el Servicio y sus relaciones desde el punto de vista de negocio.
- Su interfaz o contrato, realizando el modelado lógico mínimo para describir una interfaz de un Servicio genérico con operaciones, mensajes, tipos, errores y localización.

La razón de esta división está fundamentada en dos supuestos principales:

- No todos los Servicios han de estar implementados, es decir, no todo los Servicios han de tener interfaz en el momento en el que se adscriben al Catálogo de Servicios.
- Cuando lo están (implementados o definidos a nivel de implementación) y por tanto formalizada su descripción lógica en el Catálogo de Servicios, el Servicio adquiere una granularidad más fina (un nivel de detalle mayor) debido a la descripción de las operaciones.

Esta representación permitirá varias ventajas en referencia al enfoque que se la ha dado a esta Tesis Doctoral, basada en el descubrimiento de los Servicios Candidatos y por tanto de la funcionalidad que ya está contenida en la organización.

Como se indicaba en el Capítulo III, la metodología de desarrollo elegida dentro del Modelo Objetivo SOA debía integrar el desarrollo de aplicaciones Web con el Ciclo de Vida de los Servicios. De esta manera es posible reutilizar también esta metodología para implementar Servicios identificados por la organización pero que por razones de planificación, presupuestarias o de falta de recursos, no se han podido implementar por la propia organización, de manera que el proyecto concreto, no solo desarrolla esa funcionalidad, sino que conduce el Modelo Objetivo SOA hacia su madurez y completitud.

Por tanto estos Servicios Candidatos no sólo podrán ser utilizados sino que podrán ser además, implementados, pudiendo así hacer un uso más eficaz de los recursos de la organización. Es por tanto que la instanciación de este metamodelo estará viva en el tiempo, así como la funcionalidad en la organización está viva en todo momento.

2. Metamodelo de Servicios

Esta sección describe cómo se ha resuelto en este trabajo de investigación el segundo objetivo enumerado en el Capítulo III, es decir, la definición de un metamodelo de Servicios para una organización, en aras de identificar y formalizar su negocio.

Para este propósito, esta sección se divide en tres apartados. El primero de ellos describe el contexto y planteamiento previo, mientras que en el segundo apartado se define amplia, formal y rigurosamente la solución propuesta. Por último el tercer apartado presentar un ejemplo completo de metamodelado.

2.1. Contexto y planteamiento previo

Dentro del Modelo Objetivo SOA existen muchas entidades, algunas de las cuales se podrán modelar para este metamodelo de Servicios propuesto.

Este metamodelo tiene como base el Servicio, siendo la entidad fundamental y que a través de un lenguaje textual, describirá la funcionalidad que expone la organización. Esta funcionalidad expuesta además, pertenecerá a un Dominio de la organización que podrá ser funcional o de negocio, transversal a la organización ("*cross")* o técnico.

Además cada Servicio estará tipificado con diversas palabras clave relacionadas con su funcionalidad que serán escogidas a través de una "*nube de tags*", esto hace que el Servicio pueda recibir diferentes semánticas de forma simultánea dentro de la taxonomía indicada por el Dominio.

Además, estos Servicios estarán delimitados funcionalmente por Acuerdos de Nivel de Servicio, (SLA, "*Service Level Agreement*"), que aluden a los límites en la prestación de ese Servicio hacia los consumidores o clientes. Sobre cada Servicio pueden aplicarse unas políticas que se han de ejecutar para conducir el Gobierno de los Servicios, cada Servicio podrá tener unas políticas propias de acuerdo a su funcionalidad. No es objeto de esta Tesis Doctoral, como se comentó en capítulos anteriores, la ejecución de las políticas, que se habrían de aplicar a los Servicios Candidatos.

Por otro lado los Servicios, en una organización, están gobernados por personas, por lo que será necesario en el metamodelo representar dichas relaciones. Para ello la forma de relacionar las personas con los Servicios que se escogió en la Consejería de Cultura, que por otra parte es habitual en el resto de organizaciones, fue a través de matrices RACI, que son los acrónimos de los diferentes roles propuestos en esta metodología:

- Responsable ("*Responsible*"), como responsable de la operación del Servicio.
- Dueño ("*Accountable*"), como dueño del Servicio.
- Persona que necesita ser consultada ("*Consulted*") ante los cambios y/o eventos en el Servicio.
- Persona que necesita ser informada ("*Informed*") ante los cambios y/o eventos en el Servicio.

Con este resumen de las principales entidades enunciadas en el Capítulo III, se procederá a realizar la definición formal del metamodelo propuesto para la definición formalizada de un Servicio.

2.2. Definición del metamodelo

Los conceptos definidos en este metamodelo están agrupados en un paquete al que denominaremos «Services Structural». La Figura IV.2 muestra este metamodelo y las relaciones entre sus elementos.

En la siguiente sección se describen en detalle los elementos del metamodelo, siguiendo el estilo y la plantilla de organización de documentación utilizado por el OMG para la documentación de sus estándares y normas (como es el caso del lenguaje UML).

Antes de continuar, es importante aclarar que la sintaxis utilizada para representar el metamodelo de Servicios no es lo suficientemente rica como para reflejar semánticamente las restricciones necesarias. En consecuencia, se han definido restricciones semánticas (tal y como recomienda la OMG, «*Object Modeling Group*») mediante el lenguaje de restricciones OCL [ISO/IEC 2012]. Estas restricciones limitan las posibilidades de instanciación con el propósito de construir modelos de Servicio válidos.

Este estilo se describe a continuación. En primer lugar se ofrece una breve descripción del elemento. A continuación, se indican sus generalizaciones, si existen, y las restricciones asociadas a dichas generalizaciones. A continuación se describen sus asociaciones y sus atributos. Posteriormente, se describen las restricciones a los valores de los atributos y asociaciones si las hubiera.

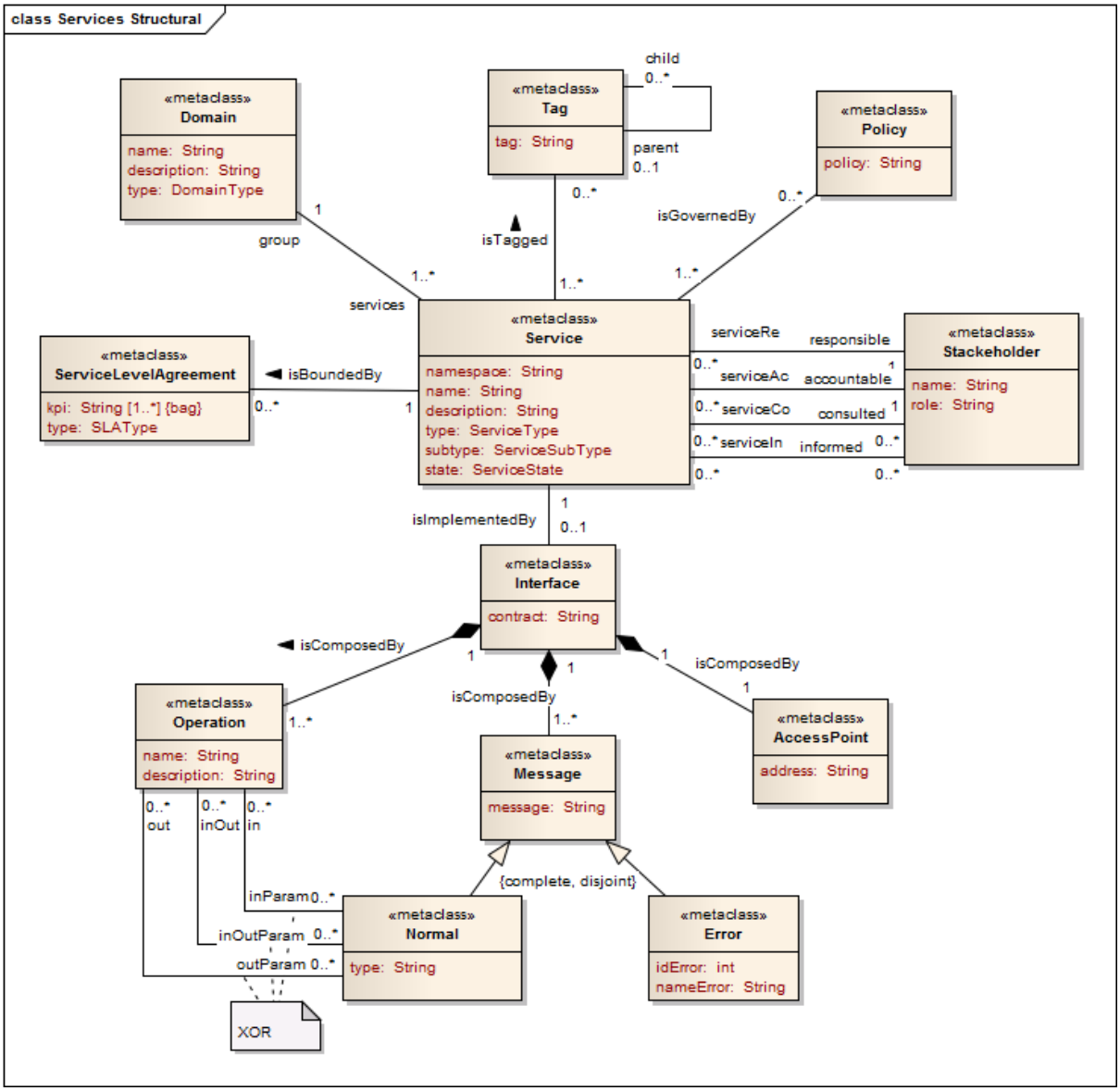

Figura IV.2. Metamodelo de Servicios.

Para la descripción detallada de cada uno de los elementos del metamodelo usaremos una ordenación alfabética, primando dicha ordenación para facilitar su lectura, sobre una posible ordenación semántica o de negocio.

2.2.1 Metaclase «AccessPoint»

Descripción: El último de los elementos que componen la interfaz es la localización o punto de acceso, que expone el punto de entrada al mismo. No se entrará en disquisiciones técnicas a la hora de describir el punto de acceso, habitual en el acceso a los Servicios (protocolos, puertos, etc...). Todo ello quedará contenido en los atributos de esta metaclase.

Generalización: Ninguna.

Atributos:

- address: String [1]

Campo que indica la dirección por la que se accede al Servicio y que como se ha indicado, generalizará las diferentes tecnologías que hay para la creación del punto de acceso a un Servicio.

Asociaciones:

- isCompoundBy: Interface [1] (Asociación de Composición)

Cada Punto de Acceso está asociado a una interfaz concreta y por tanto a un Servicio que se encuentra dentro de un espacio de nombres concreto, por lo que estará unívocamente relacionado con el Servicio.

Restricciones: Ninguna.

2.2.2 Metaclase «Domain»

Descripción: Esta metaclase «*Domain*» representa el modelo de Dominio de Servicios SOA en donde se estructura el árbol taxonómico de los Servicios. Cada Servicio está incardinado en uno de estos dominios.

Cada dominio además estará tipificado para una mejor distribución de los Servicios. Entre las formas más típicas de creación de los dominios se encuentran los modelos creados a partir de los aspectos de negocio de la organización (líneas de negocio, áreas temáticas o competencias administrativas entre otros) y los creados a partir de la estructura orgánica (divisiones, departamentos o estructura orgánica).

Generalización: Ninguna

Atributos:

- name: String [1]

Nombre unívoco que identifica al Dominio. Bajo este nombre se clasificarán todos los Servicios que se adscriban.

- description: String [1]

Este atributo realiza una descripción textual del Dominio para una mejor compresión y delimitación del mismo.

- type: DomainType [1]

Este tipo enumerado indica los tipos de Dominio que pueden darse en la organización y que estás divididos en tres bloques fundamentales, los dominios funcionales o de negocio, los dominios transversales a la organización o también llamados Cross y los dominios tecnológicos. Estos valores serán definidos por tanto como {«BUSSINES», «CROSS» y «TECHNICAL»}. Como ejemplo de cada uno podríamos tener:

- Como dominio de tipo BUSSINES *"Bienes Culturales – Servicios para la Gestión de los Bienes Culturales de la Consejería*".
- Como dominio de tipo CROSS y que afecte de forma multidimensional a la organización podríamos tener "*Publicación – Servicios para la publicación de contenidos en el portal corporativo*".
- Como dominio de tipo TECHNICAL podemos tener "*Firma Electrónica – Servicios para la firma electrónica de documentos*".

Asociaciones:

- services: Service [1..]*

Cada Dominio agrupa un conjunto de al menos un Servicio, todos los Dominios han de tener Servicios asociados.

Restricciones: Ninguna.

2.2.3 Metaclase «Error»

Descripción: Esta metaclase hereda de la metaclase «*Message*», por lo que es una información estructurada para indicar un error concreto, que se puede dar en el Servicio. Los Errores están tipificados en la Interfaz.

Generalización: metaclase «*Message*».

Atributos:

- idError: int [1]

Identificador numérico único asociado a un Error, está asociado a un mensaje que describe el error en concreto.

- nameError: String [1]

Nombre del error asociado unívocamente al identificador «*idError*» y que identifica por tanto a un mensaje concreto con información acerca de un mal funcionamiento del Servicio.

Asociaciones: Ninguna.

Restricciones: Ninguna.

2.2.4 Metaclase «Interface»

Descripción: Siempre que se ha realizado una definición de Servicio en el capítulo I (tablas I.2 y I.4) se ha indicado expresamente que esa funcionalidad se expone a través de un Interfaz o Contrato de forma concreta. La metaclase «*Interface*» viene a modelar ese contrato, como forma de presentar dicha funcionalidad. Como se ha recalcado ya en este trabajo, no todo Servicio identificado en el Catálogo tiene que tener su interfaz. Esto se ha justificado en la introducción de este capítulo atendiendo a la capacidad de aprovechamiento de recursos dentro de un organización en aras de poder desarrollar los Servicios a los largo del tiempo y aprovechando una serie de disparadores asociados al descubrimiento de Servicios Candidatos. Hacer notar que esta interfaz no pretende adscribirse a una tecnología concreta para la implantación de Servicios WEB que podrán ser SOAP[2], REST[3] u otras implementaciones estandarizadas o no. Por

[2] SOAP, (Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web Simple Access Object Protocol.

[3] REST, Representational State Transfer (REST) is a software architecture style for building scalable web services. REST gives a coordinated set of constraints to the design of components in a distributed hypermedia system that can lead to a higher performing and more maintainable architecture.

eso se han elegido los artefactos fundamentales, comunes y mínimos que representan la descripción lógica del Servicio y que se abstraen de cualquier implementación tecnológica.

Generalización: Ninguna.

Atributos:

- contract: String [1]

Descripción lógica, estructurada y formalizada que describe la funcionalidad que un servicio expone al exterior. Este atributo contendrá la descripción formal de todos los atributos de las clases que forman la agrupación. La forma de estructurarlos no será objeto de este trabajo de investigación ya que dicha estructuración está asociada a la tecnología.

Asociaciones:

***-** isImplementedBy: Service [1]*

Cada interfaz realiza la descripción lógica de un Servicio. Como se ha comentado, esta descripción lógica puede darse en el tiempo después de la descripción funcional expresada en las metaclases anteriores de este capítulo.

- isComposedBy: AccessPoint [1] (Asociación de Composición)

Cada interfaz tendrá un punto de acceso único al Servicio, que podrá ser publicado para que pueda accederse al Servicio y se puedan realizar búsquedas. Como se ha indicado antes, ésta es la localización del Servicio.

- isComposedBy: Operaciones [1..] (Asociación de Composición)*

Cada interfaz expone su funcionalidad mediante un conjunto de operaciones, llamadas también funciones.

- isComposedBy: Mensajes [1..] (Asociación de Composición)*

Cada interfaz contiene una serie de mensajes que son usados como la información que requieren las operaciones para variar su comportamiento. El mensaje es de ida y vuelta. Normalmente se representa mediante parámetros de entrada y/o salida.

Restricciones: Ninguna.

2.2.5 Metaclase «Message»

Descripción: Un mensaje es un conjunto de información estructurado que se intercambian entre dos sistemas permitiéndoles interpelar. En los Servicios, los mensajes representarán los tipos de datos que se intercambian en las operaciones del Servicio y o los errores que se generan. No se ha querido entrar en disquisiciones tecnológicas de la tipología y estructura de estos mensajes, ya que son tan variables como tecnologías hay. Lo que sí se ha indicado es que serán planos y estarán formados por texto.

Generalización: Ninguna.

Atributos:

- message: String [1]

Cada uno de los paquetes de información que requieren las operaciones para variar su comportamiento. La estructura de este mensaje dependerá de la tecnología que se escoja y puede ser una representación en XML, JSON, etc… Como hemos indicado, no interesa concretamente mostrar la estructura tecnológica ya que en una misma organización podrán tenerse varias plataformas tecnológicas que coexistan.

Asociaciones:

- isComposedBy: Interface [1] (Asociación de Composición)

Cada Mensaje específico está asociado a una Interfaz concreta y por tanto a un Servicio que se encuentra dentro de un espacio de nombres concreto, por lo que estará unívocamente relacionado con el Servicio.

Restricciones: Ninguna.

2.2.6 Metaclase «Normal»

Descripción: La metaclase «*Normal*» definirá la forma y contenido de todos los mensajes que no son de tipo «*Error*». Se recomienda realizar definiciones que sean estándar y serializables. Estos mensajes serán de forma explícita, los mensajes usados en las operaciones como entrada y/o salida, asemejándose a los tipos de datos o parámetros.

Generalización: metaclase «*Message*».

Atributos:

- type: String [1]

Sera el nombre único con que se denominará a esta estructuración del mensaje que entra a formar parte de la entrada y/o salida de información de las operaciones. Realmente desde el punto de vista funcional, representará de forma lógica a las entidades de negocio que estén involucradas en un Servicio y otros tipos de datos que formen parte de las entradas y salidas de las operaciones.

Asociaciones:

- in: Operaciones [0..]*

Un tipo podrá participar como un mensaje de entrada de una función u operación, de manera que ese flujo de información es recibido.

- out: Operaciones [0..]*

Un tipo podrá participar como un mensaje de salida de una función u operación, de manera que ese flujo de información es el resultado de ejecutar la operación.

- inOut: Operaciones [0..]*

Un tipo podrá participar como un mensaje de entrada y/o salida de una función u operación, de manera que ese flujo de información es recibido y mediante su transformación es el resultado de ejecutar la operación.

Restricciones:

Sobre la metaclase «Normal» resulta pertinente controlar que un mismo tipo de mensaje sólo puede estar vinculado con una operación mediante alguna de las relaciones que se modelan.

Esta restricción está controlada en el metamodelo a través del operador lógico «XOR» de UML entre dichas asociaciones.

2.2.7 Metaclase «Operation»

Descripción: Otro de los elementos que componen la interfaz son las operaciones, que exponen la funcionalidad del Servicio. También se suelen denominar funciones. Es mediante la ejecución de las operaciones que reciben o retornan mensajes (información que requieren las operaciones para variar su comportamiento) como el Servicio expone esa funcionalidad. Normalmente se representan estos mensajes mediante parámetros de entrada y salida. Estas operaciones podrán producir errores o excepciones, que son otra tipología de los mensajes.

Generalización: Ninguna.

Atributos:

- name: String [1]

Nombre de la operación o función en la que el Servicio expone la funcionalidad. Tendrá las características habituales de los nombre de las funciones.

- description: String [1]

Descripción funcional de la parte funcional del Servicio que expone esa función, será indexable y por tanto podrá ser buscada.

Asociaciones:

- isCompoundBy: Interface [1] (Asociación de Composición)

Cada Operación está asociada a una interfaz concreta, por tanto a un Servicio que se encuentra dentro de un espacio de nombres concreto, por lo que estará unívocamente relacionado con el Servicio.

- inParam: Normal [0..]*

Una operación podrá tener un conjunto de tipos como información de entrada. La cardinalidad mínima será 0, ya que puede haber operaciones que no necesiten información de entrada.

- outParam: Normal [0..]*

Una operación podrá tener un conjunto de tipos como información de salida. La cardinalidad mínima será 0, ya que puede haber operaciones que no retornen información de salida.

- inOutParam: Normal [0..]*

Una operación podrá transformar un conjunto de tipos como información de entrada y salida. La cardinalidad mínima será 0, ya que puede haber operaciones que no retornen información de salida, ni necesiten parámetros de entrada.

Restricciones: Ninguna.

2.2.8 Metaclase «Policy»

Descripción: Dentro del Modelo Objetivo SOA, propuesto en el Capítulo III, el Modelo de Gobierno SOA es uno de los componentes que está asociado con Catálogo de Servicios (junto con el Modelo de Dominio, metamodelado en apartados anteriores) y está basado en un conjunto de

Políticas de Gobierno que representan asertos que deben cumplirse. Estas políticas a su vez, están implementadas por procedimientos de Gobierno, que no entran en el ámbito de esta Tesis Doctoral como ya se ha comentado antes, ni en el ámbito del metamodelo, ya que dichos procedimientos no añaden información en orden al descubrimiento de los Servicios.

Las políticas son un conjunto de reglas, restricciones y requerimientos para controlar los comportamientos y acciones en los Servicios de manera que queden alineados con las necesidades del negocio y se obtenga un resultado exitoso en la implantación del enfoque SOA. El conjunto de políticas se clasifican en tres ámbitos:

- Políticas de Ciclo de Vida. Definen los comportamientos a seguir dentro del ciclo de vida de software y del Servicio.
- Políticas en Tiempo de Ejecución. Definen las pautas a examinar en tiempo de ejecución por parte de los Servicios.
- Políticas de Soporte, definen las tareas administrativas a realizar fuera del ciclo de vida y la necesidad de alineamiento con el Gobierno TIC.

Como se observará, al igual que los Acuerdos de Nivel de Servicio, las Políticas tienen algunas características semánticas equivalentes a requisitos no funcionales de un Servicio, por lo que en el metamodelo propuesto en el Capítulo V de requisitos ágiles basados en Valor, esta metaclase tendrá relación con aquellas que represente a las especificaciones no funcionales.

Generalización: Ninguna.

Atributos:

- policy: String [1]

Se describe de forma textual la política que hace referencia a una regla, restricción requerimiento para controlar los comportamientos y acciones en los Servicios.

Asociaciones:

- isGovernedBy: Service [1..]*

Las Políticas son genéricas para los Servicios, por lo que cada Política podrá asociarse y ejecutarse a una seria de Servicios. Cada Política debe al menos asociarse a un Servicio por lo que la cardinalidad mínima deberá ser 1.

Restricciones: Ninguna.

2.2.9 Metaclase «Service»

Descripción: Esta metaclase es el *"core"* de este metamodelo, ya que es el núcleo de la información funcional que está en el contexto de la organización. En esta metaclase estarán descritos de forma textual, los servicios que la organización ya ha identificado y/o desarrollado y que por tanto son conocidos por la organización y están tipificados como Servicio. Es además el elemento principal del metamodelo, que como se verá en el Capítulo V, tendrá una correlación directa con la Historia de Usuario, ya que ambas formulan una funcionalidad sin detenerse en el cómo, sólo se describe el qué. Esta metaclase además, si está implementada, tendrá una metaclase asociada, la metaclase «*Interface*», que representa la descripción lógica de ese Servicio y donde las operaciones, los mensajes y el punto de acceso describen la estructura del Servicio. Así mismo el resto de clases del metamodelo, acotan, especifican o definen el alcance

de este Servicio, su prestación y le confieren las reglas de gobierno así como su relación con las personas, es decir, le dan un contexto concreto dentro de la organización. La descripción del Servicio pretende reunir la mayor parte de la información relevante asociada al mismo. En este apartado se define la información que metodológicamente se recomienda conocer de un Servicio. La información se ha agrupado de modo que forme parte de entidades representativas por sí mismas enfocadas a distintas formas de examinarlo (no es lo mismo un enfoque desde el punto de vista funcional que desde el punto de vista lógico).

Generalización: Ninguna.

Atributos:

- namespace: String [1]

Agrupación taxonómica general del Servicio. Origina espacios de nombres que comienzan con la rama más genérica de la taxonomía y concluye con la más específica. Se suele representar con los nombres de cada uno de los nodos separados por puntos. El espacio de nombre es un camino unívoco para designar la nomenclatura de ese Servicio, de manera que cada elemento de la interfaz del Servicio está siempre referenciado dentro de un espacio de nombre, permitiendo así que no haya ambigüedad cuando se tienen muchos Servicios que pueden tener idéntica denominación. Es especialmente importante cuando se tiene una o más organizaciones o partes de la propia organización con autonomía para la creación de Servicios. Como ejemplo se podría indicar un "*namespace*" formado por el país, el organismo y el domino funcional: "*es.juntadeandalucia.cultura.flamenco*".

- name: String [1]

Nombre que identifica al Servicio, este nombre suele estar asociado con la funcionalidad del Servicio. Hace referencia al nombre del Servicio. Ha de ser único dentro del espacio de nombres al que pertenece. La nomenclatura a seguir para la designación de los servicios debe tener en cuenta los siguientes aspectos:

- El nombre del servicio debe tener la forma "verbo + sustantivo" donde el verbo contiene la operación que se provee con el servicio y el sustantivo indica la entidad de negocio afectada por la operación. Siguiendo el ejemplo que estamos desarrollando dentro del espacio de nombres del flamenco, el nombre del servicio sería "*altaEspectaculo*".
- En caso de existir más de una entidad de negocio relacionada con el servicio, se incluirá como sustantivo la generalización de todas las entidades afectadas, entendiéndose que las entidades de negocio incluidas dentro de un servicio tienen una relación lógica de negocio. En caso de no ser posible la generalización, se incluirá únicamente la más significativa de las entidades, aunque dentro de la metaclase «*Tag*» se indicarán todas las existentes. De esta manera al realizar una búsqueda por una entidad de negocio desde la metaclase «*Tag*» podremos navegar a la metaclase «*Service*», pudiendo obtener todos los servicios que tienen relación con esa entidad de negocio. Por ejemplo, si un servicio opera con las entidades de negocio "*espectáculo*" y "*concierto*", se puede generalizar el sustantivo a "*actividad*", entidad de orden superior que engloba a ambas, quedando "*exposición*" y "*concierto*" como *tags*.
- Si el servicio ofrece más de una operación sobre una entidad de negocio, se debe generalizar el verbo de manera que todas las operaciones queden englobadas dentro de dicho verbo. Al igual que en el punto anterior, las operaciones podrán ser descritas a

través de la metaclase «*Tag*». Por ejemplo, si un servicio ofrece las operaciones *de "alta", "baja" y "modificación"* de la entidad de negocio "*espectáculo*", el verbo puede generalizarse a "*gestionar*" o "*mantener*".

- description: String [1]

Descripción textual y exhaustiva del Servicio, en este campo se describe qué hace el Servicio, especificando la funcionalidad que se devuelve a través de la interfaz que expone, pero sin entrar en particularidades técnicas. Debe ser indexable mediante herramientas de búsqueda.

- type: ServiceType [1]

También será necesario para una mejor identificación del Servicio la clasificación lógica del Servicio, que deberá llevar los siguientes atributos {«*PROCESS*», «*FUNCTIONAL*», «*TECHNOLOGICAL*»}. Estos atributos, como se puede observar en la Tabla IV.1, tienen la siguiente semántica:

- *PROCESS,* hacen referencia a Servicios que realizan un proceso u orquestación. Como ejemplo se podría presentar el servicio "*esFuncionario*".
- *FUNCTIONAL,* hacen referencia a Servicios que realizan la lógica de negocio. Como ejemplo se podría presentar el servicio "*altaEspectaculo*".
- *TECHNOLOGICAL,* hacen referencia a Servicios de índole tecnológica independientes del negocio. Como ejemplo se podría presentar el servicio "*transformarSOAP2REST*".

-subtype: ServiceSubType [1]

Cada uno de los tipos anteriores «*Service.type*» se desgranará en subtipos, creándose el árbol de tipificación de servicios que se puede observar en la Tabla IV.1.

Tabla IV.1. Árbol de Tipificación de Servicios.

TYPE	SUBTYPE	DESCRIPCIÓN
PROCESS	COORDINATION	Se encarga de coordinar una secuencia o flujo formado por varios procesos.
	PROCESS	Se encarga de realizar funciones propias de lógica de procesos (toma de decisiones, sincronización, etc.).
FUNCIONAL	BUSINESS	Encapsula lógica de negocio atomizada.
	PROXY	Proporciona una funcionalidad de negocio remota implementada por aplicaciones distribuidas.
	WRAPPER	Encapsula funcionalidades de negocio proporcionadas por aplicaciones legacy.
TECHNOLOGICAL	CONTROL	Proporciona funciones horizontales de reparto, seguridad, sesión, etc. (Dispatcher, façades…).
	UTILITY	Proporciona funcionalidades ajenas al negocio (cálculos, helpers…).

Por lo tanto los tipos de los que constará este enumerado serán {«*COORDINATION*», «*PROCESS*», «*BUSINESS*», «*PROXY*», «*WRAPPER*», «*CONTROL*», «*UTILITY*»}.

- state: ServiceState [1]

Atributo que indica el estado en que se encuentra el Servicio dentro del catálogo, los estados son acumulativos y describen cuatro situaciones posibles para el Servicio, que están relacionados con el ciclo de vida de los mismos. Estos estados reflejarán si el servicio está identificado por la organización, si está en fase de desarrollo (incluye el análisis, diseño, pruebas y otras fases típicas del desarrollo software), si está en explotación o si está dado de baja, es decir, estuvo activo en la organización. Solamente tendrán interfaz si han sido construidos. Esto ayudará, al descubrir los Servicios Candidatos, a que su uso y/o posterior desarrollo reviertan en el Gobierno de los Servicios. Por lo tanto los valores que podrá tener el estado del servicio serán {«*IDENTIFICATED»,* «*DEVELOPED»,* «*ENABLED»,* «*DEPRECATED»*}. Estos valores tienen la siguiente semántica:

- *IDENTIFICATED,* hace referencia a que el Servicio está identificado pero no está construido todavía.
- *DEVELOPED*, hace referencia a que el Servicios está desarrollado pero no está en producción.
- *ENABLED*, hace referencia a que el Servicio está en producción.
- *DEPRECATED,* hace referencia a que el Servicio están en producción pero que no puede ser utilizado por nuevos consumidores, estos Servicios están obsoletos y se deberán ir retirando paulatinamente.

Asociaciones:

- isBoundedBy: ServiceLevelAgreement [0..]*

Los Servicios estarán modulados por unos Acuerdos de Nivel de Servicio que indican propiedades no funcionales del Servicio. Estas propiedades o restricciones no funcionales son importantes ya que podrán determinar o acotar el descubrimiento de Servicios Candidatos y su inclusión hará más precisa la búsqueda descrita en el Capítulo VI.

- group: Domain [1]

Cada Servicio pertenece a un Dominio, que como se ha indicado anteriormente, está relacionado habitualmente con la estructura organizativa o con la estrategia de negocio de la organización.

- isTagged: Tag [0..]*

Cada servicio puede tener una semántica con listas de palabras clave para las valoraciones realizadas por motores de búsqueda. Suelen ser adjetivos y sustantivos íntimamente relacionados con la funcionalidad del Servicio. También pueden ser nombres de entidades de negocio relacionadas con el mismo.

- isGovernedBy: Policy [0..]*

Cada Servicio puede tener asociadas una serie de políticas que aplican al Servicio y que están definidas en el Modelo de Gobierno SOA. A cada Servicio se le han de aplicar unas políticas, que se ejecutarían en momentos concretos de su ciclo de vida o ante eventos determinados. En lo que respecta a esta Tesis Doctoral, serían de interés aquellas políticas que se pueden ejecutar al identificar un Servicio como Servicio Candidato y que se les habría de aplicar, aunque la ejecución de estas políticas, así como la descripción detallada de las mismas, queda fuera del marco del presente trabajo.

- responsible: Stakeholder [1]

Dentro de la matriz RACI de responsabilidades en la asignación de roles entre Servicios y personas siempre habrá un responsable de la operación del Servicio.

- accountable: Stakeholder [1]

Dentro de la matriz RACI de responsabilidades en la asignación de roles entre Servicios y personas siempre habrá un dueño del Servicio.

- consulted: Stakeholder [0..]*

Dentro de la matriz RACI de responsabilidades en la asignación de roles entre Servicios y personas habrá un conjunto al que sería necesario consultar determinados eventos y/o actuaciones que requieran tomar alguna decisión.

- informed: Stakeholder [0..]*

Dentro la matriz RACI de responsabilidades en la asignación de roles entre Servicios y personas habrá un conjunto al que sería necesario informar ante determinados eventos y/o actuaciones.

- isImplementedBy: Interface [0..1]

Si el Servicio está construido será necesaria una descripción no sólo a nivel funcional sino a nivel lógico, que es lo que proporciona la interfaz. Debido a que hay estados en los que no existirá todavía dicha interfaz para el Servicio, la cardinalidad mínima será 0. Lógicamente un Servicio sólo podrá tener una interfaz. Sobre la cardinalidad de esta asociación se requiere hacer una consideración de tipo funcional cuando la cardinalidad es 0. En las organizaciones, el presupuesto para desarrollar no siempre coincide con la capacidad de análisis del mismo, de manera que es posible tener Servicios descritos textualmente, sin interfaz, a la espera de la financiación necesaria para su implementación, esta implementación se puede hacer "*ad hoc*" cuando la financiación es directa, es decir, su objeto es la materialización de esa funcionalidad analizada o indirecta, al ser escogido dentro de los Servicios Candidatos, por el desarrollo de una aplicación que los necesita y que lleva su propia financiación. Lógicamente al finalizar cualquiera de las dos vías, esta cardinalidad mutará a 1.

Restricciones:

Como se ha comentado y para que haya modelos consistentes funcionalmente, es necesario que exista una restricción que permita que solamente haya una metaclase «*Interface*» en determinados estados del Servicio. Concretamente cuando el Servicio tiene solamente el estado «*IDENTIFICATED*», es decir, no está desarrollado, no tendrá Interfaz.

Expresión IV.1. Restricción OCL para la metaclase «*Service*» (I).

```
context Service inv:
        if (self.state == "IDENTIFICATED") then self.interface->size() == 0
```

Debido al árbol de subtipos que se ha diseñado será necesario controlar que los atributos «*type*» y «*subtype*» de la metaclase «*Service*» son los correctos, para lo que se indica la restricción siguiente:

Expresión IV.2. Restricción OCL para la metaclase «*Service*» (II).

```
context Service inv:
if (self.type = "PROCESS") then  (self.subtype="PROCESS" or self.subtype = "COORDINATION")
else if (self.type = "FUNCTIONAL") then  ( self.subtype="BUSINESS" or
            self.subtype="WRAPPER" or self.subtype="PROXY")
else if (self.type = "TECHNOLOGICAL") then (self.subtype="CONTROL" or self.subtype="UTILITY")
endif endif endif
```

2.2.10 Metaclase «ServiceLevelAgreement»

Descripción: SLA es el acrónimo anglosajón para el Acuerdo a Nivel de Servicio, *"Service Level Agreement".* El acuerdo es una especificación no funcional entre dos partes, el proveedor y el consumidor o cliente de los Servicios, acerca del rendimiento, la disponibilidad del Servicio y otras características como la seguridad o la continuidad. Como ejemplo se podría indicar un acuerdo de nivel de servicio de tipo "*Disponibilidad*" (horaria y de calendario), que indica la franja o franjas horarias y los días en los que el servicio puede ser utilizado. Por ejemplo, "*de 8:00 a 20:00 de lunes a viernes no festivos"*, o *"24hx7días*". Hay que notar que estamos reflejando aquellos SLA que tiene el Servicio en sí, es decir, no estamos hablando de los SLA que se puedan realizar con un proveedor, dado que para un mismo tipo de indicador se pueden tener varios SLA diferentes con cada cliente. Estos SLA entre Servicio y Cliente, están siempre incluidos en el general. Por seguir con el ejemplo que se expone si la disponibilidad del Servicios fuese "*24x5"*, nunca podría tenerse un SLA con un cliente de "*24x7"* pero si un "*8:00 a 20:00 de lunes a viernes no festivos".* En esta metaclase nos referimos siempre a los genéricos y máximos que ofrece el Servicio ontológicamente.

Generalización: Ninguna.

Atributos:

- kpi: String [1..]*

KPI son las siglas anglosajonas de "*Key Performance Indicador"*, indicadores claves de rendimiento, que delimitan las características de la prestación y que por tanto dotan de una propiedad no funcional al Servicio. En este campo se realiza la descripción textual del indicador, que podrá ser indexada. Cada SLA podrá llevar un conjunto de KPI's que conformen ese nivel de Servicio para un tipo de SLA concreto.

- type: SLAType [1]

En este campo se reflejará el tipo de SLA, que es un enumerado que recogerá, en razón de agrupar los indicadores de rendimiento, una dimensión concreta. Estas dimensiones serán {«AVAILABILITY», «CAPACITY», «CONTINUITY», «PERFORMANCE», «TRACEABILITY»} las cuales se describen en la Tabla IV.2

Tabla IV.2. Descripción de los tipos de SLA.

TYPE	DESCRIPCIÓN
AVAILABILITY	Hace referencia a la disponibilidad del Servicio.

CAPACITY	Hace referencia a la dimensión de la capacidad del Servicio.
CONTINUITY	Hace referencia a la continuidad del Servicio.
PERFORMANCE	Hace referencia al rendimiento del Servicio.
TRACEABILITY	Hace referencia a la trazabilidad del Servicio, en orden a recoger todos los eventos y metadatos necesario para la monitorización de un servicio.

Asociaciones:

- isBoundedBy: Service [1]

Cada Acuerdo de Nivel de Servicio hace referencia unívocamente a un solo Servicio.

Restricciones: Ninguna.

2.2.11 Metaclase «Stakeholder»

Descripción: En las definiciones presentadas en el Capítulo I (tabla I.2 y I.4), al hablar de los Servicios y Servicios Gobernados, no sólo se hablada de las políticas, sino de las diferentes relaciones de las personas con los Servicios en el contexto de la organización. Estas relaciones tienen sentido en un gobierno y es necesario que estén identificadas. Sin embargo, no son relaciones asociadas al funcionamiento del Servicio en sí desde un punto de vista funcional, sino al Gobierno del mismo, ante cambios de estados, de interfaz o del ciclo de vida de los Servicios. Como suele ser una técnica habitual en el diseño de matrices de responsabilidad, se ha optado por la más común, la matriz RACI, que es el acrónimo anglosajón para:

- *Responsible* (Responsable): Responsable de la operación del Servicio.
- *Accountable* (Dueño): Nivel máximo de escalamiento en términos del Servicio.
- *Consulted* (Consultado): Quién debe ser consultado antes de tomar cualquier acción sobre el presente Servicio.
- *Informed* (Informado): Quién debe ser informado de cualquier decisión o acción que se vaya a tomar sobre el presente Servicio.

Generalización: Ninguna.

Atributos:

- name: String [1]

Nombre de la persona, generalmente se pueden poner datos de contacto asociados, pero no es relevante para este trabajo de Tesis Doctoral.

- role: String [1]

Cargo o rol de esa persona en la organización, habitualmente serán una serie de cargos los que asuman las funciones para cada Servicio, pero han de estar siempre identificados unívocamente.

Asociaciones:

- serviceRe: Service [0..]*

Una persona puede ser responsable de la operación de una serie de Servicios, siendo la cardinalidad mínima 0.

- serviceAc: Service [0..]*

Una persona puede ser dueño de una serie de Servicios, siendo la cardinalidad mínima 0.

- serviceCo: Service [0..]*

Una persona puede ser consultada ante los eventos de una serie de Servicios, siendo la cardinalidad mínima 0.

- serviceIn: Service [0..]*

Una persona puede ser informada ante los eventos de una serie de Servicios, siendo la cardinalidad mínima 0.

Restricciones: Ninguna.

Sobre la metaclase «*Stakeholder*» es conveniente establecer que una misma persona no puede ser responsable y dueño del mismo Servicio a la vez. Esta restricción deberá ser descrita mediante el lenguaje OCL al no ser posible explicitarla mediante el lenguaje de modelado UML. En el expresión IV.3

Expresión IV.3. Restricción OCL para la metaclase «*Stakeholder».*

```
context Service inv:
if (self.responsible->size() >0) then self.accountable->size() ==0
        else if (self.accountable->size()>0) then   self.responsible->size()==0
        endif
endif
```

2.2.12 Metaclase «Tag»

Descripción: Esta metaclase representa una nube de etiquetas a base de palabras claves (“*keywords”)* con las que etiquetar a los Servicios, esto hace que el Servicio pueda ser clasificado semánticamente de diferentes modos o desde puntos de vista multidimensionales. Esto añadirá riqueza a la hora de descubrir un Servicio ya que se amplía el rango de búsqueda. Esta nube de etiquetas puede estar jerarquizada llegándose así a un tesauro o diccionario en el que se puedan crear colecciones de sinónimos para tipificar el Servicio.

Generalización: Ninguna.

Atributos:

- tag: String [1]

Palabra clave que añade una semántica al Servicio, estas palabras claves se pueden crear en forma de árbol, para construir listas de términos. Lo más habitual es que estamos hablando de etiquetas de profundidad 0, es decir nodos únicamente. El objeto de las listas irá encaminado a enriquecer con sinónimos la semántica del Servicios.

Asociaciones:

- parent: Tag [0..1]

Cada metaclase «*Tag*» podrá tener como máximo un padre. Con esto se evita que haya términos que puedan crear un ciclo.

-child: Tag [0..]*

Esta relación permite mostrar el conjunto de hijos, ya que estas palabras clave, como se ha comentado, pueden tener a su vez términos que dependan de él (hijos). Con la relación expuesta en este metamodelo para un Servicios no se podrá dar un hijo que tenga más de un padre.

