UNIVERSIDAD DE DEUSTO

Facultad de Ingeniería – ESIDE



TESIS DOCTORAL

Método de Diseño para Desarrollar Composiciones de Servicios Dinámicas y Sensibles al Contexto de Ejecución

Presentada por Dña. LEIRE BASTIDA MERINO

dentro del Programa de Doctorado en CIENCIA DE LA COMPUTACIÓN

Dirigida por la Dra. REBECA CORTÁZAR GOICOECHEA

La Directora La Doctoranda

Bilbao, Mayo de 2008

RESUMEN

La motivación para la realización de esta tesis ha sido la relevancia hoy en día de la orientación a servicios, la evolución hacia un entorno más dinámico donde las soluciones software ofrecidas son más flexibles y adaptables al contexto de ejecución y las carencias metodológicas y tecnológicas presentes en el área de las composiciones de servicios.

Así, el objetivo de esta tesis es *solventar dichas limitaciones existentes en el diseño y desarrollo de servicios compuestos proporcionando una nueva metodología* apoyada en los principios de variabilidad del paradigma de Ingeniería de Líneas de Productos (PLE), y soportada por una herramienta que facilite su aplicación. Esta innovadora metodología ***ayuda al diseño y desarrollo de composiciones de servicios flexibles y dinámicas,*** capaces de autoconfigurarse y adaptarse en cada momento a las condiciones y características del contexto de ejecución.

Es reseñable que dicha metodología no pretende ser una metodología más de desarrollo de software, sino que su carácter innovador lo aporta el hecho de que se centra en la definición de aquellas características específicas de la composición de servicios. Para ello, describe en detalle las fases, procesos, tareas, pasos, roles y responsabilidades involucrados en el diseño y desarrollo de los servicios compuestos. El valor práctico, la aplicabilidad y la validez de esta nueva metodología de composición de servicios han quedado demostrados a través de su aplicación en un escenario industrial real realizado en el contexto de las telecomunicaciones y la automoción, obteniendo unos resultados satisfactorios.

ABSTRACT

This doctoral dissertation has been motivated by the current high relevant of service oriented paradigm, the evolution of software systems towards more dynamic and runtime-extensible environments where solutions need to be flexible and context-aware, and the still unresolved methodological and technical limitations about service compositions.

The main objective of this work is to *solve these constraints about supporting the development and implementation of service composition by providing a new methodology* based on variability techniques defined by Product Lines Engineering (PLE) paradigm; and also a tool that provides support to the usage of the methodology. This innovative methodology ***helps to the design and development of flexible and dynamic service compositions***, able to auto-configure depending on the execution environment.

It is worth noting this methodology does not aim to become another software development methodology; its main novelty is the focus on defining the specific characteristic of service compositions. For that, this work defines in detail a set of phases, processes, steps, roles and responsabilities related to the design and development of composed services. The practical value, the applicability and the validity of this new methodology has been proven valid in the evaluation, performed in a real industrial scenario that mixed telecommunications and automotion domains, and really satisfactory results have been obtained.

AGRADECIMIENTOS

Muchas personas han contribuido, de una u otra forma, a la elaboración de este trabajo. A todas ellas quiero expresar mi más sincero agradecimiento. De modo especial deseo expresar mi agradecimiento a la Dra. Rebeca Cortázar, cuyo inestimable apoyo e interés han hecho posible la realización de esta investigación doctoral. También quisiera mencionar a Dr. Jason Xabier Mansell, Vanesa González, Marisa Escalante, Estíbaliz Delgado y Alberto Berreteaga, cuya paciencia y comentarios me han servido de apoyo y ánimo en todo momento.

Deseo, asimismo, expresar mi reconocimiento al consorcio del proyecto europeo SECSE, en especial a Elisabetta di Nitto (CEFRIEL), Roberto Tola (Centre Research Fiat), Fco. Javier Nieto (European Software Institute), Bruno Lefever e Ivor Bosloper; por las valiosas aportaciones que han realizado para mejorar la presente investigación.

Finalmente, todo esto nunca hubiera sido posible sin el amparo incondicional de mis padres y de Juan Mora, que han estado a mi lado en los momentos difíciles y me han brindado comprensión y constante estímulo.

ÍNDICE DE CONTENIDOS

Capítulo 1: Introducción 1

1.1 Motivación e Interés del Trabajo 1

1.2 Hipótesis y Objetivos 5

1.2.1 Hipótesis 5

1.2.2 Objetivo General 6

1.2.3 Objetivos Específicos y Operacionales 8

1.3 Organización de la Tesis 9

Capítulo 2: Método de Trabajo 11

2.1 Métodos de Investigación en Ingeniería del Software 11

2.1.1 Método Investigación-Acción 14

2.2 Aplicación de Investigación-Acción en este Trabajo de Investigación 21

2.2.1 Elección de los Participantes en la Investigación 21

2.2.2 Etapas Seguidas durante la Investigación 23

Capítulo 3: Antecedentes y Análisis de las Soluciones Existentes 25

3.1 Composición de Servicios 25

3.1.1 Tipos de Composición de Servicios 26

3.1.2 Composición Dinámica de Servicios 28

3.2 Enfoques de Composición de Servicios 30
3.2.1 Enfoques Industriales 30
3.2.2 Enfoques Académicos 38
3.2.3 Comparativa de los Diferentes Enfoques 43
3.3 Ingeniería de Líneas de Producto 46
3.3.1 La Ingeniería de Líneas de Producto Aplicada al Software 48
3.3.2 Beneficios de la Ingeniería de Líneas de Producto 49
3.4 Modelado Formal de Metodologías 50
3.4.1 Descripción de SPEM 2.0 52
3.5 Marco de Trabajo NIMSAD 56
3.6 Valoración del Estado del Arte 60

Capítulo 4: Enfoque de la Tesis 63
4.1 Soporte a la Composición de Servicios 63
4.2 Ingeniería de Líneas de Producto Software aplicada a la Composición de Servicios 65

Capítulo 5: Método de Diseño para Composiciones de Servicios Dinámicas 69
5.1 Metodología de Composición de Servicios 69
5.1.1 Procesos de la Metodología de Composición 72
5.1.2 Roles de la Metodología de Composición 103
5.1.3 Productos de Trabajo de la Metodología de Composición 109
5.2 Herramienta de Soporte a la Metodología 116
5.3 Soporte a la Ejecución de Composiciones Dinámicas 117

Capítulo 6: Ejemplos de Aplicación de la Metodología Propuesta 121
6.1 Rational Unified Process 122
6.1.1 Introducción a RUP 122
6.1.2 Aplicación de la Metodología de Composición 125

6.2 Extreme Programming....132
6.2.1 Introducción a XP....132
6.2.2 Aplicación de la Metodología de Composición....135

Capítulo 7: Validación y Evaluación de la Metodología Propuesta....139
7.1 Validación de la Metodología de Composición....140
7.1.1 Definición del Escenario Industrial Real....141
7.1.2 Aplicación de la Metodología Propuesta al Escenario....144
7.1.3 Resultado de la Validación de la Metodología de Composición....172
7.2 Evaluación de la Metodología Propuesta....173
7.2.1 Evaluación....174
7.2.2 Conclusiones de la Evaluación....180

Capítulo 8: Conclusiones....183
8.1 Consecución de los Objetivos Operacionales....186
8.2 Futuras líneas de investigación....188

Capítulo 9: Bibliografía....191

Capítulo 10: Glosario....205

Anexo 1: Plantillas de la Metodología....211
A1.1 Modelo de Especificación Funcional....211
A1.2 Modelo de Especificación de la Composición....213
A1.3 Modelo de Especificación de los Puntos de Variación....214
A1.4 Modelo de Realización de la Composición....215

Anexo 2: Descripción de la Herramienta de Soporte a la Metodología Propuesta....219

Anexo 3: Modelos Ejecutables del Servicio Compuesto XTrip....225
A3.1 Modelo Ejecutable de la Composición....225
A3.2 Modelo Ejecutable de la Variabilidad....228

ÍNDICE DE FIGURAS

Figura 1-1: Inversión en SOA a nivel mundial en millones de dólares [IDC 2007b] 3

Figura 2-1: Desconexión entre Investigación y Práctica en Ingeniería del Software [Moody 2000] 12

Figura 2-2: Carácter cíclico del método Investigación-Acción 17

Figura 2-3: Dimensiones de Investigación-Acción en Sistemas de Información 21

Figura 2-4: Aplicación del método Investigación-Acción 22

Figura 3-1: Servicio Abstracto y Servicio Concreto 26

Figura 3-2: Ejemplo de una composición dinámica 28

Figura 3-3: Composición dinámica: el concepto de Re-binding 29

Figura 3-4: Composición dinámica: el concepto de Re-planning 29

Figura 3-5: Proceso WS-BPEL 32

Figura 3-6: Visión general de OWL-S 34

Figura 3-7: Relación entre OWL-S y WSDL 35

Figura 3-8: Bloques de control del modelo eFlow 37

Figura 3-9: Arquitectura de SELF-SERV [Benatallah et al. 2003] 40

Figura 3-10: Arquitectura del enfoque de Traverso y Pistore 42

Figura 3-11: Familia de productos de teléfonos móviles 46

Figura 3-12: Representación de la Variabilidad en el mundo real 47

Figura 3-13: Elementos del marco de trabajo NIMSAD 57

Figura 4-1: Modelado de la variabilidad 66

Figura 5-1: Metodología para diseñar composiciones de servicios 71

Figura 5-2: Especificación Arquitectural 73

Figura 5-3: Especificación Funcional 77

Figura 5-4: Composición en Tiempo de Diseño 83

Figura 5-5: Realización de la Composición 89

Figura 5-6: Generación de la Composición Ejecutable 97

Figura 5-7: Validación y Verificación 101

Figura 5-8: Visión global del diseño, desarrollo y ejecución de las composiciones de servicios 117

Figura 5-9: Entorno de ejecución de composiciones dinámicas 119

Figura 6-1: Rational Unified Process (Fuente: IBM Rational Software) 123

Figura 6-2: Flujo de trabajo de la disciplina Análisis y Diseño (Fuente: IBM Rational Method Composer v7.0) 126

Figura 6-3: Flujo de trabajo integrado con RUP para el Análisis y Diseño (Fuente: IBM Rational Method Composer v7.0) 127

Figura 6-4: Flujo de trabajo de la disciplina Implementación (Fuente: IBM Rational Method Composer v7.0) 128

Figura 6-5: Flujo de trabajo integrado con RUP para la Implementación (Fuente: IBM Rational Method Composer v7.0) 129

Figura 6-6: Flujo de trabajo de la disciplina Pruebas (Fuente: IBM Rational Method Composer v7.0) 130

Figura 6-7: Visión general de la metodología XP 133

Figura 7-1: Diagrama de casos de uso del servicio compuesto XTrip 148

Figura 7-2: Diagrama de casos de uso de ‘Calcular ruta’ 149

Figura 7-3: Diagrama de casos de uso de ‘Comprobar agenda’ 150

Figura 7-4: Diagrama de actividad del servicio compuesto XTrip..... 153

Figura 7-5: Asociación de servicios abstractos-concretos para el caso de prueba 1..... 168

Figura 7-6: Asociación de servicios abstractos-concretos para el caso de prueba 2..... 169

Figura 7-7: Asociación de servicios abstractos-concretos para el caso de prueba 3..... 170

Figura 7-8: Asociación de servicios abstractos-concretos para el caso de prueba 4..... 171

Figura 7-9: Asociación de servicios abstractos-concretos para el caso de prueba 5..... 172

Figura A2-1: Pantalla principal de la herramienta de soporte..... 219

Figura A2-2: Sub-menú 'Proyecto' de la herramienta 220

Figura A2-3: Sub-menú 'Documentos' de la herramienta 220

Figura A2-4: Información de la tarea ‘Analizar Puntos de Variación’..... 221

Figura A2-5: Pantalla de la tarea 'Identificar Requisitos Arquitectónicos'..... 222

Figura A2-6: Pantalla de la tarea ‘Analizar Puntos de Variación’ 222

ÍNDICE DE TABLAS

Tabla 1-1: Objetivos operacionales más relevantes y objetivo específico al que contribuyen 8

Tabla 2-1: Diferencias entre la Investigación Cualitativa y la Cuantitativa 13

Tabla 3-1: Comparativa de los principales enfoques de composición 44

Tabla 6-1: Correspondencia general entre la metodología propuesta y RUP 125

Tabla 6-2: Correspondencia de roles entre la metodología de composición y RUP 131

Tabla 6-3: Correspondencia de productos de trabajo entre la metodología propuesta y RUP 131

Tabla 6-4: Nuevos productos de trabajo a incluir en RUP 132

Tabla 6-5: Correspondencia general entre la metodología propuesta y XP 137

Tabla 6-6: Correspondencia de roles entre la metodología propuesta y XP 137

Tabla 7-1: Medidas de puntuación para los servicios concretos candidatos 159

Tabla 7-2: Cuestionario de valoración de la metodología 173

Tabla 7-3: Preguntas y respuestas sobre "la Situación del Problema" 175

Tabla 7-4: Preguntas y respuestas sobre "el Solucionador del Problema" 176

Tabla 7-5: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Formulación del Problema" 178

Tabla 7-6: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Diseño de la Solución" 179

Tabla 7-7: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Implementación del Diseño" 180

Tabla 8-1: Objetivos operacionales más relevantes y sección donde se abordan 187

Tabla 8-2: Relación entre los objetivos específicos y los objetivos operacionales 188

Tabla A2-1: Simbología utilizada en la herramienta 221

¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida más fácil, nos aporta tan poca felicidad? La respuesta es ésta: simplemente porque aún no hemos aprendido a usarla con tino.

Albert Einstein

CAPITULO 1

Introducción

1.1 Motivación e Interés del Trabajo

La orientación a servicios es una de las tecnologías más importantes de los últimos años, la cual está proporcionando oportunidades y beneficios a las empresas que deciden adoptar este nuevo enfoque.

En comparación con los paquetes de software instalados y los procesos de negocios fijos, la orientación a servicios proporciona importantes beneficios ya que facilita la integración de procesos de negocios en tiempo real, la interoperabilidad entre sistemas heterogéneos, la rapidez y el ahorro en el coste del desarrollo del producto a través de la reutilización de servicios proporcionados por terceras partes y el rápido despliegue de los servicios para proporcionar una respuesta ágil al mercado en tiempo real [Hurwitz et al. 2007].

También se reducen los costes a la hora de integrar diferentes sistemas y se incrementa la eficiencia de la comunicación entre las empresas gracias a que la tecnología de los servicios está basada en estándares abiertos, permitiendo alejarse del uso de las soluciones propietarias. Además, la utilización de servicios está facilitando la evolución hacia un mercado de negocio más automático y dinámico, el cual pretende utilizar toda la información disponible a su alcance para proporcionar siempre la funcionalidad más adecuada con la óptima calidad.

El paradigma de la orientación a servicios proporciona un enfoque distribuido ya que el uso de servicios permite definir soluciones débilmente acopladas y apoyadas en tecnologías estándar. Según el organismo de estandarización *W3C* (*World Wide Web Consortium*), una Arquitectura Orientada a Servicios (SOA) "*es un tipo específico de sistema distribuido en el que los agentes que intervienen son denominados 'servicios'. Un servicio es un agente software que realiza una acción bien definida y puede ser invocado fuera del contexto de una aplicación mayor*" [WSA 2003]. Los servicios son entidades computacionales autónomas e independientes de la plataforma que pueden describirse, publicarse, descubrirse, orquestarse y programarse utilizando lenguajes y protocolos estandarizados con el propósito de implementar redes colaborativas distribuidas dentro de una misma organización o entre distintas organizaciones [Sassen y Macmillan 2006].

Una arquitectura SOA no está ligada a ninguna tecnología específica. Sin embargo, se sustenta en diferentes tecnologías middleware, como pueden ser RPC, DCOM, CORBA y, especialmente, los servicios web. No obstante, es importante resaltar que el uso de alguna de estas tecnologías no implica tener SOA. La clave de SOA consiste en definir servicios independientes que definan interfaces abiertas para que cualquier aplicación pueda invocarlos, sin necesidad de ningún conocimiento mutuo preexistente entre la aplicación cliente y el servicio proporcionado [Bastida et al. 2008a][Bastida y Berreteaga 2007].

En la actualidad coexisten varios modelos de referencia de SOA, como el definido por el consorcio industrial *OASIS* (*Organization for the Advancement of Structured Information Standards*) [OASIS 2006] o el modelo propuesto por el consorcio *W3C* (*World Wide Web Consortium*) [WSA 2004], organismo que define los diferentes estándares de la Web.

En cuanto al futuro de SOA, según la consultora *Forrester Research*, el mercado de SOA va a experimentar un crecimiento importante hasta el año 2010 (ver Figura 1-1), con un ratio de crecimiento anual (*CAGR*) del 88 por ciento entre los años 2004 y 2009 [IDC 2007a][Waters 2005]. La compañía *IDC* estima que durante el año 2006 se invirtieron cerca de 2 billones de dólares en desarrollos y adquisiciones de software relacionado con SOA. Además, *IDC* predijo que dichas inversiones aumentarían cerca de un 48 por ciento entre los años 2006 y 2011, alcanzando una cifra de gastos en el año 2011 cercana a los 14 billones de dólares [IDC 2007b] para iniciativas basadas en SOA y 40.8 billones de dólares en la compra de servicios profesionales en torno a SOA [IDC 2007c].

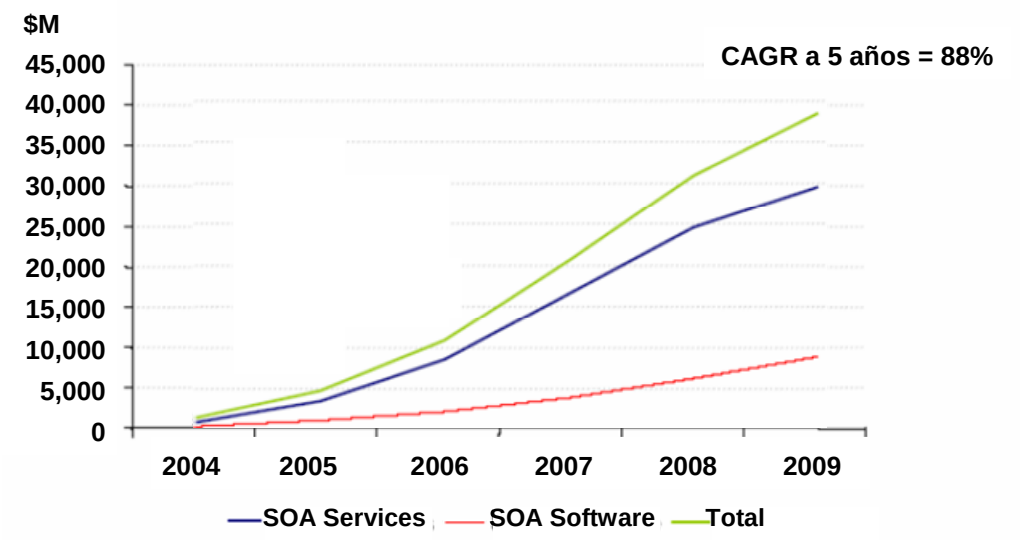


Figura 1-1: Inversión en SOA a nivel mundial en millones de dólares [IDC 2007b]

En relación con el mercado de SOA, se espera un mayor crecimiento en las aplicaciones para la gestión de procesos de negocio y las aplicaciones middleware [Gonsalves 2007].