2.3. Ejemplo de Servicio dentro del metamodelo

Como último punto de este apartado y a fin de remarcar cada una de las metaclases y atributos se presenta un ejemplo de un servicio conforme al metamodelo expuesto en las páginas anteriores. Para ello nos centraremos en un servicio tecnológico perteneciente al Gobierno Electrónico y denominado «*firmarDocumento*», cuya funcionalidad consiste en firmar un documento con un certificado electrónico.

Debido a que la descripción textual podría resultar muy extensa, se ha preferido representar dicho ejemplo mediante un tabla, la Tabla IV.3, en la que la primera columna representa las metaclases, la segunda columna los atributos y la tercera columna el valor que toma el atributo para el servicio especificado dentro del metamodelado.

Tabla IV.3. Ejemplo de metamodelado de un Servicio concreto.

Metaclase	Atributos	Valores
«Service»	*namespace*	es.juntadenadalucia.cultura.ae.firma
	name	firmarDocumento
	description	Firma de un documento con un certificado electrónico
	type	TECHNOLOGICAL
	subtype	UTILITY
	state	ENABLED
«Domain»	*name*	Gobierno Electrónico
	description	Dominio de Servicios relacionados con el Gobierno Electrónico
	type	TECHNICAL
«Tag»	*tag*	Certificado
«Tag»	*tag*	Documento
«Tag»	*tag*	Firma
«Policy»	*policy*	Este Servicio sólo puede ser usado por ciudadanos, no por personal interno a la Administración Pública.
«ServiceLevelAgreement»	*kpi*	{“Máximo tamaño del documento 10 Mb”, “Máximo de 10 documentos por minuto”}
	SLAType	CAPACITY
«Stackeholder»	*name*	Jefe de Servicio de Informática
	role	Accountable

«Stackeholder»	*name*	Responsable de la Oficina de Administración Electrónica
	role	Responsible
«Stackeholder»	*name*	Jefe de Departamento de Desarrollo
	role	Informed
«Stackeholder»	*name*	Responsable de la Firma electrónica en Hacienda
	role	Consulted
«Interface»	*contract*	<interfaz><direccion>http://www.juntadenadalucia.es/cultura/ae/servicios</direccion><tipos><tipo>Firma</tipo><tipo>Documento</tipo><tipo>Certificado</tipo></tipos><operaciones><operacion>registrarDocumento</operacion><operacion>firmarDocumento</operacion><operacion>registrarDocumento</operacion><operacion>recuperarFirma</operacion></operaciones><errores><error>23</error></errores></interfaz>
«AccesPoint»	*address*	http://www.juntadenadalucia.es/cultura/ae/firmaDocumento
«Operation»	*name*	registrarDocumento
	description	Registra un documento en la plataforma de @firma
«Operation»	*name*	firmarDocumento
	description	Firma un documento con un certificado.
«Operation»	*name*	recuperarFirma
	description	Recupera la firma de un documento de la plataforma.
«Normal»	*type*	Certificado
«Normal»	*type*	Firma
«Normal»	*type*	Documento
«Error»	*idError*	23
	nameError	Certificado caducado o revocado.

3. Sintaxis concreta de los metamodelos

Dentro de las múltiples alternativas para facilitar el uso de lenguajes basados en modelos dentro de entornos de explotación, el OMG («Object Management Group») propone definir sintaxis concretas utilizando una de las siguientes perspectivas: (i) definir un nuevo lenguaje especifico; o (ii) extender UML con el objetivo de especializar sus conceptos para definir otros nuevos, aprovechando así tanto la semántica (restricciones, propiedades y asociaciones) de todos los conceptos de UML como su propia madurez como uno de los estándares más arraigados y con mayor trayectoria dentro del mundo empresarial.

Cada una de las alternativas anteriores tiene ventajas y desventajas. Por ejemplo, al definir un nuevo lenguaje de manera ad-hoc permite una mayor expresividad y correspondencia con los conceptos de un dominio de aplicación particular. Sin embargo, el hecho de no cumplir con un estándar puede llegar a complicar su aplicación en diferentes contextos y como solución en producción dentro de una organización.

Para facilitar su aplicación en las organizaciones y contextos reales, en este trabajo se ha optado por extender el lenguaje UML como mecanismo para definir la sintaxis concreta de los metamodelos presentados en este Capítulo IV y en el Capítulo V a continuación.

Esta elección ha sido tomada porque UML proporciona un mecanismo de extensión usable, expresivo y flexible para adaptar metamodelos definidos de forma teórica dentro de entornos y herramientas empresariales. Esto es lo que se conoce como un perfil UML. Además, UML es un estándar ampliamente utilizado y reconocido en la empresa. UML proporciona flexibilidad, expresividad y un mecanismo de extensión genérico para construir modelos UML dentro de dominios específicos.

El protocolo de extensión de UML está basado en tres mecanismos básicos: (i) estereotipos o «stereotype», con los que es posible definir cada uno de los elementos de un dominio específico que deberán a su vez extender metaclases UML específicas; (ii) valores etiquetados o «tagged value», que permiten añadir propiedades particulares sobre cualquier elemento estereotipado definido dentro del perfil; y (iii) restricciones o «constraint», con las que definir las condiciones semánticas que aplican sobre los estereotipos del perfil y que condicionan la instanciación del metamodelo con el propósito de construir modelos bien definidos.

3.1. Perfil UML para el metamodelo de Servicios

Durante la definición del perfil UML que implementa el metamodelo de Servicios, se han seguido las siguientes directrices:

- definir un estereotipo por cada uno de los elementos del metamodelo de servicios expuesto en este capítulo.
- elegir y justificar qué metaclases de UML2.5 utilizar para extender cada uno de los estereotipos contemplados.

La Tabla IV.4 muestra los estereotipos del perfil UML para el metamodelo de Servicios y cuál es la metaclase UML que se ha utilizado para extender cada uno de ellos. Asimismo, en la tabla se incluye la justificación por la que se ha optado por la metaclase UML en cuestión.

Tabla IV.4. Correspondencia de UML y metamodelo de Servicios.

Metaclase UML	Estereotipo del perfil	Justificación de la elección
«Clasiffier»,	*«Domain», «Service»*	La metaclase «Clasiffier» de UML es un elemento de modelo que describe las características de comportamiento y estructura.
«Constraint»	*«ServiceLevelAgreement», «Policy»*	La metaclase «Constraint» de UML especifica restricciones, condiciones o asertos que hacen referencia a un elemento y permite su descripción en lenguaje natural, por lo que es adecuada para expresar los estereotipos propuestos.
«Port»	*«AccessPoint»*	La metaclase «Port» de UML especifica un punto de interacción a través del cual un elemento puede comunicarse con su entorno, con otros elementos o con partes internas.
«Operation»	*«Operation»*	La metaclase «Operation» de UML especifica

		una característica de comportamiento que puede ser adquirida por una interfaz, tipo de datos o clase. Las operaciones también pueden estar basadas en plantillas y se utilizan como parámetros de plantillas. Una operación puede ser invocada directamente. La operación especifica el nombre, tipo, parámetros y limitaciones para tales invocaciones.
«Parameter»	*«Normal», «Message» «Error»*	La metaclase «Parameter» de UML es un elemento que expone otro elemento como un parámetro de una operación a través de una definición formal.
«Interface»	*«Interface»*	La metaclase *«Interface»* contiene las especificaciones del Servicio como componentes de la Interfaz.
«Actor»	*«Stakeholder»*	La metaclase «Actor» de UML especifica un rol jugado por un usuario o por cualquier otro sistema que interactúa con un sujeto del modelo.
«Association»	*«isGovernedBy», «isTagged», «isComposedBy», «isImplementedBy», «isBoundedBy»*	Se ha optado por la metaclase «Association» de UML con la que extender los estereotipos indicados debido a que esta metaclase especifica una relación semántica que puede ocurrir entre dos instancias y declara que puede haber vínculos entre las instancias de los tipos asociados.

Siguiendo estas pautas, la Figura IV.3 muestra el perfil UML con el que se implementa el metamodelo de Servicios definido y estructurado formalmente en este capítulo.

Como se puede apreciar en la Figura IV.3, se ha definido un estereotipo que se instancia por cada elemento del metamodelo y que además será completado en su definición por una serie de valores etiquetados según las propiedades definidas en el elemento del metamodelo.

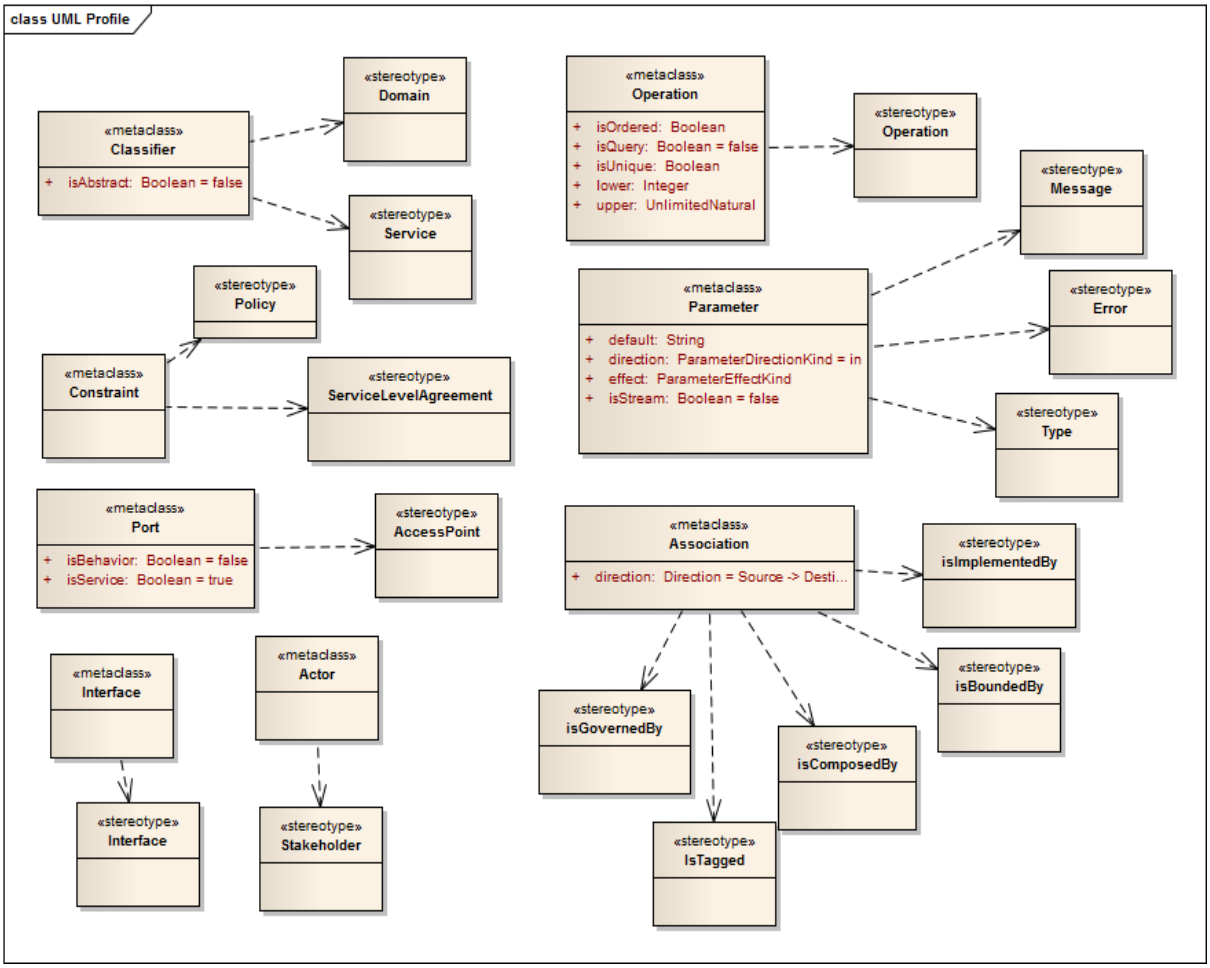

Figura IV.3. Perfil UML del metamodelo de Servicios.

4. Conclusiones

A lo largo de este capítulo se ha presentado de una manera formal el metamodelo para realizar la definición de los Servicios, tal y como quedan reflejados en el Catálogo de Servicios, en aras de la búsqueda sistemática y automatizada de aquellos Servicios formalizados con objeto de descubrir aquellos Servicios Candidatos que contienen la funcionalidad de esos requisitos.

Además, se describe una sintaxis concreta, basada en perfiles UML, para el metamodelo presentado en este capítulo. Un perfil de UML es una extensión formal del propio lenguaje UML con el objetivo de definir nuevos conceptos a partir de constructores ya existentes en UML, con el fin de dotarles de una semántica más precisa y concreta.

Esta definición y representación formal, materializada por medio de la notación de diagramas de clases UML permitirá establecer, ya en el Capítulo VI, una correlación entre este metamodelo y el metamodelo de requisitos ágiles basado en valor que se describirá en el siguiente capítulo.

Capítulo V Metamodelo Ágil de Requisitos

Como se ha comentado en el Capítulo III, la gestión de requisitos basada en técnicas ágiles, tiene la capacidad de recoger, en pocos artefactos y de forma completa, las necesidades para el desarrollo del Sistema de Información Web.

En este apartado se define la formalización de una serie de artefactos de estas técnicas mediante un metamodelo que recoge los principios expuestos en el framework ágil presentado en el Capítulo III, así como la integración de los diferentes elementos resultantes del estudio de la literatura actual en el Capítulo II.

Este metamodelo está definido y representado formalmente por medio de diagramas de clases UML y basado en el estándar de meta-metamodelado MOF («*Meta-Object Facility*») [OMG 2011] propuesto por la OMG.

A su vez en este capítulo se definirá un perfil UML para este metamodelo con objeto de poder ser utilizado en las diferentes herramientas UML y poder por tanto instanciar este metamodelo.

1. Introducción

En este metamodelo se han recogido los elementos principales que forman cada una de las líneas del "*Product Backlog"* expuesto en el Capítulo III, así como la integración de los diferentes elementos resultantes del estudio de la literatura actual en el Capítulo II, es decir, constituye la formalización de una funcionalidad indicada por el usuario, de manera que no estará reflejada la implementación, algo parecido a como hemos venido definiendo el Servicio a lo largo de la presente Tesis Doctoral. El hecho que cada una de esas líneas del "*Product Backlog"* (las Historias de Usuario) tengan que ofrecer un resultado medible y completo y que estén definidas por el usuario, nos proporciona una granularidad perecida a la de los Servicios, ya que éstos, como hemos visto en su definición, entregan una funcionalidad, encapsulando la implementación.

Siguiendo con ejemplos tomados del Gobierno Electrónico, que se desglosará en el Capítulo VIII y en el Anexo I, se puede observar que las definiciones, tanto de un Servicio como de una Historia de Usuario (sólo ya con su descripción textual) tienen un grado de coherencia muy parecido y podrían permitir búsquedas textuales. El ejemplo está referido a una notificación:

- **Historia de Usuario***: Como empleado público quiero notificar fehacientemente a los ciudadanos la concesión del carné de lector de las biblioteca públicas para que puedan usarlo en las bibliotecas.*
- **Servicio***: Servicio de Notificaciones fehacientes a los Ciudadanos.*

Ambos predicados pertenecen a ejemplos reales, como se verá en dichos capítulos, la primera a una aplicación para la obtención de la Tarjeta de Usuario de la Red de Bibliotecas y la segunda a la definición de uno de los servicios genéricos de Gobierno Electrónico.

2. Metamodelo Ágil de Requisitos

Esta sección describe cómo se ha resuelto en este trabajo de investigación el tercer objetivo enumerado en la Sección 2 del Capítulo III, es decir, la definición de un metamodelo ágil para requisitos para el modelado de necesidades funcionales.

Para este propósito, esta sección se divide en tres apartados. El primero de ellos describe el contexto y planteamiento previo, mientras que en el segundo apartado se define amplia, formal y rigurosamente la solución propuesta. Por último el tercer apartado muestra un ejemplo completo de metamodelado.

2.1. Contexto y planteamiento previo

Debido a la gran cantidad de técnicas ágiles que existen, para el presente trabajo se ha tomado como piedra angular la técnica Historias de Usuario ("*User Story*"), una de las más comunes en el agilísimo actual y que fue propuesta por Mike Cohn.

Esta técnica tiene como fundamento la descripción en lenguaje natural de una necesidad bajo la premisa "*como <rol> quiero <algo> para poder <beneficio>"*. Para la obtención de este predicado se utilizan también una serie de técnicas como "*Cards*" o "*Personas"* que ayudan a definir esta historia a través de preguntas.

Esta Historia de Usuario está asociada además a una serie de factores como son los Criterios de Aceptación, predicados en los que se indica cómo se han de probar dicha historia y que delimitan el alcance funcional y las Definiciones de Hecho que determinan cuándo una Historia de Usuario está terminada, de esta manera queda la Historia de Usuario acotada por una serie de asertos que le añaden necesidades no funcionales y que ayudan a acotar la Historia de Usuario.

En relación a la interacción del usuario, en la aplicación se introduce el modelado de las personas que participan en la Historia de Usuario, haciendo hincapié en la relación que tendrán con la misma. De esta manera podremos obtener relaciones como la navegabilidad, la usabilidad o la utilización entre otras, a través de dicha relación.

Por último cada Historia de Usuario es valorada en dos vertientes, el valor de negocio (dado por el cliente) y el esfuerzo, proporcionado por el equipo de proyecto, de manera que podremos ordenar por valor (a través del concepto del ROI, "*Return of Invesment*", que resulta de dividir el valor de negocio por el esfuerzo) de forma que podamos descomponer cada Historia de Usuario en tareas, necesarias para completar la definición de la Historia de Usuario.

El ROI será de vital importancia ya que marcará el orden de los Servicios Candidatos debido a que los Servicios Candidatos relacionados con las Historias de Usuario de más ROI serán los más importantes y los que su aplicación dará más eficiencia y eficacia al proceso.

Con este resumen de las técnicas utilizadas y descritas en el Capítulo III se procederá a realizar la definición formal del metamodelo propuesto para la elicitación ágil y completa de requisitos de un nuevo desarrollo.

2.2. Definición del metamodelo

Los conceptos definidos en este metamodelo están agrupados en un paquete que se denominará «*Value-based Agile Requirements Structu*ral». La Figura V.1 muestra este metamodelo y las relaciones entre sus elementos.

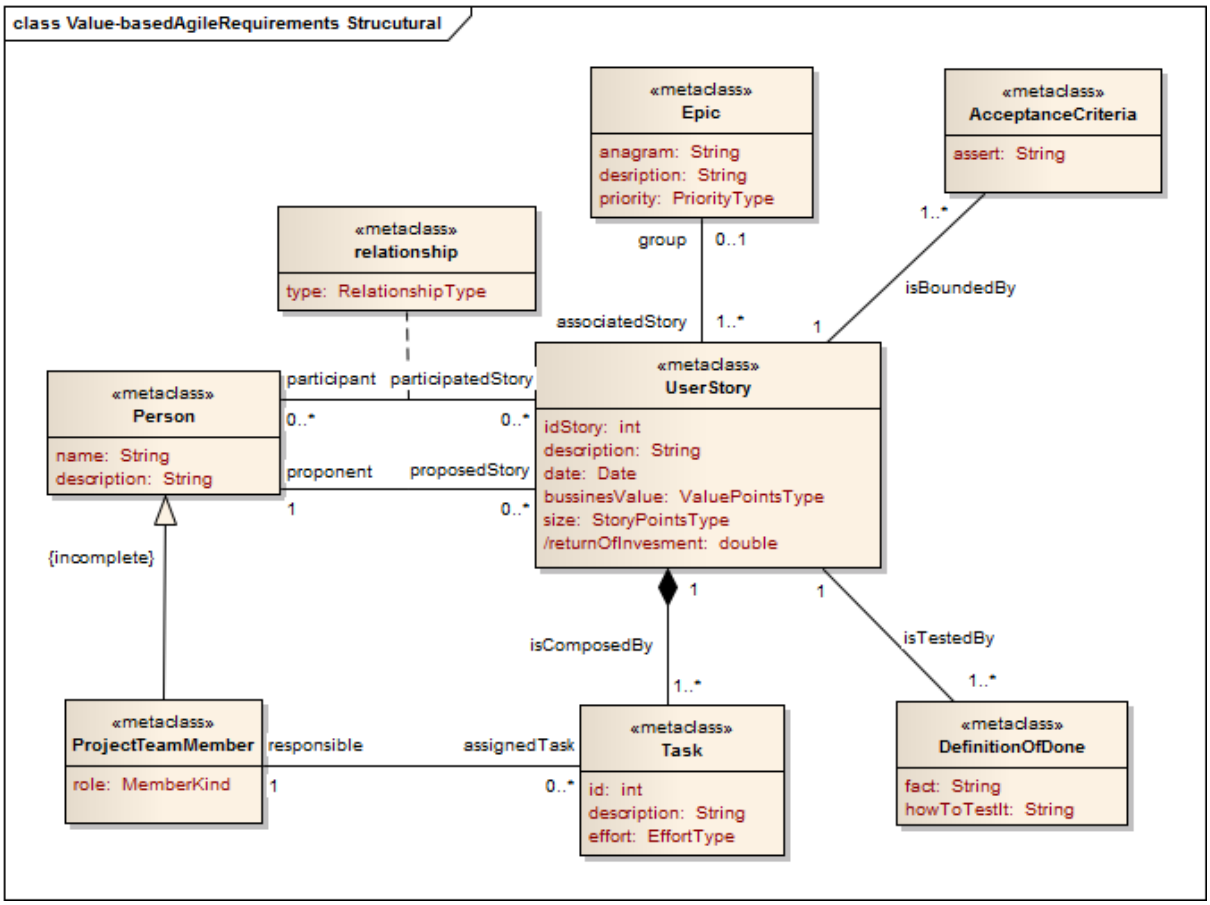

Figura V.1. Metamodelo de requisitos ágiles basado en valor.

En la siguiente sección, se describen en detalle los elementos del metamodelo, siguiendo el estilo y la plantilla de organización de documentación utilizado por el OMG para la documentación de sus estándares y normas (como es el caso del lenguaje UML).

Antes de continuar, es importante aclarar que la sintaxis utilizada para representar el metamodelo de definición de requisitos no es lo suficientemente rica como para reflejar semánticamente las restricciones necesarias. En consecuencia, se han definido restricciones semánticas (tal y como recomienda la OMG, «Object Modeling Group») mediante el lenguaje de restricciones OCL [ISO/IEC 2012]. Estas restricciones limitan las posibilidades de instanciación con el propósito de construir modelos de requisitos ágiles válidos.

Este estilo se describe a continuación. En primer lugar se ofrece una breve descripción del elemento. A continuación, se indican sus generalizaciones, si existen, y las restricciones asociadas a dichas generalizaciones. A continuación se describen sus asociaciones y sus atributos. Posteriormente, se describen las restricciones a los valores de los atributos y asociaciones si las hubiera.

Para la descripción detallada de cada uno de los elementos del metamodelo usaremos una ordenación alfabética, para facilitar su lectura, primando sobre una posible ordenación semántica o de negocio.

2.2.1 Metaclase «AcceptanceCriteria»

Descripción: Las Historias de Usuario deben ir acompañadas por unos criterios de aceptación que implicarán la aceptación de la Historia de Usuario. Estos criterios están ligados al alcance de la misma, delimitándola, o calificándola con algún aspecto no funcional. Serían equivalentes a los Requisitos no Funcionales (RNF) en la Ingeniería de Requisitos y a las pruebas de aceptación. Desde el punto de vista del metamodelo de Servicios tendrán un reflejo con los Acuerdos de Nivel de Servicio (ANS) y otras entidades del modelo de Servicios que reflejen este tipo de restricciones o alcances.

Los criterios de aceptación son las condiciones que la Historias de Usuario debe cumplir para ser aceptado por las Personas o en el caso de la funcionalidad a nivel de sistema, por los miembros del equipo de desarrollo.

Los Criterios de Aceptación son un conjunto de asertos que especifican tanto aspectos funcionales como no funcionales y que junto con las Definiciones de Hecho marcarán cuando una Historia de Usuaria queda aceptada y terminada.

En las técnicas ágiles, una vez terminada la Historia de Usuario, los equipos de trabajo presentan, en la revisión del Sprint, cada Historia de Usuario terminada mediante la demostración de los criterios de aceptación, que contienen por tanto el germen de las pruebas de usuario o de aceptación. De estos Criterios de Aceptación podrían deducirse, mediante diversas técnicas, un conjunto de pruebas automatizadas.

Es importante recalcar la diferencia entre Criterios de Aceptación y Definiciones de Hecho, en cuanto que los primeros hacen referencia a aspectos internos de la propia Historia de Usuario y los segundos hacen referencia a aspectos externos a la Historia de Usuario, referidos a cuándo se considera terminada una Historia. Como ejemplo aclaratorio, en una Historia de Usuario que fuese "*como Usuario quiero logarme en la aplicación para acceder a mi zona privada*" un criterio de aceptación podría ser que "*cuando el usuario introduzca el nombre y la contraseña correcta el sistema le deje acceder*" y la definición de hecho podría ser del tipo "*la historia de usuario estará completa cuando se haya subido el código fuente al sistema de control de versiones*".

Generalizaciones: Ninguna.

Atributos:

-assert: String [1]

Descripción textual que especifica tanto aspectos funcionales como no funcionales que deberá cumplir una Historia de Usuario y hace referencia a cualquier circunstancia de la Historia de Usuario que la acote, la clarifique, la determine o le añada cualquier semántica funcional o no funcional.

Asociaciones:

- isBoundedBy: UserStory [1]

Cada criterio se refiere a una Historia de Usuario, ya que la delimita y por tanto se refiere unívocamente a una Historia de Usuario concreta.

2.2.2 Metaclase «DefinitionOfDone»

Descripción: Las Definiciones de Hecho son una forma de declarar, en lenguaje textual, aspectos externos en la que una vez cumplidos, la Historia de Usuario se considera terminada. En las técnicas ágiles, una vez terminada la Historia de Usuario, los equipos de trabajo presentan, en la Revisión del Sprint, cada Historia de Usuario terminada mediante la demostración de las definiciones de hecho, que aseguran que todas las tareas necesarias para terminar la Historia de Usuario han sido ejecutadas. En el metamodelo de Servicios tendrán un reflejo muy tangencial con los Acuerdos de Nivel de Servicio (ANS).

Generalizaciones: Ninguna

Atributos:

-fact: String [1]

Aserto por el que se considera que una Historia de Usuario estará terminada. No confundir el hecho ("*fact*") que se indica con que la historia está hecha ("*done*").

-howToTestIt: String [1]

Descripción de la prueba que se ha de realizar para demostrar ese hecho, esto es una descripción textual de los pasos a realizar para probar el hecho ("*fact*") por el que se considera hecha ("*done*") la tarea.

Asociaciones:

- isTestedBy: UserStory [1]

Cada prueba se refiere a una Historia de Usuario, ya que prueba su completitud y por tanto se refiere unívocamente a una Historia de Usuario concreta.

2.2.3 Metaclase «Epic»

Descripción: Esta metaclase «*Epic*» modela lo que se conoce como las Historias Épicas o simplemente "épicas", que son el primer nivel de agrupación basado en Valor de la técnica Historias de Usuario. Son equivalentes a los grandes bloques funcionales que tipifican a un conjunto de Historias de Usuario. En referencia al metamodelo de Servicios se pueden equiparar a los Dominios cuando hablamos de Servicios ya que son taxonomías de los mismos. Estas historia épicas serán materializadas más tarde en Historias de Usuario.

Generalización: Ninguna

Atributos:

- anagram: String [1]

Anagrama identificativo de la Historia Épica con el que se podrá identificar unívocamente la épica de que se trate.

- description: String [1]

Descripción de la funcionalidad contenida en la historia épica. Es una descripción amplia sobre el bloque funcional, sin entrar en detalle.

- *priority: priorityType [1]*

Indicará la prioridad de la épica, su importancia para el dueño del producto ("*Product Owner")*. Es un prioridad funcional que atenderá al negocio de la organización, sin tener en cuenta aspectos técnicos, esta prioridad será escogida entre un conjunto finito de elementos del enumerado *«priorityType»* cuyos valores irán desde un valor de máxima prioridad hasta un mínimo de prioridad. Estos valores serán definidos como {«HIGH», «MEDIUM» y «LOW»}.

Operaciones: Ninguna.

Asociaciones:

- *associatedStory: UserStory [1..*]*

Conjunto de Historias de Usuario que son agrupadas funcionalmente bajo una de las épicas. Una épica tendrá al menos una Historia de Usuario.

2.2.4 Metaclase «Person»

Descripción: La metaclase *«Person»* modelará cada uno de los perfiles o personas que interactúan de alguna manera con el sistema. La persona no tiene por qué tener una interacción directa (como por ejemplo la tiene el "*WebUser*" en WebRE o el Actor en UML) con la aplicación o funcionalidad, pero si deberá tener alguna relación con la misma. Esto permite comprender muchas de las restricciones no funcionales del sistema o de la proveniencia de los requisitos dentro de la aplicación. Estará relacionada en consonancia con la matriz RACI del metamodelo de Servicios que expresa una relación entre Roles y Servicios, aunque el tipo de relación entre la metaclase *«Person» y «UserStory»* es mucho más amplia.

Generalización: Ninguna

Atributos:

- *name: String [1]*

Nombre de la Persona, las técnicas ágiles para mejorar la comprensión por parte del equipo de trabajo, ponen un nombre propio, o un cargo real y está basado en la técnica de modelado ágil "*Persona"*.

- *description: String [1]*

Descripción del tipo de Persona que es en relación al proyecto, de esta manera se podrá saber la razón da participación de esa Perona en la aplicación y por qué fue seleccionada.

Operaciones: Ninguna

Asociaciones:

- *proposedStory: UserStory [0..*]*

En esta relación se indica la identificación de la Historia de Usuario propuesta por una Persona, cada Historia de Usuario tiene siempre asociada una Persona. Como se observa por el mecanismo de herencia un *«ProjectTeamMember»* podrá proponer Historias de Usuarios, por ejemplo de carácter técnico. En consonancia con el metamodelo de Servicios expuesto en el

capítulo anterior, también existe un dominio tecnológico que agrupa a Servicios de carácter puramente técnico.

-participatedStory: UserStory [0..] (Metaclase de Asociación)*

La Persona tendrá una relación con la metaclase *«UserStory»* a través de la metaclase de asociación *«relationship»,* de manera que una Persona podrá tener una relación diferente con diversas Historias de Usuario. Podrá tener por ejemplo una relación de búsqueda, de navegación o de consulta, por poner relaciones relativas al comportamiento en WebRE o podrá tener una relación no causal, como la propiedad (puede ser el propietario de la Historia de Usuario, o el interesado o cualquiera de los roles habituales en una matriz de responsabilidad).

2.2.5 Metaclase «ProjectTeamMember»

Descripción: Dentro del subconjunto de las Personas, existe un equipo de Trabajo, que tendrá capacidad para proponer Historias de Usuario o tener relación con las mismas (podrán por ejemplo ser Historias de Usuarios meramente técnicas). No podemos olvidar, como se indicaba en al apartado 2 del Capítulo III, que en la Ingeniería de Requisitos se ha de obtener la funcionalidad de cualquier fuente disponible y el equipo es por tanto una de esas fuentes. El equipo de trabajo además es el responsable de la ejecución de las Tareas.

Generalización: Metaclase «Person».

Atributos:

- role: MemberKind [1]

En la literatura ágil, los equipos son auto organizados y sólo cuentan con tres roles posibles. El "*Product Owner",* o dueño del Producto, que es el que representa a la Dirección o al área usuaria y que es el que proporciona el Valor de Negocio. El "*Scrum Master",* que es el responsable de conducir la técnica ágil durante proyecto y proporcionar, junto con el resto de miembros del equipo el esfuerzo a las Historias de Usuario. El resto de estos miembros del equipo reciben el nombre genérico de "*Team Member".*

Asociaciones:

-assignedTask: Task [0..]*

Las Tareas estás asociadas a cada uno de los miembros que serán los responsables de su ejecución. De cara a la presente Tesis Doctoral esta relación no tendrá un valor significativo para la búsqueda de Servicios Candidatos, pero cierra la relación de identificación de Requisitos y por tanto se ha visto conveniente mostrarla por la completitud del modelo.

Restricciones:

Es necesario indicar en las restricciones que el "*Product Owner*" no puede ser responsable de una tarea ya que aunque es parte del equipo representa al área de negocio y no al área técnica.

Expresión V.1. Restricción OCL para la metaclase ProjectTeamMember

```
context ProjectTeamMember inv:
        if (self.rol = "Product Owner") then self.assignedTask->size() == 0
```

2.2.6 Metaclase «relationship»

Descripción: La metaclase *«relationship»* está concebida como una metaclase de asociación entre las clases *«Person» y «UserStory»* y refleja la riqueza de la relación entre ambas, es decir, de qué manera pueden estar relacionadas esas clases, ya sea en razón del comportamiento, de la estructura o de la propiedad entre otros.

Generalizaciones: Ninguna.

Atributos:

- type: RelationshipType [1]

Atributo de la relación, se ha utilizado un tipo enumerado para la relación. Los valores de este tipo enumerado, serían de comportamiento, de estructura o de propiedad. Los valores escogidos y que se describen en la tabla VI.1 son {«BROWSE», «SEARCH», «TRANSACT», «RESPONSIBLE», «ACCONTABLE», «CONSULTED», «INFORMED»}.

Tabla V.1. Descripción de los valores del tipo de relación.

TYPE	DESCRIPCIÓN
BROWSE	Esta relación comprende una de las actividades de navegación en la que la Persona se relacionarán con la Historia de Usuario, podrá explorar, que es la acción propia de navegar.
SEARCH	Dentro de la relación de navegación la Persona se relacionará con la Historia de Usuario de forma que podrá buscar, lo que permiten definir las consultas sobre el contenido.
TRANSACT	Dentro de la relación de navegación la Persona se relacionará con la Historia de Usuario de forma que podrá realizar transacciones y por tanto su relación irá encaminada a cambiar el contenido.
RESPONSIBLE	La Persona podrá tener una relación de responsabilidad de operaciones con la Historia de Usuario.
ACCOUNTABLE	La Persona podrá tener una relación de propiedad con la Historia de Usuario.
CONSULTED	La Persona podrá tener una relación con la Historia de Usuario en la que necesite ser consultada.
INFORMED	La Persona podrá tener una relación con la Historia de Usuario en la que necesite ser informada.

Asociaciones: Ninguna.

2.2.7 Metaclase «Task»

Descripción: Las tareas representan la descomposición de una Historia de Usuario en actividades a realizar por el *«ProjectTeamMember»,* se escoge el termino actividades o tareas ya que suponen acciones unitarias acotadas en un tiempo determinado y encaminadas a la consecución de un objetivo, la Historia de Usuario. La descripción de estas tareas es textual. En referencia al metamodelo de Servicios, la descomposición en tareas de una Historia de Usuario tiene cierta similitud con la relación existente en entre Servicio y Operaciones del metamodelo propuesto en el Capítulo IV. Así mismo, las tareas, debido a que son partes de la Historia de

Usuario y por tanto piezas funcionales, dentro del metamodelo de Servicios podrán tener una correlación también con el propio Servicio.

Generalizaciones: Ninguna

Atributos:

- id: int [1]

Identificador numérico y univoco asociado a la tarea, no tiene un significado funcional en sí.

- description: String [1]

Descripción de en qué consiste la tarea, la tarea ha de ser concretada en este atributo. Su descripción encierra una parte de la funcionalidad de la Historia de Usuario o una acción para su consecución.

- effort: EffortType [1]

El esfuerzo para valorar una Tarea es de tipo enumerado y representan las horas ideales que se necesitan para realizar dicha tarea. Como en el resto de estimaciones de las técnicas ágiles aportadas en este trabajo, al ser estimadas por métodos no algorítmicos (por comparación) y basados en decisiones de los expertos, se utiliza una escala discreta, que en el caso del framework ágil expuesto en el Capítulo III, son potencias de 2. Por lo tanto tomará los valores {«1», «2», «4», «8», «16»}. Se considera que una tarea de más de 16 horas ha de ser dividida en más tareas. El objetivo es que el equipo de trabajo pueda estimar el avance de la tarea de forma precisa y por tanto se tenga una información de grano fino del avance de la misma.

Asociaciones:

- isComposedBy: UserStory [1]

Cada tarea forma parte de una sola Historia de Usuario, ya que desgrana su funcionalidad y por tanto se refiere unívocamente a una Historia de Usuario concreta.

- reponsible: ProjectTeamMember [1]

Cada tarea tiene un miembro del equipo de proyecto que se hace responsable de la misma. A cada tarea le corresponde únicamente un miembro del equipo de proyecto que ejerza la supervisión de la misma.

2.2.8 Metaclase «UserStory»

Descripción: Esta metaclase conformará el núcleo de este metamodelo, ya que contiene las necesidades de los usuarios formalizadas en razón del valor entregado en un cuanto de tiempo dado y acordado, en el caso de las técnicas ágiles, la duración del Sprint. Este cuanto de tiempo lo que hace es delimitar el alcance de las historias, impidiendo que éstas sean demasiado ambiciosas. En esta metaclase se recogerán todos los atributos necesarios para identificar unívocamente una Historia de Usuario, así como priorizarla a través del valor de negocio y del esfuerzo. Además ofrecerá una granularidad homogénea con los Servicios siendo una de las clases que tendrá una correspondencia directa con la metaclase Servicio, dado que la técnica de modelado ágil utilizada tiene cierta similitud con la forma de definir un servicio, centrándose solamente en el valor ofrecido y no en el medio para llegar a él.

Generalización: Ninguna

Atributos:

- idStory: int [1]

Ofrece un número único para representar la historia de usuario y ayuda a encontrar y hacer referencia a ella de una manera fácil.

- description: String [1]

En él se describe la funcionalidad y/o el valor proporcionado por la historia. Debe incluir una referencia al perfil (metaclase *«Person»*) para el que la historia ofrece un valor. Habitualmente suele corresponder a un esquema fijo del tipo "*como <Persona> quero <algo> para poder <beneficio>*".

- businessValue: ValuePointsType [1]

Almacena el valor de negocio que la historia de usuario posee, dada por el propio usuario a través del "*Product Owner*" o dueño del negocio. Estos valores serán parte de un tipo enumerado que contiene una serie discreta de valores utilizados para la estimación del valor de negocio. En el caso del framework que se describió en el Capítulo III tendrán como valores {«10.000», «5.000», «1.000», «500», «100»}. Estos valores irán desde una mayor aportación de valor de negocio hasta una menor aportación de negocio. Como hemos indicado, este valor es estimado por comparación a través del conocimiento de los expertos (en este caso el del dueño del producto).

- size: StoryPointType [1]

Almacena el tamaño relativo de la historia, usando también técnicas de comparación (como puede ser "*Plannig Poker*") con el resto de historias que se encontrarían definidas en el "*Product Backlog*". Estos valores serán parte de un tipo enumerado que contienen una serie discreta de valores utilizados para la estimación del esfuerzo. En el caso del framework propuesto se utiliza la escala discreta de Fibonacci para estimar por parte del *«ProjectTeamMember»*. Los valores de este tipo serán por tanto {«1», «2», «3», «5», «8», «13»} y representan la determinación de un balanceo entre el esfuerzo y la incertidumbre por parte del equipo de trabajo para cada Historia de Usuario.

- returnOfInvesment: double (derived)

Es un atributo derivado, siendo el resultado obtenido de dividir el valor del negocio (*"bussinesValue"*) entre el esfuerzo (*"size"*) y debido a que son dos escalas discretas estimadas por comparación el ROI mayor indicará aquellas Historias de Usuario que, al tener un retorno de la inversión mayor, se han de acometer antes ya que presentan una mayor ganancia o impacto de valor para un mismo esfuerzo. De esta manera, lo Servicios Candidatos asociados a las Historias de Usuario con mayor ROI, tendrán más valor ya que su utilización redundará en una menor coste y mayor ganancia para el proyecto. Como ejemplo, si solo se pudiesen utilizar un subconjunto de los Servicios Candidatos, utilizar los que dan cobertura a las Historias de Usuario con más ROI daría más valor a los resultados.

Asociaciones:

- isBoundedBy: AceptanceCriteria [1..]*

Las Historias de Usuario son acotadas, perfiladas o aclaradas mediante una serie de criterios que marcarán su aceptación funcional y que versan sobre el alcance de la historia de usuario,

tales como las restricciones, las dependencias, las limitaciones y los casos especiales o ejemplos a tener en cuenta entre otros. Esta es una relación viva que tiene que actualizarse durante el proyecto como resultado de la colaboración entre los usuarios, clientes y equipos de desarrollo. Una Historia de Usuario debe tener siempre un conjunto de criterios que una vez cumplidos indicarán que la Historia de Usuario será aceptada desde el pinto de vista funcional y no funcional.

- idTestedBy: DefinitionOfDone [1..]*

En esta relación se indica cómo probar que la Historia de Usuario está completada o terminada. Se registra una descripción de cualquier prueba que ayuda a afirmar que la historia se ha ejecutado realmente.

- isComposedBy: Task [1..] (Asociación de Composición)*

Indica el conjunto de tareas en las que se desglosará la Historia de Usuario, es decir, las operaciones atómicas necesarias para que una Historia de Usuario se pueda ejecutar.

- proponent: Person [1]

Una Historia de Usuario ha de estar promocionada por una Persona siempre, en relación a tener la fuente de la misma y poder ser identificada ante dudas, aclaraciones u otros aspectos.

- participant: Person [0..] (Metaclase de Asociación)*

En la Historia de Usuario pueden participar Personas de una forma determinada mediante un tipo de relación que recoge el atributo "*relationship.type".* La cardinalidad mínima se ha identificado como 0 ya que podría haber una historia de índole puramente técnico que no tuviese una relación con ninguna Persona.

- group: Epic [0..]*

Una Historia de Usuario puede formar parte de una épica. En esta relación se observa que la cardinalidad mínima es 0 ya que puede haber Historias de Usuario que no estén agrupadas taxonómicamente en las épicas.