A nivel mundial, las compañías están explorando nuevos sistemas innovadores que permitan hacer frente a la creciente complejidad de la globalización y de los nuevos modelos de negocio económicos. Así, están surgiendo nuevas necesidades como:

- La creación de procesos de negocio ágiles, dinámicos y automáticos como punto clave para la supervivencia de la empresa;
- La necesidad de dar una respuesta rápida a los cambios en el negocio y a optimizar y reutilizar las aplicaciones y sistemas ya existentes en la organización;
- La gestión de sistemas heterogéneos para permitir la compartición de datos y crear un entorno de trabajo colaborativo en escenarios *business-to-business* (B2B);
- La aplicación de estándares de interoperabilidad para favorecer el intercambio de información;
- La necesidad de proporcionar un diseño de mayor calidad que ofrezca un mejor soporte a la gestión de flujos de trabajo y al ciclo de vida de los procesos de negocio.

En la actualidad, muchas empresas y organizaciones aplican el paradigma de la orientación a servicios para integrar sus sistemas a nivel interno, sin llegar a exponer parte de los servicios producidos al exterior. Esto se debe en parte al hecho de que la tecnología de los servicios, especialmente la de los servicios web, todavía es bastante difícil de utilizar y existe una falta de confianza en el consumo de servicios proporcionados por terceras partes.

A pesar de que la utilización de los servicios implica enormes beneficios, la capacidad de componer varios de ellos para obtener una funcionalidad final de mayor valor añadido es uno de los aspectos clave para evolucionar hacia una economía orientada a servicios, ya que favorece la reutilización y el desarrollo de nuevas soluciones de manera sencilla, reduciendo costes y tiempo de desarrollo. Sin embargo, todavía es necesario que madure aún más la orientación a servicios para poder llevar a la práctica las investigaciones y los trabajos que se están realizando en torno a la composición de los mismos.

Hasta ahora, las composiciones estáticas son las que han tenido un mayor auge y aceptación entre los desarrolladores, integradores y proveedores de servicios. Esto es debido a que las composiciones estáticas [Dustdar et al. 2005] garantizan en mayor medida la funcionalidad y calidad proporcionadas por el servicio compuesto al estar descrito de manera estática. Es decir, los servicios a utilizar son seleccionados en tiempo de diseño y no se admite ninguna posibilidad de cambio durante su ejecución.

Hoy en día se está evolucionando hacia un entorno más dinámico donde las soluciones software ofrecidas deben ser sensibles al contexto de ejecución proporcionando la funcionalidad más adecuada y la calidad óptima en cada momento. Por ello, surge la necesidad de diseñar y configurar composiciones dinámicas [Dustdar et al. 2005]. Sin embargo, actualmente no se ve factible su puesta en práctica porque su diseño, implementación, ejecución y gestión se ve como una tarea muy compleja de realizar manualmente e incluso de forma automática ya que hay que especificar e interpretar la variabilidad que puede presentar la composición. Esta variabilidad se define como la habilidad de cambio o de personalización de un sistema [Svahnberg et al. 2001], donde dicha variabilidad se debe principalmente a aspectos relacionados con el contexto de ejecución.

El reto subyace en el hecho de adaptar el flujo de la composición final al contexto de ejecución de una manera rápida y en la mayor brevedad de tiempo posible, en caso contrario la composición dinámica no sería práctica. La flexibilidad para adaptar dinámicamente el flujo de la composición al contexto de ejecución está limitada a la especificación de los detalles de la asociación (en inglés conocido como *binding*) entre la funcionalidad proporcionada por el servicio y la implementación que lo satisface. Además, este tipo de composición se beneficia de la búsqueda automática de servicios, bien sea a partir de un conjunto de requisitos (tanto funcionales como no funcionales), o a partir de una especificación de la funcionalidad del servicio bien detallada. Por este motivo es necesario definir, representar y modelar aquellas partes variables de la composición con el fin de favorecer el desarrollo de composiciones de servicios flexibles y dinámicas, capaces de autoconfigurarse según las condiciones del contexto de ejecución.

Sin embargo, en la actualidad existe una carencia de técnicas y metodologías que den soporte al diseño e implementación de composiciones de servicios, en especial a las composiciones dinámicas. En futuras investigaciones en el ámbito de la composición de servicios se espera proveer mecanismos que permitan especificar y automatizar las composiciones dinámicas, reducir el tiempo de desarrollo y el esfuerzo de integración, así como monitorizar las interacciones dinámicas de los servicios. Las técnicas de gestión de variabilidad de la Ingeniería de Líneas de Producto proporcionan la respuesta a gran parte de las cuestiones y retos identificados en las composiciones dinámicas de servicios.

1.2 Hipótesis y Objetivos

Con el fin de delimitar con mayor precisión el ámbito de trabajo a realizar, presentamos en este apartado la hipótesis cuya verificación promueve la presente labor de investigación, así como los objetivos fijados para el desarrollo de la misma.

1.2.1 Hipótesis

La hipótesis que se plantea en esta tesis puede resumirse de forma siguiente:

> ***"Si se proporciona soporte a la composición de servicios integrando las prácticas de la Ingeniería de Líneas de Producto, se posibilitará el diseño y desarrollo de composiciones de servicios flexibles, dinámicas y sensibles al contexto de ejecución, capaces de autoconfigurarse según las condiciones del entorno de ejecución."***

La demostración de la veracidad de esta hipótesis determinará las acciones del proceso de investigación como medios fundamentales para su validación.

1.2.2 Objetivo General

El objetivo general de la presente tesis es *mejorar la capacidad de composición e integración de servicios* que la sociedad actual y del mañana requiere para hacer realidad la evolución hacia una economía orientada a servicios.

Esto está motivado por la necesidad de solventar la limitación existente en torno al diseño y desarrollo de composiciones de servicios proporcionando un nuevo método de diseño que, haciendo uso de nuevas y prometedoras tecnologías, como la Ingeniería de Líneas de Producto (denominada con la abreviatura PLE) [Pohl et al. 2005][Clements et al. 2001] para especificar la variabilidad, facilite el desarrollo de composiciones de servicios flexibles y dinámicas, capaces de autoconfigurarse según las condiciones del entorno de ejecución.

Esta nueva metodología para desarrollar composiciones de servicios no pretende ser otra más de desarrollo de software, sino que se centra en resaltar las características más importantes en el ámbito del diseño y desarrollo de las composiciones de servicios dinámicas y sensibles al contexto de ejecución.

Por ello, nos centraremos en definir aquellas fases, procesos, tareas, pasos, roles, productos de trabajo y responsabilidades específicas de la composición de servicios que, posteriormente, podrán ser incorporadas a una metodología de desarrollo de software existente.

Esta metodología enfocará la composición de servicios desde una perspectiva multidisciplinar, completa e independiente de cualquier implementación. Por un lado, tendremos en cuenta los modelos de negocio y patrones arquitecturales más adecuados para aplicar en el diseño de la composición; y por otro lado, facilitaremos la identificación en tiempo de diseño de la variabilidad de la composición de tal modo que la composición de servicios pueda adaptarse a las condiciones del contexto de ejecución.

Así, el mapa de características generales de la metodología será el siguiente:

- **Valor práctico**. Basado en los puntos innovadores proporcionados por la metodología sobre soluciones ya existentes que no ofrecen las características de dicha metodología, permitiendo la entrada en nuevos mercados y a nuevas posibilidades.

- **Facilidad de uso**. Es decir, el grado de dificultad que la metodología presenta a la hora de comprenderla y utilizarla. Los usuarios siempre prefieren trabajar con métodos y herramientas sencillas de entender y utilizar.

- **Compatibilidad**. Según el grado de dificultad que la metodología presenta para mejorar y adaptar composiciones ya existentes, favoreciendo la descripción de la variabilidad de dichas composiciones.

- **Aplicabilidad**. En el sentido en cómo la metodología será capaz de ayudar en el diseño y desarrollo de composiciones de servicios tanto estáticas como dinámicas, capaces de adaptarse a las características del entorno de ejecución.

Para diseñar y configurar la dinamicidad o variabilidad de una composición de servicios utilizaremos el enfoque de Ingeniería de Líneas de Producto, el cual define mecanismos para gestionar la variabilidad. Además, el enfoque PLE también permite definir patrones de composiciones de servicios como líneas de productos. Es decir, es posible definir un patrón o modelo de composición a partir del cual posteriormente, según una serie de requisitos o características, se pueden definir diferentes instancias personalizadas de dicha composición.

Por último, para demostrar la validez de la hipótesis de partida procederemos a realizar la validación empírica de la metodología realizada, en un escenario industrial real que involucra la interacción y colaboración entre varias organizaciones del área de las telecomunicaciones y de la automoción, obteniendo unos resultados satisfactorios. Además, también se ha realizado una evaluación de la metodología de acuerdo al marco *Normative Information Model-based Systems Analysis and Design* (*NIMSAD*) [Jayaratna 1994].

De manera adicional, se desarrollará una herramienta que sirva como guía para aplicar los diferentes procesos y tareas definidas en nuestra metodología.

1.2.3 Objetivos Específicos y Operacionales

Los objetivos específicos de esta tesis, en consonancia con el objetivo general definido y con las acciones necesarias para la confirmación de la hipótesis enunciada, son tres:

- **Definir la solución metodológica basada en los principios de variabilidad del enfoque PLE [OE1]**. Esta metodología facilitará a los desarrolladores e integradores de aplicaciones la creación de composiciones de servicios dinámicas de una manera ágil y sencilla.

- **Construir una herramienta de soporte a la metodología de composición [OE2]**. De modo que se disponga de una utilidad software que ayude a mejorar la aplicabilidad práctica de la solución metodológica definida.

- **Verificar los resultados del uso de la metodología [OE3]**. Diseñándose para este fin experimentos o pruebas que permitan validar la presencia de las características y bondades perseguidas con la metodología.

En cuanto a los objetivos operacionales que posibilitarán la consecución de los objetivos específicos definidos y, por consiguiente, la consecución del objetivo general y la validación de la hipótesis planteada, la Tabla 1-1 recoge algunos de los más representativos y su vínculo con el objetivo específico al que contribuyen. Cabe decir que la información mostrada no pretende ser un exhaustivo plan de trabajo seguido en esta tesis, sino más bien una muestra de algunos de los objetivos operacionales más relevantes desde el punto de vista científico.

OBJETIVOS OPERACIONALES	OBJ. ESP.
OO1 - Definir un modelo de diseño de la variabilidad	OE1
OO2 - Definir una metodología de composición	OE1
OO3 - Construir una herramienta de soporte	OE2
OO4 - Realizar pruebas de uso de la metodología en escenarios industriales reales	OE3
OO5 - Evaluar la metodología desde un punto de vista del desarrollo de sistemas	OE3

Tabla 1-1: Objetivos operacionales más relevantes y objetivo específico al que contribuyen

1.3 Organización de la Tesis

La presente memoria del trabajo de investigación se compone de tres grandes secciones, que configuran las fases en las que se ha dividido el trabajo de investigación realizado: el estudio de las diferentes áreas de conocimiento implicadas en el problema y la definición del área de trabajo, el desarrollo de la solución innovadora, y la validación y evaluación de los resultados. Estas secciones se estructuran a su vez en los diferentes capítulos descritos a continuación.

El capítulo 2 presenta el método de trabajo que se ha seguido para cumplir con los objetivos marcados en esta tesis. El capítulo 3 presenta de manera detallada los antecedentes y un análisis de las soluciones existentes en el ámbito de las distintas áreas implicadas en el trabajo. En primer lugar, se describe en detalle qué es la composición de servicios y cuáles son los tipos diferentes de composición de servicios que podemos encontrar. Seguidamente, se comentan cuáles son los retos a superar en torno a la composición dinámica de servicios, explicando dos enfoques interesantes, el *re-binding* y el *re-planning*. A continuación, se resume el estado del arte de los distintos enfoques de composición de servicios que hay en la actualidad, desde un punto metodológico y técnico respecto a su aplicación en un entorno más dinámico. En este capítulo también se describen los conceptos básicos de la Ingeniería de Líneas de Producto así como su aplicación en el dominio de las Arquitecturas Orientadas a Servicios, en especial para gestionar la variabilidad de las composiciones dinámicas y flexibles de servicios. Este capítulo incluye una descripción del lenguaje de modelado SPEM 2.0 así como de los principales elementos utilizados para formalizar la metodología propuesta. El capítulo finaliza con una descripción del enfoque *NIMSAD*, que define un marco de trabajo para evaluar metodologías de desarrollo de Sistemas de Información.

El capítulo 4 está dedicado a proporcionar una descripción general sobre la solución propuesta y cómo se han aplicado las técnicas de variabilidad definidas en el enfoque de la Ingeniería de Líneas de Producto Software en el dominio de la composición dinámica de servicios.

En el capítulo 5 se recoge la descripción de la metodología de composición de servicios, compuesta por un conjunto de procesos y tareas, y la herramienta de soporte a la metodología que ha sido desarrollada.

La idea fundamental sobre el formato utilizado para presentar esta metodología está orientada a facilitar a las organizaciones su aplicación y uso. Este capítulo también incluye una descripción de cómo se ejecutarían las composiciones de servicios dinámicas desarrolladas siguiendo la metodología propuesta.

En el capítulo 6 se incluyen dos ejemplos sobre cómo aplicar la metodología propuesta en una organización que ya tiene implantada una metodología general de desarrollo de software. Dichos ejemplos se basan en *Rational Unified Process* (*RUP*) y *Extreme Programming* (*XP*).

El capítulo 7 presenta la validación del modelo a través de la presentación de un escenario industrial real que define la colaboración entre empresas de dominios diferentes, como son las telecomunicaciones y la automoción. Al mismo tiempo, se proporciona una evaluación de la metodología propuesta dentro del enfoque *NIMSAD*. Este capítulo concluye con un análisis sobre los resultados obtenidos tras la validación y evaluación de la metodología propuesta.

En los capítulos 8 y 9 se recogen las conclusiones finales del trabajo y se describen las diversas líneas de investigación que quedan abiertas, así como la bibliografía utilizada y consultada durante este trabajo de investigación. El capítulo 10 incluye un glosario con los términos más importantes que se han utilizado a lo largo de esta tesis.

La memoria se completa con tres anexos que recogen, respectivamente, las plantillas asociadas a los principales resultados generados por la metodología de composición de servicios, una descripción de la herramienta de soporte a la metodología y los modelos ejecutables del servicio compuesto asociados al escenario industrial explicado en el capítulo 7.

La ciencia más útil es aquella cuyo fruto es el más comunicable.

Leonardo Da Vinci

CAPITULO 2

Método de Trabajo

En este capítulo se presenta el método de investigación denominado Investigación-Acción (*Action-Research*) como el método de trabajo seguido para cumplir con los objetivos marcados en esta tesis doctoral, ya que es uno de los principales métodos de investigación cualitativa en el campo de los Sistemas de Información y en la Ingeniería del Software.

Este capítulo se divide a su vez en dos secciones. En primer lugar, se presentan las características especiales que deben ser tenidas en cuenta al utilizar el método Investigación-Acción (IA) en proyectos de Ingeniería del Software. Posteriormente, se detalla cómo hemos aplicado de manera concreta dicho método en esta tesis doctoral.

2.1 Métodos de Investigación en Ingeniería del Software

Comprender una disciplina implica aprendizaje, es decir, observación, reflexión y encapsulación de conocimiento, construcción de modelos (del dominio de aplicación, de los procesos para resolver problemas, etc.), experimentación, y evolución de los modelos con el tiempo [Basili 2000]. Este paradigma ha sido utilizado en campos tan diversos como la medicina, la física, o la industria manufacturera; pero la única diferencia entre dichos campos estriba en cómo se construyen y analizan los modelos, y cómo se lleva a cabo la experimentación.

Desde un punto de vista científico, las áreas de Sistemas de Información e Ingeniería del Software son de carácter experimental. La investigación dentro de estas áreas se centra en conocer la naturaleza de los procesos y de los productos, y cómo se interrelacionan entre ellos en el contexto de un sistema software o de un sistema organizacional (en el caso de Sistemas de Información).

La Ingeniería del Software, como cualquier otra disciplina científica aplicada, por ejemplo la química o la medicina, tiene dos objetivos fundamentales [Philips 1998]: un objetivo teórico que consiste en aumentar el conocimiento para entender por qué las cosas ocurren en un área particular de interés; y un objetivo práctico orientado a mejorar las prácticas de forma que los resultados de la investigación sean útiles. Por lo tanto, es fundamental establecer una buena base teórica en Ingeniería del Software, pero siempre teniendo también como objetivo la obtención de resultados útiles en la práctica. Sin embargo, ésta no es la situación habitual, ya que a menudo se proponen nuevas ideas, métodos, etc., sin aplicación en la práctica, existiendo una clara desconexión entre la investigación teórica y su aplicación [Moody 2000] tal y como se puede observar en la Figura 2-1.

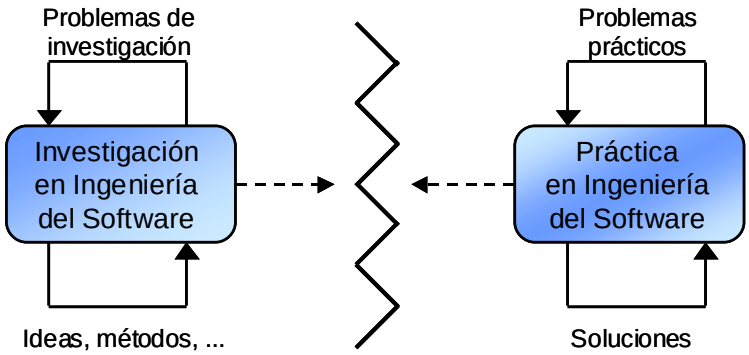


Figura 2-1: Desconexión entre Investigación y Práctica en Ingeniería del Software [Moody 2000]

Si el objetivo es eliminar esta barrera entre la teoría y la práctica en Ingeniería del Software y favorecer la transferencia de dicho conocimiento hacia su puesta en práctica, es necesario que la investigación esté orientada a objetivos prácticos, y que la industria del software aplique los resultados obtenidos en la investigación. Para ello, en Ingeniería del Software existen métodos de trabajo enfocados según el tipo de investigación que se está realizando. Aunque los métodos de investigación pueden clasificarse de diversas formas, una clasificación ampliamente aceptada es la definida por [Myers 1997], en la cual divide a los métodos en cuantitativos y cualitativos.

Los métodos de investigación cuantitativos son especialmente adecuados para el estudio de fenómenos u objetos naturales, y en este tipo de métodos se podrían encuadrar los métodos deductivos y empíricos. El estudio de fenómenos culturales y sociales requiere otro tipo de métodos, que no se basen en experimentos ni teorías formales, sino en entrevistas, cuestionarios, documentos, impresiones y reacciones del investigador, etc. Estos métodos se encuadran dentro de lo que se conoce como métodos cualitativos y ejemplos de estos métodos son la etnografía y el método investigación-acción.

Las principales diferencias entre la investigación cualitativa y la investigación cuantitativa se encuentran en la naturaleza de los datos y en el análisis realizado. En la investigación cualitativa los datos son poco estructurados, obtenidos mediante entrevistas y discusiones en grupo y no se suelen analizar de forma estadística, mientras que en la investigación cuantitativa los datos son más estructurados y se utilizan técnicas estadísticas para su análisis. La siguiente tabla (Tabla 2-1) resume los diferentes aspectos en los que difiere la investigación cualitativa de la cuantitativa.

ASPECTO	INVESTIGACIÓN CUALITATIVA	INVESTIGACIÓN CUANTITATIVA
Propósito	Proveer conocimiento sobre un problema y sus soluciones	Generalizar resultados desde un caso de estudio a la población de interés
Orientación	Hacia la verificación	Hacia el descubrimiento
Muestra	Tamaño pequeño	Tamaño grande
Recogida de datos	Datos poco estructurados obtenidos mediante entrevistas, observaciones y discusiones en grupo	Datos muy estructurados
Análisis	No estadístico	Estadístico

Tabla 2-1: Diferencias entre la Investigación Cualitativa y la Cuantitativa

Gran parte de la investigación llevada a cabo en las áreas de Sistemas de Información e Ingeniería del Software es de tipo cuantitativo y está basada en técnicas estadísticas. Sin embargo, la aplicación de los métodos cualitativos [Glass et al. 2002], y en especial, de investigación-acción es bastante reciente y de hecho, hasta finales de los años 90 no ha recibido la atención y aceptación de la comunidad investigadora en Sistemas de Información, como se muestra en los trabajos de [Avison et al. 1999] y [Seaman 1999].