2.3. Ejemplo de Requisito ágil basado en Valor dentro del metamodelo

Como último punto de este apartado y a fin de remarcar cada una de las metaclases y atributos, se presenta un ejemplo de un requisito ágil conforme al metamodelo expuesto en las páginas anteriores. Para ello nos centraremos en una historia de usuario que pertenece a la construcción de la Oficina Virtual y que permitirá al ciudadano poder firmar documentos con su certificado electrónico.

Debido a que la descripción textual podría resultar muy extensa, se ha preferido representar dicho ejemplo mediante una tabla, la Tabla V.2, en la que la primera columna representa las metaclases, la segunda columna los atributos y la tercera columna el valor que toma el atributo para el requisito ágil basado en valor especificado dentro del metamodelado.

Tabla V.2. Ejemplo de metamodelado de un requisito ágil concreto.

Metaclase	Atributos	Valores
«UserStory»	*idStory*	1

	description	Como ciudadano quiero firmar un documento con mi certificado electrónico para presentarlo a la administración.
	date	01/12/2016
	bussinesValue	10000
	size	5
	ROI	2000
«Epic»	*anagram*	OFICINA_VIRTUAL
	description	Creación de la Oficina Virtual para que los ciudadanos puedan relacionarse de forma telemática con la Administración Pública.
	priority	HIGH
«Person»	*name*	Ciudadano Kane
	descrption	Ciudadano que quiere relacionarse con la administración por medios telemático para presentar solicitudes.
«relationship»	*type*	TRANSACT
«AceptanceCriteria»	*assert*	La firma realizada por el ciudadano debe ser validada en la plataforma de validación corporativa de firma.
«DefinitionOfDone»	*fact*	El código fuente ha de estar subido al sistema de gestión de versiones.
	howToTestIt	El equipo de desarrollo ha de enseñar las diferencias de código entres las versiones subidas al sistema de gestión de versiones.
«Task»	*id*	1
	description	El ciudadano ha de adjuntar el fichero a firmar en la Oficina Virtual.
	effort	4
«Task»	*id*	2
	description	El ciudadano ha de poder firmar el documento adjuntado con el certificado electrónico que tiene en el almacén de claves del ordenador.
	effort	16
«Task»	*id*	3
	description	El ciudadano puede obtener un resguardo de su firma y descargarse el fichero de firma electrónica.
	effort	8
«ProjectTeamMember»	*name*	Jefe
	description	Responsable de llevar a cabo la gestión del proyecto.
	role	Scrum Master
« ProjectTeamMember »	*name*	Indio
	description	Encargado de ejecutar las tareas del proyecto
	role	Team Member

3. Perfil UML para el metamodelo ágil de requisitos basado en Valor

La definición del perfil UML para dar soporte al metamodelo ágil de requisitos basado en Valor ha sido realizada siguiendo las mismas directrices que las tomadas durante la definición del perfil UML descrito en el capítulo anterior.

Siguiendo estas pautas la Tabla V.3 muestra de manera resumida los estereotipos del perfil UML de definición de requisitos ágiles basados en Valor y cuál es la metaclase UML que se ha utilizado para extender cada uno de ellos. Asimismo, en la tabla se incluye la justificación por la que se ha optado por la metaclase UML en cuestión.

Tabla V.3. Correspondencia de UML y metamodelo de Requisitos Ágiles basado en Valor.

Metaclase UML	Estereotipo del perfil	Justificación de la elección
«Clasiffier»	*«UserStory», «Task», «Epic»*	La metaclase «Clasiffier» de UML es un elemento de modelado que describe las características de comportamiento y estructura.
«Constraint»	*«DefinitionOfDone», «AcceptanceCriteria»*	La metaclase «Constraint» de UML especifica restricciones, condiciones o asertos que hacen referencia a un elemento y permite su descripción en lenguaje natural, por lo que es adecuada para expresar los estereotipos propuestos.
«Actor»	*«ProjectTeamMember», «Person»*	La metaclase «Actor» de UML especifica un rol jugado por un usuario o por cualquier otro sistema que interactúa con un sujeto del modelo.
«Dependency»	*«relationship»*	Para extender el estereotipo indicado se ha optado por la metaclase «Dependency» de UML. Esta metaclase establece una relación semántica entre los objetos enlazados de tal forma que la semántica del objeto de origen no está completa sin el objeto destino.
«Association»	*«isBoundedBy», «isTestedBy», «isComposedBy»*	Se ha optado por la metaclase «Association» de UML con la que extender los estereotipos indicados debido a que esta metaclase especifica una relación semántica que puede ocurrir entre dos instancias y declara que puede haber vínculos entre las instancias de los tipos asociados.

La Figura V.2 muestra la sintaxis concreta (mediante un perfil UML) con el que se define el metamodelo ágil de requisitos basado en Valor, definido formalmente en este capítulo.

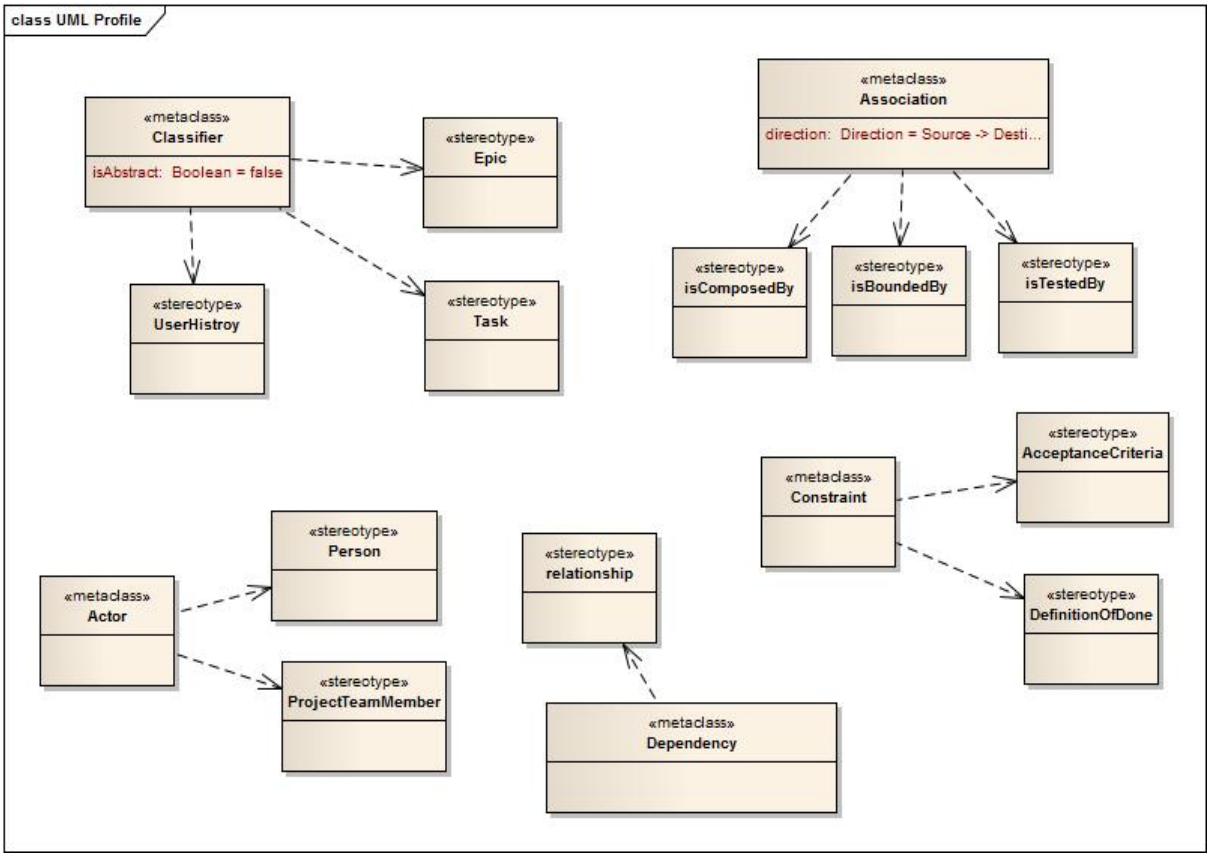

Figura V.2. Perfil UML del metamodelo de requisitos ágiles basado en Valor.

4. Conclusiones

A lo largo de este capítulo se ha presentado de una manera formal el metamodelo para realizar la definición ágil de los requisitos basada en Valor, cuyo objetivo era la formalización, dentro de la Ingeniería de Requisitos, a partir de una serie de técnicas ágiles, en aras del posterior tratamiento para la búsqueda sistemática y automatizada de aquellos Servicios formalizados y estructurados en el metamodelo del Capítulo IV con objeto de descubrir aquellos Servicios Candidatos que contienen la funcionalidad de estos requisitos.

Además, se describe una sintaxis concreta, basada en perfiles UML, para el metamodelo presentado en este capítulo.

Esta definición y representación formal, materializada por medio de la notación de diagramas de clases UML permitirá establecer, en el siguiente capítulo, una correlación entre los metamodelos, de manera que a través de una serie de técnicas algorítmicas basadas en búsquedas textuales podemos descubrir Servicios a partir de Historias de Usuario.

A lo largo de este capítulo se han ido esbozando posibles relaciones entre los elementos de ambos metamodelos que se justificarán de forma detallada en el capítulo siguiente, cuyo objetivo final consistirá en el descubrimiento de los Servicios Candidatos.

Capítulo VI Descubrimiento de Servicios Candidatos

En este capítulo se detallará el proceso para el descubrimiento de los Servicios Candidatos, comenzando en primera instancia por la descripción de la relación de los metamodelos y su correlación, así como la importancia de cada uno de los atributos dentro de esa relación. Así mismo en este capítulo se describirán las consultas a realizar y el concepto de puntuación, necesario para el diseño del algoritmo de búsqueda de semejanzas. Por último, se describirá la secuencia de pasos necesaria para el descubrimiento de Servicios Candidatos a partir de un requisito ágil basado en valor, de manera que a partir de dicha funcionalidad, de forma sistemática, se pueda llegar a descubrir qué Servicios dentro del contexto de la organización son susceptibles de contenerla. El capítulo termina con unas conclusiones relativas al proceso de descubrimiento de Servicios Candidatos.

1. Introducción

En esta sección se tratará la relación entre metamodelos a dos niveles diferentes. En un primer nivel se describirá la correlación existente entre las diferentes entidades de cada modelo (metaclases), de manera que se pueda realizar una asociación conceptual entre cada una de las entidades de los metamodelos propuestos y por tanto una relación entidad a entidad. En un segundo nivel, una vez establecida la relación entre las entidades, se establecerán las relaciones entre los diferentes atributos de dichas entidades. En la Figura VI.1 se puede observar de forma gráfica estas relaciones en los dos niveles.

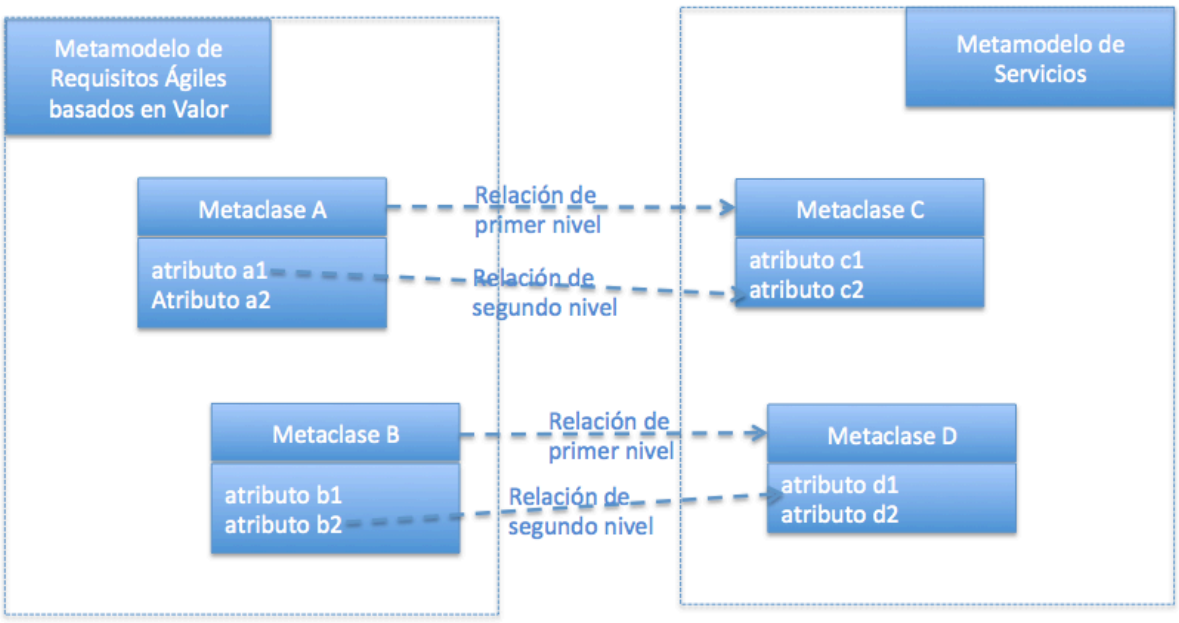

Figura VI.1. Niveles de relación entre metamodelos.

Para ello partiremos siempre como fuente del metamodelo ágil de requisitos basado en valor, debido a que será el que estructurará la consulta ("*Query*") que posteriormente se confrontará

con el Catálogo de Servicios que hará las veces de base de datos, como se describió de manera somera en el Capítulo I y que se detallará en los siguientes apartados del presente capítulo.

Es importante constatar que estas relaciones son de tipo descriptivo y están obtenidas y refinadas a través del método "*Wideband-Delphi*" que potencia la experiencia y opinión de los expertos y es por tanto empírica. En el Capítulo VIII se describirá de forma detallada la implantación del proceso, ya ajustada, en un entono real, la Consejería de Cultura de la Junta de Andalucía. Esto ha permitido mejorar la precisión de las relaciones, de manera empírica, a través de la obtención de los Servicios Candidatos con una serie de funcionalidades dadas y posteriormente compararlos con los resultados reales obtenidos para realizar el refinamiento del modelo.

La relación entre las entidades y los atributos de cada metamodelo es una relación por tanto semántica y estará condicionada por la descripción, en lenguaje natural, que cada equipo de desarrollo realice. Es por ello que las metodologías ágiles han resultado de gran ayuda, al normalizar a través de las técnicas comentadas en el Capítulo III, la forma de describir las funcionalidades usando el lenguaje natural. Al estar trabajando con el lenguaje natural será necesario que la búsqueda esté relacionada con términos, de manera que desde el punto de vista arquitectónico nos encontraremos con un sistema de recuperación y búsqueda de información cuyo modelo sea capaz de operar con una relación entre términos.

2. Relación entre metamodelos

Cada uno de los metamodelos propuestos en los capítulos IV y V cuentan con una serie de entidades o metaclases que han sido detalladamente descritas en los mismos, de manera que en esta sección nos centraremos en describir las relaciones semánticas entre ellas, atendiendo a su ontología, es decir a su naturaleza, y a la interpretación que tienen en cada metamodelo. Para ello se acudirán a relaciones de semejanza desde el punto de vista del negocio (funcional) entre las mismas. Gran parte de esta semejanza vendrá dada, como hemos dicho, por su naturaleza dentro de cada uno de los paradigmas en los que se encuadran.

A su vez y debido a que no todas las entidades tiene el mismo peso de negocio en el metamodelo, dividiremos las entidades entre entidades principales, es decir, cuya semejanza juega un papel predominante en el proceso de descubrimiento (si una Historia de Usuario es semejante a un Servicio, éste tendrá más posibilidad de llegar a ser un Servicio Candidato que si hay una correlación entre una etiqueta y un dominio, por ejemplo) y entidades secundarias o adyacentes cuya semejanza por sí sola no justifica la relación directa con un Servicio Candidato, pero que ayuda a su identificación. Para una mejor comprensión definiremos estos conceptos en las Tablas VI.1 y VI.2 respectivamente.

Tabla VI.1. Definición de entidad principal.

« Una entidad principal es aquella cuya semejanza funcional (o de negocio) dentro de la relación entre los metamodelos justifica por sí sola que la relación entre servicio y requisito tenga que ser considerada y por tanto pueda convertirse, dentro del proceso de descubrimiento, en una relación que cristalice en un Servicio Candidato. »

Tabla VI.2. Definición de entidad adyacente.

« Una entidad adyacente es aquella cuya semejanza funcional (o de negocio) dentro de la relación entre los metamodelos no justifica por sí sola la relación entre servicio y requisito, pero que acompañando a una relación entre entidades principales potencia la probabilidad de establecer esa relación como Servicio Candidato dentro del proceso de descubrimiento. »

2.1. Relación entre entidades del metamodelo

Para la descripción de esta relación basada en la semántica, se utilizará una tabla que indicará por un lado las entidades y junto a ellas, una justificación basada en la experiencia y en la naturaleza semántica de las mismas. Cada una de estas relaciones nos dará, ya en el apartado siguiente de este capítulo, la estructura de la consulta ("*Query*") necesaria para realizar el descubrimiento de los Servicios Candidatos.

En la Tabla VI.3 se describirá la correspondencia entre las entidades principales, entendiendo por principales aquellas en las que la semejanza entre servicio y requisito sea considerada por sí misma, dentro del proceso de descubrimiento, necesaria para que el servicio pueda ser considerado un Servicio Candidato.

Tabla VI.3. Correspondencia entre entidades principales en cada metamodelo.

Requisito Ágil basado en Valor	Servicio	Justificación de la relación
«UserStory»	*«Service», «Operation»*	La entidad o metaclase «UserStory» contiene, como se indicó en los capítulos III y V, el valor a entregar al cliente, es decir, la funcionalidad que produce valor, independientemente de la implementación que sea necesaria para llevarla a cabo. Semánticamente coincide por naturaleza con la definición de Servicio, por lo que una semejanza alta contendrá un probabilidad alta de considerarlo un Servicio Candidato. Así mismo y debido a que la granularidad de ambos pudiese ser diferente, una Historia de Usuario podría equivaler a una Operación dentro del metamodelo de los Servicios, o viceversa, por lo que una semejanza a este nivel también aumentará la probabilidad de que el Servicio obtenido sea un Servicio Candidato.
«Task»	*«Operation»*	La metaclase «Task» es la entidad que contiene los pasos o tareas necesarias para la realización de una Historia de Usuario, pero por si mismas no representan valor una vez. Por tanto su correlación semántica se mantendrá al nivel de las operación dentro de un Servicio. La semejanza entre Operación y Tarea será suficiente para el uso de parte de un Servicio y por tanto del Servicio en sí, siendo entonces mayor la probabilidad de que el servicio sea candidato.

En la Tabla VI.4 se describirá la correspondencia entre estas entidades adyacentes. Estas entidades son denominadas adyacentes ya que complementan al resto de entidades principales, siendo aquellas cuya semejanza por sí misma no constituye una posibilidad de obtener un Servicio Candidato, aunque su presencia aumenta la posibilidad de que pueda darse un Servicio Candidato siempre que haya una semejanza coetánea a una entidad principal.

Tabla VI.4. Correspondencia entre entidades adyacentes en cada metamodelo.

Requisito Ágil basado en Valor	Servicio	Justificación de la elección
«Epic»	*«Domain», «Tag»*	Las entidad Épica hace referencia a una taxonomía superior que engloba a diferentes Historias de Usuario, de la misma manera que el Dominio en SOA engloba a un conjunto de Servicios, además dado que la Épica es la descripción en lenguaje natural de la funcionalidad global, sus términos podrían estar contenidos en la nube de etiquetas ("*tags*") que incorpora el Servicio y que lo cualifica.
«Person»	*«Stakeholder»*	Todo los Actores están correlacionados ya que la semejanza de Actores iguales podrá ser un indicio de que haya probabilidad en que el Servicio sea Candidato, aunque como se ha indicado anteriormente, sin la semejanza en una entidad principal, no será indicativo por sí solo.
«AcceptanceCriteria», «DefinitionOfDone»	*«Policy», «ServiceLevelAgreement»*	Las entidades o metaclases que delimitan cada una de las entidades principales (Historia de Usuario y Servicio) podrán coadyuvar, en caso de semejanza, cuando la haya en sus entidades principales, en un aumento de probabilidad a la hora del descubrimiento de un Servicio Candidato. Tanto las Políticas como los Acuerdos de Nivel de Servicio tiene una relación con los diferentes Criterios de Aceptación a modo de asertos en las Historias de Usuario y con las Definiciones de Hecho, de manera que su semejanza potencia la probabilidad.

2.2. Relación entre atributos de las entidades del metamodelo

Dentro de cada una de las relaciones que se han descrito anteriormente existe a su vez una relación entre los atributos de cada entidad o metaclase. Estos son los campos que se utilizarán como términos dentro del sistema de búsqueda. De cada relación entre entidades antes mencionada, se indicarán qué atributos serán utilizados en la búsqueda de la semejanza de cara a obtener los Servicios Candidatos. Estas semejanzas serán textuales, ya que la descripción de dichos campos está basada en el lenguaje natural. Al igual que se ha representado en el anterior apartado, se mostrarán de cada relación univoca entre las entidades, aquellos atributos que van a ser usados en la búsqueda. En la Tabla VI.5 se describe la relación de atributos entre cada una de las entidades principales. La notación escogida para identificar el atributo seguirá el patrón siguiente ("*Objeto.atributo"),* usando el punto como operador de navegación entre objeto y atributo.

Tabla VI.5. Correspondencia entre atributos de entidades principales.

Requisito Ágil basado en Valor	Servicio	Relación de Atributos
«UserStory»	*«Service»*	• UserStory.name -> Service.name • UserStory.description -> Service.description
«UserStory»	*«Operation»*	• UserStory.name -> Operation.name • UserStory.description -> Operation.description
«Task»	*«Operation»*	• Task.description -> Operation.description

Del mismo modo que con las principales, para la entidades adyacentes, la Tabla VI.6 muestra la relación entre los atributos de cada una de las entidades, desplegando, al igual que el caso anterior, las combinaciones posibles de correspondencia.

Tabla VI.6. Correspondencia entre atributos de entidades adyacentes.

Requisito Ágil basado en Valor	Servicio	Relación de Atributos
«Epic»	*«Domian»*	• Epic.description -> Domain.description
«Epic»	*«Tag»*	• Epic.description -> Tag.tag
«Person»	*«Stakeholder»*	• Person.name -> Stakeholder.name • Person.description -> Stakeholder.role
«AcceptanceCriteria»	*«Policy»*	• AcceptanceCriteria.assert -> Policy.policy
«AcceptanceCriteria»	*«ServiceLevelAgreement»*	• AcceptanceCriteria.assert -> ServiceLevelAgreement.kpi
«DefinitionOfDone»	*«Policy»*	• DefinitionOfDone.fact -> Policy.policy
«DefinitionOfDone»	*«ServiceLevelAgreement»*	• DefinitionOfDone.fact -> ServiceLevelAgreement.kpi

2.3. El concepto de semejanza y el concepto de puntuación en motores de búsqueda

Debido a que nos basamos en el lenguaje natural y que en el capítulo siguiente, en el que se expondrá la arquitectura tecnológica que soporta el proceso, se describe un sistema de

búsqueda y recuperación de información (se ha optado en esa propuesta por Lucene[4] uno de los motores de búsqueda textuales más habituales).

Aunque una de las características más importante de un sistemas de búsqueda y recuperación de información está precisamente en la rapidez del algoritmo de puntuación y en que oculta casi toda la complejidad al usuario, sin embargo y debido a su importancia para el proceso de descubrimiento de Servicios Candidatos se describirá como se realizan estás búsquedas.

Se han mencionado anteriormente los conceptos de puntuación ("score") y de semejanza ("*similarity*"), que están profundamente relacionados con los motores de búsqueda. Aunque para la definición del proceso de descubrimiento de servicios estos conceptos actuarán como una caja negra, ya que la puntuación será adjudicada a cada resultado de una consulta de términos, se estima conveniente exponer de manera somera los conceptos involucrados en esta puntuación ya que son la base para encontrar las relaciones de semejanza en lenguaje natural.

En el campo de la estadística y cuando se trata de campos relacionados, una medida de similitud o función de similitud es una función de valor real que cuantifica la similitud entre dos objetos. Aunque no existe una sola definición de una medida de similitud, por lo general estas medidas son en cierto sentido la inversa de las métricas de distancia: toman grandes valores para objetos similares y pequeños valores o un valor negativo para objetos muy diferentes.

Dentro de los motores de búsqueda y recuperación de información, se conoce como Modelo de Espacio Vectorial a un modelo algebraico utilizado para filtrado, recuperación, indexado y cálculo de relevancia de información. Este modelo representa documentos en lenguaje natural de una manera formal mediante el uso de vectores en un espacio lineal multidimensional.

Muchas de las tareas de recuperación de información como la búsqueda, agrupamiento o categorización de textos tienen como primer objetivo procesar documentos en lenguaje natural. El problema que surge es que los algoritmos que pretenden resolver estas tareas necesitan representaciones internas explícitas de los documentos. En el área de recuperación de información normalmente se usa una expresión vectorial, donde las dimensiones del vector representan términos, frases o conceptos que aparecen en el documento. En este aspecto la representación más adoptada es la conocida como bolsa de palabras: una colección de documentos compuesta por n documentos indexados y m términos representados por una matriz documento-término de n x m. Donde los n vectores representan los n documentos y el valor asignado a cada componente refleja la importancia o frecuencia ponderada que produce el término, frase o concepto en la representación semántica del documento.

La puntuación por tanto expresa el grado de semejanza entre un documento y la consulta correspondiente. Como veremos más adelante, para nuestro sistema de descubrimiento de servicios un documento será una instancia del metamodelo de servicios, es decir, un servicio.

Lucene, el motor de búsqueda y recuperación utilizado en esta Tesis Doctoral, combina el "*Boolean model of Information Retrieval*" [Lashkari et al. 2009] con el "*Vector Space Model of Information Retrieval*" [Salton et al. 1975] implementando para el "*score*" (grado de semejanza) o puntuación, el algoritmo TF-IDF (del inglés "*Term frequency – Inverse document frequency*"),

[4] Apache Lucene es una API de código abierto para recuperación de información, originalmente implementada en Java por Doug Cutting. Está apoyado por el Apache Software Foundation y se distribuye bajo la Apache Software License. Lucene tiene versiones para otros lenguajes incluyendo Delphi, Perl, C#, C++, Python, Ruby y PHP.

frecuencia de término – frecuencia inversa de documento (o sea, la frecuencia de ocurrencia del término en la colección de documentos), que es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección. Esta medida se utiliza a menudo como un factor de ponderación en la recuperación de información y la minería de texto. El valor tf-idf aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras. Una de las funciones de ranking más sencillas se calcula como la suma de los valores tf-idf de cada término de la consulta. Muchas funciones de ranking más complejas constituyen variaciones de este simple modelo. La función que se muestra en la Expresión VI.1 donde "t" es el término, "q" es la consulta y "d" es el documento, representa la variante de la función de "score" para Apache Lucene.

Expresión VI.1. Función para score en Lucene [Apache Lucene 2012].

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \text{ in } q} (tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d))$$

Cada termino de esta función representa una variable, que se define de la siguiente manera:

- COORD, número de términos en la consulta que se encontraron en el documento.
  - *coord(q,d)*
- QUERYNORM o factor de normalización para que las consultas puedan compararse.
  - *queryNorm(q) = queryNorm(sumOfSquaredWeights) = 1/ sumOfSquaredWeights½*
  - *sumOfSquaredWeights = q.getBoost()* ^{2} $\cdot \sum_{(t \text{ in } q)}$*(idf(t) · t.getBoost())* ^{2}
- TF, frecuencia de término en el documento, es la medida de la frecuencia con que aparece un término en el documento (para nuestro trabajo de investigación, un "*servicio"*).
  - *tf(t in d) = frequency½*
- IDF, frecuencia inversa del documento, es la medida de la frecuencia con que aparece el término a través de todo el índice.
  - *idf(t) = 1 + log (numDocs /docFreq+1)*
- BOOST, en el que se tiene el impulso del campo en tiempo de indexación ("*boost index*") y el impulso del campo en tiempo de consulta ("*boost query*").
- NORM, encapsula tanto los factores de impulso y longitud en tiempo de indexación como los factores de impulso y longitud del campo ("*field boost"*) antes de agregar el campo a un documento. Así mismo calcula la longitud normalizada ("*lengthNorm")* del campo cuando el documento se agrega al índice, de acuerdo con el número de tokens de este campo en el documento, de modo que los campos más cortos contribuyen más a la puntuación. En definitiva es la medida de la importancia de un término según el número total de términos en el atributo.
  - *norm(t,d) = lengthNorm ·* $\prod_{(field\ f\ in\ d\ named\ as\ t)}$ *· f.boost().*

2.4. Estructuración de las consultas ("Querys")

En esta sección y una vez definida la relación entre las entidades de ambos metamodelos y haber definido qué atributos serán utilizados en las búsquedas, se deberá proponer una consulta conforme se ha indicado en el apartado anterior, para realizar el descubrimiento de Servicios, que devuelva una lista de Servicios. Como se expondrá en las secciones siguientes, será el resultado de un proceso en el que se decidirá, a partir de la puntuación ("*score"*)

obtenida en la búsqueda, si ese servicio puede ser un Servicio Candidato que cubra todo o parte de dicha funcionalidad. Para ello se deberán crear una serie de consultas que nos indiquen la probabilidad que un Servicio sea un Servicio Candidato, atendiendo a las entidades principales y posteriormente comprobando la variación de esa probabilidad usando las entidades adyacentes, de manera que se diseñarán dos consultas, una para las entidades principales y otras para las entidades adyacentes.

La notación que se ha utilizado es la más habitual en los motores de búsqueda textuales, de manera que se realizarán consultas usando la distancia entre los términos cuando sean campos de poca longitud (nombres, títulos, etc.…) y de forma abierta en las descripciones textuales (descripciones, asertos, indicadores, etc.…) de manera que complemente a las anteriores. Así mismo se indicará el peso que tendrá cada campo en la puntuación final.

Para ello, además de los operadores lógicos habituales, se van a utilizar dos operadores de las búsquedas de texto indexadas que son:

- **~(distancia),** operador que indica la distancia entre cada palabra permitida. Por ejemplo si tenemos un Servicio cuyo título es "*Notificaciones electrónicas fehacientes a los ciudadanos*" y queremos buscar "*Notificaciones a los Ciudadanos*", con una distancia de 2 se encontrará "*Notificaciones* ***electrónicas fehacientes*** *a los Ciudadanos*", los términos señalados en negrita están incluidos en la distancia (2) permitida y por tanto la búsqueda sería positiva[5].
- **^(impulso),** operador que multiplica la puntuación de ese campo cuando se hallan las coincidencias. Se ha tomado como máximo 10 y se va dividiendo en función de la importancia del campo de manera que siempre el impulso que toma la puntuación ("*score"*) es el doble que en el anterior nivel.

De esta manera en la expresión VI.2 podemos observar la estructura de la consulta para las entidades principales:

Expresión VI.2. Query para las entidades principales.

```
query := " (Service.name: UserStory.name~2)^10 OR Service.description: UserStory.description ^5 OR (Operation.name: (UserStory.name)~2)^5 OR Operation.description: UserStory.description^2,5 OR Operation.description: Task.description^1,25"
```

Así mismo se deberá realizar una segunda consulta con las entidades adyacentes de manera que en la expresión VI.3 podemos observar la estructura de la consulta para dichas entidades:

Expresión VI.3. Query para las entidades adyacentes.

```
Query := "Tag.tag: Epic.description OR Domain.description: Epic.description OR Stakeholder.name: Person.name~1 OR Stakeholder.role: (Person.description) OR Policy.policy: (AcceptanceCriteria.assert DefinitionOfDone.fact) OR ServiceLevelAgreement.kpi: (AcceptanceCriteria.assert DefinitionOfDone.fact)"
```

En este caso se buscará cualquier coincidencia en las entidades de manera que puedan captarse los matices más lejanos de la semejanzas, sin especificar una distancia entre términos, salvo en el nombre del interesado, que lleva la distancia mínima posible.

[5] Las partículas como artículos, preposiciones y otras similares son eliminadas en la configuración del motor de búsqueda, tanto al hacer las consultas como al indexar el contenido.

3. Proceso de descubrimiento de Servicios Candidatos

Una vez que se han identificado las relaciones básicas entre los metamodelos, tanto a nivel de entidad como de sus atributos y se han estructurado las consultas para cada uno de los tipos de entidades propuestos, en este apartado se describirá, de forma detallada, el proceso de descubrimiento de Servicios Candidatos al que pretende dar respuesta este trabajo de Tesis Doctoral.

En este punto, aunque ya se ha indicado en otros lugares a lo largo de este trabajo, se hace notar que la creación del Catálogo de Servicios y por tanto la conformación de los Servicios al metamodelo expuesto en el Capítulo IV, es algo anterior en el tiempo y no forma, por tanto, parte del proceso de descubrimiento de Servicios Candidatos en sí, aunque es necesario que se haya dado para su correcta ejecución.

3.1. El descubrimiento de servicios como actividad dentro del desarrollo ágil

En este apartado es muy importante recalcar, como se pudo observar en el Capítulo III, que nos encontramos usando una metodología de desarrollo ágil basada en "*Scrum*" y por tanto, según su propia naturaleza, iterativa, de manera que este proceso se repetirá tantas veces y con la periodicidad que esté previsto en dicha metodología.

Esto es importante porque el proceso de descubrimiento de los Servicios Candidatos es una actividad inmersa en este proceso ágil y que se ejecuta dentro de la elicitación de requisitos, y como se puede deducir del Capítulo III, serán una actividad dentro del "*Sprint*", una vez que se ha ejecutado el "*Product Backlog Grooming*" y el "*Sprint Planning*", momento en el que se realiza la gestión de las Historia de Usuario y se desglosa cada Historia de Usuario en tareas y por tanto tenemos toda la información necesaria para modelar el requisito ágil de forma completa. La descripción de este proceso se hace, por tanto, para un requisito modelado como una Historia de Usuario.

Como se especificó en el Capítulo V, en el metamodelo de requisitos ágil basado en valor, en la metaclase *«UserStory»,* existen una serie de atributos encaminados a calcular el retorno de la inversión (ROI, "*Return of Investment*"). Estos atributos se usarán para ordenar las Historias de Usuario a razón de maximizar ese retorno de la inversión, consiguiendo el mayor valor de negocio (atributo *«bussinesValue»*) con el menor esfuerzo posible (atributo *«size»).* Esto implica que en caso de restricciones que afecten a los parámetros del desarrollo de aplicaciones (coste, tiempo, esfuerzo) podremos identificar de forma eficiente y sencilla y con un bajísimo coste, sobre qué requisitos del conjunto de requisitos vamos a actuar y por ende, a lanzar el proceso de descubrimiento de Servicios Candidatos, pudiendo adaptar el proceso a las características particulares de las condiciones y del entorno de ese desarrollo en particular.

Esta priorización, valoración y ordenación de los requisitos, aunque son parte del proceso ágil, no entran a formar parte del proceso de descubrimiento de Servicios Candidatos, que como se ha comentado anteriormente, comenzará cuando se formalice ese requerimiento.

Gráficamente y para encuadrar el momento en el tiempo en el que se ejecutará el proceso, se muestra en la figura VI.2, dentro del framework ágil descrito en el Capítulo III, el momento concreto en que se ejecutará la actividad en cada iteración del proceso.

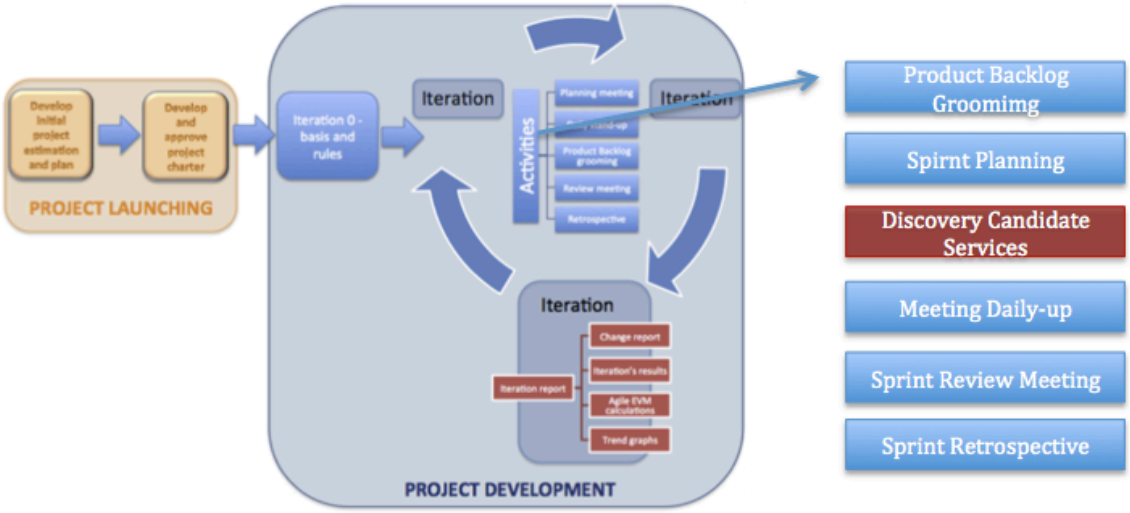

Figura VI.2. Actividad para el descubrimiento de Servicios dentro de una iteración.

3.2. Fases del proceso de descubrimiento de Servicios Candidatos

A continuación y una vez enmarcada esta actividad, se irán describiendo cada una de las fases del proceso de descubrimiento de Servicios Candidatos por orden cronológico y secuencial. Este proceso comenzará a raíz de la identificación de una nueva funcionalidad y terminará con la propuesta de los posibles Servicios Candidatos, como se puede observar en la Figura VI.3.

Figura VI.3. Diagrama conceptual del proceso de descubrimiento de Servicios Candidatos.

Los pasos indicados se pueden describir funcionalmente como:

1. **Formalización del requisito.** El equipo de trabajo, una vez identificado el requisito, elige realizar el proceso de descubrimiento de Servicios Candidatos. El primer paso del proceso consiste por tanto en el envío y formalización del requisito en orden a conformarse con el metamodelo ágil de requisitos basados en valor. Para ello y usando las técnicas ágiles descritas en el capítulo III, se modelará esa Historia de Usuario hasta obtener cada una de las entidades que soporten dicha funcionalidad.
2. **Diseño y ejecución de las consultas.** Una vez formalizado el requisito, será necesario construir y estructurar las consultas, es decir, realizar una transformación desde el requisito a la consulta, de manera que podemos tener las consultas que se ejecutarán en un paso posterior. Estas habrán de ser lanzadas contra el Catálogo de Servicios y se habrán de depurar los resultados, para ello se lanzarán las dos consultas en orden cronológico, primero la de entidades principales y después la de entidades adyacentes. Ambas consultas nos devolverán un conjunto de Servicios con su puntuación ("*score*").
3. **Preparación de los Servicios.** Una vez obtenidas estas listas de Servicios con sus puntuaciones, será necesario realizar un tratamiento algorítmico sobre las mismas, de manera que se realizará una unión de los conjuntos en las que solamente saldrán

aquellos Servicios que hayan tenido alguna incidencia en las entidades principales. Una vez seleccionados dichos Servicios se sumarán las puntuaciones obtenidas en cada consulta y se escogerán aquellos que estén por encima del valor de corte escogido en las entidades principales. En los sistemas de búsqueda y recuperación de información basado en texto rara vez se hace una consulta sin recibir una puntación, por lo que será necesario tomar un valor de referencia, que se denominará FILTRO_REFERENCIA, por debajo del cual se considerará que la relación de semejanza no existe y por tanto ese Servicio será filtrado y eliminado del proceso y no podrá constar como un Servicio que pueda llegar a ser Candidato.

4. **Análisis de los Servicios.** Una vez ejecutado el paso anterior, se obtendrá el conjunto de posibles Servicios Candidatos asociados a un requisito ágil dado. Este algoritmo tendrá a su vez otro valor de corte que vendrá dado por una función de probabilidad basada en una distribución normal, que puede ser refinado por parte de cada organización, llegando a usar incluso umbrales discretos, si el organismo lo dispone así. Este umbral deberá calcularse o indicarse a partir de la experiencia, debido a que el contexto de la organización, el uso del lenguaje natural y la capacidad de descripción por parte de los equipos de desarrollo no es siempre idéntica. Como todo método empírico, tendrá que realizarse un ajuste para cada organización. Para este trabajo de investigación y una vez obtenida la puntación de los Servicios que han pasado el filtro de referencia, se calculará un segundo punto de corte basado en las distancias entre la puntuación a través de la distribución de una curva de Gauss, proponiendo como regla general la media aritmética menos la desviación típica, es decir, la desviación estándar. Escogemos como Servicios Candidatos aquellos cuya puntuación es mayor que la desviación estándar.

Una vez ejecutados todos pasos del proceso, tendremos los Servicios Candidatos y por tanto aquello servicios dentro del Catálogo de Servicios que cubren un requisito ágil dado.

De esta manera y en pseudocódigo el proceso completo viene implementado en el algoritmo que se propone a continuación en la Expresión VI.4. para obtener, a partir de un requisito ágil, la lista definitiva de Servicios Candidatos.