2.1.1 Método Investigación-Acción

Entre los diversos métodos de investigación cualitativa, el más utilizado en Sistemas de Información e Ingeniería del Software es el denominado Investigación-Acción (IA).

El término 'Investigación-Acción' describe una forma de investigación que podía enlazar el enfoque experimental de las ciencias sociales con programas de acción social que respondieran a los problemas sociales principales de entonces. Mediante la Investigación-Acción, se argumentaba que se podían lograr de forma simultánea avances teóricos y cambios sociales. Este método ha obtenido una amplia aceptación y aplicación en la investigación en Ingeniería del Software en los últimos años, desde que fue introducida en el año 1985 por [Wood-Harper 1985]. En realidad Investigación-Acción no se refiere a un método de investigación concreto, sino a una clase de métodos que tienen en común varias características [Baskerville 1999]: orientación a la acción y al cambio, focalización en un problema, un modelo de proceso "orgánico" que engloba etapas sistemáticas y algunas veces iterativas, y fomento de la colaboración entre los participantes.

Existen diversas definiciones de Investigación-Acción [McTaggart 1991], [French y Bell 1996] o [Wadsworth 1998]. De estas definiciones se puede deducir que Investigación-Acción tiene una doble finalidad: generar un beneficio al "cliente" de la investigación y, al mismo tiempo, generar "conocimiento de investigación" relevante [Koch y Lau 2001]. Por tanto, Investigación-Acción es una forma de investigar de carácter colaborativo que busca unir teoría y práctica entre investigadores y usuarios mediante un proceso de naturaleza cíclica. Investigación-Acción está orientada a la producción de nuevo conocimiento útil en la práctica, que se obtiene mediante el cambio y/o búsqueda de soluciones a situaciones reales que le ocurren a un grupo de usuarios [Avison et al. 1999]. Esto se consigue gracias a la intervención de un investigador en la realidad del mencionado grupo. Los resultados de esta experiencia deben ser beneficiosos para el investigador y para los usuarios.

Una premisa fundamental en esta forma de investigar es que los procesos sociales complejos (y el uso de tecnologías de la información en organizaciones es de este tipo) pueden ser estudiados mejor introduciendo cambios en dichos procesos y observando los efectos de dichos cambios [Baskerville 1999].

En el campo de los Sistemas de Información, el cliente de una investigación es normalmente una organización a la cual el investigador suministra servicios tales como consultoría, ayuda para cambiar o desarrollo de software, a cambio de tener acceso a datos de interés para la investigación y, en muchos casos, de recibir financiación [Koch y Lau 2001]. En cualquier caso, el investigador que utiliza Investigación-Acción en Sistemas de Información (IA-SI) sirve a dos entidades diferentes: al cliente de la investigación y a la comunidad científica de Sistemas de Información. Las necesidades de ambos suelen ser muy diferentes y, a veces, opuestas entre sí. Intentar satisfacer ambas demandas es el principal desafío que todos los investigadores de IA-SI tienen que enfrentar. Sirviendo tanto a las necesidades de los usuarios como a las del conocimiento científico, se añaden nuevos elementos a la investigación que hace que sea más deseable.

En un análisis más formal de los participantes en Investigación-Acción, [Wadsworth 1998] identifica los siguientes cuatro tipos de roles en este método (en algunas ocasiones la misma persona o equipo puede desempeñar más de un rol):

- El **investigador**, el individuo o grupo que lleva a cabo de forma activa el proceso investigador.

- El **objeto investigado**, es decir, el problema a resolver.

- El **grupo crítico de referencia**, aquél para quien se investiga en el sentido de que tiene un problema que necesita ser resuelto y que también participa en el proceso de investigación (aunque menos activamente que el investigador). En él hay tanto personas que saben que están participando en la investigación, como otras que participan sin saberlo.

- El **beneficiario**, aquél para quien se investiga en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participa directamente en el proceso. Puede ser el receptor de documentos, informes, etc.

  En este grupo, por ejemplo, caben tanto las empresas que se benefician de un nuevo método para resolver problemas en tecnologías de la información, como los técnicos que aplican dicha metodología.

Desde sus orígenes se han distinguido formas diferentes de aplicar Investigación-Acción, como en [Chein et al. 1948]. [French y Bell 1996] propone cuatro variantes que dependen principalmente de las características del proyecto de investigación:

- **De diagnóstico**: El investigador se adentra en una situación problemática, la diagnostica y realiza recomendaciones al grupo crítico de referencia, pero sin que haya un control posterior de sus efectos.

- **Empírica**: El grupo crítico de referencia realiza un registro amplio y sistemático de sus acciones y sus efectos. Esta característica hace que esta variante sea difícilmente aplicable.

- **Participativa**: El grupo crítico de referencia pone en práctica las recomendaciones realizadas por el investigador, compartiendo con él sus efectos y resultados.

- **Experimental**: Consiste en evaluar las diferentes opciones que existen para conseguir un objetivo. El principal inconveniente de esta variante reside en la dificultad de poder medir objetivamente las diversas opciones, ya que por lo general serán, o bien aplicadas en organizaciones con distintas características que enturbian los resultados de la investigación, o bien en una sola organización pero en distintos momentos, con lo que el entorno experimental habrá variado.

Un proceso de investigación que emplea Investigación-Acción se halla compuesto de grupos de actividades organizadas formando un ciclo característico. [Padak y Padak 2005] identifican los siguientes pasos (ver Figura 2-2), que deben seguirse en las investigaciones que utilicen este método:

- **Planificación**: Identificar las cuestiones relevantes, que guiarán la investigación, que deben estar directamente relacionadas con el objeto que se está investigando y ser susceptibles de encontrarles respuesta. En esta actividad se buscan caminos alternativos, líneas a seguir o reforzar algo existente.

  El resultado es que se definen claramente otros problemas o situaciones a tratar. Algunos autores como [Baskerville 1997] distinguen entre diagnóstico (identificar los

problemas iniciales) y planificación (especificar acciones para resolver dichos problemas).

- **Acción**: Variación de la práctica, cuidadosa, deliberada y controlada. Se efectúa una simulación o prueba de la solución. Es cuando el investigador interviene sobre la realidad.

- **Observación**: Recoger información, tomar datos, documentar lo que ocurre. Esta información puede proceder prácticamente de cualquier sitio (bibliografía, medidas, resultados de pruebas, observaciones, entrevistas, documentos, etc.). También se conoce como "evaluación".

- **Reflexión**: Compartir y analizar los resultados con el resto de interesados, de tal manera que se invite al planteamiento de nuevas cuestiones relevantes y, como añade [Wadsworth 1998], "*a profundizar en la materia que se está investigando para proporcionar conocimientos nuevos que puedan mejorar las prácticas, modificando éstas como parte del propio proceso investigador, para luego volver a investigar sobre estas prácticas una vez modificadas*". También se conoce como "especificación del aprendizaje". En algunas variantes de Investigación-Acción no es una etapa realmente, sino un proceso continuo que ocurre durante todo el tiempo.

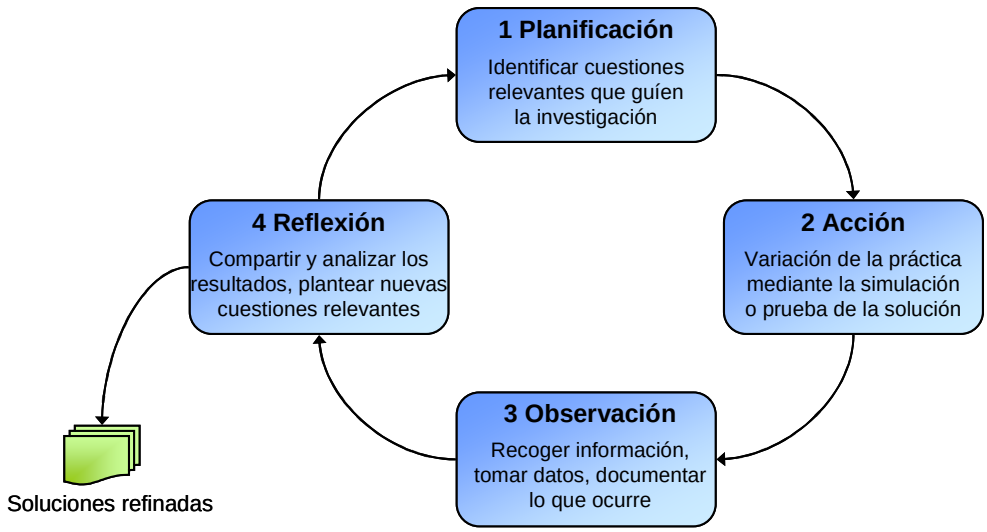


Figura 2-2: Carácter cíclico del método Investigación-Acción

Con estas características, el proceso definido por Investigación-Acción es iterativo, de forma que se va avanzando en soluciones cada vez más refinadas mediante la completitud de ciclos, en cada uno de los cuales se ponen en marcha nuevas ideas, que son puestas en práctica y comprobadas en el ciclo siguiente, tal como se muestra en la Figura 2-2.

Este ciclo caracteriza Investigación-Acción como un proceso reflexivo de aprendizaje y búsqueda de soluciones. El carácter cíclico supone volver a reevaluar o replantear las acciones o caminos a seguir ponderando diagnóstico y reflexión.

En los últimos años el método Investigación-Acción ha sido reconocido como uno de los métodos de investigación (cualitativa) más potentes en el ámbito de los Sistemas de Información (SI). Ahora bien, la comunidad de especialistas ha detectado diversos problemas en su aplicación que se deben principalmente a:

- La falta de método con que los investigadores y usuarios utilizan y conciben su aplicación en el ámbito de los Sistemas de Información debido a la ausencia de un modelo de proceso de investigación definido que indique los pasos a seguir.
- El contexto de consultoría utilizado, que impone una perspectiva demasiado restrictiva al implicar responsabilidades contractuales e intereses organizacionales que pueden ir en contra de lo propuesto por Investigación- Acción.

Todo lo anterior puede tener como consecuencia una falta de rigor en el proceso de investigación. Para solventar estos problemas se han propuesto, entre otras, las siguientes alternativas:

- Conducir la investigación usando una perspectiva de gestión de proyectos,
- Incluir criterios de calidad especialmente concebidos para el caso en cuestión,
- Analizar los factores que inciden en la formalización del proceso, y
- Organizar el proceso con una estructura de proyecto.

Combinando varias de estas ideas, [Estay y Pastor 2000] y [Estay y Pastor 2000b], han propuesto "*usar la gestión de proyectos para mejorar el rigor de un proyecto de Investigación-*

Acción aplicado a Sistemas de Información (IA-SI), lo cual se ha traducido en generar una estructura de proyecto que contenga los principales elementos de IA-SI". Para lograr lo anterior, estos autores plantean la necesidad de adoptar prácticas de gestión, adecuadas a IA-SI, basadas en el *PMBOK* (*Guide to the Project Management Body of Knowledge*), que es el modelo de gestión de proyectos más difundido a nivel internacional propuesto por el *Project Management Institute* [PMB 2000]. Para Estay y Pastor, Investigación-Acción y proyecto son conceptos equivalentes, ya que ambos son experiencias de trabajo únicas con resultados finales únicos y, además, comparten la idea de intervención, es decir, ambos suponen una alteración voluntaria de la realidad.

Aunque la intervención en Investigación-Acción produce alteraciones en una práctica de trabajo, también es una forma de obtener datos de la experiencia real que son necesarios para el proceso de investigación. Los mismos autores también han propuesto un modelo de madurez basado en [CMM 1995], aplicando prácticas de gestión de proyectos de forma incremental con objeto de garantizar una mejora del rigor y calidad del uso de Investigación-Acción en Sistemas de Información [Estay y Pastor 2001].

En el contexto de la investigación cualitativa en Sistemas de Información se puede considerar que existen dos realidades, la científica/académica y práctica, las cuales interactúan pero que se mueven en planos diferentes. El método Investigación-Acción aplicado a Sistemas de Información opera sobre esta realidad dual, que se concreta en dos tipos de ciclos de Investigación-Acción para dos tipos de proyectos:

- **Ciclos orientados a resolver problemas** dentro de proyectos de Sistemas de Información. Estos proyectos consisten en el desarrollo de una solución informática (son proyectos informáticos, de desarrollo de software, de implantación y/o mantenimiento de sistemas informáticos, etc.).

  En este caso el investigador se encarga de resolver un problema e Investigación-Acción aparece como una herramienta más para el desarrollo de sistemas de información.

- **Ciclos orientados a investigar** dentro de proyectos de investigación. Estos proyectos son esfuerzos intencionados buscando un resultado. En este caso Investigación-Acción

nos ofrece un método de trabajo y una justificación para acercarnos a una determinada realidad con fines de probar una teoría o hipótesis.

Por otro lado, la estructura de proyecto de IA-SI propuesta por [Estay y Pastor 2000b] define dos ciclos característicos:

- **Ciclo orientado a construir una solución** para generar nuevo conocimiento útil a usuarios y mejorar su práctica. El investigador se conecta con la realidad mediante una intervención. La investigación se utiliza para construir modelos, teorías o conocimiento de manera informada e influida por la realidad. En este ciclo es el interés por resolver un problema lo que origina el interés por la investigación.

- **Ciclo orientado a gestionar la investigación** para producir nuevo conocimiento para la disciplina de Sistemas de Información y mejorar la práctica de los investigadores. En este ciclo es el interés por la investigación el que origina esfuerzo por resolver ciertos problemas.

En [Lau 1997] se puede encontrar un resumen del uso de Investigación-Acción aplicado a Sistemas de Información, comentando diversos ejemplos publicados por distintos autores referidos a la construcción y desarrollo de sistemas de información, y más especialmente, al análisis, diseño, desarrollo e implementación de software y a los procesos asociados.

En [Baskerville 1999] se hace una introducción al uso del método Investigación-Acción en los Sistemas de Información indicando diez formas de utilización y cuatro características que determinan dicha forma de uso: modelo de proceso (iterativo, reflexivo, lineal); estructura (rigurosa, fluida); rol del investigador (colaborador, usuario, experto); y objetivos principales (desarrollo organizacional, diseño de sistemas, conocimiento científico, entrenamiento). [Baskerville y Wood-Harper 1996] señalan siete estrategias básicas para llevar a cabo IA-SI: utilizar el "paradigma del cambio", establecer un acuerdo o contrato formal de investigación, proporcionar un marco de trabajo teórico, planificar los métodos de captura de datos, mantener la colaboración y el aprendizaje mutuo entre investigador y grupo crítico de referencia, incentivar las iteraciones del ciclo típico, y buscar la generalización de las soluciones.

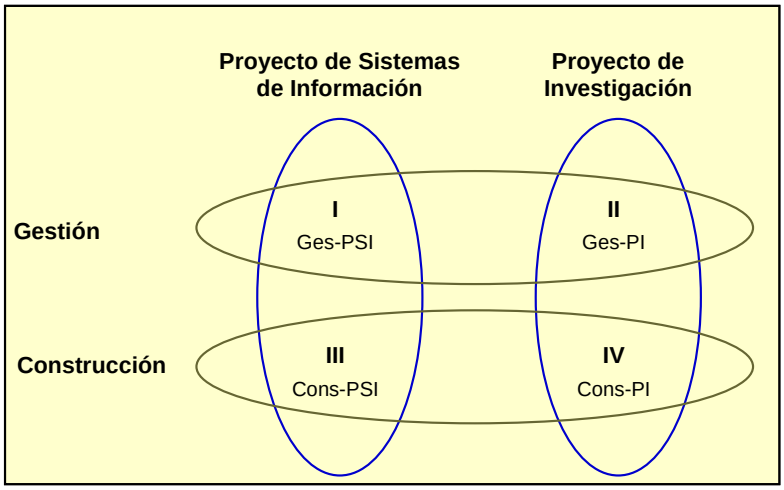


Figura 2-3: Dimensiones de Investigación-Acción en Sistemas de Información

En resumen, IA-SI puede analizarse desde dos dimensiones complementarias: una dimensión "vertical" en función del tipo de proyecto; y una dimensión "horizontal" en función del bi-ciclo típico de la estructura de un proyecto de IA-SI.

2.2 Aplicación de Investigación-Acción en este Trabajo de Investigación

Teniendo en cuenta que el objeto de investigación de la tesis es la de ***ayudar al diseño y desarrollo de composiciones de servicios flexibles y dinámicas, capaces de autoconfigurarse según el entorno de ejecución,*** hemos considerado la variante participativa del método Investigación-Acción como la más adecuada.

2.2.1 Elección de los Participantes en la Investigación

Los cuatro tipos de participantes, identificados en [Wadsworth 1998] se concretan en esta tesis a través de los siguientes participantes:

- **Investigador**: El rol de investigador lo asume Leire Bastida en compañía del equipo de European Software Institute, formado por expertos de diferentes dominios de dentro del área de I+D.

- **Objeto investigado**: La composición de servicios dinámicos y sensibles al contexto de ejecución aplicando la Ingeniería de Líneas de Producto para ayudar a especificar la variabilidad de la composición.

- **Grupo crítico de referencia (GCR)**: Para realizar la validación de la propuesta recogida en esta tesis, se ha contado con especialistas en el área de las Arquitecturas Orientadas a Servicios (SOA) y en la Ingeniería de Líneas de Producto. Estos especialistas han desarrollado diversos proyectos en los que se abordaba el tema de la composición de servicios. En especial, este grupo de referencia ha participado dentro del proyecto europeo SeCSE [SeCSE 2008].

- **Beneficiarios**: Serán todas aquellas organizaciones que pueden ser beneficiadas por los resultados del trabajo. No sólo son todas aquellas empresas de desarrollo de software interesadas en la creación, integración y provisión de nuevos servicios de manera rápida y fácil, sino también aquellas empresas que quieren obtener una mayor funcionalidad sin necesidad de invertir tiempo, dinero y esfuerzo en nuevos desarrollos.

El cumplimiento de los roles que le corresponden a cada uno de los participantes en la investigación permitirá satisfacer la hipótesis planteada en esta tesis. La Figura 2-4 ilustra las relaciones entre los diferentes actores.