Expresión VI.4. Algoritmo para la obtención de la lista de Servicios Candidatos.

```
RequistoAgile aRE <UserStory, Task, Epic, Person, AcceptanceCriteria, DefinitionOfDone>

Resulset <Service, Real> resulsetQ1, resulsetQ2;

List <Service> serviciosCandidatos;

Real scoreTotal, medScore;

Query q1 = "(Service.name: aRE.UserStory.name~2)^10 OR Service.description: aRE.UserStory.description^5 OR (Operation.name: (aRE.UserStory.name)~2)^5 OR Operation.description: aRE.UserStory.description^2,5 OR Operation.description: aRE.Task.description^1,25";

Query q2 = "Tag.tag: aRE.Epic.description OR Domain.description: aRE.Epic.description OR Stakeholder.name: aRE.Person.name~1 OR Stakeholder.role: (aRE.Person.description) OR Policy.policy: (aRE.AcceptanceCriteria.assert aRE.DefinitionOfDone.fact) OR ServiceLevelAgreement.kpi: (aRE.AcceptanceCriteria.assert aRE.DefinitionOfDone.fact)"

resulsetQ1 := execute.query(q1);

resulsetQ2 := execute.query (q2);

for(int i=0; i <resulsetQ1.length; i++){

        if (resulsetQ1[i].getScore > FILTRO_REFERENCIA){
```

```
                if (resulsetQ2.contains(resulsetQ1[i]).services)){
              aux = resulsetQ2.getService(resulsetQ1[i].service);
              resulsetQ1[i].score +:=aux.getScore();
        } else {
        resulsetQ1[i].remove(); }
}

medScore := desviacionEstandar(resulsetQ1);
for (int i =0 ; i < sp.length; i ++){
        if (resulset [i].score >= medScore){
                serviciosCandidatos.add(resulset.Service);
        }
}
return serviciosCandidatos;
// Función que realiza el cálculo de la probabilidad que se tomará como umbral
function desviacionEstandar(Resulset <Service, Real> resulset) {
Real sumatorio = 0.0;
Real media = 0.0;
Real varianza = 0.0;
Real desviacion = 0.0;
Real nServicios = resulset.length;
for (int i = 0; i < nServicios; i++) {
     sumatoria += lista.get(i).getScore();
}
media = sumatorio / nServicios;
for (int i = 0; i < nServicios; i++) {
       double rango;
        rango = Math.pow(lista.get(i).getScore() - media, 2);
        varianza = varianza + rango;
}
varianza = varianza / 10f;
desviacion = Math.sqrt(varianza);
return media - desviacion;
}
```

Para una mejor comprensión del algoritmo anterior se aporta un diagrama de secuencia del proceso desde la perspectiva funcional. La Figura VI.4 muestra gráficamente este proceso de descubrimiento de Servicios Candidatos desde el punto de vista funcional, atendiendo a los elementos que se han ido presentando en las secciones anteriores y mostrando como se relacionan entre ellos en tiempo de ejecución. Se ha optado por representar tres líneas de tiempo, la de los miembros del equipo, la que hace referencia al trabajo con los requisitos ágiles y la que hace referencia al trabajo con los Servicios (con la información del Catálogo de Servicios que ha sido indexada, para ser más exactos).

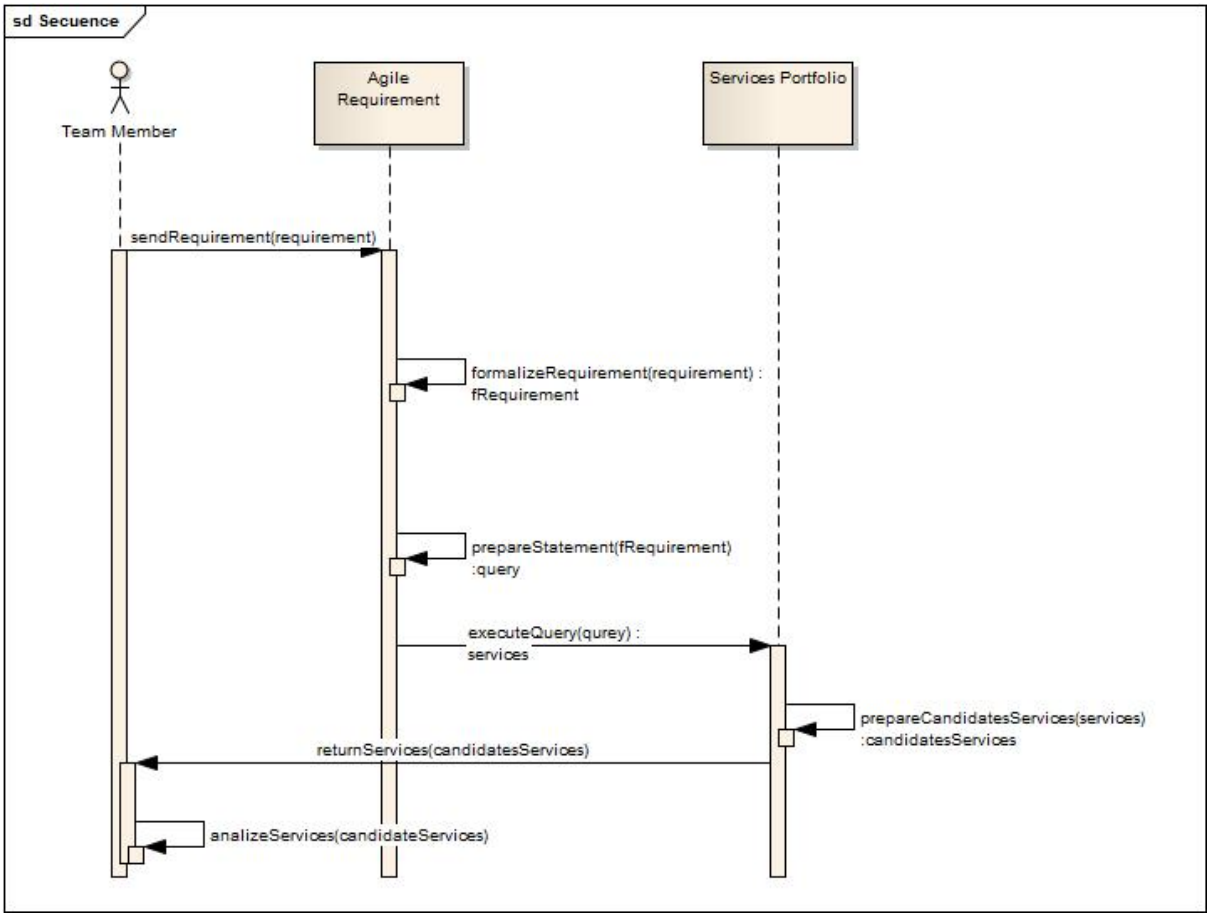

Figura VI.4. Diagrama de secuencia funcional del proceso de descubrimiento de Servicios.

Una vez ejecutado el proceso descrito en este apartado se obtendrá el conjunto de posibles Servicios Candidatos asociados a un requisito ágil dado. Por tanto, con la obtención de la lista de Servicios Candidatos lista para su análisis de viabilidad, concluye el proceso de descubrimiento propuesto para este trabajo de Tesis Doctoral.

En la validación de los trabajos de investigación de esta Tesis Doctoral, descrita en el Capítulo VIII, se ha utilizado un FILTRO_REFERENCIA de valor 10 y una puntuación de corte que es la media aritmética menos la desviación típica (calculada sobre los valores resultantes una vez pasado el filtro de referencia), que como se ha indicado, es la definición de la desviación estándar y que estadísticamente contiene el 68% de las observaciones de la distribución.

Cabe recalcar, aunque no sea una actividad automática, que una vez obtenida la lista de los Servicios Candidatos, serán los equipos de desarrollo los encargados de analizar la viabilidad del uso de estos Servicios, ya que entran a formar parte de este análisis cuestiones que quedan fuera del alcance de esta Tesis Doctoral y que son de índole política, jurídica, económica, técnica, funcional o práctica y que impiden que se pueda automatizar la decisión del uso del Servicio Candidato. Así mismo una vez elegido, el proceso de ejecución de las políticas del Ciclo de Vida de los Servicios y del Gobierno de los mismos, así como el de incorporación al desarrollo ya planteado en el nuevo Sistema de Información, quedan fuera del alcance de esta Tesis Doctoral y podrán constituir parte de los trabajos futuros que se expondrán en el Capítulo IX.

Dentro de este aparatado será necesario mencionar que la automatización de alguna de estas tareas suponen un profundo ahorro de costes en la organización así como homogeniza dicho

proceso. La automatización de los diversos pasos se describirá de manera detallada en el capítulo siguiente.

4. Conclusiones

A lo largo de este capítulo se ha descrito la solución adoptada para, junto con los metamodelos propuestos en los Capítulos IV y V, poder realizar el proceso de descubrimiento de los Servicios Candidatos para un requisito dado. Así mismo se ha presentado, dentro de la metodología ágil de desarrollo expuesta en el Capítulo III, el lugar exacto donde se ejecuta este proceso dentro del conjunto de actividades para cada iteración.

Se ha descrito en profundidad en este capítulo el proceso de descubrimiento de Servicios Candidatos para un requisito ágil dado, describiendo con lenguaje natural cada uno de sus pasos, proponiendo un algoritmo en pseudocódigo que lo implementa y mostrando el proceso funcional a través de un diagrama de secuencia.

Además, en este capítulo, se han incoado los principales elementos que formarán parte de la arquitectura tecnológica que dan soporte a esta solución y que se describirán con todo detalle en el capítulo siguiente, en el que se expondrá, de manera completa, el framework que soporta el proceso de descubrimiento de Servicios Candidatos, de manera que dicha implementación se pueda desplegar en una organización real concreta.

Capítulo VII Soporte tecnológico: Framework DS4aRE

En este capítulo, una vez detallado el proceso para el descubrimiento de los Servicios Candidatos en el capítulo anterior, se realizará una descripción del entorno tecnológico que posibilita las bases para la realización de dicho proceso, así como la justificación de las tecnologías y herramientas escogidas, aportado aquellas características, tanto intrínsecas como extrínsecas, que han sido relevantes para su elección.

Se comenzará con la descripción de cada uno de los elementos o sistemas que conforman el framework denominado "DS4aRE" ("*Discovering Services for agile Requirements")* y que implementa la definición del proceso de descubrimiento de Servicios Candidatos expuesto en el anterior capítulo. Una vez descritos en profundidad los elementos de este framework, se expondrá cómo trabajan de forma coherente y sistemática en dicho proceso de descubrimiento, así como el grado de automatización alcanzado.

Por último, el capítulo termina con unas conclusiones acerca del framework que soporta el proceso de descubrimiento de Servicios Candidatos.

1. Introducción

A continuación y una vez descrito el proceso de descubrimiento de los Servicios Candidatos en el capítulo anterior, se describirá la arquitectura tecnológica que da soporte a la solución planteada en este trabajo de Tesis Doctoral para el descubrimiento de los Servicios Candidatos usando metodologías y técnicas ágiles.

Se ha utilizado el termino framework, ya que no existe actualmente ninguna herramienta específica que realice todo el proceso completo y porque hay elementos que son preexistentes en el tiempo y que por tanto tienen necesidad de su propia arquitectura. Para ello se irán describiendo cada uno de los elementos que forman parte de la arquitectura y una vez descritos, se mostrará cómo participan de forma coherente y sistemática dentro del proceso de descubrimiento de los Servicios Candidatos.

2. Elementos del framework DS4aRE

En este apartado se describirán los diferentes elementos usados en la arquitectura tecnológica de la solución aportada y se indicarán cuáles han sido las razones de su elección. Algunas de estas motivaciones hacen referencia a la idoneidad y características del elemento tecnológico en sí y otras a la idiosincrasia del contexto donde ha sido desarrollado y probado este proceso de descubrimiento de Servicios Candidatos, es decir, la Consejería de Cultura de la Junta de Andalucía, debido a las limitaciones tecnológicas impuestas por el propio contexto o sus destinatarios inmediatos.

Dado que alguna de las herramientas aquí expuestas son de propósito general, algunas de ellas podrán ser sustituidas por otras de funcionalidad similar, con el objeto de adaptarse al contexto de la organización donde se quiera implantar dicho proceso de descubrimiento.

2.1. Enterprise Architect

En este trabajo de Tesis Doctoral se ha optado por la herramienta de modelado Enterprise Architect [SparxSystems 2016] como herramienta base sobre la que modelar y desarrollar los metamodelos y los perfiles UML propuestos en los capítulos IV y V del presente trabajo de Tesis Doctoral.

Esta decisión se ha tomado después de tener en cuenta las conclusiones de un trabajo [IWT2 2008] realizado por el grupo de investigación Ingeniería Web y Testing Temprano de la Universidad de Sevilla y la Consejería de Cultura.

Este trabajo consistió en la realización de un estudio comparativo de nueve herramientas de modelado y su propósito era identificar aquella que mejor se adecuaba a las necesidades dicho organismo. Para alcanzar este objetivo, el estudio establecía un esquema de caracterización que proporcionó un marco estándar sobre el cual fundamentar la elección tomada. Estas características fueron las siguientes:

1. Soporte a mecanismos de extensión de UML con los que sea posible la definición de perfiles UML.
2. Soporte a mecanismos que posibiliten la generación sistemática de modelos y código a partir de modelos.
3. Soporte para la generación de documentación de forma automática, personalizada y flexible.
4. Soporte para la gestión del ciclo de vida software de grandes proyectos.
5. Soporte para gestionar de manera completa el ciclo de vida software teniendo cuenta las fases de estudio de viabilidad, captura de requisitos, análisis, diseño, implementación, pruebas y mantenimiento.
6. Compatibilidad con UML.

La Tabla VII.1 resume los resultados del estudio comparativo. Por simplicidad y legibilidad, sólo se ha indicado qué característica, de las indicadas anteriormente, tiene cada herramienta.

Tabla VII.1. Estudio comparativo de herramientas de modelado.

	NDT-Tool	ArgoUML	StarUML	Poseidon for UML	Rational Rose	Rational Software Modeler	EA	IRqA	Doors
1	*x*		*x*		*x*	*x*	*x*	*x*	
2	*x*		*x*		*x*	*x*	*x*	*x*	
3	*x*				*x*	*x*	*x*	*x*	x
4		*x*	*x*	*x*	*x*	*x*	*x*	x	x
5		*x*	*x*	*x*	*x*	*x*	*x*		
6	*x*	*x*		*x*		*x*	*x*		

Dentro de Enterprise Architect (EA) se han instalado los Perfiles UML descritos en los Capítulos IV y V de cada uno de los metamodelos, que son utilizados para el modelado de los Servicios y los Requisitos Ágiles basados en Valor.

Así mismo desde el Enterprise Architect son exportados a formato XML, a través de la utilidad de Generación XSD, que convierte un modelo de clases UML en un esquema de XML W3C (XSD). Esto permite a los equipos de desarrollo comenzar a trabajar a nivel conceptual en UML, dejando los aspectos tediosos de creación de XSD a EA. La generación de esquemas se puede personalizar si fuera necesario, usando el "Perfil UML para XML" provisto por defecto en la herramienta. Esto permitirá en definitiva tener en un formato XML las instancias de los metamodelos que deberán ser utilizados en los diferentes elementos de la arquitectura.

2.2. Apache Solr

Dentro de la arquitectura tecnológica propuesta, destaca como pieza central un sistema de búsqueda y recuperación de información ("*Information Retrieval System*", *IR System*), que actuará como motor de búsqueda e indexación. Para esta pieza se ha elegido la implementación de Apache Solr [Apache Solr 2016], ya que es la plataforma JAVA más moderna que usa el motor de indexación y búsqueda de Apache Lucene.

Solr es la plataforma de búsqueda NoSQL más popular, más rápida y de fuentes abiertas del proyecto Apache Lucene. Sus características principales incluyen una búsqueda muy potente y completa de texto, clustering dinámico, integración de base de datos, soporte para documentos de texto enriquecido y búsqueda geoespacial. Solr es altamente escalable, proporcionando búsquedas distribuidas y es tolerante a fallos. En la Figura VII.1 se puede observar la interfaz que Solr expone para su trabajo.

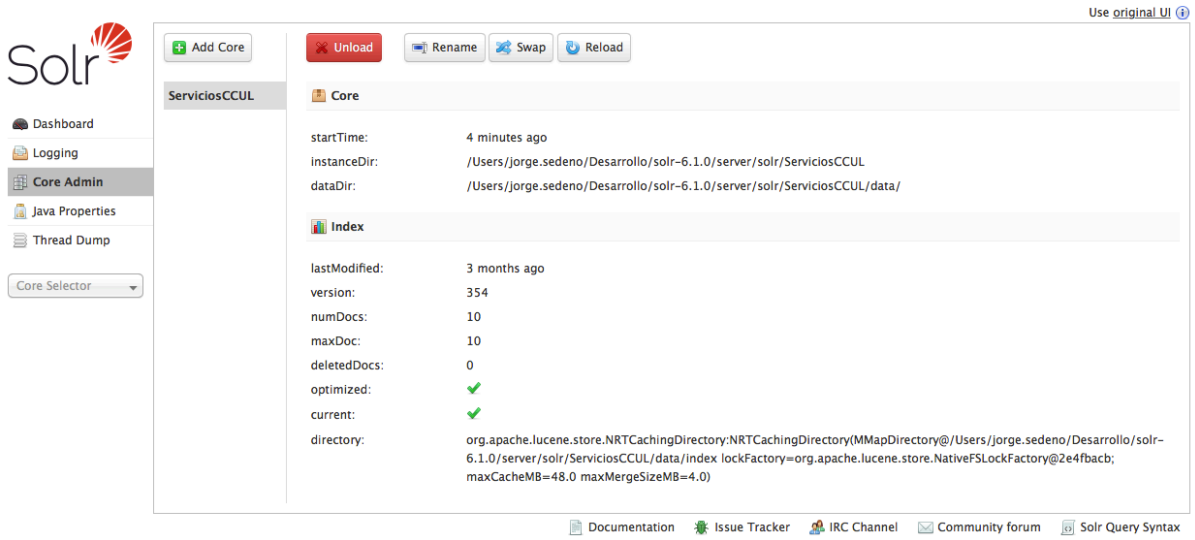

Figura VII.1. Interfaz de Solr para configuración y administración.

Además Solr está escrito en JAVA y se ejecuta como servidor de búsqueda independiente dentro de un contenedor de servlets (como puede ser Apache Tomcat). Solr utiliza la librería de Lucene como núcleo para la indexación y búsqueda de texto completo y ofrece interfaces tanto REST como HTTP y XML o JSON que hacen que sea fácil de usar prácticamente desde cualquier lenguaje de programación. La potente configuración externa de Solr permite que pueda adaptarse a casi cualquier tipo de aplicación. Además cuenta con una arquitectura de plugins que permite extender su funcionalidad, cuando se requiera una personalización más avanzada.

Por su potencia y funcionalidad en la indexación y búsqueda, desde la Consejería de Cultura de la Junta de Andalucía se venía utilizando este motor para otra tipología de proyectos, los portales web y los tramitadores de expedientes administrativos, para mejorar las búsquedas, una vez indexada, de la información de los mismos.

Por lo tanto la elección de Solr viene supeditada no sólo por su características tecnológicas, sino debido a su conocimiento dentro de la organización en la que se encuentra implantada. Además, desde el punto de vista de las tecnologías que utiliza (JAVA, XML, JSON, REST, …) estaba alineado a la perfección con el conocimiento de los equipos de desarrollo de software que habitualmente trabajan como proveedores de dicho organismo, así como del conocimiento del personal interno de la organización.

En este contexto Solr es utilizado para dos cuestiones fundamentales dentro el proceso de descubrimiento de Servicios propuesto en este capítulo:

- Solr es utilizado como motor de indexación donde se encuentra contenido el Catálogo de Servicios de la organización, es decir, cada instancia del metamodelado de un Servicio que se modela en Enterprise Architect es indexado en este motor de manera que contiene toda la información necesaria para la búsqueda sobre los Servicios. Cada Servicio representa un documento en el motor de indexación. Se indexará por tanto el Catálogo de Servicios de la organización que está en el Enterprise Architect.
- Es el motor de búsqueda sobre el que se lanzan las consultas que se han estructurado en el capítulo anterior, de manera que cada una de ellas devolverá un conjunto de documentos con una puntuación. Como cada documento representa un Servicio que está indexado y cada resultado tendrá una puntuación específica, podremos ejecutar el algoritmo indicado en el capítulo anterior.

En el contexto de esta Tesis Doctoral nos centraremos en estas funcionalidades, necesarias para el descubrimiento de Servicios, en la que tendremos una instancia de Solr en la que se indexarán todos los servicios disponibles en el Catálogo de Servicios (Enterprise Architect) de la organización.

Es necesario hablar aquí del concepto de indexación enmarcado dentro de una herramienta de búsqueda y recuperación de información, como es en este caso Apache Solr. La indexación es la operación que se realiza para construir un índice de términos a partir de un esquema de información, de manera que cada texto en lenguaje natural que lleva un documento es filtrado, analizado y descompuesto en términos para poder crear dicho índice, de manera que se puedan realizar luego búsquedas sobre él y por tanto devolver los documentos con la puntuación adecuada a ese índice creado, aplicando de forma interna por la herramienta las funciones presentadas en el capítulo anterior sobre la semejanza y la puntuación.

Para ello será necesario definir en Apache Solr una serie de elementos como los esquemas, los filtros y los *tokenizers*, de acuerdo con la organización concreta donde se instancie. En el Capítulo VIII se describirá en profundidad como se han definido cada uno de estos elementos, que aunque no participan directamente en el proceso de descubrimiento que se expone en este capítulo (ya que han de ser preexistentes en el tiempo), sí son necesarios para realizar la indexación de los Servicios dentro de la herramienta y su posterior búsqueda.

2.3. Algoritmo DS4aRE

La tercera pieza de este framework es un proyecto MAVEN desarrollado en JAVA que implementa el algoritmo definido y diseñado en el capítulo anterior y que desde el punto de vista funcional, teniendo como entrada la información de un requisito modelado mediante el metamodelo de Requisitos Ágiles basado en Valor presentado en el Capítulo IV, devolverá una lista de Servicios Candidatos, una vez indexados los Servicios y realizando las llamadas necesarias al motor de búsqueda Apache Solr.

En la Figura VII.2 podemos observar el diagrama de clases del proyecto JAVA DS4sRE con las diferentes clases que intervienen y que conforman su arquitectura.

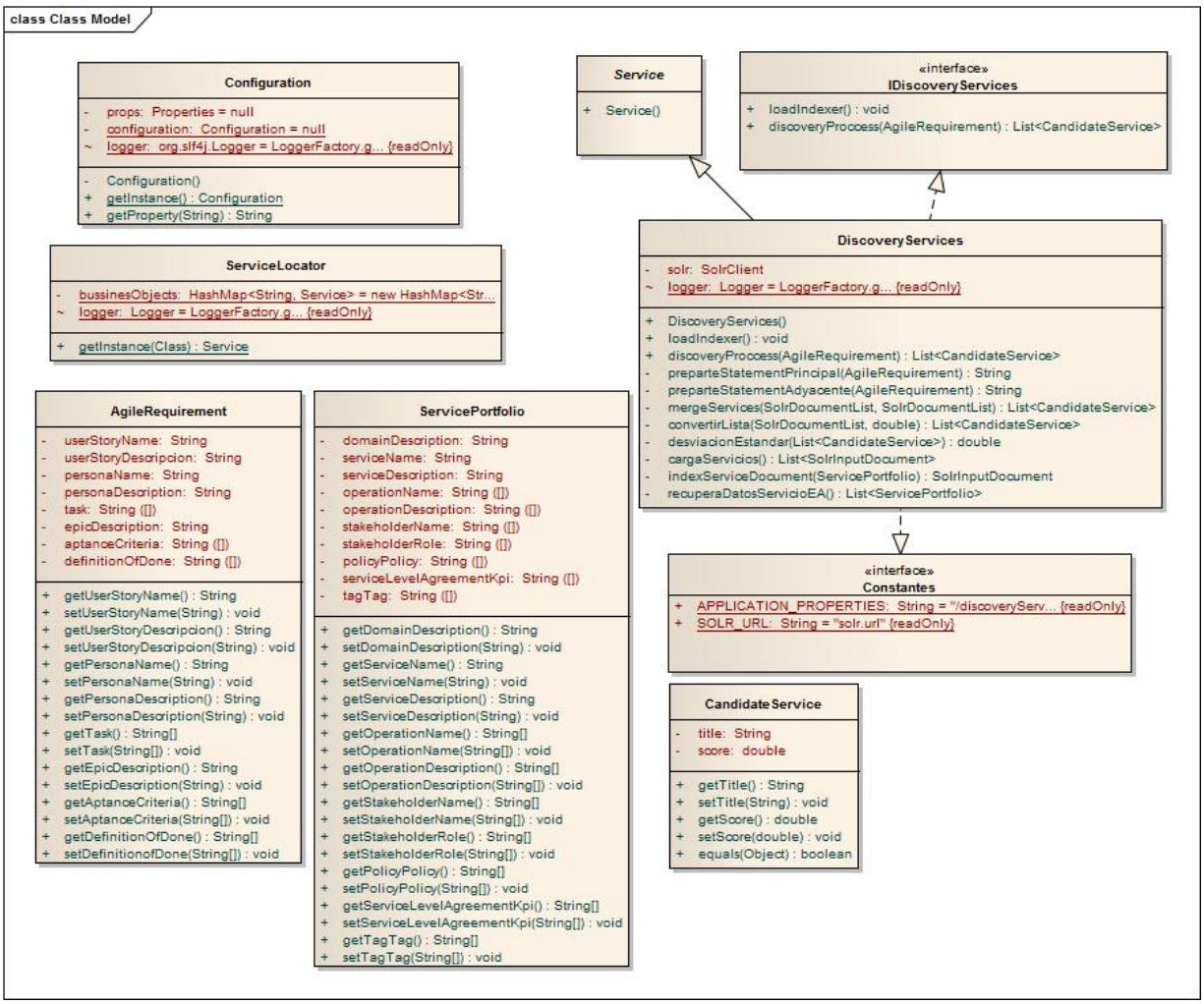

Figura VII.2. Diagrama de Clases del proyecto DS4aRE.

Como se ha indicado anteriormente, este proyecto se ha desarrollado en Java (versión 8), se ha estructurado mediante MAVEN[6], que es una herramientas de gestión de proyectos y ha sido

[6] Apache Maven es una herramienta de gestión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede gestionar la compilación, la elaboración de informes y la documentación de un proyecto a partir de una información central. Se puede consultar toda la información referente a este producto en http://maven.apache.org

desarrollado usando el IDE de JAVA Eclipse[7] (versión Neón), herramienta que forma parte del entorno estándar de desarrollo instalado en la Consejería de Cultura. Para ello además de mostrar el diagrama de clases, en la Figura VII.3 se presenta la estructura del proyecto DS4aRE dentro del IDE de Eclipse.

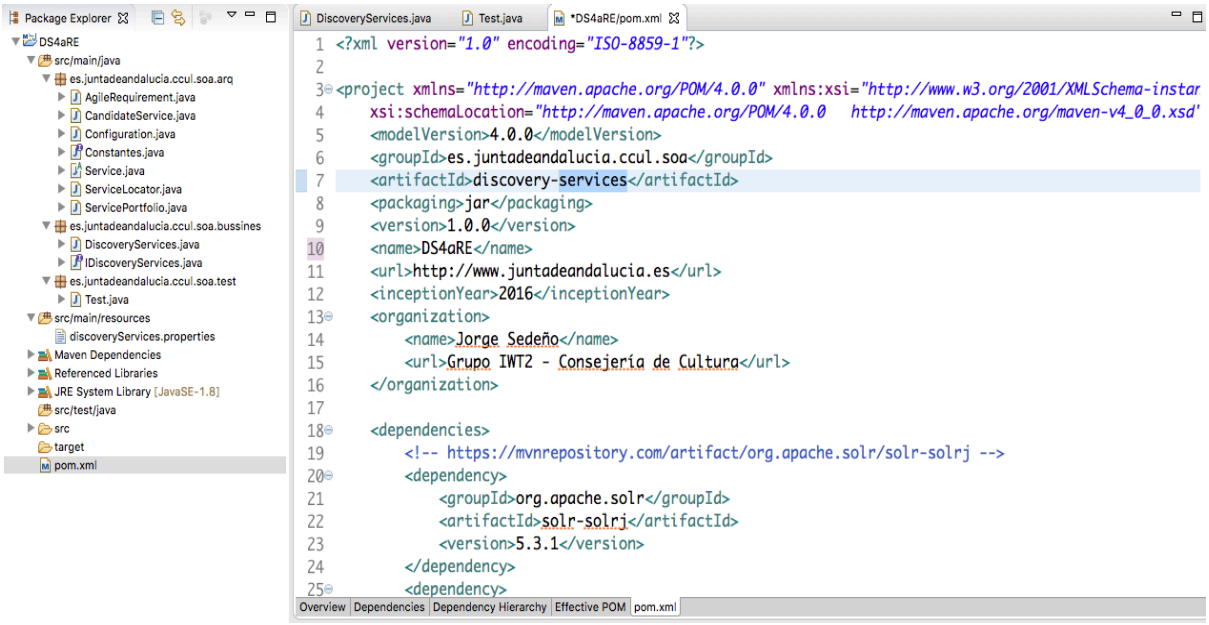

Figura VII.3. Estructura MAVEN del proyecto DS4aRE con su fichero de configuración.

A continuación y una vez descrito el proyecto JAVA en general, nos centraremos en describir la funcionalidad de cada una de las clases junto con el código fuente de las mismas.

El orden en el que se describen cada una de estas clases ha sido considerado de forma lógica, desde aquellas clases más tangenciales al proceso de descubrimiento de Servicios Candidatos hasta las que implementan directamente la lógica del mismo:

- Clases referentes a la configuración del proyecto y al diseño del patrón "*ServiceLocator*" («Contantes» , «Configuration», «Service» y «ServiceLocator»).
- Clases planas que sirven para implementar el modelado de los requisitos ágiles y de los servicios («AgileRequirement», «ServicePortfolio», «CanditateService»,).
- Clases que implementan directamente el algoritmo para el descubrimiento de los Servicios Candidatos («IDiscoveryServices» y «DiscoveryServices»).

2.3.1 Interface «Constantes»

Esta interfaz contiene una serie de constantes para facilitar el proceso de descubrimiento. En la Expresión VII.1 podemos observar el código fuente de dicha interfaz.

[7] Eclipse proporciona IDEs y plataformas para casi todos los idiomas y arquitecturas. Somos famosos por nuestros IDE de Java IDE, C / C ++, JavaScript y PHP construidos sobre plataformas extensibles para crear IDEs de escritorio, Web y de la nube. Estas plataformas ofrecen la más amplia colección de herramientas complementarias disponibles para los desarrolladores de software. (https://eclipse.org/ide/)

Expresión VII.1. Código fuente JAVA para la interfaz «Constantes».

```
package es.juntadeandalucia.ccul.soa.arq;
public interface Constantes {
        public static final String APPLICATION_PROPERTIES="/discoveryServices.properties";
        public static final String SOLR_URL="solr.url";
        public static final Double FILTRO_REFERENCIA = 10f;
}
```

2.3.2 Class «Configuration»

Esta clase contiene el patrón "*singleton*" para la lectura del fichero de propiedades. En la Expresión VII.2 podemos observar el código fuente de dicha clase.

Expresión VII.2. Código fuente JAVA para la clase «Configuration».

```
package es.juntadeandalucia.ccul.soa.arq;
import java.util.Properties;
import org.slf4j.LoggerFactory;
public class Configuration {
        private Properties props = null;
        private static Configuration configuration= null;
        final static org.slf4j.Logger logger = LoggerFactory.getLogger(Configuration.class);

        private Configuration() {
                props = new Properties();
                try {
        props.load(Configuration.class.getResourceAsStream(Constantes.APPLICATION_PROPERTIES));
                } catch (Exception e) {
                        logger.error("No se puede leer el fichero de propiedades",e);
                }
        }
        public static Configuration getInstance() {
                if (configuration == null)
                        configuration =new Configuration();
                return configuration;
        }
        public String getProperty (String key){
                return props.getProperty(key);
        }
}
```

2.3.3 Class «Service»

Esta clase abstracta será padre de los todos los servicios de negocio que serán instanciados. En la Expresión VII.3 podemos observar el código fuente de dicha clase.

Expresión VII.3. Código fuente JAVA para la clase «Service».

```
package es.juntadeandalucia.ccul.soa.arq;
public abstract class Service{
        public Service() {
                super(); }
}
```

2.3.4 Class «ServiceLocator»

Esta clase modela el patrón "*ServiceLocator*" que se utilizará para instanciar los objetos de tipo "*Service*" (servicios de negocio). En la Expresión VII.4 podemos observar el código fuente de dicha clase.

Expresión VII.4. Código fuente JAVA para la clase «ServiceLocator».

```
package es.juntadeandalucia.ccul.soa.arq;
import java.util.HashMap;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class ServiceLocator {
    private static HashMap<String, Service> bussinesObjects = new HashMap<String, Service>();
    final static Logger logger = LoggerFactory.getLogger(ServiceLocator.class);
    /**
     * Obtiene una instancia de la clase
     * @param classname Nombre de la clase de la que se quiere obtener la instancia
     * @return Instancia de la clase
     */
    static public Service getInstance(@SuppressWarnings("rawtypes") Class classname) {
        Service service = (Service) bussinesObjects.get(classname.toString());
        if(service==null){
            try{
                service = (Service)classname.newInstance();
                bussinesObjects.put(classname.toString(),service);
            }catch(Exception e){
                String error="Error al instanciar la Lógica de negocio: "+classname+".
"+e.getMessage();
                logger.error(error,e);
                throw new IllegalArgumentException(error);
            }
        }
```

```
        return service;
    }
}
```

2.3.5 Class «AgileRequirement»

Esta clase JAVA es de tipo POJO ("*Plain Old Java Object*") y modela un objeto java que contiene la información necesaria para representar en plano un objeto de un requerimiento ágil basado en el metamodelo expuesto en el Capítulo V. En la Expresión VII.5 podemos observar el código fuente de dicha clase.

Expresión VII.5. Código fuente JAVA para la clase «AgileRequirement».

```
package es.juntadeandalucia.ccul.soa.arq;
/**
 * @author jorge.sedeno
 */
public class AgileRequirement {
        private String userStoryName;
        private String userStoryDescripcion;
        private String personaName;
        private String personaDescription;
        private String [] task;
        private String epicDescription;
        private String []acceptanceCriteria;
        private String []definitionOfDone;
        /**
         * @return the userStoryName
         */
        public String getUserStoryName() {
                return userStoryName;
        }
        /**
         * @param userStoryName the userStoryName to set
         */
        public void setUserStoryName(String userStoryName) {
                this.userStoryName = userStoryName;
        }
        /**
         * @return the userStoryDescripcion
         */
        public String getUserStoryDescripcion() {
                return userStoryDescripcion;
```

```
	}
	/**
	 * @param userStoryDescripcion the userStoryDescripcion to set
	 */
	public void setUserStoryDescripcion(String userStoryDescripcion) {
		this.userStoryDescripcion = userStoryDescripcion;
	}
	/**
	 * @return the personaName
	 */
	public String getPersonaName() {
		return personaName;
	}
	/**
	 * @param personaName the personaName to set
	 */
	public void setPersonaName(String personaName) {
		this.personaName = personaName;
	}
	/**
	 * @return the personaDescription
	 */
	public String getPersonaDescription() {
		return personaDescription;
	}
	/**
	 * @param personaDescription the personaDescription to set
	 */
	public void setPersonaDescription(String personaDescription) {
		this.personaDescription = personaDescription;
	}
	/**
	 * @return the task
	 */
	public String[] getTask() {
		return task;
	}
	/**
	 * @param task the task to set
	 */
	public void setTask(String[] task) {
```

```
                this.task = task;
        }
        /**
         * @return the epicDescription
         */
        public String getEpicDescription() {
                return epicDescription;
        }
        /**
         * @param epicDescription the epicDescription to set
         */
        public void setEpicDescription(String epicDescription) {
                this.epicDescription = epicDescription;
        }
        /**
         * @return the acceptanceCriteria
         */
        public String[] getAcceptanceCriteria() {
                return acceptanceCriteria;
        }
        /**
         * @param acceptanceCriteria the acceptanceCriteria to set
         */
        public void setAcceptanceCriteria(String[] acceptanceCriteria) {
                this.acceptanceCriteria = acceptanceCriteria;
        }
        /**
         * @return the definitionofDone
         */
        public String[] getDefinitionOfDone() {
                return definitionOfDone;
        }
        /**
         * @param definitionofDone the definitionofDone to set
         */
        public void setDefinitionofDone(String[] definitionOfDone) {
                this.definitionOfDone = definitionOfDone;
        }
}
```

2.3.6 Class «ServicePortfolio»

Esta clase JAVA es de tipo POJO ("*Plain Old Java Object*") y modela un objeto java que contiene la información necesaria para representar en plano una instancia de un servicio del Catálogo de Servicios de la organización y que una vez indexado podrá formar parte de los Servicios Candidatos. Está basado en el metamodelo expuesto en el Capítulo IV. En la Expresión VII.6 podemos observar el código fuente de dicha clase.

Expresión VII.6. Código fuente JAVA para la clase «ServicePortfolio».

```
package es.juntadeandalucia.ccul.soa.arq;
public class ServicePortfolio {
        private String domainDescription;
        private String serviceName;
        private String serviceDescription;
        private String [] operationName;
        private String [] operationDescription;
        private String [] stakeholderName;
        private String [] stakeholderRole;
        private String [] policyPolicy;
        private String [] serviceLevelAgreementKpi;
        private String [] tagTag;
        public String getDomainDescription() {
                return domainDescription;
        }
        public void setDomainDescription(String domainDescription) {
                this.domainDescription = domainDescription;
        }
        public String getServiceName() {
                return serviceName;
        }
        public void setServiceName(String serviceName) {
                this.serviceName = serviceName;
        }
        public String getServiceDescription() {
                return serviceDescription;
        }
        public void setServiceDescription(String serviceDescription) {
                this.serviceDescription = serviceDescription;
        }
        public String[] getOperationName() {
                return operationName;
        }
```

```
	public void setOperationName(String[] operationName) {
		this.operationName = operationName;
	}
	public String[] getOperationDescription() {
		return operationDescription;
	}
	public void setOperationDescription(String[] operationDescription) {
		this.operationDescription = operationDescription;
	}
	public String[] getStakeholderName() {
		return stakeholderName;
	}
	public void setStakeholderName(String[] stakeholderName) {
		this.stakeholderName = stakeholderName;
	}
	public String[] getStakeholderRole() {
		return stakeholderRole;
	}
	public void setStakeholderRole(String[] stakeholderRole) {
		this.stakeholderRole = stakeholderRole;
	}
	public String[] getPolicyPolicy() {
		return policyPolicy;
	}
	public void setPolicyPolicy(String[] policyPolicy) {
		this.policyPolicy = policyPolicy;
	}
	public String[] getServiceLevelAgreementKpi() {
		return serviceLevelAgreementKpi;
	}
	public void setServiceLevelAgreementKpi(String[] serviceLevelAgreementKpi) {
		this.serviceLevelAgreementKpi = serviceLevelAgreementKpi;
	}
	public String[] getTagTag() {
		return tagTag;
	}
	public void setTagTag(String[] tagTag) {
		this.tagTag = tagTag;
	}
}
```

2.3.7 Class «CandidateService»

Esta clase JAVA es de tipo POJO ("*Plain Old Java Object*") y modela un objeto java que contiene la información necesaria para representar en plano un Servicio Candidato. En la Expresión VII.7 podemos observar el código fuente de dicha clase.

Expresión VII.7. Código fuente JAVA para la clase «CandidateService».

```
package es.juntadeandalucia.ccul.soa.arq;
/**
 * Clase que contiene un POJO de Servicio Candidato
 * @author jorge.sedeno
*/
public class CandidateService {
	/**
	 * Título del Servicio
	 */
	private String title;
	/**
	 * Puntuación del Servicio
	 */
	private double score;
	/**
	 * @return the title
	 */
	public String getTitle() {
		return title;
	}
	/**
	 * @param title the title to set
	 */
	public void setTitle(String title) {
		this.title = title;
	}
	/**
	 * @return the score
	 */
	public double getScore() {
		return score;
	}
	/**
	 * @param score the score to set
	 */
```

```
	public void setScore(double score) {
		this.score = score;
	}
	/* (non-Javadoc)
	 * @see java.lang.Object#equals(java.lang.Object)
	 */
	@Override
	public boolean equals(Object obj) {
		if (this == obj) {
			return true;
		}
		if (obj == null) {
			return false;
		}
		if (!(obj instanceof CandidateService)) {
			return false;
		}
		CandidateService other = (CandidateService) obj;
		if (title == null) {
			if (other.title != null) {
				return false;
			}
		} else if (!title.equals(other.title)) {
			return false;
		}
		return true;
	}
}
```

2.3.8 Interface «IDiscoveryServices»

Esta interfaz modela el comportamiento que ha de realizar principalmente el proceso de descubrimiento de servicios y que consistirá en la indexación de los servicios para su búsqueda y el proceso de descubrimiento a partir de un requisito ágil.

En la Expresión VII.8 podemos observar el código fuente de dicha interfaz.

Expresión VII.8. Código fuente JAVA para la interfaz «IDiscoveryServices».

```
package es.juntadeandalucia.ccul.soa.bussines;
import java.util.List;
import es.juntadeandalucia.ccul.soa.arq.AgileRequirement;
import es.juntadeandalucia.ccul.soa.arq.CandidateService;
```

```
public interface IDiscoveryServices {
        /**
         * Método para indexar el Catálogo de Servicios
         */
        public void loadIndexer() throws Exception;
        /**
         * Proceso de descubrimiento de Servicios Candidatos
         * @param agileRE
         * @return
         * @throws Exception
         */
        public List<CandidateService> discoveryProccess(AgileRequirement agileRE) throws Exception;
}
```

2.3.9 Class «DiscoveryServices»

Esta clase implementa el proceso de descubrimiento que consistirá en la indexación de los servicios y en el proceso de descubrimiento a partir de un requisito y que implementa a la interfaz «*IDiscoveryServices*». A su vez esta clase extiende de la clase «*Service*», necesaria para implementar el patrón "*ServiceLocator,"* y se apoya en una interfaz para el uso de las constantes del proyecto, la interfaz «*Constantes*». En esta clase se implementa el algoritmo para el descubrimiento de Servicios Candidatos definido y diseñado en el Capítulo VI.