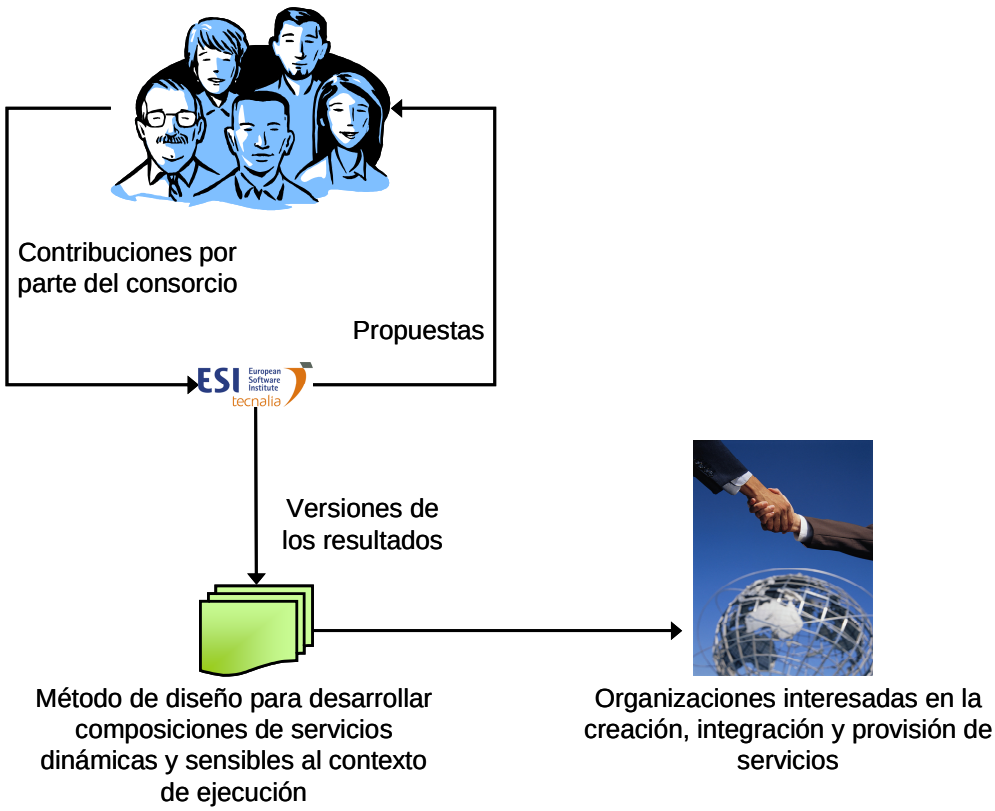


Figura 2-4: Aplicación del método Investigación-Acción

2.2.2 Etapas Seguidas durante la Investigación

La realización de cada uno de los pasos mencionados en la Figura 2-2 es la siguiente:

- **Planificación**. Conocida la ausencia de un marco adecuado para la composición de servicios, la solución es plantear una aproximación basada en diversos mecanismos para desarrollar composiciones de servicios sensibles al contexto. Dicha solución consiste en una metodología de composición de servicios que incorpora la Ingeniería de Líneas de Producto para describir la variabilidad de la composición.

- **Acción**. Una vez definida la metodología, que consiste en un proceso de composición de servicios y unas guías de uso, todo ello soportado por una herramienta para facilitar su aplicación, se procede a aplicarlos probando la solución en determinados casos de estudio.

- **Observación**. A partir de la propuesta y aplicación a los casos de estudio se procede a optimizar cada una de las propuestas, producto de una evaluación del trabajo realizado y su aplicación. De esta manera, se logra obtener desde una perspectiva multidisciplinar, completa e independiente de cualquier implementación, una metodología de composición de servicios que utiliza los principios de la variabilidad para definir los aspectos dinámicos.

- **Reflexión**. Siguiendo la filosofía cíclica de la Investigación-Acción, los resultados detallados en el punto anterior no se han obtenido de una vez, sino que han sido fruto de las distintas iteraciones seguidas.

  Por otro lado, a la hora de compartir resultados se ha hecho a distintos niveles en función de la madurez de los resultados:

  — Primero se han compartido los resultados entre el propio grupo de investigación, es decir, entre el doctorando y el director de tesis.

  — Luego, se han contrastado los resultados con la comunidad investigadora nacional e internacional dedicada al dominio de las Arquitecturas Orientadas a Servicios, los servicios web y la Ingeniería de Líneas de Producto.

Esto se ha realizado mediante la participación en el proyecto europeo SeCSE y su retroalimentación ha surgido tanto a partir de las revisiones previas de los trabajos, como de los comentarios en las presentaciones y de los contactos posteriores con los investigadores interesados en el trabajo.

Además, se han realizado diversas publicaciones y presentaciones en congresos y conferencias de relevancia tanto a nivel nacional como internacional:

- 2nd International Workshop on Systems Development in SOA Environments (SDSOA 2008).
- 4th International Conference Interoperability for Enterprise Software and Applications (I-ESA 2008).
- 7th IEEE International Conference on Composition-Based Software Systems (ICCBSS 2008).
- 3th International Conference Interoperability for Enterprise Software and Applications (I-ESA 2007).
- III Jornadas Científico-Técnicas en Servicios Web (JSWEB 2007).
- 2th International Conference Interoperability for Enterprise Software and Applications (I-ESA 2006).
- II Jornadas Científico-Técnicas en Servicios Web (JSWEB 2006).
- 4th International Conference on COTS-Based Software Systems (ICCBSS 2005).

— Finalmente, los resultados se han compartido con el grupo crítico de referencia.

Aprender sin pensar es tiempo perdido; pensar sin aprender es peligroso.

Confucio

CAPITULO 3

Antecedentes y Análisis de las Soluciones Existentes

En este capítulo se presenta un resumen de los antecedentes y del estado del arte de las áreas de conocimiento implicadas en este trabajo de investigación y utilizadas como base del mismo. Las áreas de conocimiento fundamentales son la composición de servicios, la Ingeniería de Líneas de Producto, el modelado formal de metodologías y la evaluación de metodologías.

3.1 Composición de Servicios

A partir de nuestra participación en diversos proyectos de investigación, hemos identificado que gracias a la reciente explosión de la tecnología de los servicios, las empresas están descubriendo nuevas oportunidades de negocio a través de la integración de sus procesos y servicios, permitiendo compartir costes, capacidades y recursos. La capacidad para compartir servicios a través de la red de un modo eficiente y efectivo supone un paso crítico hacia el desarrollo de una economía orientada a servicios.

A pesar de los beneficios que implica el uso de los servicios, la capacidad de componer varios de ellos para obtener una funcionalidad final de mayor valor añadido es uno de los aspectos clave para maximizar sus potencialidades individuales a través de la integración de datos y de aplicaciones.

La composición de servicios presenta muchas oportunidades pero la tecnología aún no está lo suficientemente madura. De hecho, el desarrollo de servicios compuestos todavía es una tarea ad-hoc que consume mucho tiempo y requiere un considerable esfuerzo de programación a bajo nivel. El hecho de utilizar soluciones ad-hoc implica confiar en soluciones propietarias, lo cual dificulta la coordinación e interoperabilidad entre las diferentes empresas y servicios.

Desde la perspectiva del desarrollador, la composición de servicios ofrece posibilidades de reutilización, ya que permite desarrollar nuevos servicios de valor añadido reduciendo costes y tiempo de desarrollo. Desde la perspectiva del usuario, la composición de servicios permite acceder de manera fácil y transparente a una gran variedad de servicios complejos. En la actualidad, se ha identificado un gran potencial en cuanto a la aplicabilidad de la composición de servicios a diferentes sectores de la sociedad, como por ejemplo pueden ser la domótica, las telecomunicaciones o la automoción.

3.1.1 Tipos de Composición de Servicios

Un servicio (ver Figura 3-1) está compuesto de una especificación que define su funcionalidad, y de una implementación que permite ejecutar dicho servicio. En esta tesis, nos vamos a referir a la especificación de un servicio con el concepto de **servicio abstracto**, ya que sólo define la funcionalidad del servicio independientemente de la selección de la tecnología usada para su implementación. De este modo, un mismo servicio abstracto puede ser representado por múltiples implementaciones, a las cuales nos referiremos en esta tesis como **servicios concretos**.

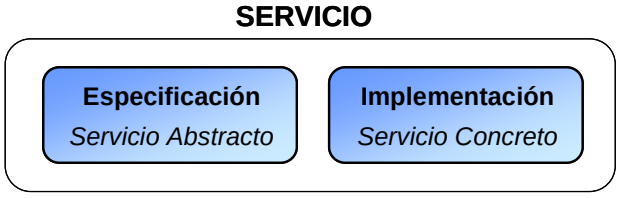


Figura 3-1: Servicio Abstracto y Servicio Concreto

Hoy en día, la mayor parte de las composiciones de servicios están enfocadas a crear servicios compuestos estáticos. Por estático se entiende que la asociación entre las especificaciones y las implementaciones o servicios concretos a utilizar es predefinida en tiempo de diseño y no se favorece la adaptación y los cambios en dicha composición

durante su ejecución, incluso a pesar de que haya problemas con algunos de los servicios concretos utilizados en la composición; como por ejemplo, que no esté disponible el servicio concreto o que no se comporte como se espera. Este tipo de composición es conocida como **composición estática**.

Sin embargo, en la actualidad está surgiendo la necesidad de añadir cierta dinamicidad a las composiciones de servicios de tal modo que se puedan obtener composiciones más adecuadas y adaptadas a las necesidades de cada momento y contexto de ejecución. De este modo, se podrían proporcionar servicios más flexibles y auto-configurables. Este tipo de composición se denomina **composición dinámica** y se caracteriza porque la asociación entre los servicios abstractos y los servicios concretos puede variar dinámicamente durante la ejecución de dicha composición según un conjunto de decisiones o alternativas. Dichas decisiones pueden depender de aspectos tales como las características del contexto de ejecución, fallos o errores en los servicios concretos que se utilizan, propiedades de calidad, etc.

La clasificación de una composición en estática o dinámica es compleja ya que una composición de servicios puede tener una parte estática y otra parte dinámica. Un buen ejemplo de composición estática y dinámica es una agencia de viajes. Una agencia de viajes ofrece un servicio completo de reserva de viajes, el cual es capaz de localizar un vuelo y un hotel para los días que el cliente desea.

En este caso, la agencia utiliza un servicio concreto de una compañía aérea asociada para reservar los billetes y otro servicio concreto de una cadena de hoteles para hacer la reserva de la habitación. La composición conoce bien la localización y acceso a dichos servicios concretos o implementaciones. Sin embargo, también tiene una parte dinámica ya que el sistema de la agencia puede expandir sus funcionalidades a servicios concretos ofrecidos por otras compañías aéreas o cadenas de hoteles en el caso de que no se puedan realizar las reservas con los servicios concretos asociados y predefinidos durante el diseño de la composición de servicios.

Teniendo en cuenta la hipótesis planteada en esta tesis, nuestra investigación se ha centrado en la composición dinámica de servicios.

3.1.2 Composición Dinámica de Servicios

Cuando se habla de composiciones dinámicas de servicios surgen tres ideas o retos principales a tener en cuenta:

- La composición de servicios debe de ser capaz de adaptarse y autoconfigurarse según el contexto de ejecución. Para ello, aparecen dos enfoques interesantes a tener en cuenta, los cuales permiten realizar una asociación dinámica de servicios concretos. Estos enfoques son conocidos como **re-binding y re-planning**.