En la Expresión VII.9 podemos observar el código fuente de dicha clase.

Expresión VII.9. Código fuente JAVA para la clase «DiscoveryServices».

```
package es.juntadeandalucia.ccul.soa.bussines;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.apache.solr.client.solrj.SolrClient;
import org.apache.solr.client.solrj.SolrQuery;
import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.impl.HttpSolrClient;
import org.apache.solr.client.solrj.response.QueryResponse;
import org.apache.solr.client.solrj.response.UpdateResponse;
import org.apache.solr.common.SolrDocumentList;
import org.apache.solr.common.SolrInputDocument;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import es.juntadeandalucia.ccul.soa.arq.AgileRequirement;
```

```
import es.juntadeandalucia.ccul.soa.arq.CandidateService;
import es.juntadeandalucia.ccul.soa.arq.Configuration;
import es.juntadeandalucia.ccul.soa.arq.Constantes;
import es.juntadeandalucia.ccul.soa.arq.Service;
import es.juntadeandalucia.ccul.soa.arq.ServicePortfolio;
/**
 * @author jorge.sedeno
 *
 */
public class DiscoveryServices extends Service implements IDiscoveryServices, Constantes {
	private SolrClient solr;
	final static Logger logger = LoggerFactory.getLogger(DiscoveryServices.class);
	public DiscoveryServices() {
		super();
		try {
			String urlSolr = Configuration.getInstance().getProperty(Constantes.SOLR_URL);
			solr = new HttpSolrClient(urlSolr);
		} catch (Exception e) {
			e.printStackTrace();
			logger.error("Se ha producido un error al inicializar los servicios");
		}
	}
	/*
	 * (non-Javadoc)
	 * @see
	 * es.juntadeandalucia.cccul.soa.bussines.IDiscoveryServices#loadIndexer()
	 */
	public void loadIndexer() throws Exception {
		List<SolrInputDocument> servicios = null;
		servicios = cargaServicios();
		Iterator<SolrInputDocument> iter = servicios.iterator();
		try {
			UpdateResponse response = solr.deleteByQuery("*:*");
			System.out.println("Respuesta:" + response);
			solr.commit();
			solr.add(iter);
			solr.commit();
		} catch (SolrServerException e) {
			logger.error("Se ha producido un error al borrar el indexador los servicios");
			throw e;
		} catch (IOException e) {
```

```
                    logger.error("Se ha producido un error al borrar el indexador los servicios");
                    throw e;
            }
    }
    /*
     * (non-Javadoc)
     * @see es.juntadeandalucia.cccul.soa.bussines.IDiscoveryServices#
     * discoveryProccess()
     */
    public List<CandidateService> discoveryProccess(AgileRequirement agileRE) throws Exception {
            SolrQuery query = new SolrQuery();
            List<CandidateService> serviciosCadidatos = null;
            SolrDocumentList serviciosPrincipales = null;
            SolrDocumentList serviciosAdyacentes = null;
            QueryResponse responsePrin = null;
            QueryResponse responseAdya = null;
            query.set("fl", "service.name,score");
            String queryP = preparteStatementPrincipal(agileRE);
            String queryA = preparteStatementAdyacente(agileRE);
            try {
                    query.set("q", queryP);
                    responsePrin = solr.query(query);
                    serviciosPrincipales = responsePrin.getResults();
                    query.set("q", queryA);
                    responseAdya = solr.query(query);
                    serviciosAdyacentes = responseAdya.getResults();
            } catch (SolrServerException e) {
                 logger.error("Se ha producido un error al recuperar las entidades principales");
                 throw e;
            } catch (IOException e) {
                 logger.error("Se ha producido un error al recuperar las entidades principales");
                    throw e;
            }
            serviciosCadidatos = mergeServices(serviciosPrincipales, serviciosAdyacentes);
            return serviciosCadidatos;
    }

    /**
     * Preparación de la Query principal
     * @param agileRE
     * @return
```

```
	 */
	private String preparteStatementPrincipal(AgileRequirement agileRE) {
		String statement = new String();
		statement = "service.name: " + agileRE.getUserStoryName() + "~2^10 "
			  + "OR service.description: "
				+ agileRE.getUserStoryDescripcion()
				+ "^5 " + "OR operation.name: "
				+ agileRE.getUserStoryName()
				+ "~2^5  "
				+ "OR operation.description: "
				+ agileRE.getUserStoryDescripcion() + "^2,5 ";
		for (int i = 0; i < agileRE.getTask().length; i++) {
			statement += "OR operation.description: " + agileRE.getTask()[i] + "^1,25 ";
		}	}
	return statement;
	}
	/**
	 * Preparación de la Query adyacente
	 *
	 * @param agileRE
	 * @return
	 */
	private String preparteStatementAdyacente(AgileRequirement agileRE) {
		String statement = new String();
		statement = "tag.tag: " + agileRE.getEpicDescription()
			    + " OR domain.description: "
				+ agileRE.getEpicDescription()
				+ " OR stakeholder.name: "
				+ agileRE.getPersonaName() + "~1"
				+ " OR stakeholder.role: " + agileRE.getPersonaDescription();
		for (int i = 0; i < agileRE.getAptanceCriteria().length; i++) {
			statement += " OR policy.policy: "
				+ agileRE.getAptanceCriteria()[i];
			statement += " OR serviceLevelAgreement.kpi: "
				+ agileRE.getAptanceCriteria()[i];
		}
		for (int i = 0; i < agileRE.getDefinitionOfDone().length; i++) {
			statement += " OR policy.policy: "
				+ agileRE.getDefinitionOfDone()[i];
			statement += " OR serviceLevelAgreement.kpi: "
				+ agileRE.getDefinitionOfDone()[i];
```

```
		}
		return statement;
	}
	/**
	 * Devuelve la lista de servicios candidatos
	 * @param serviciosPrincipales
	 * @param serviciosAdyacentes
	 * @return
	 */
	private List<CandidateService> mergeServices(SolrDocumentList serviciosPrincipales,
			SolrDocumentList serviciosAdyacentes) {
		double medScore = 0f;
		List<CandidateService> servP = new ArrayList<CandidateService>();
		List<CandidateService> servA = new ArrayList<CandidateService>();
		List<CandidateService> servCadidatosFinal = new ArrayList<CandidateService>();
		servP = convertirLista(serviciosPrincipales,Constantes.FILTRO_REFRENCIA);
		servA = convertirLista(serviciosAdyacentes, 0f);
		for (int i = 0; i < servA.size(); i++) {
			CandidateService aux = servA.get(i);
			if (servP.contains(aux)) {
				int j = servP.indexOf(aux);
				double scoreAux = servP.get(j).getScore() + aux.getScore();
				servP.get(j).setScore(scoreAux);
			}
		}
		medScore = desviacionEstandar(servP);
		for (int i = 0; i < servP.size(); i++) {
			if (servP.get(i).getScore() >= medScore) {
				servCadidatosFinal.add(servP.get(i));
			}
		}
		return servCadidatosFinal;
	}
	/**
	 * Conversión del Resulset de Solar a una lista normal
	 *
	 * @param servicios
	 * @return
	 */
	private List<CandidateService> convertirLista(SolrDocumentList servicios, double corte) {
		List<CandidateService> aux = new ArrayList<CandidateService>();
```

```
            for (int i = 0; i < servicios.size(); i++) {
               CandidateService sc = new CandidateService();
              if (new Double(servicios.get(i).get("score").toString()).doubleValue() > corte){
                  sc.setTitle((String) servicios.get(i).get("service.name"));
                  sc.setScore(new Double(servicios.get(i).get("score").toString()).doubleValue());
                  aux.add(sc);
              }
            }
          return aux;
    }

    private double desviacionEstandar(List<CandidateService> lista) {
            int nServicios = lista.size();
            Collections.sort(lista, new Comparator<CandidateService>() {
                    @Override
                    public int compare(CandidateService a, CandidateService b) {
                            return new Double(b.getScore()).compareTo(new Double(a.getScore()));
                    }
            });
            double sumatoria = 0.0;
            double media = 0.0;
            double varianza = 0.0;
            double desviacion = 0.0;
            for (int i = 0; i < nServicios; i++) {
                    sumatoria += (double) lista.get(i).getScore();
            }
            media = sumatoria / (double) nServicios;
            for (int i = 0; i < nServicios; i++) {
                    double rango = 0.0;
                    if (lista.get(i).getScore() < media) {
                            rango = Math.pow(lista.get(i).getScore() - media, 2f);
                    }
                    varianza = varianza + rango;
            }
            varianza = varianza / 10f;
            desviacion = Math.sqrt(varianza);
            System.out.println("La desviación estándar es: " + desviacion);
            System.out.println("La media ponderada es: " + media);
            return media-desviacion;
    }
```

```
	private List<SolrInputDocument> cargaServicios() {
		List<SolrInputDocument> listaServicios = new ArrayList<SolrInputDocument>();
		List<ServicePortfolio> catalogo = recuperaDatosServicioEA();
		for (int i = 0; i < catalogo.size(); i++) {
			SolrInputDocument document = new SolrInputDocument();
			document = indexServiceDocument(catalogo.get(i));
			listaServicios.add(document);
		}
		return listaServicios;
	}

	private SolrInputDocument indexServiceDocument(ServicePortfolio sp) {
		SolrInputDocument document = new SolrInputDocument();
		document.addField("domain.description", sp.getDomainDescription());
		document.addField("service.name", sp.getServiceName());
		document.addField("service.description", sp.getServiceDescription());
		for (int i = 0; i < sp.getOperationName().length; i++) {
			document.addField("operation.name", sp.getOperationName()[i]);
		}
		for (int i = 0; i < sp.getOperationDescription().length; i++) {
			document.addField("operation.description", sp.getOperationDescription()[i]);
		}
		for (int i = 0; i < sp.getStakeholderName().length; i++) {
			document.addField("stakeholder.name", sp.getStakeholderName()[i]);
		}
		for (int i = 0; i < sp.getStakeholderRole().length; i++) {
			document.addField("stakeholder.role", sp.getStakeholderRole()[i]);
		}
		for (int i = 0; i < sp.getPolicyPolicy().length; i++) {
			document.addField("policy.policy", sp.getPolicyPolicy()[i]);
		}
		for (int i = 0; i < sp.getServiceLevelAgreementKpi().length; i++) {
		  document.addField("serviceLevelAgreement.kpi", sp.getServiceLevelAgreementKpi()[i]);
		}
		for (int i = 0; i < sp.getTagTag().length; i++) {
			document.addField("tag.tag", sp.getTagTag()[i]);
		}
		return document;
	}

	private List<ServicePortfolio> recuperaDatosServicioEA() {
```

```
                List<ServicePortfolio> servicesPortfoloio = new ArrayList<ServicePortfolio>();
                // Se cargan los servicios de la fuente.
                Return servicesPortfoloio;
}
```

2.3.10 Ejemplo de Clase Principal

Una vez expuesto el proyecto JAVA y para comprender mejor el funcionamiento de cada una de las clases, en la Figura VII.10 se muestra un ejemplo de ejecución en la que se tendrá modelado un requerimiento ágil. Se muestra por tanto como es invocado, desde el punto de vista de la ejecución, el algoritmo para el descubrimiento de Servicios Candidatos.

Expresión VII.10. Código fuente JAVA para la clase «Test».

```
package es.juntadeandalucia.ccul.soa.test;
import java.util.ArrayList;
import java.util.List;
import es.juntadeandalucia.ccul.soa.arq.AgileRequirement;
import es.juntadeandalucia.ccul.soa.arq.CandidateService;
import es.juntadeandalucia.ccul.soa.arq.ServiceLocator;
import es.juntadeandalucia.ccul.soa.bussines.DiscoveryServices;
import es.juntadeandalucia.ccul.soa.bussines.IDiscoveryServices;

/**
* @author jorge.sedeno
*
*/
public class Test {
        public static void main(String[] args) {

        List<CandidateService> serviciosCanidatos = null;
        IDiscoveryServices                dss                =                (IDiscoveryServices)
ServiceLocator.getInstance(DiscoveryServices.class);
                try {
                        dss.loadIndexer();
                } catch (Exception e) {
                        System.out.println("Error al realizar la indexación del catálogo");
                        e.printStackTrace();
                }
                List<AgileRequirement> productBacklog = new ArrayList<>();
                AgileRequirement agileRE1 = new AgileRequirement();
                agileRE1.setUserStoryName("Solicitud del Carnet de Lector");
```

```
		agileRE1.setUserStoryDescripcion("El ciudadano accede con su certificado digital a la aplicación y rellena un formulario con sus datos, este formulario se firma con certificado electrónico por el ciudadano y se almacena como lector");

		agileRE1.setPersonaName("Usuario de la Biblioteca");

		agileRE1.setPersonaDescription("Biblioteca de Andalucía");

		agileRE1.setEpicDescription("Solicitud de Carnet");

		agileRE1.setAptanceCriteria(new String[] { "Se obtiene una solicitud firmada electrónicamente con certificado digital", "Se comprueba que el lector está dado de alta en la BBDD"});


		agileRE1.setDefinitionofDone(new String[] { "El documento está firmado con un certificado válido" });

		agileRE1.setTask(new String[] { "El usuario rellena el formulario para la solicitud del carnet", "Se convierte ese formulario a una solicitud PDF y se firma con el certificado digital del usuario", "Se almacena la solicitud firmada en la custodia de documentos electrónicos firmados", "El lector se graba en la BBDD"});

		productBacklog.add(agileRE1);

		try {

		for (int i=0; i<productBacklog.size(); i++){

			AgileRequirement aux = productBacklog.get(i);

			System.out.println("*********************************************************");

			System.out.println("* UserStory "+(i+1)+" :"+aux.getUserStoryName()+" *");

			System.out.println("*********************************************************");

			serviciosCanidatos = dss.discoveryProccess(aux);

			for (int j=0; j< serviciosCanidatos.size(); j++) {

				System.out.println("Servicio Candidato: " + serviciosCanidatos.get(j).getTitle() + " [" + serviciosCanidatos.get(j).getScore() + "]");

			}

		}

		} catch (Exception e) {

			System.out.println("Error al realizar el descubrimiento de Servicios");

			e.printStackTrace();

		}

	}

}
```

3. Visión unitaria del Framework DS4aRE

Una vez que se han descrito en profundidad cada uno de los elementos que integran el framework DS4aRE, se mostrará mediante dos diagramas, uno estático y otro dinámico, cómo trabajan y se relacionan estos elementos entre sí, mostrando así la visión unitaria del framework propuesto para el descubrimiento de Servicios Candidatos a partir de requisitos modelados con las técnicas ágiles expuestas en anteriores capítulos.

En la Figura VII.4 se muestra cómo se relacionan los diferentes elementos que componen la arquitectura tecnológica del framework, de manera que se pueda obtener una visión estática del mismo.

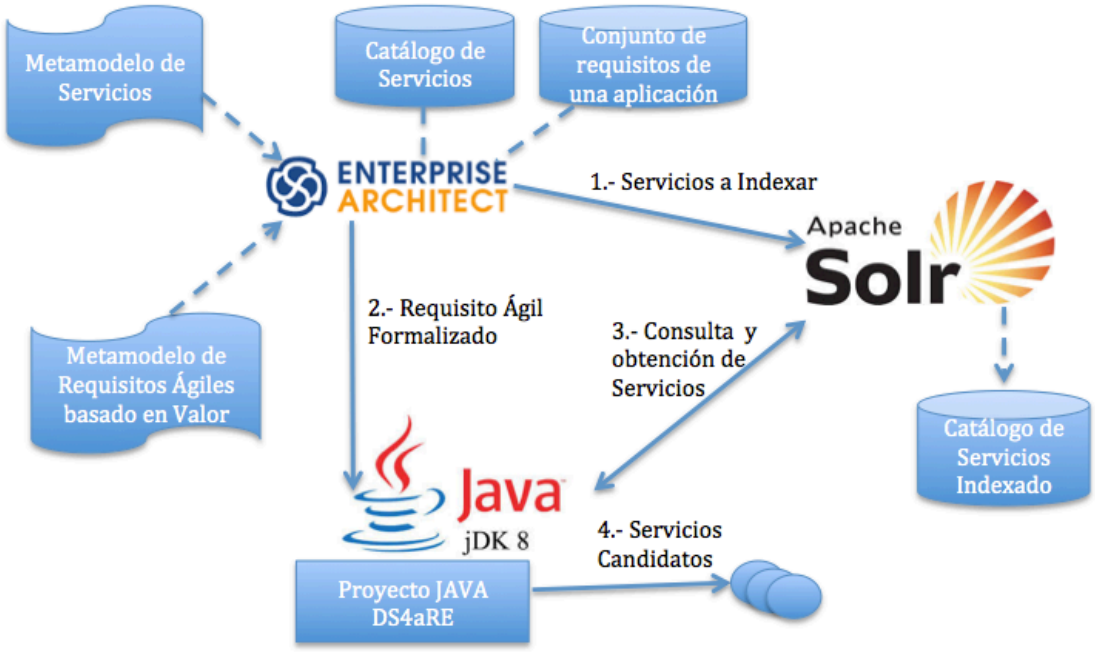

Figura VII.4. Arquitectura tecnológica del framework DS4aRE.

Como se puede observar en dicha figura:

- De Enterprise Architect se obtienen los XML con la información de los servicios que van a indexarse en Solr, es decir, en Enterprise Architect está la descripción funcional de los Servicios que será preparada para ser enviada a la herramienta Solr, con objeto de que esos servicios (que como se ha indicado anteriormente serán documentos para Solr) sean indexados en Solr para crear así el Catálogo de Servicios indexado y poder realizar búsquedas sobre ellos.
- Es desde el proyecto JAVA DS4aRE desde donde se obtiene la lista de Servicios Candidatos a través de las sucesivas llamadas a Solr. Una vez que se tiene modelado un requerimiento ágil, que estará recopilado en el Enterprise Architect, se obtendrá como salida la lista de Servicios Candidatos que podrían cubrir ese requisito.
- Es importante recalcar en esta figura que el Catálogo de Servicios está en Enterprise Architect y por tanto en Solr estará solamente el Catálogo de Servicios indexado (en tiempo de ejecución, cuando se necesite para la realizar la búsqueda y por tanto mientras se está ejecutando el proceso de descubrimiento de los Servicios Candidatos).
- Como se indicó en la descripción de la sección anterior sobre Solr, la indexación consiste en la creación de un índice de búsqueda con el contenido de los documentos, es decir, con el contenido de los Servicios que están descritos en Enterprise Architect.

Desde el punto de vista dinámico, tendremos cada uno de los elementos (Enterprise Architect, Solr y la proyecto DS4aRE) interactuando entre sí de manera coherente y sistemática. Para recalcar este relación en el tiempo de los elementos, en la Figura VII.5 se muestra un diagrama

de secuencia que nos indicará la ejecución de los pasos necesarios para la obtención de los Servicios Candidatos.

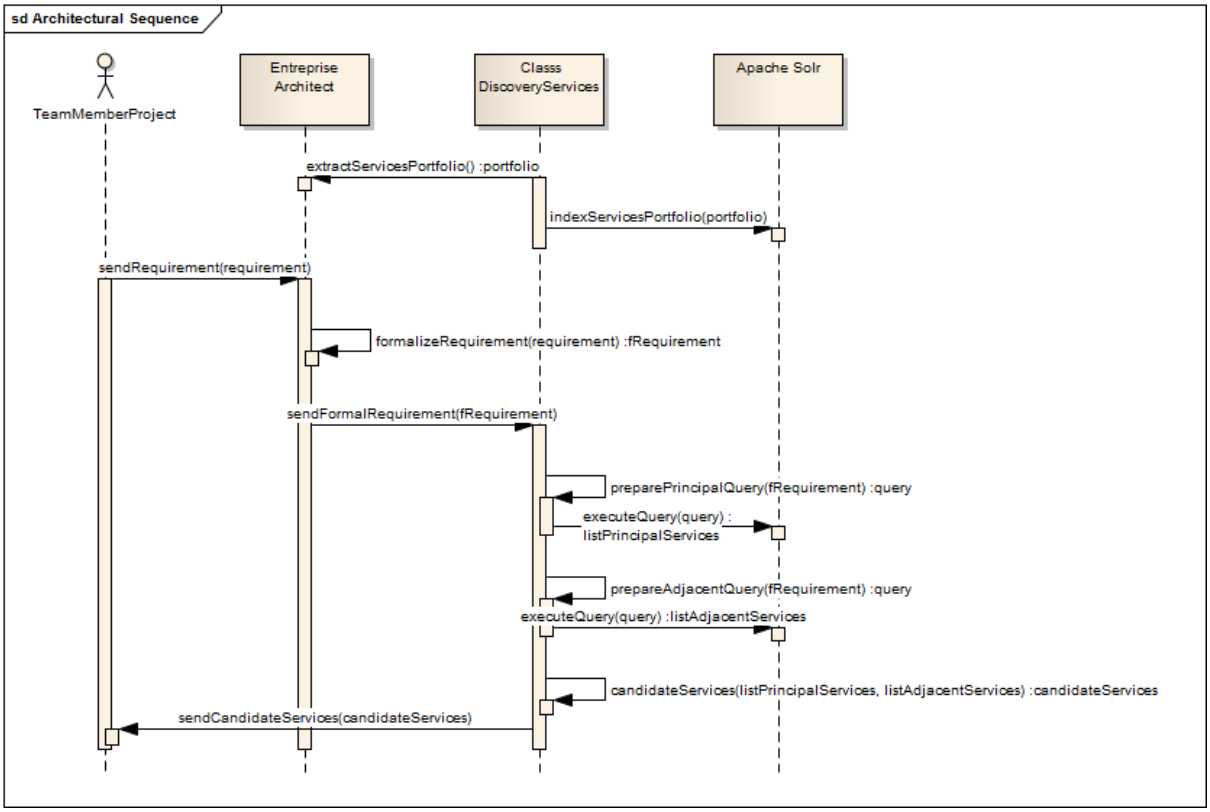

Figura VII.5. Diagrama de Secuencia del framework DS4aRE.

Estos pasos, desde el punto de vista técnico, se corresponde con la funcionalidad del Proyecto DS4aRE para el proceso de Servicios Candidatos implementado en la clase «DiscoveryServices» y son:

1. Proceso de indexación, que consistirá en extraer la información necesaria de los Servicios recogidos en Enterprise Architect e indexarlos en el herramienta Solr (es decir, indexar los Servicios para obtener el catálogo de servicios indexado en tiempo de ejecución). Se corresponde en la figura con:
    a. extractServicesPortfolio().
    b. indexServciesPortfolio(portfolio).
2. Proceso de formalización de requisitos ágiles, que consistirá en la formalización a través de las técnicas expuestas en este trabajo de investigación de un requisito y su formalización para ser usado en el proyecto DS4aRE y su posterior envío para iniciar el procedo de búsqueda o descubrimiento de Servicios Candidatos. Se corresponde en la figura con:
    a. sendRequirement(requirement).
    b. formalizeRequirement(requirement).
    c. sendFormalRequirement(fRequirement).
3. Proceso de consulta a Solr, que consistirá en el diseño de las consultas que se ejecutarán contra el repositorio de servicios indexados, a partir del requisito ágil que llega como entrada, obteniéndose así los servicios que hacen referencia a las entidades principales y a las entidades adyacentes. Se corresponde en la figura con:

    a. preparePrincipalQuery(fRequirement).
    b. executeQuery(query).
    c. prepareAdjacentQuery(fRequirement).
    d. executeQuery(query).
4. Proceso para la preparación de los Servicios Candidatos, que consistirá en la ejecución del algoritmo estadístico de las dos listas de Servicios anteriores para obtener definitivamente la lista de Servicios Candidatos para ese requerimiento ágil. Se corresponde en la figura con:
    a. canditateServices(listPrincipalServices, listAdjacentServices)
    b. sendCandidateServices(candidateServices)

Una vez expuesta la estructura estática del Framework DS4aRE y la forma en que trabajan de forma cohesionada en el tiempo sus diferentes elementos, queda descrito de forma completa y exhaustiva el proceso de descubrimiento de los Servicios Candidatos para un requisito ágil dado.

3.1. Grado de automatización del proceso de descubrimiento

Como se indicó en el capítulo anterior, desde el proceso completamente manual hasta esta propuesta para el descubrimiento de Servicios Candidatos se ha conseguido un grado de automatización muy elevado.

Como se han indicado a lo largo de la presente Tesis Doctoral, el diseño del Catálogo de Servicios no es parte del proceso de descubrimiento de Servicios Candidatos y aunque la presentación del metamodelo de Servicios aporte un cierto grado de automatización a dicha actividad, al no ser parte del proceso, no se tratará en este apartado. Así mismo, la decisión final de incluir dentro del desarrollo de la nueva aplicación un Servicio Candidato, tampoco está contabilizada en este apartado debido a que al contener otra serie de variables diferentes del aspecto funcional (que pueden ser de índole económica, política o estructural entre otras) queda fuera del proceso propuesto para este trabajo de investigación.

Con estos presupuestos, se describirá a través de la Tabla VII.2 el grado de automatización que se ha alcanzado con el framework DS4aRE comparando las actividades realizadas de forma manual con las realizadas por el proceso propuesto. En la primera columna se tendrá la descripción de la actividad manual, en la segunda el componente del framework que realizar esa actividad actualmente con la descripción de cómo se realiza esa actividad y en la tercera columna el grado de automatización alcanzado tomando la escala de valores:

- "MANUAL", cuando en el nuevo proceso no se ha podido automatizar esa actividad.
- "SEMIAUTOMÁTICO" cuando en el nuevo proceso haya un cierto grado de automatización o apoyo de herramientas que faciliten la tarea.
- "AUTOMÁTICO", cuando en el nuevo proceso se haya automatizado totalmente y de una forma desatendida dicha actividad manual.

Tabla VII.2. Grado de automatización del proceso de descubrimiento.

Actividad manual inicial	Framework DS4aRE	Grado Automatización
El responsable de los Servicios ha de estudiar todo el Catálogo de Servicios de la organización.	SOLR, Proyecto DS4aRE*: se indexan para la búsqueda todos los Servicios de la organización.*	AUTOMÁTICO

El responsable del proyecto de la nueva aplicación a desarrollar presenta los requisitos obtenidos al responsable de Servicios de la organización.	EA, Proyecto DS4aRE: *El responsable de proyecto modela los requisitos en EA a través del perfil UML propuesto y que será usado por el proyecto DS4aRE y formalizado para la búsqueda.*	SEMIAUTOMÁTICO
El responsable de los Servicios indica qué servicios podrían ser los Servicios Candidatos para un requisito dado.	Proyecto DS4aRE, SOLR: *Para el requisito formalizado, el proyecto DS4RE consulta SOLR y obtiene los Servicios Candidatos a través del tratamiento algorítmico propuesto.*	AUTOMÁTICO

Como se puede deducir de la Tabla VII.2, si se ha de ceñirse estrictamente al proceso de descubrimiento de Servicios en sí, el proceso es prácticamente automático ya que solamente se encuentra cierta acción manual en la formalización y modelado de los requisitos para lanzar el proceso de descubrimiento de Servicios propuesto.

Ya dentro del Capítulo IX se reflejará, dentro de los trabajos futuros, completar el grado de automatización de dicho proceso de descubrimiento para que los equipos de desarrollo no tengan interacción alguna con los Servicios de forma manual. Por otra parte se remarcará también, dentro de los trabajos futuros, tanto la integración dentro de este proceso de descubrimiento del análisis de los Servicios Candidatos a través de la ejecución automática de las políticas y las reglas necesarias, como la metodología para incluir estos servicios definitivos dentro del nuevo desarrollo.

4. Conclusiones

A lo largo de este capítulo se ha descrito el framework DS4aRE que da soporte al proceso de descubrimiento de los Servicios Candidatos para un requisito ágil dado. Así mismo en este capítulo se han descrito con cierto detalle los principales elementos que forman parte de la arquitectura tecnológica que da soporte a esta solución y se han aportado aquellas características, tanto intrínsecas como extrínsecas, de estas herramientas, que han sido relevantes para su elección.

Se ha aportado dentro de este capítulo el código fuente del proyecto JAVA DS4aRE así como un ejemplo para su ejecución y se ha descrito en profundidad cada una de las clases JAVA que intervienen, así como la relación coherente y sistemática, estructurada en el tiempo, entre los elementos de dicho framework. Por último, se ha mostrado el grado de automatización alcanzado por esta propuesta en comparación con el proceso manual, dato relevante para demostrar el ahorro de costes en una organización a la hora de la ejecución de este proceso.

En el capítulo siguiente se expondrá el contexto completo, es decir, la implantación y personalización de este proceso en una organización real en la que se ha validado este marco teórico con resultados positivos, constituyendo dicha implantación la validación del modelo teórico presentado en este trabajo de investigación.

Capítulo VIII Validación en un entorno real

En este capítulo se describirá la implantación en un entorno real del marco teórico expuesto en los anteriores capítulos, y que constituye la piedra de toque para la validación de dicha propuesta. Este entorno es la Consejería de Cultura de la Junta de Andalucía, organismo en el que se detectó el problema planteado en el Capítulo III.

Para la descripción de este entorno real se estructurará el capítulo en tres apartados. Primero, una introducción al propio organismo para conocer el contexto. En segundo lugar, los antecedentes, en los que se describirán los principales hitos a los que se ha ido enfrentando el organismo hasta la detección del problema planteado en esta Tesis Doctoral. Por último, la situación actual, describiendo los procesos de implantación y validación del modelo teórico expuesto en este trabajo de investigación.

El capítulo termina con unas conclusiones sobre la implantación de este modelo teórico para una organización concreta.

1. Introducción

Para enmarcar la Consejería de Cultura, organismo en donde ha sido realizada la validación del marco teórico propuesto, es necesario encuadrar primero la Junta de Andalucía, el gobierno de una región situada en el sur de España. La Junta de Andalucía es el nombre del gobierno autonómico de Andalucía que cuenta con más de 220.000 empleados y es uno de los principales actores económicos en dicha comunidad autónoma [Pérez et al. 2011].

Dentro de la Junta de Andalucía se encuentra la Consejería de Cultura, que tiene encomendada por parte del ejecutivo el desarrollo y coordinación de las políticas públicas en materia de Cultura, Patrimonio Histórico, Archivos, Cine, Música, Bibliotecas, Museos y otros bienes culturales de gran internes artístico y monumental, tanto materiales como inmateriales. También es responsable del apoyo económico e institucional a las industrias culturales, junto con otros actores públicos y privados. El Servicio de Informática, dentro de la consejería, tiene la responsabilidad de todas las políticas sobre tecnologías de la información. Es principalmente responsable de proveer toda la infraestructura de TI necesaria para el funcionamiento de los sistemas de información, así como desarrollar y mantener todos los servicios que se ofrecen a la ciudadanía dentro del denominado Gobierno Electrónico.

Es importante destacar que como las demandas de acceso a los servicios públicos por parte de los ciudadanos han aumentado drásticamente en los últimos años, cada administración ha puesto en marcha un proceso de administración electrónica [Ley 11/2007] [Ley 39/2015] [COM 743/2010] [COM 179/2016]. Debido al compromiso de la Consejería de Cultura con esta estrategia de gobierno electrónico, el número de sistemas bajo la responsabilidad de su Servicio de Informática se ha incrementado de forma exponencial en los últimos 6 años hasta contar con un parque de unos 100 Sistemas de Información y un Catálogo de Servicios de 80 servicios propios con aproximadamente 900 operaciones (o microservicios) identificadas en el mismo.

2. Antecedentes

Para entender de donde surge el problema planteado en la presente Tesis Doctoral es necesario realizar un somero itinerario por los antecedentes de la Consejería de Cultura en materia de Servicios, desarrollado en el Trabajo Fin de Master del autor [Sedeño 2012] y en otros artículos posteriores [Sedeño et al. 2013][Sedeño et al. 2014], así como la implantación de la metodología ágil descrita en el Capítulo III y plasmada en sucesivos artículos [Torrecilla et al. 2013a], [Torrecilla et al. 2013b], [Torrecilla et al. 2014] hasta su completa publicación a mediados de 2015 [Torrecilla et al. 2015].

Sobre este itinerario solamente se describirán con cierto detalle aquellos aspectos relevantes para la presente Tesis Doctoral, pues tienen un reflejo patente en los metamodelos propuestos en los capítulos IV y V.

2.1. La transformación (2008-2012)

A finales de 2008 se comienza en la Consejería de Cultura la transformación de dicho organismo mediante la metodología propuesta en el Capítulo III para adaptarlo a una organización capaz de operar bajo un paradigma SOA. A partir del contexto de negocio de la Consejería de Cultura, se comienza la transformación hasta adoptar el Modelo Objetivo SOA, instanciando cada uno de los elementos de dicho modelo.

El proceso de transformación no es objeto de la presente Tesis Doctoral, por lo que no se describirán los pormenores de dicho proceso ni los entregables intermedios, ya que fueron expuestos ampliamente en otros trabajos del autor [Sedeño 2012].

Como resultado de esta transformación se adopta en la Consejería de Cultura una arquitectura orienta a Servicios que permite a la misma desplegar los principales Servicios dentro del Gobierno Electrónico y a su vez usarlos para constituir un apoyo al desarrollo de los grandes sistemas de tramitación para el ciudadano, como son el Depósito Legal de Andalucía [Deposito 2008], el sistema de Gestión del Patrimonio Histórico de Andalucía, denominado Mosaico [Mosaico 2012] o el Sistema de Gestión para el Registro de la Propiedad Intelectual, llamado Ninfa [Ninfa 2009], entre otros.

Así mismo, se desarrollan y se implantan las políticas del Ciclo de Vida de los Servicios, de la que se hará una breve reseña al final de esta sección.

Dentro del Modelo Objetivo SOA implantado, se mostrarán a continuación qué elementos se instanciaron y de qué forma dentro de la Consejería de Cultura y qué importancia han tenido en el desarrollo del metamodelo de Servicios planteado en el Capítulo IV.

Por comenzar desde lo más general a lo más particular, el Modelo de Dominio de la Consejería de Cultura quedó fijado como se expone en las Figuras VIII.1, VIII.2 y VIII.3, cubriendo todas las áreas funcionales y técnicas del organismo y sirviendo como primera taxonomía para los Servicios.

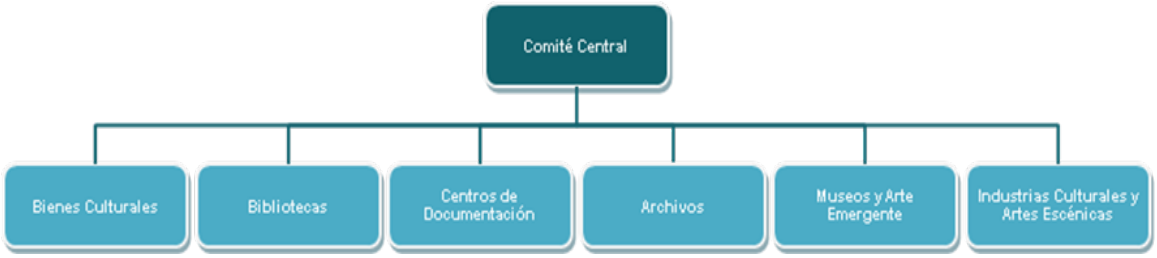

Figura VIII.1. Dominios Funcionales de Negocio.

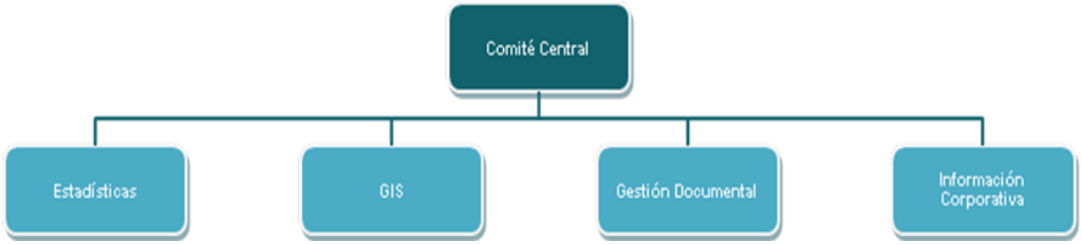

Figura VIII.2. Dominios Cross u Horizontales a la organización.

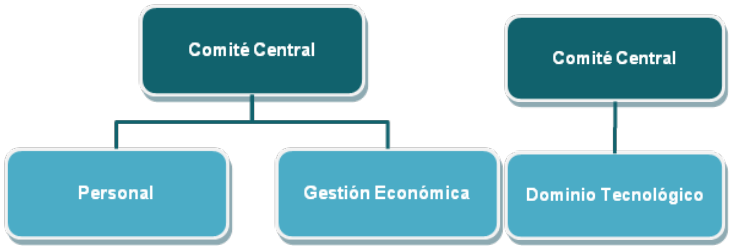

Figura VIII.3. Dominios de Soporte y Tecnológicos.

En estos dominios están recogidas todos las posibles agrupaciones de Servicios, asignándoles también una responsabilidad, pues cada Dominio tiene un responsable de Dominio y dependen todos de un Comité Central.

Avanzando en la instanciación del Modelo Objetivo SOA, se describió la ficha de Servicio, artefacto que recogía todas las características funcionales de un Servicio y que servía para describir cada uno de los Servicios dentro de cada Dominio.

Como se puede observar en la ficha que se muestra en la Tabla VIII.1, muchas de las entidades presentes en el metamodelo de Servicios, propuesto en el Capítulo IV, tienen su reflejo en la información recogida en ella. Así, la descripción del Servicio, las palabras claves, las entidades relacionadas, las operaciones o los responsables dentro de la matriz RACI, han servido como germen del metamodelo de Servicios propuesto.

A continuación, en dicha Tabla VIII.1 se puede observar la ficha de un servicio concreto ya instanciado como ejemplo. Es un servicio que pertenece a un Dominio Cross u Horizontal, concretamente al de Información Corporativa y es el servicio cuya funcionalidad está asociada a la gestión de las diferentes unidades organizativas de la consejería, desde el que se centraliza su gestión y cuya información es usada por todas las aplicaciones que necesiten la estructura orgánica de la consejería, tanto para exponerla como para gestionarla.

Tabla VIII.1. Ficha de un Servicio concreto.

Descripción conceptual del servicio. Dominio de Información Corporativa	
Cabecera	
Espacio	*es.juntadeandalucia.ccul.InformacionCorporativa*
Nombre	*Gestión de Unidades*
Descripciones	
Descripción	*Este servicio permite la gestión de la información de las Unidades Organizativas, entendidas como las entidades que componen la estructura de servicio de la Consejería de Cultura.*
Definición	*Este servicio permite la gestión de la información de las Unidades Organizativas, entendidas como las entidades que componen la estructura de servicio de la Consejería de Cultura. Permite la consulta y actualización de su información básica, así como la consulta de las dependencias existentes entre las distintas unidades.*
Taxonomía	*InformaciónCorporativa*
Semántica	*Unidad, Unidad organizativa, plantilla, horario, estructura organizativa,*
Entidades de Negocio	*UNIDADES*
Descripción lógica del servicio	
Tipificación técnica	*Tecnológico.Datos*
Interfaz	
Operaciones	• *Búsqueda de Unidades Organizativas.* • *Recuperar detalle de Unidad Organizativa.* • *Actualizar información básica de Unidad Organizativa* • *Dar de alta Unidad Organizativa.* • *Recuperar Horario de Unidad Organizativa.* • *Recuperar Árbol de dependencias.*
Metainformación	
RACI	
Responsable (Responsible)	*Responsable Técnico del Dominio "Información Corporativa" (a definir por la Consejería de Cultura).*
Dueño (Accountable)	*Responsable del Dominio "Información Corporativa" (a definir por la Consejería de Cultura).*

Cada Servicio además, tenía unas políticas que se aplicaban en diversos momentos de su Ciclo de Vida. De estas políticas, aunque no son parte del alcance de esta Tesis Doctoral, se estima necesario al menos explicar dos de ellas ya que su ejecución está prevista en el tiempo después de la identificación de los Servicios Candidatos. Estas políticas entran a formar parte de las Políticas de Dominio (Gestión de los Servicios) y son:

- Uso de un Servicio.
- Modificación / Versionado de un Servicio.

Uso de Servicio. Este procedimiento es invocado cuando se detecta que un Servicio existente dentro del repositorio va a ser reutilizado, es decir, cuando esté dentro del conjunto de Servicios Candidatos.

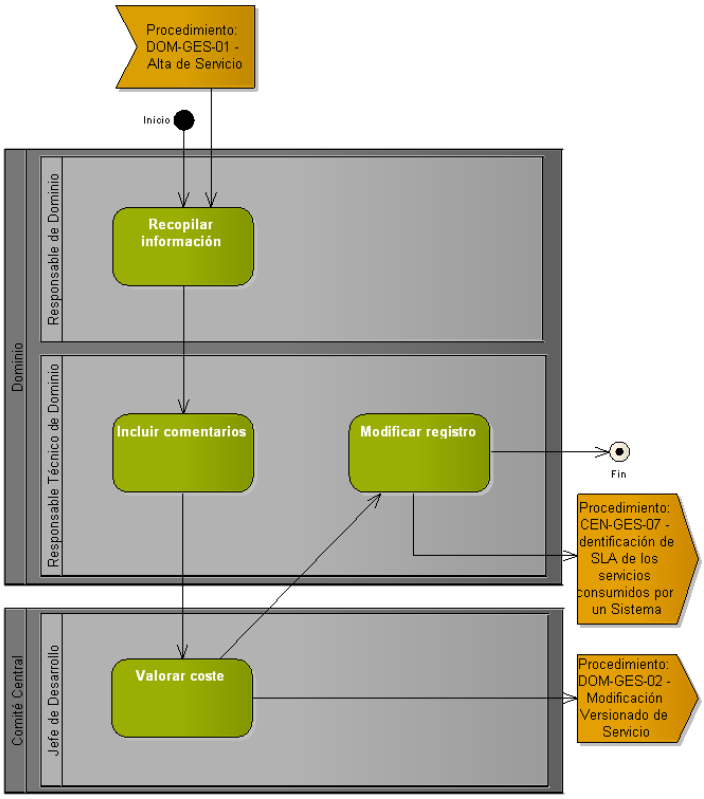

Figura VIII.4. Procedimiento para el uso de un Servicio.

Las actividades que componen este flujo de trabajo son:

- **Recopilar información:** el Responsable de Dominio recopilará la información relativa del nuevo consumidor del servicio. Se deberá identificar la siguiente información:
  - Funcionalidad existente utilizada del servicio.
  - Modificaciones necesarias a realizar por parte del servicio para cubrir la funcionalidad requerida por el nuevo consumidor.
- **Incluir comentarios:** el Responsable Técnico de Dominio, tras estudiar la solicitud, debe aprobar e indicar los comentarios que estime oportunos sobre la información facilitada para el uso del servicio.

Valorar coste: el Jefe de Desarrollo revisará los comentarios realizados por el Responsable Técnico de Dominio y será el responsable de valorar el coste del uso del servicio por el nuevo consumidor. Para ello se tendrán en cuenta los siguientes puntos:

- Necesidad de modificación del servicio: en este caso de invocará al procedimiento "Modificación / Versionado de Servicio".
- Necesidad de pruebas de regresión: el nuevo consumidor incrementa el número de peticiones al servicio, corriendo el riesgo de disminuir los SLA del servicio, por lo que será necesario realizar pruebas de sistemas del servicio.

- Modificar registro: una vez realizado el coste asociado al uso del servicio por el nuevo consumidor, el Responsable Técnico de Dominio modificará el registro del servicio para incluir al nuevo consumidor.

Los roles participantes son:

- **Responsable de Dominio:** se encarga de recibir la solicitud de uso del servicio y de recopilar la información necesaria para la valoración del coste asociado a que un nuevo consumidor quisiese utilizar el servicio.
    - Recopilar información
- **Responsable Técnico de Dominio:** se encargará de estudiar la solicitud realizada de uso del servicio, añadiendo los comentarios que estime oportunos. Una vez valorado el coste del uso del servicio, se encargará de modificar el registro para incluir al nuevo consumidor del servicio.
    - Incluir comentarios
    - Modificar registro
- **Jefe de Desarrollo:** se encargará de valorar el coste asociado al uso del servicio por un nuevo consumidor.
    - Valorar coste

Modificación / Versionado de un Servicio. Este procedimiento se invoca cuando un responsable notifica que va a utilizar un servicio existente, sin embargo, tras un análisis preliminar, son necesarias modificaciones en el mismo para que se adecue al nuevo consumidor del servicio.

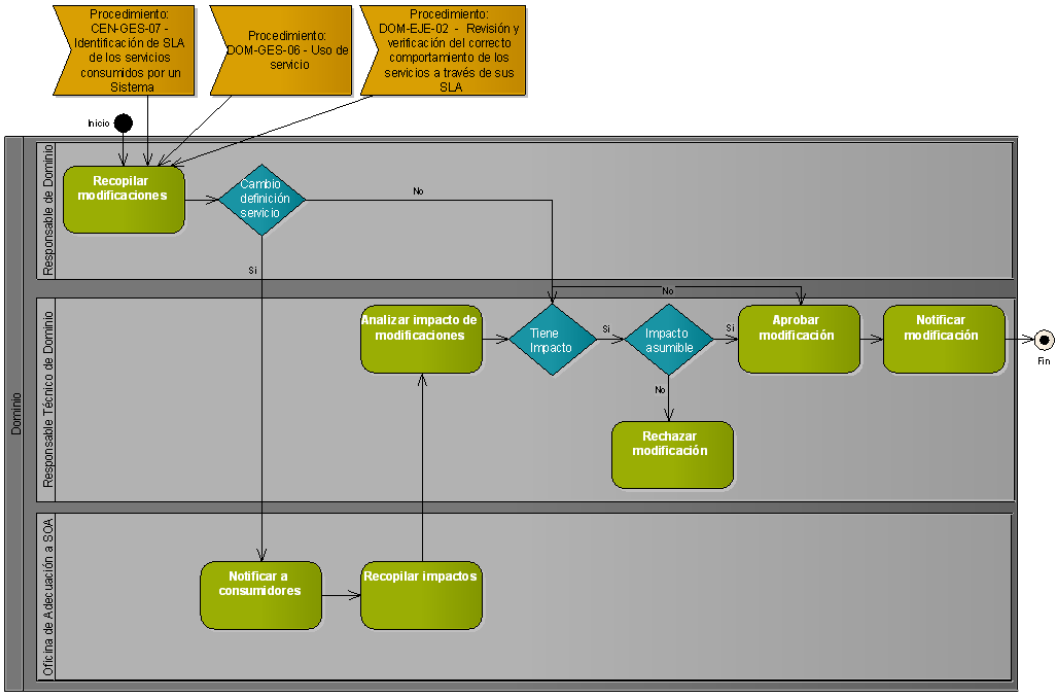

Figura VIII.5. Procedimiento para la modificación / versionado de un Servicio.

No se describirá este segundo procedimiento de la política ya que es auto explicativo por sí mismo, pero es importante recalcar que el hecho de que un Servicio se presente como un

Servicio Candidato y funcionalmente se adapte, no implica que sea utilizado en el desarrollo ya que por motivos, por ejemplo económicos o de saturación, podría ser contraproducente su uso.

Además, esto requiere un análisis no automático por parte de los responsables del procedimiento y otros actores, lo que imposibilita una automatización completa sin concurso humano.

Llegado a este punto, el Catálogo de Servicios de la organización estuvo formado pues por el conjunto, agrupado por Dominios, de cada una de las fichas de Servicio. En el metamodelo de Servicios propuesto, existe una relación entre Servicios y Políticas que no se da en esta fase de la transformación, de manera que el metamodelo propuesto aumenta las características de trazabilidad dentro del Modelo Objetivo SOA instanciado en la Consejería de Cultura.

Dentro de este Modelo Objetivo SOA, una de las piezas fundamentales es la referida a la metodología de desarrollo y especialmente en qué manera se integra en el desarrollo de nuevas aplicaciones, los servicios y su gobierno.

Durante estos años en la Consejería de Cultura, se venía utilizando la metodología de desarrollo NDT ("*Navigational Development Techniques*") [Escalona et. al 2008], metodología encuadrada dentro de la Ingeniería Web Guiada por Modelos y pensada para el desarrollo de aplicaciones Web. Se definieron dos aspecto referentes a la integración con los Servicios:

- En primer lugar la ampliación de las actividades dentro del Ciclo de Vida del desarrollo, especialmente en las fases de la Ingeniería de Requisitos, el Análisis y Diseño Detallado, pero con el enfoque encaminado a la creación de Servicios a partir de nuevas funcionalidades, sin hacer hincapié en la identificación de los Servicios existentes que cubrían parte de ese desarrollo. La razón fundamental era que en ese momento del tiempo, no había demasiados Servicios y por lo tanto el foco y el esfuerzo estaban puestos en el desarrollo de dichos servicios y no es su reutilización.
- En segundo lugar se diseñó un repositorio en Enterprise Architect donde se iban almacenado y modelando los Servicios educidos durante el desarrollo para su posterior reutilización. Este repositorio es una de las razones fundamentales por las que se utilizó el Enterprise Architect, ya que estaba implantado en la Consejería de Cultura y los equipos de desarrollo tenían conocimiento sobre su uso. En ese momento, los Servicios no se modelaban conforme al metamodelo propuesto, sino que se representaban con un artefacto *Interface* nativo de UML para modelar ese Servicio y poder así reutilizarlo en el modelado de las aplicaciones con NDT. Por lo tanto la información del Servicio estaba documentada solamente en papel. La estructura de este repositorio se puede observar en la Figura VIII.6

Figura VIII.6. Repositorio de Servicios en conexión con los Sistemas de Información.

La búsqueda de Servicios dentro del repositorio era una búsqueda manual, hecha sobre la documentación, de manera que no existía ningún proceso para el descubrimiento de Servicios dada una funcionalidad concreta.

Como se ha podido ver a lo largo de este apartado, los elementos resultantes de la transformación de la Consejería de Cultura así como los procesos vistos derivan en dos conclusiones que afectan a este trabajo de Tesis Doctoral:

- En primer lugar, marcan el origen de la creación del metamodelo de Servicios, dotando de semántica a cada una de las entidades que hay provistas en él y se propone por tanto una formalización teórica para agrupar e interconectar toda la información de los Servicios, con un perfil UML completo para su modelado y la posibilidad de automatizar procesos al estar definidos de la misma forma en un único repositorio.
- En segundo lugar, presenta uno de los puntos de partida del problema planteado en esta Tesis Doctoral, el crecimiento de los Servicios en una organización sin un procedimiento sistemático de descubrimiento de los mismos (fuera de una búsqueda manual por parte de los equipos de desarrollo) y por tanto el alto de grado de dificultad para la reutilización de la funcionalidad que ya existe en el contexto.

Estos dos puntos expuestos anteriormente y la manera de resolverlos mediante el marco teórico propuesto en esta Tesis Doctoral son considerados como una aportación nueva al estado del arte en el descubrimiento de Servicios que se justificará en el capítulo siguiente.

2.2. El uso de metodologías ágiles (2012-2015)

A partir del año 2012 se comenzó en la Consejería de Cultura la adopción de una nueva metodología de desarrollo basada en metodologías ágiles, combinando el uso de "*Scrum*" con diversas técnicas, como EVM *("Earned Value Management")* o técnica para la gestión del valor ganado, con el objeto de hacer más eficiente el desarrollo de aplicaciones Web así como

extender a la gestión de proyectos (fuera del ámbito del desarrollo) dichos conceptos metodológicos.

Esta metodología se enfocó en un primer momento al desarrollo de aplicaciones de Administración Electrónica dentro del Gobierno Electrónico ("*e-government*") que dieron un excelente resultado, ajustando las estimaciones iniciales con los resultados finales [Torrecilla et al. 2013ab] y se extendió a otros ámbitos de TI, como los relacionados con la infraestructura y la estrategia, que también dieron buenos resultados debido a la precisión de las estimaciones con respecto al coste final [Torrecilla et. al 2014]. Esto derivó en la creación de una metodología que integraba, dándoles cohesión, diversas técnicas y procesos ágiles y que estaba enfocada a la estimación, planificación y gestión de proyectos. Esta metodología fue publicada en la revista *"Information and Software Technology"* [Torrecilla et al. 2015].

Esta metodología es la que se ha expuesto en el Capítulo III de esta Tesis Doctoral y es la que actualmente ha adoptado la Consejería de Cultura para sus desarrollos ágiles, tanto internos como con los proveedores con quienes trabaja.

Básicamente y por resumir este apartado y encajarlo dentro del contexto de esta Tesis Doctoral, a la hora de comenzar el proyecto, se realiza una primera iteración llamada "*inception*" donde se lanza el proyecto y que tiene como principal entregable el "*Product Backlog*" o pila de producto mencionado a lo largo de este trabajo de investigación.

Una vez que se comienza cada iteración, empezando por la primera, gestionando la pila de producto a partir del ROI y eligiendo las Historias de Usuario que entrarán a formar parte de esa iteración *("Sprint"),* desglosándolas en tareas y estimándolas (lo que se conoce en la metodología como el *"Sprint Planning")* es donde la Consejería de Cultura encontraba el problema para conocer de manera automática si había ya en la propia organización (el contexto) Servicios que cubriesen esa funcionalidad que se deseaba implementar.

Es por ello que se requería una nueva actividad para integrar, dentro de la metodología de desarrollo ágil de la Consejería, el proceso para el descubrimiento de los Servicios Candidatos.

Para posibilitar esto, desde el marco teórico de este trabajo de Tesis Doctoral se necesitaba formalizar esas Historias de Usuario a través de un metamodelo que recogiese la información contenida de los dos artefactos ágiles ya mencionados, el "*Product Backlog*" (que se crea una vez y se revisa antes de comenzar cada Sprint y que contiene la descripción de alto nivel de las Historias de Usuario para todo el proyecto) y el "*Sprint Backlog*" (que contiene el desglose en taras de cada Historia de Usuario y que se va creando en cada Sprint y cuya vida va asociada a la duración de dicha iteración).

Como se puede observar, esto representa, junto con lo expuesto en el apartado anterior, la completitud del problema expuesto en el Capítulo III, el descubrimiento de Servicios Candidatos usando metodologías ágiles en orden a cubrir una serie de requisitos.

Así mismo y dentro de las aportaciones que realiza este trabajo de Tesis Doctoral al estado del arte, destaca la formalización de un requisito ágil en orden a participar en otros procesos que no forman parte de la metodología de desarrollo en sí y que permite integrarlo, de manera formal, en otros procesos, en nuestro caso el de descubrimiento de los Servicios Candidatos. En el capítulo siguiente se tratará esta aportación con más detalle debido a los trabajos futuros que se pueden derivar de ella.

3. La Situación Actual

En esta sección, y una vez expuestos los antecedentes, se describirá cómo se ha implantado el marco teórico propuesto en esta Tesis Doctoral y cómo se ha llevado a cabo la adopción de la arquitectura tecnológica, que como se explicaba en el Capítulo VII, forma parte intrínsecamente de la misma, en la Consejería de Cultura.

3.1. Implantación del marco teórico y del framework DS4aRE

Llegados a este punto y habiendo descrito el problema concreto del organismo en cuestión, se comenzó a implantar a finales de 2015 la solución teórica aportada en el marco de la presente Tesis Doctoral, partiendo de las carencias expuestas en la sección anterior:

- No tener un Sistema de Información preparado para la consulta de la funcionalidad de los Servicios de la organización, debido a que dicha funcionalidad se encontraba desnormalizada y en formato papel, siendo ésta la única manera de obtenerla.
- La falta de un proceso sistemático para la consulta de la información sobre los Servicios existentes ante un requisito dado.
- La falta de un proceso que permitiese de forma sistemática la integración de los requisitos ágiles (Historias de Usuario) con otros procesos, debido a que no existía una descripción formal del mismo en ningún lenguaje de modelado.

Para ello se realizaron dos pasos fundamentales que quedaron como sustrato del presente proceso de descubrimiento de Servicios.

- La creación del repositorio de Servicios con la descripción funcional de los mismos modelada y preparada para poder ser indexada y por tanto realizar búsquedas sobre ella.
- El desarrollo de un proceso JAVA que implementaría dicho algoritmo de búsqueda ante un requisito dado y que constituiría el proceso para el descubrimiento de Servicios.
- La inclusión de una actividad dentro de la metodología ágil con objeto de modelar la Historia de Usuario y poder ejecutar el proceso antes mencionado.

La Consejería de Cultura tenía amplia experiencia en el trabajo con contenidos indexados, debido fundamentalmente a la gran cantidad de portales Web que desarrolla y debido a la necesidad de búsquedas dentro de los tramitadores de expedientes que forman parte de la Administración Electrónica. Para ello y de forma corporativa utilizaba Solr, basado en Lucene y descrito en el capítulo anterior.

Por tanto como primer paso, se modelaron los Servicios en Enterprise Architect, una vez cargado en dicha herramienta el perfil UML propuesto junto con el metamodelo de Servicios y se preparó el motor de búsqueda para indexar dicha información (de carácter textual y en lenguaje castellano).

Como segundo paso se preparó el proceso de indexación para lo que se realizó la parametrización del motor de búsqueda e indexación. Para ello:

1. Se desarrolló un tipo de dato denominado "*texto_es_ccul*", que se muestra en la Expresión VIII.1, que extendía del tipo genérico "*text_es*" y que permitía por cada campo de tipo "*texto_es_ccul*" lo siguiente:
    a. Se adaptaba al lenguaje castellano.

b. Eliminaba las partículas de las descripciones que no tenían valor semántico.
c. Creaba términos a partir de palabras compuestas como operaciones, cuando había cambio de minúsculas a mayúsculas o viceversa y diversos separadores.
d. Generaba los términos ("*tokens*") necesarios a partir de la configuración de los analizadores mencionados anteriormente.

Expresión VIII.1. Configuración del tipo de dato text_es_ccul.

```
<fieldType name="text_es_ccul" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.WordDelimiterFilterFactory"
catenateNumbers="1" generateNumberParts="1" splitOnCaseChange="1"
generateWordParts="1" preserveOriginal="1" catenateAll="0"
catenateWords="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory"
format="snowball" words="lang/stopwords_es.txt" ignoreCase="true"/>
      <filter class="solr.SpanishLightStemFilterFactory"/>
    </analyzer>
  </fieldType>
```

2. Dentro del esquema de campos en Solr, se definieron cada uno de los tipos de datos que entrarían a formar parte de las consultas, estos campos se muestran en Expresión VIII.2. Estos campos están acordes con el metamodelo de Servicios y constituyen un parte del mismo.

Expresión VIII.2. Campos que forman parte del documento Servicio.

```
<field name="domain.description" type="text_es_ccul" indexed="true"
  required="true" stored="true"/>
<field name="operation.description" type="text_es_ccul" indexed="true"
  stored="true"/>
<field name="operation.name" type="text_es_ccul" multiValued="true"
  indexed="true" required="false" stored="true"/>
<field name="policy.policy" type="text_es_ccul"
  multiValued="true" indexed="true" stored="true"/>
<field name="service.description" type="text_es_ccul" indexed="true"
  required="true" stored="true"/>
<field name="service.name" type="text_es_ccul" indexed="true"
  required="true" stored="true"/>
<field name="serviceLevelAgreement.kpi" type="text_es_ccul"
  multiValued="true" indexed="true" required="false" stored="true"/>
<field name="stakeholder.name" type="text_es_ccul" indexed="true"
  required="true" stored="true"/>
<field name="stakeholder.role" type="text_es_ccul" indexed="true"
```

```
  required="true" stored="true"/>
```

Se creó un proceso para cargar (indexar) la información sobre los Servicios contenida en Enterprise Architect (esta rutina se puede personalizar para cada entorno).

Solamente se diseñaron en el motor de indexación, como se muestra en la Expresión VIII.3, aquellos campos necesarios para la búsqueda, por lo que es Enterprise Architect quien alberga toda la información de los Servicios y por tanto constituye en última instancia el Catálogo de Servicios de la organización.

Expresión VIII.3. Ejemplo de documento de JSON para la indexación de cada Servicio.

```
{
        “domain.description”:”Nombre del Dominio”,
        “service.name”:”Nombre del Servicio”,
        “service.description”:”Descripción del Servicio”,
        “operation.name”:”Operacion1”,
        “operation.description”:”Descripción de la operación 1”,
        “operation.name”:”Operacion2”,
        “operation.description”:”Descripción de la operación 2”,
        “stakeholder.name”:”Nombre del Interesado 1”,
        “stakeholder.role”:”Rol del interesado 1”,
        “stakeholder.name”:”Nombre del Interesado 2”,
        “stakeholder.role”: ”Rol del interesado 2”,
        “serviceLevelAgreement.kpi”:”Indicador Clave de Rendimiento 1”,
        “serviceLevelAgreement.kpi”:” Indicador Clave de Rendimiento 2”,
        “plociy.policy”:”Política 1”,
        “plociy.policy”:”Política 1”,
        “tag.tag”:”tag 1”,
        “tag.tag”:”tag 2”,
}
```

De esta manera, de cada consulta que se lanza a Solr, se recupera el título del Servicio y la puntuación (“score”) facilitada por Solr en la respuesta, la estructura de la respuesta, que es utilizada en el proceso de descubrimiento de los Servicios Candidatos, se describe en la Expresión VIII.4.

Expresión VIII.4. Estructura de la respuesta en JSON que devuelve Solr al lanzar una Query.

```
{
  "responseHeader":{
    "status":0,
    "QTime":2,
    "params":{
      "q":"*:*",
      "indent":"on",
      "fl":"service.name,score",
      "wt":"json",
```

```
      "_":"1472032002050"}},
  "response":{"numFound":1,"start":0,"maxScore":1.0,"docs":[
      {
        "service.name":"Nombre del Servicio",
        "score":1.0}]
  }
}
```

Por último se ajustaron los parámetros de los algoritmos y de las consultas para adaptarlos a la Consejería de Culturas, tomando el valor de corte FILTRO_REFERENCIA el valor 10, la función de probabilidad la desviación estándar y los valores de impulso y distancia los expuestos en las consultas presentadas en los anteriores capítulos.

3.2. Validación de la solución

Como se ha indicado anteriormente, el modelo se refinó en base a la experiencia, por lo que una vez indexados los Servicios, se realizó la ejecución del proceso de descubrimiento de Servicios Candidatos con el framework DS4aRE, tomando como entrada el conjunto de requisitos de las necesidades de Gobierno Electrónico de los tres Sistemas de Información ya construidos y mencionados anteriormente (*NINFA*, *DEPÓSITO LEGAL* y *MOSAICO*), obteniéndose para ese conjunto de requisitos los servicios candidatos y realizando de forma paralela dicho proceso de forma manual a través del estudio la documentación para obtener también los servicios candidatos sin usar el framework DS4aRE. Por último, como estos sistemas estaban ya construidos se obtuvo de forma inmediata los servicios que estaban implementados ya en dicho código.

Los resultados de estudio se recogen en la Tabla VIII.2, cuya semántica es la siguiente:

- La primera columna denominada Número de requisitos del Sistema de Información muestra el número de requisitos de Administración Electrónica de cada uno de los Sistemas de Información. No se describirán estos requisitos de forma textual ya que resultaría innecesario para entender el proceso.
- La segunda columna denominada Número de Servicios Usados, indicaba el número de servicios realmente usados en cada uno de los Sistema de Información.
- La tercera columna denominada Número de Servicios Candidatos Manuales, indica el número de Servicios Candidatos que se han descubierto estudiando la documentación que se tiene sobre los servicios de modo manual.
- La cuarta columna denominada Número de Servicios Candidatos DS4aRE, indica el número de Servicios Candidatos que han sido descubiertos mediante la ejecución del nuevo proceso de descubrimiento de Servicios Candidatos usando el framework DS4aRE.

Tabla VIII.2. Resultados con los principales sistemas construidos.

Nº de requisitos del Sistema de Información	Nº Servicios Usados	Nº Servicios Candidatos Manuales	Nº Servicios Candidatos DS4aRE
NINFA [11 requisitos de administración	[7] Fachada DSS, Firma DSS, Registro	[11] Fachada DSS, Firma DSS, Firma	[11] Fachada DSS, Firma DSS, Firma

electrónica]	Telemático, Notificaciones Temáticas, Pago de Tasas, Sellado de Documentos, Generación de Informes de Firma	Delegada DSS, Registro Telemático, Notificaciones Temáticas, Pago de Tasas, Envío de Mail, Supresión de Certificados en Soporte Papel, Sellado de Documentos, Generación de Informes de Firma, Envío a Portafirmas.	Delegada DSS, Registro Telemático, Notificaciones Temáticas, Pago de Tasas, Envío de Mail, Supresión de Certificados en Soporte Papel, Sellado de Documentos, Generación de Informes de Firma, Envío a Portafirmas.
DEPÓSITO LEGAL [8 requisitos de administración electrónica]	[7] Fachada DSS, Firma DSS, Registro Telemático, Notificaciones Temáticas, Sellado de Documentos, Generación de Informes de Firma	[8] Fachada DSS, Firma DSS, Firma Delegada DSS, Registro Telemático, Envío de Mail Notificaciones Temáticas, Sellado de Documentos, Generación de Informes de Firma.	[8] Fachada DSS, Firma DSS, Firma Delegada DSS, Registro Telemático, Notificaciones Temáticas, Sellado de Documentos, Envío de Mail, Generación de Informes de Firma.
MOSAICO [15 requisitos de administración electrónica]	[5] Autenticación con Fachada DSS, Registro Telemático, Notificaciones Temáticas, Sellado de Documentos, Generación de Informes de Firma	[9] Fachada DSS, Firma DSS, Registro Telemático, Notificaciones Temáticas, Envío de Mail, Supresión de Certificados en Soporte Papel, Sellado de Documentos, Generación de Informes de Firma, Envío a Portafirmas	[9] Fachada DSS, Firma DSS, Registro Telemático, Notificaciones Temáticas, Envío de Mail, Supresión de Certificados en Soporte Papel, Sellado de Documentos, Generación de Informes de Firma, Envío a Portafirmas

Sobre los resultados de esta tabla se pueden sacar las siguientes conclusiones:

- El número de Servicios Candidatos obtenidos con el framework DS4aRE es el mismo que realizando un estudio exhaustivo de los requisitos y del Catálogo de Servicios, lo que nos indica dos aspectos:
    - El proceso propuesto cumple su propósito ya que obtiene los resultados teóricos que debe obtener. El número de Servicios Candidatos que se pueden usar realizando el proceso manual es el mismo que usando el proceso propuesto.
    - El proceso propuesto tiene un coste ínfimo comparado con el proceso manual, ya que las tareas que se han automatizado de forma total o parcial, como se vio en el capítulo anterior, son todas las que entran a formar parte del estudio. Lógicamente, el coste es completamente diferente, siendo el proceso de descubrimiento de Servicios mucho más eficiente, dado que tarda unos milisegundos en comparación con las jornadas de trabajo necesarias para que los consultores puedan realizar el descubrimiento manual.
- En cuanto a los Servicios Usados, estos sistemas son más antiguos que el propio proceso de descubrimiento y que algunos de los propios Servicios actuales, por lo que lógicamente no todos los servicios pudieron integrarse, sin embargo, independientemente de lo realizado en el diseño de estos Sistemas de Información (que

son de la época de la descrita transformación de este organismo para operar bajo el paradigma SOA), sí se observa que alguno de los servicios existentes en aquella época no han sido integrados por fallos en el descubrimiento de los mismos.

Esta prueba se repitió con otra aplicación más, la Tarjeta de Usuario de la Red de Bibliotecas Públicas de Andalucía [Turbo 2016].

Esta aplicación llevaba en producción desde el 25 de enero de 2009, pero en el año 2016 se sometió a una reingeniería para adaptarlo a la nueva normativa de Administración Electrónica, para lo que se usó el proceso de descubrimiento de Servicios antes mencionado y soportado por el framework DS4aRE, ya que al cambiar el procedimiento, había que cambiar los requisitos de Administración Electrónica de la aplicación.

Para ello al realizar el proceso completo, se mostrará en este trabajo de investigación cada uno de los requisitos, las consultas a Solr y el tratamiento de los resultados obtenidos.

Para poder seguir el proceso, además de la descripción textual, se muestra de nuevo el diagrama de secuencia del proceso de descubrimiento de Servicios Candidatos (Figura VIII.7) a fin de recalcar de forma gráfica los pasos descritos a continuación.

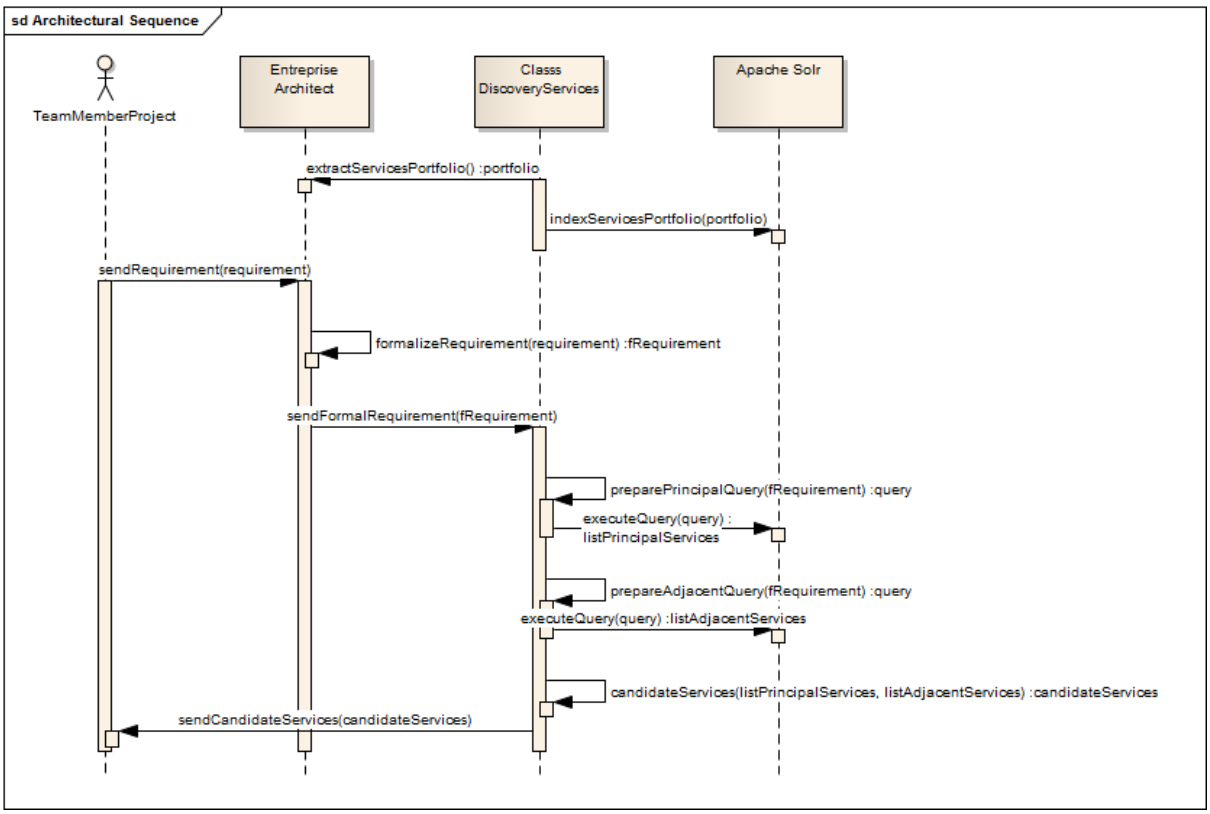

Figura VIII.7. Secuencia del proceso de descubrimiento de Servicios Candidatos.

En primer lugar, y una vez que se han indexado los Servicios en Solr, se muestra en la Tabla VIII.3 un ejemplo de la información de un requisito ágil basado en Valor una vez conformado con el metamodelo.

Tabla VIII.3. Instancia de un requisito ágil basado en Valor.

Entidad	Atributo	Valor
UserStrory	name	*Envío de Notificaciones Telemáticas*
	description	*Enviar una notificación telemática a los ciudadanos con la resolución firmada y sellada de su expediente administrativo*
Persona	name	*Nombre del responsable de las Bibliotecas*
	description	*Biblioteca de Andalucía*
Epic	*description*	*Notificaciones*
AceptanceCriteria	*assert*	*{ "La notificación enviada ha sido leída por el ciudadano o han pasado 10 días sin ser leída" }*
DefinitionOfDone	*fact*	*{ "El código de notificación está dado de alta en Hacienda", "El código fuente está en Subversion" }*
Task	*description*	*{ "Se da de alta como abonado al ciudadano", "Se envía a Portafirmas la resolución", "Se sella la resolución con el sistemas Sell@", "Se notifica de forma fehaciente la resolución de forma telemática", "El ciudadano lee la notificación", "El sistema comprueba si se ha leído la notificación", "Si no la ha leído, se publica en BOJA" }*

A continuación se muestra la consulta en formato JSON que será lanzada para las entidades principales con la información obtenida del metamodelo anterior. Esta consulta tendrá la estructura indicada en la Expresión VIII.5 y devolverá una lista de servicios con el título y el score indicados en la Expresión VIII.6.

Expresión VIII.5. Consulta en JSON para las entidades principales.

```
queryPrincipal := { service.name: Envío de Notificaciones Telemáticas~2^10 OR service.description: Enviar notificaciones telemáticas a los ciudadanos con la resolución de portafirmas^5 OR operation.name: Envío de Notificaciones Telemáticas~2^5 OR operation.description: Enviar notificaciones telemáticas a los ciudadanos con la resolución de portafirmas^2,5 OR operation.description: Abonar al ciudadano al servicio de notificación^1,25 OR operation.description: Se envía a Portafirmas la resolución^1,25 OR operation.description: Se sella la resolución con el sistemas Sella^1,25 OR operation.description: Se notifica de forma fehaciente la resolución de portafirmas de forma telemática ^1,25 OR operation.description: El sistema comprueba si se ha leído la notificación^1,25 }
```

Expresión VIII.6. Respuesta en JSON para las entidades principales.

```
//TODO – Respuesta JSON principales {
  "responseHeader":{
    "status":0,
    "QTime":35,
    "params":{
      "q":" queryPrincipal ",
      "indent":"on",
      "fl":"service.name, score",
      "wt":"json",
      "_":"1472111760559"}},
```

```
  "response":{"numFound":10,"start":0,"maxScore":88.11103,"docs":[
      {
        "service.name":"Notificaciones Telemáticas",
        "score":88.11103},
      {
        "service.name":"Envío a Portafirmas",
        "score":46.477116},
      {
        "service.name":"Registro Telemático de entrada y salida",
        "score":23.96941},
      {
        "service.name":"Generación de Documentos con Plantill@",
        "score":9.851344},
      {
        "service.name":"Firma electrónica con certificado digital",
        "score":9.321022},
      {
        "service.name":"Generación de Formularios con Formula@",
        "score":9.037504},
      {
        "service.name":"Login con certificado en la Fachada de Ticket",
        "score":7.215325},
      {
        "service.name":"Publicación en BOJA",
        "score":6.2786894},
      {
        "service.name":"Sellado de documentos",
        "score":5.2364364},
      {
        "service.name":"Firma de Servidor",
        "score":4.7580214}]
  }
}
```

De forma semejante y como se observa en las Expresiones VIII.7 y VIII.8 se mostrarán la consulta y la respuesta correspondiente para las entidades adyacentes.

Expresión VIII.7. Consulta en JSON para las entidades adyacentes.

```
queryAdyacente := { tag.tag: Notificaciones OR domain.description: Notificaciones~1 OR stakeholder.name: Manuela Sánchez Macarro~1 OR stakeholder.role: Biblioteca de Andalucía OR policy.policy: La notificación enviada ha sido leída por el ciudadano o han pasado 10 días sin ser leída OR serviceLevelAgreement.kpi: La notificación enviada ha sido leída por el ciudadano o han pasado 10 días sin ser leída OR policy.policy: El código de notificación está dado de alta en Hacienda OR serviceLevelAgreement.kpi: El código de notificación está dado de alta en Hacienda OR policy.policy: El código fuente está en Subversion OR serviceLevelAgreement.kpi: El código fuente está en Subversion
}
```

Expresión VIII.8. Respuesta en JSON para las entidades adyacentes.


```
{
  "responseHeader":{
    "status":0,
    "QTime":3,
    "params":{
      "indent":"on",
      "fl":"service.name, score",
      "wt":"json",
      "_":"1472111760559"}},
  "response":{"numFound":3,"start":0,"maxScore":20.058418,"docs":[
      {
        "service.name":"Notificaciones Telemáticas",
        "score":35.950325},
      {
        "service.name":"Firma de Servidor",
        "score":10.079586},
      {
        "service.name":"Publicación en BOJA",
        "score":8.898813},
      {
        "service.name":"Firma electrónica con certificado digital",
        "score":8.058745},
      {
        "service.name":"Envío a Portafirmas",
        "score":6.2992773},
      {
        "service.name":"Registro Telématico de entrada y salida",
        "score":5.129067},
      {
        "service.name":"Login con certificado en la Fachada de Ticket",
        "score":3.5113442},
      {
        "service.name":"Generación de Formularios con Formula@",
        "score":2.7738729},
      {
        "service.name":"Sellado de documentos",
        "score":2.3645697},
      {
        "service.name":"Generación de Documentos con Plantill@",
        "score":2.2042482}]
```

```
}}
```

De esta forma y aplicando los tratamientos algorítmicos indicados en el Capítulo VII a los resultados de ambas consultas, podemos concluir que los Servicios Candidatos, para cubrir el siguiente requisito, dentro del Catálogo de Servicios de la Consejería de Cultura serían:

- ***Servicio Candidato: Notificaciones Telemáticas [120.551399]***
- ***Servicio Candidato: Envío a Portafirmas [52.7763933]***

En el estudio en paralelo sobre la documentación en papel se llegaba de nuevo a la concusión que estos Servicios Candidatos coinciden con los Servicios Candidatos obtenidos con el framework propuesto, validando por tanto el proceso de descubrimiento.

En el caso del ejemplo, al combinar las listas y ejecutar el primer filtro (FILTRO_REFERENCIA) realizando el corte de los resultados de las consultas, los servicios seleccionados con sus pesos han sido:

- Notificaciones Telemáticas [*120.551399*]
- Envío a Portafirmas [*52.7763933*]
- Registro Telemático de entrada y salida [*29.098477*]

Al aplicar el segundo corte del algoritmo, la media aritmética menos la desviación típica (desviación estándar) ha resultado ser de *32.368171345486175*, quedando por tanto fuera:

- Registro Telemático de entrada y salida [*29.098477*]

Hecho el estudio en paralelo, los Servicios Candidatos que cubren el requisito anterior son el de "*Notificaciones Telemáticas*" y el de "*Envío a portafirmas*". Este proceso se haría sobre todos los requisitos sobre los que se desee saber qué cobertura se tendrá por parte de los Servicios, obteniendo así los Servicios Candidatos.

Nótese que se está trabajando en estos ejemplos, para lo expresado en este capítulo de validaciones de esta Tesis Doctoral, con Servicios del ámbito de la Administración Electrónica, que forman parte del Dominio Tecnológico y por tanto la responsabilidad recae sobre el Servicio de Informática y pueden presentarse por tanto como parte de este trabajo de investigación, gracias al permiso concedido para mostrar dichos resultados. El Catálogo de Servicios completo es propiedad de la Consejería de Cultura y por tanto no es publicable de forma completa en este trabajo de investigación, quedando además como se ha indicado anteriormente, fuera del alcance de dicho trabajo. En el Anexo I de este trabajo se describen estos Servicios de Administración Electrónica a modo de catálogo tal y como han sido indexados en el motor de búsqueda.

Para concluir la validación, se expondrá el desarrollo completo de los requisitos formalizados de la Tarjeta de Usuario, descritos en las siguiente tablas, y a continuación, se mostrarán los resultados de aplicar el proceso de descubrimiento de Servicios Candidatos a dicho conjunto de requisitos como se ha realizado en los pasos anteriores.

Tabla VIII.4. Requisito ágil de Solicitud del Carnet de Lector.

Entidad	Atributo	Valor
UserStrory	name	*"Solicitud del Carnet de Lector"*
	description	*"El ciudadano accede con su certificado digital a la aplicación y rellena un formulario con sus datos, este formulario se firma con certificado electrónico por el ciudadano y se almacena como lector"*
Persona	name	*"Nombre de la Persona"*
	description	*"Biblioteca de Andalucía"*
Epic	*description*	*"Solicitud de Carnet"*
AceptanceCriteria	*assert*	*{ "Se obtiene una solicitud firmada electrónicamente con certificado digital", "Se comprueba que el lector está dado de alta en la BBDD"}*
DefinitionOfDone	*fact*	*{ "El documento está firmado con un certificado válido" }*
Task	*description*	*{ "El usuario rellena el formulario para la solicitud del carnet", "Se convierte ese formulario a una solicitud PDF y se firma con el certificado digital del usuario", "Se almacena la solicitud firmada en la custodia de documentos electrónicos firmados", "El lector se graba en la BBDD"}*

Tabla VIII.5. Requisito ágil de Solicitud del Carnet de Lector de un autorizado.

Entidad	Atributo	Valor
UserStrory	name	*"Solicitud del Carnet de Lector de un autorizado"*
	description	*"El ciudadano accede con su certificado digital a la aplicación y rellena un formulario con los datos del autorizado, este formulario se firma con certificado electrónico por el ciudadano y el autorizado se almacena como lector"*
Persona	name	*"Manuela Sánchez Macarro"*
	description	*"Biblioteca de Andalucía"*
Epic	*description*	*"Solicitud de Carnet"*
AceptanceCriteria	*assert*	*{ "Se obtiene una solicitud firmada electrónicamente con certificado digital", "Se comprueba que el lector está dado de alta en la BBDD" }*
DefinitionOfDone	*fact*	*{ "El documento está firmado con un certificado válido", "El autorizado debe ser menor de 14 años" }*
Task	*description*	*{ "El usuario rellena el formulario para la solicitud del carnet con los datos del autorizado", "Se convierte ese formulario a una solicitud PDF y se firma con el certificado digital del usuario", "Se almacena la solicitud firmada en la custodia de documentos electrónicos firmados", "El lector se graba en la BBDD"}*

Tabla VIII.6. Requisito ágil de Consulta de Solicitudes.

Entidad	Atributo	Valor
UserStrory	name	*"Consulta de Solicitudes"*
	description	*"El ciudadano puede consultar las solicitudes que ha firmado y entregado y recuperarlas"*
Persona	name	*"Manuela Sánchez Macarro"*
	description	*"Biblioteca de Andalucía"*
Epic	*description*	*"Consulta"*
AceptanceCriteria	*assert*	*{ "Se puede acceder a todas las solicitudes firmadas" }*
DefinitionOfDone	*fact*	*{ "N/A"}*
Task	*description*	*{ "El usuario se loga con su certificado digital", "Se muestra un lista con las solicitudes", "Puede descargar la solicitud firmada electrónicamente"}*

Tabla VIII.7. Requisito ágil de Notificación Telemática del Carnet.

Entidad	Atributo	Valor
UserStrory	name	*"Notificación Telemática del Carnet"*
	description	*"Enviar notificaciones telemáticas a los ciudadanos con la resolución del carnet firmada por el servidor"*
Persona	name	*"Manuela Sánchez Macarro"*
	description	*"Biblioteca de Andalucía"*
Epic	*description*	*"Notificaciones"*
AceptanceCriteria	*assert*	*{ "Se comprueba que le llega la notificación", "El ciudadano ha leído la notificación" }*
DefinitionOfDone	*fact*	*{ "El ciudadano debe de estar dado de alta como abonado antes del envío de las notificación", "El lector tiene que existir en la BBDD" }*
Task	*description*	*{ "Abonar al ciudadano al servicio de notificación", "Se envía la notificación con el carnet firmado con un certificado de servidor", "El sistema comprueba si se ha leído la notificación"}*

Una vez lanzado el proceso completo de descubrimiento de Servicios Candidatos para el conjunto de requerimientos descrito, de la misma manera que se ha detallado con el anterior requisito, se puede observar en la Expresión VIII.9 los resultados obtenidos al ejecutar el proyecto DS4aRE.

Expresión VIII.9. Servicios Candidatos para la Tarjeta de Usuario.

```
******************************************************
* UserStory 1 :Solicitud del Carnet de Lector *
******************************************************
```

```
La desviación estándar es: 10.563517850097158
La media ponderada es: 40.694176462499996
El corte se produce con score menor de :[30.13065861240284]
Servicio Intermedio: Firma electrónica con certificado digital [68.880352]
Servicio Intermedio: Generación de Formularios con Formula@ [58.251235]
Servicio Intermedio: Login con certificado en la Fachada de Ticket [49.7854645]
Servicio Intermedio: Firma Electrónica de Servidor [45.573864]
Servicio Intermedio: Generación de Documentos con Plantill@ [35.473215]
Servicio Intermedio: Sellado de documentos con Sello [30.000506]
Servicio Intermedio: Publicación en BOJA [21.5388878]
Servicio Intermedio: Envío a Portafirmas [16.0498874]
Servicio Candidato: Firma electrónica con certificado digital [68.880352]
Servicio Candidato: Generación de Formularios con Formula@ [58.251235]
Servicio Candidato: Login con certificado en la Fachada de Ticket [49.7854645]
Servicio Candidato: Firma Electrónica de Servidor [45.573864]
Servicio Candidato: Generación de Documentos con Plantill@ [35.473215]
******************************************************
* UserStory 2 :Solicitud del Carnet de Lector de un autorizado *
******************************************************
La desviación estándar es: 11.693915186876216
La media ponderada es: 45.34464010000001
El corte se produce con score menor de :[33.65072491312379]
Servicio Intermedio: Firma electrónica con certificado digital [74.744675]
Servicio Intermedio: Generación de Formularios con Formula@ [65.591343]
Servicio Intermedio: Login con certificado en la Fachada de Ticket [55.476653]
Servicio Intermedio: Firma Electrónica de Servidor [52.030847]
Servicio Intermedio: Generación de Documentos con Plantill@ [38.685412]
Servicio Intermedio: Sellado de documentos con Sello [32.582865]
Servicio Intermedio: Publicación en BOJA [27.004432800000004]
Servicio Intermedio: Envío a Portafirmas [16.640893]
Servicio Candidato: Firma electrónica con certificado digital [74.744675]
Servicio Candidato: Generación de Formularios con Formula@ [65.591343]
Servicio Candidato: Login con certificado en la Fachada de Ticket [55.476653]
Servicio Candidato: Firma Electrónica de Servidor [52.030847]
Servicio Candidato: Generación de Documentos con Plantill@ [38.685412]
******************************************************
* UserStory 3 :Consulta de Solicitudes *
******************************************************
La desviación estándar es: 0.0
La media ponderada es: 14.7612195
El corte se produce con score menor de :[14.7612195]
Servicio Intermedio: Firma electrónica con certificado digital [14.7612195]
Servicio Candidato: Firma electrónica con certificado digital [14.7612195]
******************************************************
* UserStory 4 : Notificación Telemática del Carnet *
```

```
*****************************************************
La desviación estándar es: 8.94922149979136
La media ponderada es: 41.35702175
La mitad es: 38.210210000000004
El corte se produce con score menor de: [32.40780025020864]
Servicio Intermedio: Notificaciones Telemáticas [76.42042000000001]
Servicio Intermedio: Firma Electrónica de Servidor [46.064205]
Servicio Intermedio: Publicación en BOJA [23.711829]
Servicio Intermedio: Login con certificado en la Fachada de Ticket [19.231633]
Servicio Candidato: Notificaciones Telemáticas [76.42042000000001]
Servicio Candidato: Firma Electrónica de Servidor [46.064205]
```

Una vez contrastado el proceso automático con el manual, podemos indicar que el grado de adecuación desde el punto de vista funcional es muy alto, es decir, la idoneidad de los Servicios Candidatos que se incluyen y la de los Servicios que se excluyen coinciden a cómo debería ser en la realidad, cuando se compara con dicho estudio realizado de forma manual para realizar la validación.

Como se puede ver en la Tabla VIII.8, sobre un total de 10 Servicios de Administración Electrónica pertenecientes al Catálogo de Servicios y que han sido indexados para su búsqueda, los Servicio incluidos y excluidos por error suponen 0, es decir, el proceso muestra los resultados que debe obtener. La semántica de esta tabla es la que se indica a continuación:

- Servicios Candidatos, es el número de servicios que se han devuelto después de ejecutar el proceso de Servicios Candidatos con el framework DS4aRE.
- Servicios Candidatos con Error, es el número de servicios que se han devuelto después de ejecutar el proceso de Servicios Candidatos con el framework DS4aRE y que no tendrían que haberse devuelto como Servicios Candidatos.
- Servicios Excluidos, es el número de servicios que han sido excluidos después de ejecutar el proceso de Servicios Candidatos con el framework DS4aRE.
- Servicios Excluidos con Error, es el número de servicios, de entre los Servicios Excluidos, que no se debían de haber excluido por parte del proceso.

Tabla VIII.8. Resultados para la aplicación Tarjeta de Usuario.

Requerimiento Ágil	Servicios Incluidos	Servicios Excluidos
Solicitud del Carnet de Lector	Servicios Candidatos: 5 Servicios Candidatos con Error: 0	Servicios Excluidos: 5 Servicios Excluidos con Error: 0
Solicitud del Carnet de Lector de un autorizado	Servicios Candidatos: 5 Servicios Candidatos con Error: 0	Servicios Excluidos: 5 Servicios Excluidos con Error: 0
Consulta de Solicitudes	Servicios Candidatos: 1 Servicios Candidatos con Error: 0	Servicios Excluidos: 9 Servicios Excluidos con Error: 0
Notificación Telemática del Carnet	Servicios Candidatos: 2 Servicios Candidatos con Error: 0	Servicios Excluidos: 8 Servicios Excluidos con Error: 0

Como se puede observar en la Tabla VIII.8, después de la ejecución del proceso de Servicios Candidatos con el framework DS4aRE, se advierte que no se ha dado ningún error en cuanto a recuperar Servicios Candidatos de forma incorrecta, así mismo, para este aplicativo, no se da ningún fallo en cuanto a la exclusión de posibles servicios que hubiesen debido ser Servicios Candidatos, proporcionando una fiabilidad a la organización suficiente para utilizar este proceso de forma habitual en el desarrollo de aplicaciones.

Por último y para concluir este apartado se presentan una serie de datos para reflejar el ahorro de costes que supone a la organización este proceso. Para ello en la Tabla VIII.9 se expondrán las horas invertidas para el proceso de descubrimiento de Servicios Candidatos de forma manual y para el proceso usando el framework DS4aRE. Los costes se medirán en horas.

En esta tabla se han seleccionado las actividades del proceso de descubrimiento de Servicios Candidatos y se ha comparado su coste manual con el coste de realizar esa misma actividad mediante el framework DS4aRE, para un requisito de la Tarjeta de Usuario usado como ejemplo en esta validación.

Tabla VIII.9. Comparativa de costes económicos para un requisito.

Coste	Proceso Manual	Proceso con el Framework DS4aRE	Coste
15 horas	El responsable de los Servicios de la organización analiza los Servicios de Administración Electrónica.	SOLR, Proyecto DS4aRE*: se indexan para la búsqueda todos los Servicios de Administración Electrónica*	0 horas
21 horas	El responsable de la nueva aplicación a desarrollar presenta el requisito ágil obtenido en la Tarjeta de Usuario.	EA, Proyecto DS4aRE: *El responsable de proyecto modela el requisito en EA a través del perfil UML propuesto y que será usado por el proyecto DS4aRE y formalizado para la búsqueda.*	10 horas
20 horas	El responsable de los Servicios indica qué servicios podrían ser los Servicios Candidatos para ese requisito dado.	Proyecto DS4aRE, SOLR: *Para el requisito formalizado, el proyecto DS4RE consulta SOLR y obtiene los Servicios Candidatos a través del tratamiento algorítmico propuesto.*	0 horas

Para un requisitos dado, con el catálogo de Servicios de Administración Electrónica presentado, el coste total para el proceso manual sería de 66 horas y el coste del proceso automatizado 10 horas, suponiendo un ahorro en el proceso del **75%** aproximadamente. Este caso hace referencia al coste directo traducido en horas por perfil, pero además habría que añadir los costes indirectos ahorrados debidos a la reutilización del software.

Pero no sólo presenta una serie de ventajas este proceso desde el punto de vista estrictamente económico sino que además presenta un impacto grande en el trabajo de los perfiles de las personas que participan, ya que pueden dedicarse a tareas que generen valor para el negocio. Sirva como ejemplo:

- Para los equipos de desarrollo, no es necesario que tengan que conocer todo el Catálogo de Servicios de la organización, ya que el proceso propuesto hace ese trabajo por ellos de forma instantánea, reduciendo los costes del equipo.
- Por parte de los equipos dedicados al gobierno de los Servicios, tampoco será necesario, gracias al proceso propuesto, que conozcan cada requisito de una aplicación a desarrollar para valorar la adecuación entre los Servicio y ese requisito.

Por otra parte la ejecución de este proceso para todos los requisitos de todas las aplicaciones (histórico), también redundará en la eficacia y eficiencia en el gobierno de los Servicios, ya que se podrán mejorar una serie de situaciones como pueden ser:

- Si hay Servicios Candidatos que llegan a elegirse y no deberían serlo o servicios excluidos que deberían de incluirse, se podría detectar que la definición textual de los mismos no es adecuada y por tanto podrán actualizar o mejorar dicha definición para un mejor descubrimiento.
- Si hay requisitos recurrentes para los que no se descubren Servicios Candidatos, podrá ser indicativo de que se ha de definir uno o varios Servicios para darles cobertura.

Todo esto por tanto redunda en la eficacia y eficiencia del desarrollo de aplicaciones con metodologías ágiles desde las etapas más tempranas de dicho desarrollo debido a que la reutilización del software en la organización se maximizará, se aumentará la calidad y se reducirán los costes del mismo. Así mismo redundará en una mejora del Gobierno TI en el área de Servicios, mejorando la integración del ciclo de vida de los mismos dentro del crecimiento del parque de aplicaciones de la organización.

4. Conclusiones

Como se ha ido presentado a lo largo del presente capítulo, se ha podido llevar a cabo la implantación y validación del marco teórico de la presente Tesis Doctoral en un entorno real con resultados satisfactorios.

En necesario remarcar sin embargo, que la solución propuesta requiere de cierta adaptación cuando se vaya a implantar en otro entorno diferente ya que depende de una serie de factores como la arquitectura tecnológica, la metodología de desarrollo usada en el organismo en cuestión, la madurez de los equipos de desarrollo, etc.… Es decir, para realizar la adopción del proceso de descubrimiento de Servicios Candidatos expuesto en el presente trabajo de investigación, es necesario que primero se adapte al contexto al que se reproduce en la presente Tesis Doctoral.

Es necesaria también la parametrización del proceso de descubrimiento de Servicios de manera que se deberán ajustar estos valores con algunas pruebas iniciales, ya que para cada entorno dependerá fundamentalmente de la redacción en lenguaje natural, tanto del Catálogo de Servicios, como de los requisitos.

Sin embargo y a pesar de lo expuesto en los párrafos anteriores, el proceso tiene un carácter universal, ya que se podrá adaptarse a otras metodologías de desarrollo siempre que se tenga la información mínima necesaria y con la estructura y modelado que se explicita a lo largo de este trabajo de investigación para el proceso de descubrimiento de Servicios Candidatos.

Capítulo IX Aportaciones, trabajos futuros y conclusiones

En los capítulos anteriores se han sentado las bases para la definición de un proceso encaminado al descubrimiento de Servicios Candidatos dentro del contexto de una organización que cubra los requisitos de una nueva aplicación en las fases más tempranas del desarrollo de software con metodologías ágiles en un entorno basado en Servicios. Una vez analizada la situación actual, se han presentado las influencias sobre las que se ha fundamentado el desarrollo de esta Tesis Doctoral así como las motivaciones que llevaron al planteamiento del problema expuesto en dicho trabajo. Además, se ha mostrado la validación de este marco en un organismo cuya implantación ha sido considerada como satisfactoria y que ha supuesto la sistematización de la búsqueda de Servicios que ya están en el contexto de la organización en orden a cubrir los requisitos de las nuevas aplicaciones a desarrollar.

Además, se han propuesto y analizado una serie de metamodelos y se ha desarrollado un algoritmo para la identificación de dichos Servicios Candidatos. Así mismo, se ha expuesto la arquitectura tecnológica que sustenta dicho proceso, el framework DS4aRE, y se ha identificado la actividad necesaria dentro de la metodología ágil de desarrollo para la integración de dicho proceso dentro de su ciclo de vida.

Se ha presentado por último el proceso de implantación de esta arquitectura tecnológica en un entono real en donde se han podido instanciar, validar y ejecutar los artefactos, sistemas y procesos sobre los que ha versado el presente trabajo y que ha resultado satisfactorio.

También, durante los capítulos anteriores, se han comentado y referenciado trabajos e influencias que han servido como fuente de inspiración para la consecución de los objetivos marcados en este trabajo de Tesis Doctoral. Sin embargo, es necesario dejar constancia detallada de cuáles han sido las aportaciones de este trabajo. Éste es uno de los propósitos de este capítulo.

Este capítulo prosigue con el planteamiento de los trabajos futuros que continúan la línea de investigación abierta con la realización de esta Tesis Doctoral y finaliza describiendo el marco de investigación en el que se ha desarrollado esta Tesis Doctoral y aportando una serie de conclusiones finales.

1. Aportaciones de este trabajo de Tesis

Como se ha descrito en capítulos anteriores, este trabajo de Tesis Doctoral se ha nutrido de ideas que provienen de otros trabajos ya existentes.

Cabe destacar, en primer lugar, los trabajos referenciados y estudiados en el Capítulo II, que abordan la relación entre los paradigmas de las arquitecturas orientadas a servicios en orden al descubrimiento de los Servicios Candidatos dentro de un contexto concreto, en entornos de desarrollo con metodologías ágiles, en aras de cubrir los requisitos de nuevos desarrollos.

En segundo lugar se encuentran ya en el Capítulo III las influencias expuestas sobre la transformación de organizaciones para hacerlas capaz de operar con Servicios, así como un

conjunto de proceso y técnicas ágiles imbricados entre sí que conforman una metodología para la estimación, planificación y gestión para el desarrollo de Sistemas de Información Web.

Tomando como fuentes todas estas influencias, esta Tesis Doctoral culmina en una serie de aportaciones que plantean una solución original para abordar de forma sistemática y coherente la formalización de un requisito Web fundamentándolo en procesos y técnicas ágiles (debido a su agilidad y completitud) y que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios, dentro del contexto, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito, es decir, poder realizar el proceso para el descubrimiento de los Servicios Candidatos.

A modo de resumen, a continuación se describen estas aportaciones.

1.1. Estudio del Estado del Arte

Antes de proponer una solución original resulta conveniente analizar y estudiar qué es lo que han propuesto otros autores en el mismo ámbito científico para identificar puntos de mejora o las deficiencias de esas propuestas. Ésta ha sido la primera aportación de esta tesis.

Debido a que el descubrimiento de Servicios Candidatos dentro de un contexto concreto, en organizaciones que utilizan las metodologías ágiles para el desarrollo de aplicaciones Web, era muy específico, se ha constatado que no existía en la literatura actual ningún trabajo anterior que englobara completamente este proceso.

Ha sido por tanto dividido este estudio del estado del arte en dos partes, la primera acerca de la ingeniería de requisitos para metodologías ágiles, que parte de la revisión sistemática de la literatura realizada en colaboración con otro doctorando dentro del grupo de investigación IWT2 [Schön et al. 2017], así como de otros trabajos relacionados con este tema, dándoles un enfoque encaminado a la formalización del requisito ágil.

La otra parte de este estudio ha versado sobre la modelización funcional de Servicios incardinados en una organización concreta de cara a poder realizar el gobierno de los mismos y poder tener esa funcionalidad disponible de forma normalizada hacia el exterior.

Con esta aportación acerca del estudio concreto de la literatura existente en el proceso que se describe en esta Tesis Doctoral, se cubre el primero de los objetivos definidos en el Capítulo III.

1.2. Formalización del Catálogo de Servicios

Para el descubrimiento de la funcionalidad que ya está en el contexto, es necesario exponer la misma hacia el exterior de una manera formal, a este respecto se aporta en esta Tesis Doctoral la formalización funcional de los Servicios de una organización, de manera que dicho Catálogo de Servicios, a través del metamodelo propuesto, expondrá de manera formal la funcionalidad de dichos Servicios Gobernados. Se expone en un mismo Catálogo información de los servicios y del entorno en el que se incardinan, siendo necesaria esa meta-información añadida al concepto puro de Servicio para saber cuando se podrá cubrir un nuevo requisito con ese Servicio, ya que parte de esa información está también en el contexto.

Con estas aportaciones se cubre el segundo objetivo propuesto en el capítulo correspondiente del presente trabajo de investigación.

1.3. Formalización de requisitos Web mediante procesos y técnicas ágiles

En este campo y como recogía el tercer objetivo del Capítulo III, este trabajo de Tesis Doctoral aporta un metamodelo que permite la formalización de requisitos que ha sido educido a partir de la aplicación de los principales procesos y técnicas ágiles, que cooperan organizadamente entre sí dentro de un metodología que las engloba y que permite, de una manera formal, poder presentar esos requisitos ágiles para otros procesos, dentro de la ingeniería de requisitos, pudiendo ser un punto de enlace de las metodologías ágiles con el resto de paradigmas de la Ingeniería de Software.

En el caso que nos ocupa nos permitirá descubrir qué requisitos están ya implementados en el contexto y por tanto se maximizará la reutilización del software dentro de un organización.

1.4. Proceso para el descubrimiento de los Servicios Candidatos

Este trabajo de investigación aporta un proceso para el descubrimiento de los Servicios Candidatos para cubrir la funcionalidad de un requisito ágil basados en valor, primero, a través de la propuesta de una relación entre los metamodelos propuestos, tanto a nivel de entidades como de atributos, seguido por un algoritmo que permite realizar una serie de consultas sobre el Catálogo de Servicios, una vez indexado en una herramienta especializada en búsqueda y recuperación de información textual, así como el tratamiento de los resultados, describiéndose los pasos necesarios para la realización de este proceso completo, que abarca desde la formalización del requisito hasta la devolución para su estudio de una lista de servicios que puedan dar cabida a ese requisito, es decir, los Servicios Candidatos.

Queda cubierto por tanto el cuarto objetivo propuesto en el Capítulo III.

1.5. Framework DS4aRE

El objetivo quinto queda cubierto por la presentación del framework DS4aRE que permite la ejecución del proceso de descubrimiento de Servicio Candidatos, constituyendo el soporte tecnológico necesario de dicho proceso a través del uso de una serie de herramientas software y de la implementación del algoritmo mencionado en el capítulo anterior por medio de un proyecto JAVA.

Además y coincidiendo con este quinto objetivo propuesto en el Capítulo III, a parte del framework descrito, entre las aportaciones realizadas en esta Tesis Doctoral se encuentran la creación de lenguajes de dominio específicos para cada uno de los metamodelos propuestos, a través del desarrollo de perfiles UML (así como su uso dentro de la herramienta Enterprise Architect) que aporta la capacidad de poder trabajar con dichos metamodelos en entornos reales.

1.6. Validación en un entorno real

Por último y coincidiendo con el sexto y último objetivo propuesto en esta Tesis Doctoral, este trajo de investigación aporta una validación del modelo teórico en la Consejería de Cultura de la Junta de Andalucía, realizando la implantación en dicha organización de los procesos y herramientas tecnológicas descritos en profundidad a lo largo de este documento y obteniéndose una prueba satisfactoria de la usabilidad de este proceso, así como de la arquitectura tecnológica propuesta, que ha supuesto para dicho organismo la mejora en la

eficiencia y eficacia de su metodología de desarrollo, adoptando una serie de actividades que le ha permito maximizar la reutilización del software que tiene implementado a través de servicios, así como mejorar en el gobierno de los mismos. Esto ha supuesto minimizar los costes en recursos y tiempo del desarrollo de nuevas aplicaciones Web usando metodologías ágiles, integrando, a través de los metamodelos y el proceso propuesto, los diferentes trabajos tanto de los equipos que desarrollan software como de los equipos responsables del gobierno y la gestión de los servicios.

Es por tanto obligado indicar que estas aportaciones han supuesto mejoras empíricas y cuantificables en la identificación y uso de los servicios por parte de nuevos desarrollos que realiza el organismo, así como una mejora en el gobierno de dichos servicios, al entrar esas actividades dentro de un proceso sistemático y con un grado de automatización suficiente que redunda en el ahorro de costes y aumenta la eficiencia y eficacia de dicha actividad dentro de la organización.

2. Trabajos futuros y nuevas líneas de investigación

Una vez establecidas las aportaciones, resulta conveniente explicitar que esta Tesis Doctoral no es un trabajo cerrado y de hecho, sus resultados abren nuevas líneas de investigación. En este sentido, a lo largo de este documento se han ido mencionando algunas de estas líneas, pero es en esta sección donde se retoman y detallan de una manera más concreta.

Una de las principales líneas de investigación está relacionada con la formalización de requisitos Web en entornos de desarrollo en los que se utilizan diferentes metodologías y técnicas ágiles, el metamodelo de Requisitos basados en Valor, que tiene la virtud de formalizar una serie de artefactos ágiles usados habitualmente en el desarrollo y podrá constituir en un futuro el punto de conexión entre las metodologías ágiles y el paradigma del desarrollo guiado por modelos, que es uno de los retos futuros de la Ingeniería Web guiada por modelos (MDWE) [Rossi et al. 2016], ya que constituye una propuesta que permitirá, a través del desarrollo de las respectivas transformaciones a partir del metamodelo, poder integrar los desarrollos ágiles dentro de otras metodologías o abordar otros problemas dentro de la Ingeniería de Requisitos que requieran que la información este formalizada.

Gracias a esto podrían integrase técnicas del campo del ASD ("*Agile Software Development*"), como por ejemplo todas las técnicas relacionadas con la integración de los diferentes interesados (responsables, usuarios, consumidores, equipos de desarrollo, etc..), como puede ser la técnica "Persona", pudiendo combinarlas de manera eficiente con otras metodologías dentro del campo de la ingeniería guiada por modelos.

El propio trabajo de investigación expuesto en los capítulos anteriores representa una forma de abordar la idea anterior, en cuanto que se ha integrado dentro de las etapas más tempranas del desarrollo, con metodologías agiles, la integración con el Gobierno de los Servicios dentro de una Arquitectura Orientada a Servicios y que de por sí requiere una formalización. Este metamodelo ha sido el punto que ha posibilitado la unión de ambos paradigmas.

Así mismo, el propio metamodelo de Servicios y el trabajo con los Servicios Candidatos, dentro del proceso propuesto, arroja otra línea de investigación que amplía este trabajo. Nos referimos a la ejecución automatizada de las políticas asociadas al Ciclo de Vida de los Servicios mencionado en este trabajo de Tesis Doctoral.

Esta línea de trabajo enriquecería el metamodelo, desarrollando las relaciones pormenorizadas para la ejecución de políticas una vez escogidos los Servicios. De la misma manera, este nuevo metamodelo y la posibilidad de ejecución de políticas, abriría una línea para un nuevo diseño en la arquitectura tecnológica que permitiese ejecutar dichas políticas dentro del proceso para el descubrimiento de los Servicios Candidatos.

Por último, la mejora continua, ya en el estadio del marco tecnológico, nos llevaría a abrir una línea de trabajo para el diseño de una herramienta que integrase en una única aplicación todos los procesos que actualmente están diseminados en una serie de elementos arquitectónicos, expuestos en el Capítulo VII y que redundaría en una mejora en la implantación de dicho proceso en organizaciones que desarrollan software y trabajen con Servicios de una manera más eficiente, reduciendo los costes de implantación.

Por otra parte y ya en el marco de la transferencia de conocimiento hacia organizaciones empresariales, la extensión de esta metodología e implantación de las herramientas en otras consejerías de la Junta de Andalucía y en otras administraciones públicas supone otros de los trabajos futuros de esta Tesis Doctoral en cuanto que es habitual que las administraciones públicas comparten las experiencias de éxito surgidas en cada una de ellas.

Así mismo y dentro de este marco, la adopción de esta metodología por empresas de desarrollo de software, será uno de los trabajos futuros, debido en parte a que por un lado estas empresas colaboran con las administraciones para la implantación de las metodologías y a su vez, debido a la incorporación de las mismas en los diferentes equipos de trabajo, podrán extenderla a otros clientes, generándose una difusión más amplia de este proceso de descubrimiento de Servicios Candidatos para cubrir las nuevas necesidades software de los clientes.

Será objeto, por último, de los trabajos futuros la adaptación de esta metodología para una mejor adopción por parte de las diferentes organizaciones involucradas.

3. Marco estratégico en el que se ha desarrollado este trabajo de investigación

El marco estratégico en el que se ha desarrollado este trabajo de investigación está relacionado, por una parte, con el trabajo de investigación que se desarrolla dentro del grupo IWT2 acerca de la relación de las nuevas metodologías ágiles con la Ingeniería de Software y que constituye una línea de investigación que ha ido teniendo cada vez más relevancia y de la que se han realizado o se están realizando varias tesis doctorales que presentan alguna relación con la presente. Así mismo, por otra parte, este marco estratégico engloba el trabajo profesional realizado por el autor dentro de la Consejería de Cultura en el campo de los Servicios, el Gobierno Electrónico y las metodologías ágiles.

3.1. Relación con el grupo de investigación IWT2

Como se indicaba en la introducción de este apartado, en el seno del grupo de investigación IWT2 las metodologías ágiles han tomado cuerpo como una línea de investigación dentro del ámbito de la Ingeniería de Software.

En relación con esta Tesis Doctoral se están llevando en el seno del grupo otros trabajos de investigación relacionados con el presente, como son el de Eva Schön, dirigido por Jörg Thomaschewski y María José Escalona sobre un metamodelo para la definición de contexto de

uso de metodologías ágiles en las que la participación de los interesados permita abordar problemas de la Ingeniería de Requisitos ágiles y de la que la presente tesis doctoral puede presentar una incipiente aplicación de dichos principios y la tesis titulada "*A mature Agile approach in Web Engineering contexts*" del doctor Carlos Torrecilla, dirigida también por Manuel Mejías y María José Escalona y que versa sobre la definición de la metodología NDT-Agile, que presenta la madurez en el uso de las metodologías ágiles por parte de las organizaciones hasta llegar a la adopción de los modelos de calidad CMMI-Dev, cuyos procesos quedan encuadrados dentro del ciclo de vida NDT-Agile, el cuál agrupa de forma coherente y estructurada un conjunto de metodologías ágiles, muchas de las cuales se han desarrollado como influencia en el Capítulo III de este trabajo de investigación y que han sido cruciales para definir un metamodelo que pueda ser instanciado en una organización real.

3.2. Trabajos realizados en la Consejería de Cultura

Este trabajo ha sido posible gracias a las autorizaciones de las diferentes Jefaturas de Servicio de Informática de la Secretaría General Técnica de la Consejería de Cultura de la Junta de Andalucía, en las épocas en que se realizó el presente estudio, para la publicación de los resultados expuestos a lo largo del Capítulo VIII de este trabajo.

A lo largo dicho capítulo se han ido exponiendo las diferentes iniciativas del autor, que ha ido adoptando la Consejería de Cultura, así como la experiencia y la capacidad para ejecutar proyectos de dicho organismo de las que el autor se ha nutrido para la realización de este trabajo.

La sintonía con la Consejería de Cultura, así como su vocación de servicio a la ciudadanía, a través de la mejora continua de sus procesos, ha llevado a que el "*feedback*" entre la investigación y el desarrollo del trabajo se llevase, en la persona del autor, a su máxima expresión, beneficiándose ambos del resultado de los trabajos, ya que los fundamentos teóricos para la resolución de problemas reales que existen en este organismo aportados por la investigación, se han podido validar y refinar en dicho organismo, llevando así la colaboración entre investigación y el tejido empresarial, en este caso la administración pública, pero por difusión a sus interesados (ciudadanía, personal interno y empresas proveedoras en el ámbito de las tecnologías de la información y comunicaciones) a resultados satisfactorios en la resolución de problemas reales a través del estudio del marco teórico, sus propuesta y su posterior implantación y validación. Es por tanto necesario recalcar que se ha habilitado un canal para la transferencia de conocimiento desde la investigación hacia la administración con la retroalimentación de las experiencias empíricas que supone la implantación de dicho conocimiento.

Sin esta relación no hubiese sido posible la realización de este trabajo de investigación ya que la Consejería de Cultura ha aportado a este trabajo tres elementos fundamentales:

- El contexto, es decir, el entorno real donde se constata la evolución de dos paradigmas que se interrelacionan de una manera determinada.
- El problema, es decir, el marco donde se advierte que existe una carencia que requiere una solución.
- La validación, es decir, el lugar donde se prueban y refinan los planteamientos teóricos para demostrar su validez en un entorno real.

Así mismo la Consejería se beneficia, gracias al grupo de investigación, de una serie de elementos fundamentales también expuestos en esta Tesis Doctoral como son:

- La definición formal del problema, a través de un proceso de inducción que formaliza un problema concreto dotándole de la universalidad necesaria.
- La solución, ya que es en el trabajo de investigación donde se plantea la solución teórica al marco presentando y una vez estudiado el estado del arte del mismo.
- La arquitectura tecnológica, ya que como parte del trabajo de investigación, se proponen de un conjunto de herramientas que puedan ser implantadas en un entorno real y den soporte a la solución teórica planteada.

4. Conclusiones

A lo largo de los capítulos que articulan la presente Tesis Doctoral se ha descrito la propuesta para la formalización de un requisito, fundamentándolo en técnicas ágiles (debido a su agilidad y completitud) y que pueda ser gestionado contra un Catálogo de Servicios, a fin de descubrir qué Servicios, dentro del contexto, son susceptibles de ser incorporados en el desarrollo de la nueva aplicación para dar cobertura a ese requisito.

El cuerpo de esta Tesis Doctoral pues, se cimenta sobre la definición de una serie de metamodelos. Se define un metamodelo en el que se formalizarán los Servicios pertenecientes al Catálogo de Servicios de dicha organización, que contendrá la funcionalidad identificada y viva, en el contexto, de forma normalizada. A su vez, se define un metamodelo de requisitos que permita la formalización ágil, temprana y completa de los nuevos requisitos. Se define por último un algoritmo que permita realizar las consultas diseñadas con la relación propuesta entre los metamodelos mencionados anteriormente.

Como aportación de la presente Tesis Doctoral obtenemos un proceso sistemático y coherente para el descubrimiento de los Servicios Candidatos, que nos permita descubrir qué Servicios dentro del Catálogo de Servicios dan cobertura a un conjunto total o parcial de los requisitos, es decir, identificar los Servicios Candidatos para su análisis.

Además se obtiene en este trabajo el framework DS4aRE que soporta e implementa dicho proceso y es desplegable en una organización real.

En conclusión, esta Tesis Doctoral plantea una solución a un problema específico: realizar el proceso sistemático y coherente para el descubrimiento de los Servicios Candidatos dentro del contexto de una organización que presta Servicios, a través de la formalización de requisitos usando técnicas y metodologías ágiles a fin de identificar dentro del Catálogo de Servicios de la organización que funcionalidad de los nuevos requisitos está ya contenida.

Uno de los beneficios fundamentales de esta gestión ágil de requisitos dentro el Gobierno de los Servicios, desde las etapas más tempranas del desarrollo, desembocará en la eficacia y eficiencia de los recursos, del propio desarrollo software y de una mejor prestación de dichos Servicios.

Desde las etapas más tempranas de la elicitación de requisitos, se conocerá qué Servicios, dentro de la organización, cubren parte de la funcionalidad, por lo que la reutilización del software se maximizará para estos desarrollos, con el consiguiente ahorro en tiempo y coste y aumentando la calidad de las nuevas aplicaciones.

Así mismo, se mejora, dentro del Gobierno de TI, el gobierno de los Servicios debido a que la temprana identificación de su uso, hace posible que se puedan ejecutar las políticas adecuadas a su ciclo de vida, redundando así en un mejor control de los Servicios y por tanto mejorando la prestación de los mismos y minimizando el impacto de los sucesivos cambios de estado dentro de su ciclo de vida.

Por último, el trabajo presenta la evaluación de los resultados teóricos obtenidos en un entorno de producción basado en la instanciación de la solución, cuya aplicación ha sido considerada como satisfactoria.

Referencias Bibliográficas

[Abdul-Manan & Hyland 2013] Abdul-Manan M., Hyland P., 2013. A Framework for Assessing Enterprise-Wide SOA Implementation Readiness. International Journal of Intelligent Information Technologies (IJIIT), 9(2), 21-37. doi:10.4018/jiit.2013040103.

[Alver, 2012] Alver, M. O. (2012). JabRef reference Manager. http://jabref.sourceforge.net/. Último acceso diciembre 2016.

[Alwadain et al. 2013] Alwadain, A., Fielt, E., Korthaus, A., & Rosemann, M., 2013. A Comparative Analysis of the Integration of SOA Elements in Widely-Used Enterprise Architecture Frameworks. International Journal of Intelligent Information Technologies (IJIIT), 9(2), 54- 70. doi:10.4018/jiit.2013040105.

[Ambler 2002] Ambler, S.W. "Lessons in Agility from Internet-based Development." IEEE Software, pp. 66-73. Mar-Apr, 2002.

[Ameller et al. 2015] Ameller, David; Burgus, Xavier; Collell, Oriol; Costal, Dolors; Franch, Xavier & Papazoglou, Mike P. 2015. Development of service-oriented architectures using model-driven development: A mapping study. Information and Software Technology. Vol. 62, pp. 42 – 66.

[Anderson 2004] Anderson, D.J. Making the Business Case for Agile Management - Simplifying the Complex System of Software Engineering, in: Motorola S3 Symposium, 2004: pp. 1–13.

[Anderson 2010] Anderson, D.J. Kanban - Successful Evolutionary Change for your Technology Business**.** Blue Hole Press, 2010**.**

[Apache Lucene 2012] Apache Software Foundation (ASF). Class TFIDFSimilarity (2000-2012). http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html. Último acceso enero 2017.

[Apache Solr 2016] Perteneciente a la Apache Software Foundation (ASF). Página Oficinal. http://lucene.apache.org/solr/. Último acceso septiembre 2015.

[Arni-Bloch & Ralyte 2009] Arni-Bloch, Nicolas; Ralyte, Jolita. 2009. MISS: A Metamodel of Information System Service. Information Systems Development: Towards a Service Provision Society, pp.177-186.

[Arsanjani et al. 2007] Arsanjani, A.; Zhang, L. J.; Ellis, M.; Allam, A.; Channabasavaiah, K. 2007. S3: A Service-Oriented Reference Architecture. IT Professional. Vol.91, pp. 1-17.

[Balazs et al. 2013] Simon, Balazs; Goldschmidt, Balazs; Kondorosi, Karoly. A Metamodel for the Web Services Standards. Journal of Grip Computing. Vol.: 11 Núm.: 4 Pág.: 735-752. December 2013.

[Baresi et al., 2003] Baresi L., Garzotto F., Paolini P.: Extending UML for Modelling Web Applications. In: Annual Hawaii Int. Conf. on System Sciences, pp. 1285–1294. Miami, USA (2001)

[Barrett et al. 2015] Barrett, M., Davidson, E., Prabhu, J., and Vargo, S. L. (2015). "Service Innovation in the Digital Age: Key Contributions and Future Directions". MIS Quarterly, 39 (1), 135–154

[Basak, 2015] Basak, S. K. (2015). A comparison of three reference management software: Jabref, zotero, and endnote. International Journal of Research in Information Technology, 3(4):223–231.

[Beck & Andres 2004] Beck, K.; Andres, C. "Extreme Programming Explained: Embrace Change, Second Edition". Boston: Addison-Wesley. 2004.

[Beck et al. 2001] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, Manifesto for Agile Software Development, (2001) http://www.agilemanifesto.org/ Ultimo Acceso diciembre de 2016.

[Benguria et al. 2007] Benguria, Gorka; Larrucea, Xabier; Elveser, Brian; Neple, Tor; Beardsmore, Anthony; Friess, Michael, 2007. A platform independent model for service oriented architectures. Enterprise Interoperability. Pp 23-32. Springer

[Beyer 2010] H. Beyer, "User-centered agile methods", Morgan & Claypool Publishers, [San Rafael, Calif.], 2010.

[Bourimi et al. 2010] Bourimi, M., Barth T., Thaake, J.M., Ueberschär, B., Kesdogan, D., "AFFINE for enforcing earlier consideration of NFRs and human factors when building socio-technical systems following agile methodologies". Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 6409 LNCS (2010) 182–189.

[Braga et al. 2016] Braga, Rosana T. Vaccare; Feloni, Daniel; Pacini, Karen; Filho, Domenico Schettini; Gottardi, Thiago. 2016. AIRES: An Architecture to Improve Software Reuse. Software Reuse: Bridging with Social-Awareness. Springer. http://dx.doi.org/10.1007/978-3-319-35122-3_16

[Braytee et al. 2015] Braytee, A., Gill, AQ., Kennedy, PJ., Hussain, FK.: A Review and Comparison of Service E-Contract Architecture Metamodels. 22nd International Conference on Neural Information Processing (ICONIP). Istanbul, TURKEY (09-12, Nov 2015)

[Castellano et al. 2005] Castellano M., Pastore N., Arcieri F., Summo V., de Grecis G. B., 2005. An E-Government Cooperative Framework for Government Agencies, System Sciences. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference vol., no., pp. 121.

[Cauvet 2010] Cauvet, Corine. 2010. Method Engineering: A Service-Oriented Approach. Intentional Perspectives on Information Systems Engineering. Springer. http://dx.doi.org/10.1007/978-3-642-12544-7_19

[Chehili et al. 2013] Chehili, H., Seinturier, L., Boufaida, M.: "FASOAD: A Framework for Agile Service-Oriented Architectures Development," 24th International Workshop on Database and Expert Systems Applications, Los Alamitos, CA, 2013, pp. 222-226. 2013. doi: 10.1109/DEXA.2013.28

[Cockburn 2004] Cockburn, A. "Crystal Clear: A Human-Powered Methodology for Small Teams". Boston: Addison- Wesley. 2004.

[Cohn 2004] Cohn, M. "User Stories Applied: For Agile Software Development". Boston: Addison-Wesley. 2004.

[Cohn 2005] Cohn, M. "Agile Estimating and Planning". NJ: Addison-Wesley. 2005.

[COM 179/2016] Plan de Acción Europeo sobre Administración Electrónica 2016-2020. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2016:0179:FIN:ES:PDF Último acceso junio 2016.

[COM 743/2010] Plan de Acción Europeo sobre Administración Electrónica 2011-201. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0743:FIN:ES:PDF Último acceso junio 2016.

[Comerio et al. 2015] Comerio, Marco; Batini, Carlo; Castelli, Marco; Grega, Simone; Rossetti, Marco; Viscusi, Gianluigi. 2015. Service portfolio management: A repository-based

framework. Journal of Systems and Software. Vol. 104, pp. 112-125. http://www.sciencedirect.com/science/article/pii/S0164121215000333

[Delgado et al. 2013] Delgado H., Losavio F., Matteo A., 2013. Goal oriented techniques and methods: Goal refinement and levels of abstraction. Computing Conference (CLEI), XXXIX Latin American , vol., no., pp.1,12, 7-1.

[Depósito 2008] Sistema de Tramitación del Deposito Legal de Andalucía. https://ws096.juntadeandalucia.es/deposito/acceso.do . Último acceso enero 2017.

[Deshpande et al. 2002] Deshpande Y., Marugesan S., Ginige A., Hanse S., Schawabe S., Gaedke M., White B., "Web engineering", J. Web Eng. 1 (1) (2002) 3–17.

[Dragicevic et al. 2014] Dragicevic, S., Celar, S., Novak, L.; "Use of Method for Elicitation, Documentation, and Validation of Software User Requirements (MEDoV) in Agile Software Development Projects" in: 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks, IEEE, 2014: pp. 65–70.

[Duddy et al. 2010] Duddy, Keith; Henderson, Michael; Metke-Jimenez, Alejandro; Steel, Jim. 2010. Design of a Model-generated Repository As a Service for USDL. Proceedings of the 12th International Conference on Information Integration and Web-based Applications Services, pp. 707-713. New York, NY, USA. ACM.

[Dybá & Dingsoyr 2008] T. Dybá, T. Dingsoyr, Empirical studies of agile software development: a systematic review, InfSoftwTechnol 50 (9–10) (2008) 833–859.

[Elyacoubi et. al 2009] Elyacoubi, Naima Erchidi; Belouadha, Fatima-Zahra; Roudies, Ounsa. A metamodel of WSDL Web services using SAWSDL semantic annotations. 7th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-09). Rabat, MOROCCO (13-10 Mayo, 2009).

[Emig et al. 2007] Emig, Christian; Krutz, Karsten; Link, Stefan; Momm, Christof; Abeck, Sebastian. 2007. Model-driven development of SOA services. Universit at Karlsruhe (TH). Karlsruhe 2007.

[Escalona & Knoch 2007] Escalona, M. J., Koch, N. . Metamodeling the requirements of web systems. In Web Information Systems and Technologies (pp. 267-280). Springer Berlin Heidelberg. 2007

[Escalona & Knoch 2012] Escalona, M.J., Koch, N.: Requirements Engineering for Web Applications: A Comparative Study. Journal on Web Engineering 2(3), 193–212 (2012)

[Escalona 2004] Escalona, M.J.: Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software. Ph. Thesis University of Seville (October 2004)

[Escalona et al. 2008] Escalona MJ, Aragón G. *NDT. A model-driven approach for web requirements*. IEEE Transactions on software engineering, vol: 34, pp. 377-394. 2008.

[Galster et al. 2013] Galster M., Laurens L., Paris A., 2013. Service-oriented architecture in variability-intensive environments: pitfalls and best practices in the example of local e-government. IEEE Software. IEEE computer Society Digital Library. IEEE Computer Society.

[Gichoya 2005] Gichoya, David. "Factors affecting the successful implementation of ICT projects in government." the Electronic Journal of e-government 3.4 (2005): 175-184.

[Gill et al. 2016] Gill, Asif Qumer; Henderson-Sellers, Brian; Niazi, Mahmood. 2016. Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. Information Systems Frontiers. Vol.1. Springer. http://dx.doi.org/10.1007/s10796-016-9672-8.

[Glazer et al. 2008] H. Glazer, J. Dalton, D. Anderson, M. Konrad, S. Dhrum, "CMMI or agile: why not embrace both!", in: CMU/SEI-2008-TN-003, Pittsburgh, 2008.

[Goudos et al. 2007] Goudos S. K., Loutas N., Peristeras V., Tarabanis K., 2007. Public Administration Domain Ontology for a Semantic Web Services E-Government Framework Services Computing. SCC 2007. IEEE International Conference on, vol., no., pp.270-277.

[Hahn & Slomic 2008] Hahn, Christian ; Slomic, Ismar. 2008. Agent-based Extensions for the UML Profile and Metamodel for Service-oriented Architectures. 12th Enterprise Distributed Object Computing Conference Workshops. IEEE.

[Hahn et al. 2008] Hahn, Christian; Slomic, Ismar. 17th International Conference on Information Systems Development Ubicación: Univ Cyprus, Dept Comp Sci, Paphos, CYPRUS Fecha: AUG 25-27, 2008.

[Hall 2007] Hall J. L., 2007. Implications of success and persistence for public sector performance. Public Organization Review, 7(3), 281-297.

[Hock-Koon & Oussalah 2010] Hock-Koon, Anthony; Oussalah, Mourad. 2010. Composite service metamodel and auto composition. Journal of Computational Methods in Sciences and Engineering. Vol. 10, pp. 215-229. IOS Press.

[Hynes 2015] Haynes, Philip. Managing complexity in the public services. Routledge, 2015.

[Inayat et al. 2015] I. Inayat, S.S. Salim, S. Marczak, M. Daneva, S. Shamshirband.; A systematic literature review on agile requirements engineering practices and challenges, Computers in Human Behavior. 51 (2015) 915–929.

[Insfrán et al. 2002] Insfrán, E., Pastor, O., Wieringa, R.: Requirements Engineering-Based Conceptual Modelling. Requirements Engineering Journal 7(1) (2002)

[ISO/IEC 2012] ISO/IEC. ISO/IEC 19507:2012 Information technology - Object Management Group Object Constraint Language (OCL). International Organization for Standardization, formal/2012-05-09, 2012.

[IWT2 2008] "Comparativa herramientas de modelado". Consejería de Cultura de la Junta de Andalucía. Proyecto Calidad. (2008). http://www.iwt2.org. Ultimo acceso julio 2014.

[IWT2 2017] Grupo de Investigación "Ingeniería Web y Testing Temprano" de la Universidad de Servilla. http://www.iwt2.org. Ultimo acceso diciembre de 2016.

[Kappel et al. 2003] Kappel, G., Pröll, B., Reich, S., Retschizegger, W.: Web Engineering, dpunkt Verlag (2003)

[Kitchenham & Brereton 2013] Kitchenham, B. y Brereton, P. (2013). A systematic review of systematic review process research in software engineering. Information and Software Technology, 55(12):2049 – 2075.

[Kitchenham & Charters 2007] Kitchenham, B. y Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

[Koch & Kraus, 2002] Koch, N., Kraus, A.: The expressive Power of UML-based Web Engineering. In: Second Int. Workshop on Web-oriented Software Technology (IWWOST02), pp. 105–119. Málaga,Spain (2002)

[Koumaditis et al. 2013] Koumaditis K., Marinos T., Rupino Da Cunha P., 2013. SOA implementation critical success factors in healthcare. Journal of Enterprise Information Management, Vol. 26 Iss: 4, pp.343 - 362.

[Krogdahl et al. 2005] Krogdahl ,P., Luef, G., Steindl, C,: "Service-oriented agility: an initial analysis for the use of agile methods for SOA development," 2005 IEEE International

Conference on Services Computing (SCC'05) Vol-1, 2005, pp. 93-100 vol.2. doi: 10.1109/SCC.2005.86

[Kruchten 2004] Kruchten, P. The Rational Unified Process: An Introduction, 3rd ed., Addison-Wesley, 2004.

[Lashkari et al. 2009] Lashkari, A.H., Mahdavi F. and Ghomi V., "A Boolean Model in Information Retrieval for Search Engines," Information Management and Engineering, 2009. ICIME '09. International Conference on, Kuala Lumpur, 2009, pp. 385-389. doi: 10.1109/ICIME.2009.101

[Lethrech et al. 2007] Lethrech, M.; Elmagrouni, I.; Nassar, M.; Kriouile, A.; Kenzi, A. 2013. A Generic Metamodel for Adaptable Service Oriented Systems Modeling using DSM Approach. 3rd International Symposium Isko-maghreb. ISKO.

[Ley 11/2007] Ley 11/2007 de Acceso Electrónico de los Ciudadanos a los Servicios Públicos. 2007. http://www.boe.es/boe/dias/2007/06/23/pdfs/A27150-27166.pdf. Último acceso diciembre 2016.

[Ley 39/2015] Ley 39/2015 del Procedimiento Administrativo Común de las Administraciones Públicas. 2015. https://www.boe.es/boe/dias/2015/10/02/pdfs/BOE-A-2015-10565.pdf. Último acceso diciembre 2016.

[Lowe & Hall 1999] Lowe, D., Hall, W.: Hypermedia and the Web. An Engineering approach. John Wiley & Son, Chichester (1999)

[Maguire 2013] Maguire, M.; "Using human factors standards to support user experience and agile design" in: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013.

[Mefteh & Benhassen 2015] Mefteh H., Lobna Benhassen L., "Impact of Information Technology and Communication on Economic Growth". International Journal of Economics, Finance and Management. VOL. 4, NO. 2, March 2015

[Mosaico 2012] Sistema de Gestión del Patrimonio Histórico de Andalucía. https://ws096.juntadeandalucia.es/mosaico/faces/jspx/portadaMosaico.jspx. Último acceso enero 2017.

[Näkki et al. 2011] Näkki, P., Koskela, K., Pikkarainen, M.; "Practical model for user-driven innovation in agile software development" in: Proceedings of the 2011 17th International Conference on Concurrent Enterprising (ICE 2011), IEEE, 2011: pp. 1–8.

[Nawrocki et al. 2014] Nawrocki, J., Ochodek, M., Jurkiewicz, J., Kopczyńska, S., Alchimowicz, B.; "Agile requirements engineering: A research perspective" in: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2014: pp. 40–51.

[Ninfa 2009] Sistema de Gestión para el Registro del Propiedad Intelectual de Andalucía. https://ws096.juntadeandalucia.es/ninfa . Último acceso enero 2017.

[OASIS 2006] Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 12 October 2006. http://docs.oasis-open.org/soa-rm/v1.0/. Último acceso enero 2015.

[OMG 2011] OMG. Meta Object Facility (MOFTM) Core. Object Management Group, http://www.omg.org/spec/MOF/. 2011b.

[Ott et al. 2011] Ott, Christian; Korthaus, Axel; Bauhmann, Tilo; Rosemann, Michael; Krcmar, Helmut. 2011. Foundations of a Reference Model for SOA Governance. Information Systems Evolution. Springer. http://dx.doi.org/10.1007/978-3-642-17722-4_4

[Palmer 2002] S.R. Palmer, A Practical Guide to Feature-Driven Development, Prentice-Hall, NJ, 2002.

[Peinado et al. 2015] Peinado, S., Ortiz, G., Dodero, JM.A.: A metamodel and taxonomy to facilitate context-aware service adaptation. COMPUTERS & ELECTRICAL ENGINEERING (Vol: 44 Pág: 262-279). DOI: 10.1016/j.compeleceng. .2015.02.004 Mayo de 2015.

[Pérez et al. 2011] Pérez García F., Cucarella Tormo V., Fernández García A., Hernández Lahiguera L. "Las Diferencias Regionales del Sector Público Español". Madrid. Fundación BBVA.

[Pikkarainen et al. 2008] M. Pikkarainen et al., The Impact of Agile Practices on Communication in Software Development, Empirical Software Engineering, Springer, 2008 (May 2008).

[Piprani et al. 2008] Piprani, Baba; Wang, Chong; He, Keqing. 2008. A Metamodel for Enabling a Service Oriented Architecture. On the Move to Meaningful Internet Systems: OTM 2008.

[Popescu et al. 2013] Popescu D., Nirvana P., Ciprian D., 2012. E-Frameworks to Optimize Public Administration Services. Digital Democracy: Concepts, Methodologies, Tools and Applications (3 Vols.). IGI Global.

[Poppendieck 2003] Poppendieck, M.; Poppendieck, T. "Lean Software Development. An Agile Toolkit". Boston: Addison-Wesley. 2003.

[Postina et al. 2010] Postina, M.; Trefke, J.; Steffens. 2010. An EA-approach to Develop SOA Viewpoints. 14th IEEE International Enterprise Distributed Object Computing Conference2010, pp. 37-46.

[Pressman 2000] Pressman, R.S. "What a Tangled Web We Weave". IEEE Software, pp. 18-21. Jan.-Feb. 2000.

[Ramadour et al. 2010] Ramadour, P.; Cauvet, C. & Ferrarini, A. 2010. Towards services paradigm: principles and models. 4th International Conference on Research Challenges in Information Science (RCIS), pp. 109-120.

[Ran et al. 2008] Ran, H. et al. 2008. "Agile Web Development with Web Framework". In proceedings of 4th International Conference on Wireless Communications, Networking and Mobile Computing. (Dalian, China, October 12 – 17 2008) WiCOM '08. IEEE.

[Repa 2006] Repa V., 2006. Modeling business processes in public administration. Advances in information systems development (pp. 107-118). Berlin. Springer.

[Rossi et al. 2016] Rossi G., Urbieta M., Distante D., Rivero J.M., Firmenich S. 2016. "25 Years of Model-Driven Web Engineering. What we achieved, What is missing". Clei Electronic Journal Vol 19, nº 3, paper 1. Diciembre 2016.

[Rivero et al. 2014] Rivero, J.M., Grigera, J., Rossi, G., Robles Luna, E., Montero, F., Gaedke, M.; "Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering" . Information and Software Technology. 56 (2014) 670–687.

[Ruas de Oliveira et al. 2010] Ruas de Oliveira L. B., Romero K., Feitosa D., Yumi E., 2010. Reference Models and Reference Architectures Based on Service-Oriented Architecture: A Systematic Review. ECSA2010. In proceeding of: Software Architecture, 4th European Conference, Copenhagen, Denmark. Proceedings Source: DBLP.

[Salton et al. 1975] G. Salton, A. Wong, and C. S. Yang. (1975): A vector space model for automatic indexing. Commun. ACM 18, 11 (November 1975), 613-620. DOI=http://dx.doi.org/10.1145/361219.361220

[Schön 2016] Schön, E.M.; "A Methodology for Agile Requirements Engineering based on a Pattern Approach" CAiSE 2016 Doctoral Consortium, At Ljubljana, Slovenia. June 2016.

[Schön et al. 2015] Schön, E.M.; Escalona, M.J.; Thomaschewski, J. "Agile Values and Their Implementation in Practice," International Journal of Interactive Multimedia and Artificial Intelligence. 3 (2015) 61–66.

[Schön et al. 2017] Schön, E.M.; Thomaschewski, J.; Escalona, M.J.; "Agile requirements engineering: A Systematic Literature Review". Computer Standards & Interfaces. Vol. 49, pp. 79-91. http://dx.doi.org/10.1016/j.csi.2016.08.011. Jenuary 2017.

[Schwabe & Rossi 2005] Schwabe, D., Rossi, G.: An Object Oriented Approach to Web-Based Application Design. In: Theory and Practice of Object Systems, vol. 4(4), Wiley and Sons, New York, USA (1998)

[Schwaber 1996] Schwaber, K. "Controlled Chaos: Living on the Edge". American programmer. April 1996.

[Schwaber 1997] Schwaber, K. SCRUM Development Process, in: J. Sutherland, C. Casanave, J. Miller, P. Patel, G. Hollowell (Eds.), Business Object Design and Implementation, London, 1997.

[Schwaber 2004] Schwaber, K. Agile Project Management with Scrum. Microsoft, 2004.

[Sedeño 2012] Sedeño J., "Implantación de una arquitectura SOA en Administraciones Públicas", Máster de Gestión de las Tecnologías de la Información y las Comunicaciones 2011-2012.Trabajo Fin de Máster. Universidad de Sevilla. Julio 2012.

[Sedeño et al. 2013] Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "Propuesta para la transformación de Administraciones Públicas en organizaciones basadas en SOA". XVIII Jornadas de Ingeniería del Software y Bases de Datos, pp. 105-110. (Madrid, Esp.). 2013.

[Sedeño et al. 2014a] Sedeño J., Torrecilla-Salinas C.J., Escalona M.J., Mejías M. "An approach to transform Public Administration into SOA-based organizations". 10th International Conference on Web Information Systems and Technologies, pp. 135-142 (Barcelona, Esp., 2-4 Abr 2014).

[Sedeño et al. 2014b] Sedeño J. ,Torrecilla Salinas, C.J., Escalona,M.J., Mejías,M. "Propuesta de modelado de requerimientos en paradigmas de Ingeniería Web Ágil guiada por modelos". XIX Jornadas de Ingeniería del Software y Bases de Datos, pp. 273-278. (Cadiz, Spain, 16-19 Sep 2014).

[Sedeño et al. 2014c] Sedeño J. Escalona,M.J., Mejías,M. "An approach to extend NDT in the development of Web Applications into Services based organizations". 17 International Conference on Model Driven Engineering Languages and Systems (Valencia, Spain 1-3 Oct 2014)

[Silva-Lepe et al. 2008] Silva-Lepe, I; Subramanian, R; Rouvellou, I; Mikalsen, T; Diament, J; Iyengar, A.: SOAlive Service Catalog: A Simplified Approach to Describing, Discovering and Composing Situational Enterprise Services. 6th International Conference on Service-Oriented Computing. Sydney, AUSTRALIA (01-05, Dic, 2008).

[Soares et al. 2015] H.F. Soares, N.S.R. Alves, T.S. Mendes, M. Mendonca, R.O. Spinola, Investigating the Link between User Stories and Documentation Debt on Software Projects, in: 2015 12th International Conference on Information Technology - New Generations, IEEE, 2015: pp. 385– 390.

[SparxSystems 2016] SparxSystems. Enterprise Architect. Sitio web: http://www.sparxsystems.com.au, último acceso agosto 2016.

[Sutherland & Schwaber 2011] Sutherland, J.; Schwaber, K. "The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game". 2011. http://www.scrum.org/Scrum-Guides. Accessed 2013.

[Takeuchi et al. 1986] Takeuchi, H,; Nonaka, I.; Takeuchi, H.: "The new new product development game". Harvard Business Review. 64 (1986) 137–146.

[Torrecilla et al. 2012] Torrecilla-Salinas, C. J., Escalona, M. J., & Mejías, M.. A scrum-based approach to CMMI maturity level 2 in web development environments. In Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services (pp. 282-285). ACM. (2012, December)

[Torrecilla et al. 2013a] Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. Agile in Public Administration: Oxymoron or reality? An experience report. CAiSE. Valencia , Jul 2013

[Torrecilla et al. 2013b] Torrecilla-Salinas, C.J.; Sedeño J.; Escalona M.J.; Mejías M. "Using an Agile framework to deliver e-Government services in public administrations". 21nd Annual Software Quality Management (London, 2-5, Sep, 2013). SQM2013.

[Torrecilla et al. 2014a] Torrecilla-Salinas, C.J.; Sedeño J.; Escalona M.J.; Mejías M. "Using Agile methods for infrastructure projects: A practical experience". 22nd International Conference on Information Systems Development (Seville, 2-4, Sep, 2013).

[Torrecilla et al. 2014b] Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M.,"Mapping Agile Practices to CMMI-DEV Level 3 in Web Development Environments." International Conference on Information Systems Development (ISD). Croatia 2014

[Torrecilla et al. 2015] Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2015). "Estimating, planning and managing Agile Web development projects under a value-based perspective". Information and Software Technology, Volume 61, Pages 124-144**.**

[Torrecilla et al. 2016] Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2016)., "*Agile, Web Engineering and Capability Maturity Model Integration: A systematic literature review*", Information and Software Technology, Volume 71 Pages 92-107.

[Tregear & Jenkins 2007] Tregear R., Jenkins T., 2007. Government process management. http://www.w.bptrends.com/publicationfiles/10-07-ART-Govt. ProcessMgt. -Tregear and Jenkins-ph.pdf.

[Turbo 2016] Sistema para la obtención del Carnet de Bibliotecas Públicas de la Junta de Andalucía. https://ws096.juntadeandalucia.es/tarjetaUsuarioBibliotecas. Último acceso enero 2017.

[W3C 2004] W3C. Word Wide Web Consortium. Web Services. http://www.w3.org/TR/ws-arch/. Último acceso enero 2015.

[Wieringa 2005] Wieringa, R.: Requirement Engineering: Problem Analysis and Solution Specification. In: Sri- kanthan, T., Xue, J., Chang, C.-H. (eds.) ACSAC 2005. LNCS, vol. 3740, pp. 13–16. Springer, Heidelberg (2005).

[Wohlin & Prikladniki 2013] Wohlin, C. y Prikladniki, R. (2013). Systematic Literature Reviews in Software Engineering. Information and Software Technology, (0).

[Zhang et al. 2011] Zhang, H., Babar, M. A., y Tell, P. (2011). Identifying relevant studies in software engineering. Information and Software Technology, 53(6):625–637.

[Zhang y Ali Babar 2013] Zhang, H. y Ali Babar, M. (2013). Systematic reviews in software engineering: An empirical investigation. Information and Software Technology, 55(7):1341– 1354.

Anexo I Servicios de Administración Electrónica

Como se indicaba en el Capítulo VIII, en este anexo se exponen los Servicios pertenecientes al Dominio de Administración Electrónica y que forman parte del Catálogo de Servicios de la Consejería Cultura. De forma análoga a como se estructuró la información con las entidades, atributos y valores para la representación de los requisitos ágiles basados en valor presentados en el Capítulo VIII, se estructurará mediante una tabla, sólo aquella información de dichos Servicios que ha de ser indexada en el motor de búsqueda a fin de realizar el proceso para el descubrimiento de Servicios.

1. Información de los Servicios a indexar

En este único apartado del Anexo se describirán, como se ha indicado, mediante una tabla, los Servicios de Administración Electrónica que han serán indexados en el motor de búsqueda y que han sido utilizados en la validación presentada en el Capítulo VIII. Debido a su descripción textual y auto comprensiva no será necesario realizar ninguna aclaración adicional sobre los mismos.

Tabla AI.1. Servicio de Notificaciones Telemáticas.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{ "Notificaciones", "Abonados", "Procedimiento Administrativo" };
Service	name	"Notificaciones Telemáticas"
	description	"Envío de notificaciones telemática fehacientes a los ciudadanos"
Operation	*name*	{"altaAbonado", "bajaAbonado", "envioNotificacionAbonado", "isLeida" }
	description	{ "Alta de un abonado en el sistema de notificaciones", "Baja de un abonado en el sistema de notificaciones", "Envío de una notificación a un abonado", "Indica si un abonado a leído la notificación" }
Stakeholder	*name*	{ "Servicio Informática", "Consejería de Hacienda" }
	role	{ "Responsable", "Dueño" }
Policy	*policy*	{ "Preguntar a Hacienda para su uso" }
ServiceLevelAgreement	*kpi*	{ "Una notificación cada 15 minutos", "El código de notificación está dado de alta en Hacienda" }

Tabla AI.2. Servicio de Registro Telemático de entrada y salida.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{ "Registro", "Entrada", "Salida", "Intranet", "Extranet", "Aries" }
Service	name	"Registro Telemático de entrada y salida"
	description	"Registro telemático único para los asientos de entrada y salida tanto Intranet como Extranet"
Operation	*name*	{ "altaRegistroEntradaIntranet", "altaRegistroEntradaExtranet", "altaRegistroSalidaIntranet", "altaRegistroSalidaEXtranet" }
	description	{ "Alta de un registro de entrada en el libro para intranet", "Alta de un registro de salida en el libro para intranet", "Alta de un registro de entrada en el libro para extranet", "Alta de un registro de salida en el libro para intranet" };
Stakeholder	*name*	{ "Servicio de Informática", "Consejería de Hacienda" }
	role	{ "Responsable", "Dueño" }
Policy	*policy*	{ "Debe tener la IP dada de alta" }
ServiceLevelAgreement	*kpi*	{ "Cada asunto diferente ha de estar registrado" }

Tabla AI.3. Servicio de Login con certificado en la Fachada de Ticket.

Entidad	Atributo	Valor
Domain	description	"Firma Electrónica"
Tag	*tag*	"Firma", "Fachada", "Certificado", "@FIRMA", "Tickets" , "Certificado"
Service	name	"Login con certificado en la Fachada de Tickets"
	description	"El usuario puede logarse con su certificado electrónico haciendo uso de la fachada de tickets"
Operation	*name*	{ "getTicket", "getCertificado", "generarURL" }
	description	"Devuelve un ticket que estará en la plataforma de firma electrónica ", "Devuelve el certificado del usuario", "Compone la URL para la llamada a la plataforma de firma electrónica"
Stakeholder	*name*	{ "Servicio de Informática", "Consejería de Hacienda" }
	role	{ "Responsable", "Dueño" }
Policy	*policy*	{"Alta del usuario en la Consejería de Hacienda", "Para aplicaciones que no tienen firma electrónica pero necesitan login con certificado"}
ServiceLevelAgreement	*kpi*	{ "Los reintentos son cada 30 minutos" }

Tabla AI.4. Servicio de Firma electrónica con certificado digital.

Entidad	Atributo	Valor
Domain	description	"Firma Electrónica"
Tag	*tag*	{“Firma", "Cofirma", "Contrafirma", "@FIRMA", "Certificado Electrónico"}
Service	name	{"Alta del usuario en la Consejería de Hacienda"}
	description	{"El certificado electrónico tiene que ser válido y reconocido" }
Operation	*name*	{"firma", "cofirma", "contrafirma" }
	description	{"Firma electrónica con certificado de un documento", "Cofirma electrónica con certificado de un documento con certificado electrónico", "Contrafirma electrónica con certificado o firma en cascada "}
Stakeholder	*name*	{"Servicio de Informática", "Consejería de Hacienda" }
	role	{"Responsable", "Dueño" }
Policy	*policy*	{"Alta del usuario en la Consejería de Hacienda"}
ServiceLevelAgreement	*kpi*	{"El certificado electrónico tiene que ser válido" }

Tabla AI.5. Servicio de Firma Electrónica de Servidor.

Entidad	Atributo	Valor
Domain	description	"Firma Electrónica”
Tag	*tag*	{"Firma", "Cofirma", "Contrafirma", "@FIRMA", "Servidor" }
Service	name	"Firma Electrónica de Servidor"
	description	"Se firma la documentación en Servidor ya sea firma, cofirma o contrafirma"
Operation	*name*	{"firmaServidor", "cofirmaServidor", "contrafirmaServidor" }
	description	{"Firma electrónica de servidor", "Cofirma electrónica de servidor de un documento con certificado de sello", "Contrafirma electrónica de servidor o firma en cascada con certificado de sello”}
Stakeholder	*name*	{"Servicio de Informática", "Consejería de Hacienda" }
	role	{"Responsable", "Dueño" }
Policy	*policy*	{"Alta del usuario en la Consejería de Hacienda"}
ServiceLevelAgreement	*kpi*	{"El certificado electrónico tiene que ser válido", "Tiene que ser un certificado de sello"}

Tabla AI.6. Servicio de Envío a Portafirmas.

Entidad	Atributo	Valor
Domain	description	"Firma Electrónica"
Tag	*tag*	{ "Firma", "Portafirmas", "Informe de Firma", "@FIRMA" }
Service	name	"Envío a Portafirmas"
	description	"Envío de documentación para la firma en el Sistema Portafirmas"
Operation	*name*	{ "envioPortafirma", "añadirFirmante", "descargarInformeFirma"}
	description	{"Envío de documentación para firmar en portafirmas", "Añadir un firmante a la petición enviada al Portafirmas", "Descargar el Informe de firma, que es la representación física de la firma"}
Stakeholder	*name*	{ "Servicio de Informática"}
	role	{ "Dueño" }
Policy	*policy*	{"El documento ha de estar en formato ENI"}
ServiceLevelAgreement	*kpi*	{ "Sólo pueden firmar funcionarios y personal de la administración" }

Tabla AI.7. Servicio de Sellado de documentos.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{ "Sello", "Registro" , "Documentos"}
Service	name	"Sellado de documentos"
	description	"Servicio diseñado para estampar un sello con los valores del registro de entrada y salida sobre un documento PDF"
Operation	*name*	{ "sellarDocumento"}
	description	{ "Se envía un documento con los parámetros de fecha, hora, servicio y número de registro y devuelve un documento sellado"}
Stakeholder	*name*	{ "Servicio de Informática"}
	role	{ "Dueño" }
Policy	*policy*	{ "El documento ha de estar en formato PDF" }
ServiceLevelAgreement	*kpi*	{ "Se puede usar en versión standalone" }

Tabla AI.8. Servicio de Generación de Documentos con Plantill@.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{ "Generación", "Documentos", "Plantillas" , "Plantill@"}
Service	name	"Generación de Documentos con Plantill@"
	description	"Genera un documento PDF a partir de una plantilla en ODT y un mapa con los campos y los valores que se sustituirán en la plantilla"
Operation	*name*	{ "generarDocumentoPlantilla"}
	description	{"Se genera un documento en PDF dada una plantilla de Plantill@ y un mapa con los valores"}
Stakeholder	*name*	{ "Servicio de Informática"}
	role	{ "Dueño" }
Policy	*policy*	{ "El documento ha de estar en formato ODT" }
ServiceLevelAgreement	*kpi*	{ "N/A" }

Tabla AI.9. Servicio de Generación de Formularios con Formul@.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{ "Generación", "Formularios", "Formul@" }
Service	name	"Generación de Formularios con Formul@"
	description	"Se diseña un formulario con Formul@ que puede ser invocado y usado en una aplicación "
Operation	*name*	{"cargaFormualrio", "guardarFormulario","recuperarFormulario"}
	description	{"Carga en la aplicación un formulario diseñado con la herramienta Formul@", "Se guardan los datos de ese formulario en un mapa y persiste en XML", "Se recuperar un formulario con los datos que hay almacenados en el sistema"}
Stakeholder	*name*	{ "Servicio de Informática"}
	role	{ "Dueño" }
Policy	*policy*	{ "El formulario tiene que llevar asociado el CSS para el estilo" }
ServiceLevelAgreement	*kpi*	{ "N/A" }

Tabla AI.10. Servicio de Publicación en BOJA.

Entidad	Atributo	Valor
Domain	description	"Administración Electrónica"
Tag	*tag*	{"BOJA", "Publicación", "Insértese", "Disposición" }
Service	name	"Publicación en BOJA"
	description	"Servicio para la publicación de resoluciones en el BOJA"
Operation	*name*	{ "envioDisposición", "envioInsertese", "consultaDisposicion" }
	description	{ "Envío de la disposición firmada por el alto cargo", "Envío de la orden de inserción en BOJA firmada por el insertaste del organismo", "Consulta del estado de la disposición"}
Stakeholder	*name*	{ "Servicio de Informática", "Consejería de Presidencia" }
	role	{ "Responsable", "Dueño" }
Policy	*policy*	{"Se ha de solicitar un usuario en Presidencia"}
ServiceLevelAgreement	*kpi*	{ "Los insertantes deben estar dados de alta previamente en Presidencia" }

Anexo II Actividad Profesional e Investigadora

En este anexo se refleja la trayectoria profesional e investigadora del autor de la presente Tesis Doctoral, haciendo hincapié en el periodo de desarrollo de este trabajo de investigación, especialmente en lo que se refiere a la trayectoria profesional durante los años de gestación del presente trabajo, así como las actividades académicas más relevantes de dicho periodo.

En la trayectoria profesional se hará un breve esbozo del currículo del autor para pasar a continuación a describir aquellas tareas, de su actividad habitual, de relevancia para el presente trabajo de Tesis Doctoral.

Dentro da la actividad investigadora del doctorando, se recopila, catalogándola en diferentes apartados: participación en eventos de divulgación científica, publicaciones, proyectos y redes de investigación en las que ha participado.

En cuanto a las publicaciones, éstas han sido catalogadas según su tipología: capítulo de libro, artículo de revista, artículo publicado en conferencia internacional y artículo publicado en conferencia nacional.

1. Actividad Profesional

El autor es Ingeniero Superior en Informática desde 2002, obteniendo el grado de Master en la Universidad de Sevilla (España) en 2012 con el Master de Gestión de las Tecnologías de la Información y Comunicaciones. Ha trabajado como Analista, Consultor y Jefe de Proyecto en varias organizaciones privadas; nacionales e internacionales. Desde el 2006 trabaja como Responsable de la Oficina de Administración Electrónica de la Consejería de Cultura (Junta de Andalucía, España) y combina su trabajo con la investigación, en el seno del grupo de investigación Ingeniería Web y Testing Temprano (IWT2) de la Universidad de Sevilla, en las áreas de Metodologías Ágiles, Gobierno Electrónico, Servicios e Ingeniería del Software.

Durante el periodo de esta Tesis Doctoral, el autor ha ejercido, como Jefe de Departamento de la Agencia Andaluza de Instituciones Culturales (Consejería de Cultura), los siguientes trabajos, los cuáles han sido especialmente relevantes para el objeto de la Presente Tesis doctoral.

- Responsable de la Oficina de Administración Electrónica; tareas de coordinación, estrategia, análisis y desarrollo de las plataformas de firma electrónica y administración electrónica de la Junta de Andalucía (@firma, @ries, trew@, Solicit@, eCO, ptW@nda y VEA). Racionalización de procedimientos administrativos. Legislación en materia de administración electrónica. Implementación de tramitadores de procedimientos administrativos.
- Gestión de Proyectos Ágiles; Gestión de Proyectos mediante Metodologías Ágiles tanto en la parte de planificación ("*Scrum*") como en la de gestión económica (basadas en la técnica EVM-Agile, "*Earned Value Management*") usando métricas de productividad y técnicas de retrospectiva de proyectos ágiles como "*satisfaction histogram*", "*radar*", "*color dot voting"*, "*Ishikawa diagrams*" o "*5 whys*"). Entre estos proyectos destacan la implantación del BOJA

electrónico y el replanteo de la infraestructura de servicios hardware y software de la Consejería de Cultura.

- Responsable de la Oficia de Interoperabilidad. Implantación de la metodología SOA en la organización. Integración del paradigma SOA. Responsable del Catálogo de Servicios de la organización y de la arquitectura de Interoperabilidad del mismo. Participa también el la implantación del ESB ("*Enterprise Services Bus*") y de la arquitectura del producto WSO2 para el gobierno, operación y desarrollo de los Servicios.

Así mismo y en menor relación con el presente trabajo de tesis Doctoral también ejerce las siguientes tareas y funciones:

- Coordinador Técnico del Registro de la Propiedad Intelectual. Coordinador del Sistema de Gestión del Registro de la Propiedad Intelectual de Andalucía [Ninfa 2009], técnico adscrito al Registro Central del Ministerio de Educación, Cultura y Deporte en materia de Propiedad Intelectual.
- Implantación de una Arquitectura de integración continua basada en "*Subversión", "Artifactory", "Jenkins", "Maven"* y "*Sonar*" para el departamento de desarrollo. Coordinador de la formación de Administración Electrónica y JEE.
- Implantación de ERP ("*Enterprise Resource Planning*"). Implantación de la solución "*Epsilon RH*" para Nómina, Formación y Gestión del Tiempo y de "*Microsoft Dynamics NAV*" para la gestión económico-financiera.
- Diseño de cuadros de mando integrales (CMI) dentro de la disciplina de BI ("*Bussines Inteligence*") para la toma de decisiones de la dirección con la herramienta Qlik.

2. Eventos de interés científico

Durante la elaboración de esta Tesis Doctoral ha participado y colaborado en los siguientes eventos de divulgación científica:

- Durante 2016-2017 formó parte del comité de programa como experto en *Agile* del LASD 2017 (*«1st International Conference on Lean and Agile Software Development»* celebrado en Praga (República Checa).
- Durante 2016-2017 formó parte del comité de programa de la conferencia APMDWE 2017 (*«2nd International Workshop on Advanced practices in Model-Driven Web Engineering»)* dentro de la conferencia *«13th International Conference on Web Information Systems and Technologies»* celebrada en Oporto (Portugal).
- Durante 2016-2017 formó parte del comité local de organización de la conferencia PLM 2017 (*«14th International Conference on Product Lifecycle Management»)* celebrada en Sevilla (España).
- Durante el 2016, dentro de la revista de ELSEVIER, TECHNOVATION, *The International Journal of Technological Innovation, Entrepreneurship and Technology Management*, actuó como revisor del área de metodologías ágiles.
- En junio de 2016 formó parte del comité local de organización de la conferencia PAAM'S 2016 (*«14th International Conference on Practical Applications of Agents and Multi-Agent Systems»)* celebrada en Sevilla (España).

- Durante el 2015 formó parte del comité de programa de la conferencia MODELS 2015 (*«ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems»*) dentro del track del Doctoral Symposium que tuvo lugar en Ottawa (Canadá).
- En octubre de 2014 participó en el programa del Doctoral Symposium en el Congreso MODELS 2014 (*«ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems»*) que se celebró en Valencia (España).
- En septiembre de 2014 participó en el programa del Doctoral Symposium en el Congreso JISBD 2014 (*«XIX Jornadas de Ingeniería del Software y Bases de Datos»*) que se celebró en Cádiz (España).
- En abril 2014, entró a formar parte del comité científico como investigador asesor de la revista iberoamericana GTI (Gerencia Tecnológica Informática). Además de realizar tareas de difusión, esta participación ha implicado la revisión de diferentes artículos de investigación enviados a la revista antes mencionada.
- En el año 2013 formó parte del comité organizador de la conferencia ISD 2013 (*«International Conference on Information Systems Development»*) que se llevó a cabo en Sevilla (España).

3. Publicaciones

Los siguientes apartados catalogan la producción investigadora del doctorando según la tipología de la misma. Además, esta producción se lista de manera cronológica descendente en cada apartado.

3.1. Capítulos de libro

C.J. Torrecilla-Salinas, J. Sedeño, M.J. Escalona, M. Mejías, "*Using Agile Methods for Infrastructure Projects: A Practical Experience*", Information System Development, Volume 1, Pages 459-471, Year 2014. ISBN: 978-3-319-07214-2. DOI: 10.1007/978-3-319-07215-9_37.

3.2. Revistas

C.J. Torrecilla-Salinas, J. Sedeño, M.J. Escalona, M. Mejías, "*Agile, Web Engineering and Capability Maturity Model Integration: A systematic literature review*", Information and Software Technology, Volume 71, March 2016, Pages 92-107, ISSN 0950-5849, doi:10.1016/j.infsof.2015.11.002

C.J. Torrecilla-Salinas, J. Sedeño, M.J. Escalona, M. Mejías, "*Estimating, planning and managing Agile Web development projects under a value-based perspective*", Information and Software Technology, Volume 61, May 2015, Pages 124-144, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2015.01.006.

3.3. Conferencias internacionales

Sedeño J., Schön E.M., Torrecilla-Salinas C., Thomaschewski, J., Escalona M.J., Mejías M. "*Eliciting Agile requirements using Context-based Persona Stories*". Presentada al 13th International Conference on Web Information Systems and Technologies. Porto (Portugal). April, 2017. Actualmente en proceso de revision.

Sedeño J., Escalona M.J., Mejías M. "*An approach to extend NDT in the development of Web Applications into Services based organizations*". In Proceedings of the Doctoral Symposium at MODELS'14, 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014). Valencia (Spain). October, 2014.

Sedeño J., Torrecilla-Salinas C., Escalona M.J., Mejías M. "*An Approach to Transform Public Administration into SOA-based Organizations*". In Proceedings of the 10th International Conference on Web Information Systems and Technologies, pages 135-142. DOI: 10.5220/0004794701350142. Barcelona (Spain). April, 2014.

Torrecilla-Salinas C.J., Sedeño J., Escalona M.J., Mejías M. *"Mapping Agile Practices to CMMI-DEV Level 3 in Web Development Environments*". Information Systems Development: Transforming Organisations and Society through Information Systems (ISD2014 Proceedings). ISBN: 978-953-6071-43-2. Varaždin (Croatia). September, 2014.

Torrecilla-Salinas C.J, Sedeño J., Escalona M.J., Mejías M. "*Agile in Public Administration: Oxymoron or Reality? An Experience Report*". Proceedings of the Industrial Track of the Conference on Advanced Information Systems Engineering 2013 (CAiSE'13). Valencia (Spain). June, 2013.

Torrecilla-Salinas C.J, Sedeño J., Escalona M.J., Mejías M. "*Using an Agile framework to deliver e-Government services in public administrations*". 21st Annual Software Quality Management. London (England). September, 2013.

3.4. Conferencias nacionales

Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "*Propuesta de modelado de requerimientos en paradigmas de Ingeniería Web Ágil guiada por modelos*". XIX Jornadas de Ingeniería del Software y Bases de Datos, pp. 273-278. Cádiz, (Spain), 16-19 Septiembre, 2014.

Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "*NDT-SOA, extensión de la metodología NDT para el desarrollo de Aplicaciones Web en organizaciones públicas basadas en SOA*". XIX Jornadas de Ingeniería del Software y Bases de Datos (Doctoral Consortium), pp. 273-278. Cádiz, (Spain), 16-19 Septiembre, 2014.

Torrecilla-Salinas C.J.; Sedeño J.; Escalona M.J.; Mejías M. "*Una aproximación Ágil a los niveles de madurez 2 y 3 de CMMI-DEV en entornos de desarrollo Web*". XIX Jornadas de Ingeniería del Software y Bases de Datos, pp. 273-278. Cádiz, (Spain), 16-19 Septiembre, 2014.

Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "Propuesta para la transformación de Administraciones Públicas en organizaciones basadas en SOA". XVIII Jornadas de Ingeniería del Software y Bases de Datos, pp. 105-110. Madrid, (Spain), 16-19 Septiembre, 2013.

4. Proyectos de Investigación

Durante la carrera investigadora, el doctorando ha trabajo en los proyectos de investigación denominados:

- **POLOLAS (TIN2016-76956-C3-2-R).** Explorando soluciones guiadas para sistematizar el aseguramiento de la calidad del software.
    - Investigador responsable: María José Escalona Cuaresma (Universidad Sevilla).
    - Entidad/es financiadora/s: Ministerio de Economía y Competitividad

- Nombre del programa: otros programas del plan nacional i+d, Ministerio de Economía y Competitividad
- Fecha de inicio: 01-01-2016 / 31-12-2019
- Cuantía total: 181.200,00 EUR.
- Calidad en que ha participado: Investigador/a

- **MeGUS (TIN2013-46928-C3-3-R).** Mecanismos Guiados en Etapas Tempranas para la Mejora del Software.
  - Investigador responsable: María José Escalona Cuaresma (Universidad Sevilla).
  - Entidad/es financiadora/s: Ministerio de Economía y Competitividad
  - Nombre del programa: otros programas del plan nacional i+d, Ministerio de Economía y Competitividad
  - Fecha de inicio: 01-01-2014 / 31-12-2016
  - Cuantía total: 148.830,00 EUR.
  - Calidad en que ha participado: Investigador/a

- **TEMPROS (TIN2010-20057-C03-02)**. Testing temprano y modelos de simulación híbrida en la producción software, cuyos datos son:
  - Investigador responsable: María José Escalona Cuaresma (Universidad Sevilla).
  - Entidad/es financiadora/s: Ministerio de Ciencia e Innovación
  - Nombre del programa: Otros programas del plan nacional i+d, ministerio de ciencia y tecnología.
  - Fecha de inicio: 01-01-2011 / 31-12-2013.
  - Cuantía total: 90.400,00 EUR
  - Calidad en que ha participado: Investigador/a.

5. Redes de Transferencia e investigación

Durante la carrera investigadora del doctorando, éste también ha formado parte de las siguientes redes de transferencia e investigación.

- **Redes nacionales:**
  - Red CaSA- Calidad del software Aplicada (TIN2010-12312-E).
  - TEDIS - Red Temática en Tecnologías para el Desarrollo Industrial de Software (TIN2011-15009-E).
  - Red Temática para el desarrollo de soluciones software de calidad en entornos PLM (SoftPLM: TIN2015-71938-REDT)
- **Redes internacionales:**
  - Red Temática Mexicana en Ingeniería del Software (244467) del Consejo Nacional de Ciencia y Tecnología de México en la convocatoria Conacit Redes Temáticas 2014.