  Cuando hablamos de **re-binding** estamos haciendo referencia a la tarea de reemplazar un servicio concreto por otro similar para cubrir la misma funcionalidad. Durante la fase de diseño de la composición, se pueden identificar varios servicios concretos candidatos para ofrecer la misma funcionalidad requerida en la composición o simplemente, durante la ejecución de la composición se puede realizar una búsqueda dinámica de servicios concretos existentes que satisfagan la especificación y los requisitos requeridos. El **re-planning** hace referencia al reemplazo de un servicio concreto por una composición de servicios que al final proporcione la misma funcionalidad. Es decir, la funcionalidad se puede dividir de tal modo que sea cubierta por diferentes servicios concretos que, al componerse, proporcionan la funcionalidad deseada.

  Supongamos que tenemos una composición que identifica tres funcionalidades: A, B y C. Cada una de dichas funcionalidades está asociada con un servicio concreto, (S1, S2 y S3, respectivamente). Pero además, en este caso, la funcionalidad B incluye una lista de posibles servicios concretos candidatos a utilizar en el caso de que ocurra algún problema con el servicio concreto S2.

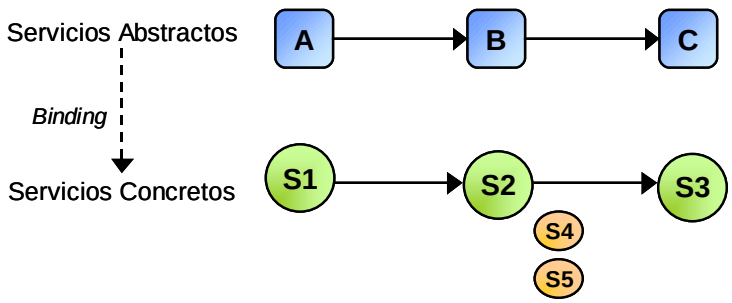


Figura 3-2: Ejemplo de una composición dinámica

Puede ocurrir que durante la ejecución de la composición, el servicio concreto S2 asociado en tiempo de diseño tenga que ser reemplazado, por ejemplo debido a un error. En este caso, el proceso de *re-binding* seleccionaría un nuevo servicio concreto de la lista de servicios concretos candidatos y modificaría la composición asociando la especificación B con el nuevo servicio concreto, en este caso sería el servicio concreto S4. También sería posible que se buscara dinámicamente un servicio concreto candidato.

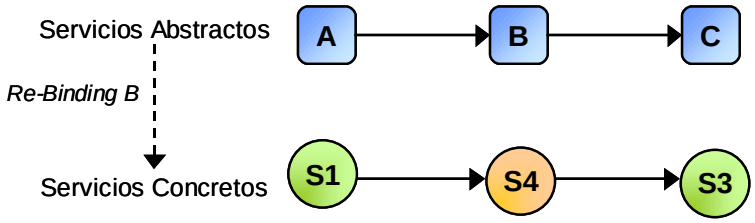


Figura 3-3: Composición dinámica: el concepto de Re-binding

En el caso de que no hubiera ningún servicio concreto disponible que cubriese dicha especificación, sería necesario hacer un *re-planning*. Tras realizar una búsqueda dinámica y analizar varios servicios concretos candidatos, se descubre que la sub-composición S2' formada por los servicios concretos S6 y S7 cubre la especificación B. Así, sustituyendo el servicio concreto S2 por la sub-composición S2' se realiza un *re-planning* de la composición final.

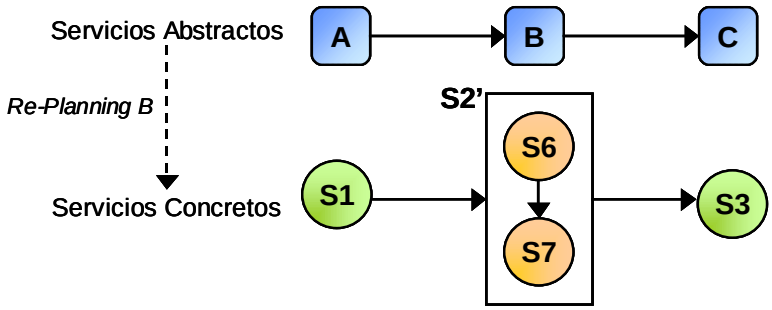


Figura 3-4: Composición dinámica: el concepto de Re-planning

Es importante resaltar que para poder explotar todo el potencial de las composiciones dinámicas, es necesario que se proporcione soporte a la búsqueda dinámica de servicios en base a requisitos y/o a especificaciones bien detalladas, con el fin de localizar servicios concretos candidatos satisfactorios en tiempo de ejecución.

- Una composición de servicios puede obtener diferentes servicios concretos candidatos para asociar a un servicio abstracto de dos maneras: buscándolos dinámicamente en tiempo de ejecución a partir de un conjunto de requisitos o, en el caso de que se haya asociado a dicho servicio abstracto una lista de diversos servicios concretos candidatos descubiertos en tiempo de diseño, utilizando dicha lista en tiempo de ejecución. Dicha lista también puede ser generada y asociada al servicio abstracto en tiempo de ejecución para utilizarla en caso de que fuera necesario.

- La asociación entre los servicios abstractos que forman la composición y los servicios concretos a utilizar es **estática** si se realiza en tiempo de diseño; y es una asociación **dinámica** si se realiza justo antes o durante la ejecución de la composición de servicios.

3.2 Enfoques de Composición de Servicios

La carencia de un mayor nivel de abstracción, y la necesidad de adaptarse al contexto de ejecución cuando se desarrollan composiciones de servicios han originado un considerable número de iniciativas e investigaciones tanto a nivel industrial como académico. Dichas iniciativas tienen como objetivo definir y obtener productos, estándares, plataformas y prototipos que cubran aspectos complementarios o carencias en el desarrollo de los servicios compuestos.

3.2.1 Enfoques Industriales

La industria ha tenido una gran influencia en el desarrollo y aceptación de diferentes iniciativas para la composición de servicios. Los enfoques industriales más importantes de la actualidad, los cuales están orientados a la tecnología específica de los servicios web, son WS-BPEL [WS-BPEL 2007], OWL-S [OWL-S 2004] y eFlow [Casati 2001].

WS-BPEL

Web Services Business Process Execution Language [WS-BPEL 2007], también conocido con el acrónimo WS-BPEL, es un estándar apoyado por el organismo de estandarización llamado *OASIS* desde Septiembre del 2002. WS-BPEL es un lenguaje de ejecución basado en XML que permite describir procesos de negocio sobre servicios web [Bastida y Benguría 2005]. Fue publicado inicialmente con el nombre de *Business Process Execution*

Language for Web Services (*BPEL4WS*) en Junio del 2002 por un grupo de trabajo formado por *BEA*, *IBM* y *Microsoft*.

BPEL4WS nació tomando propiedades de dos lenguajes propietarios con características diferentes pero complementarias. Uno de estos lenguajes es *WSFL* (*Web Services Flow Language*), desarrollado por *IBM*, el cual se basa en gráficos de actividades y conectores de control para definir un modelo global describiendo las interacciones entre los servicios web existentes y permitiendo la definición de nuevos servicios a partir de los existentes. WSFL soporta el manejo de excepciones pero no proporciona soporte directo para transacciones.

El otro lenguaje es *XLANG* desarrollado por *Microsoft*, el cual basa su funcionamiento en una notación orientada al comportamiento de intercambio de mensajes a través de los servicios web participantes, de esta manera automatiza el proceso de negocio. XLANG incluye soporte para la gestión de excepciones y para transacciones largas a través del uso transacciones de compensación. BPEL4WS combina el estilo orientado en gráficos de WSFL y el estilo algebraico propuesto por XLANG.

En Abril del año 2003 el organismo de estandarización OASIS formó un grupo de trabajo denominado *Web Services Business Process Execution Language* con el objetivo de continuar el trabajo sobre un lenguaje de proceso de negocio realizado por la especificación BPEL4WS 1.0.

En Mayo del 2003, BEA, IBM, Microsoft, SAP y Siebel publicaron la versión 1.1 de la especificación BPEL4WS a la vez que hicieron una propuesta formal a este grupo de trabajo formando en OASIS, que actualmente dirige la evolución y desarrollo de esta especificación. La versión 1.1 de BPEL4WS captó la atención de varios proveedores de software, favoreciendo la aparición en el mercado de diferentes motores de orquestación compatibles con la especificación BPEL4WS, como por ejemplo WebSphere Process Server de IBM o BPEL Process Manager de Oracle.

En Septiembre del 2005 se publicó el primer borrador de la segunda versión de esta especificación que pasó a llamarse *Web Services Business Process Execution Language* (WS-BPEL). En Mayo del 2007 WS-BPEL v2.0 es aprobado oficialmente como un estándar de OASIS.

WS-BPEL tiene como objetivo primordial la "orquestación" de servicios web, es decir, la definición de un nuevo servicio web a partir de servicios web existentes. Para ello, WS-BPEL define un modelo y una gramática para describir el comportamiento de un proceso de negocio basado en las interacciones entre el proceso y sus socios. La gramática definida por esta especificación está basada en XML y permite describir la lógica de control necesaria para coordinar los diferentes servicios web que componen un flujo de composición. La gramática puede ser interpretada y ejecutada por un motor de ejecución, el cual coordina las diferentes actividades que conforman el flujo del proceso.

Además WS-BPEL soporta extensiones a través de atributos o elementos con un espacio de nombres cualificado, con la única condición de que no deben alterar la semántica de ningún elemento o atributo del espacio de nombre de la especificación. Aunque este estándar proporciona mecanismos de gestión de transacciones y errores a través de compensaciones, no permite especificar características tan importantes en los procesos *Business-to-Business* (B2B) como son, por ejemplo, la autenticación y el no repudio.

De modo gráfico, un proceso definido en WS-BPEL tiene la siguiente forma:

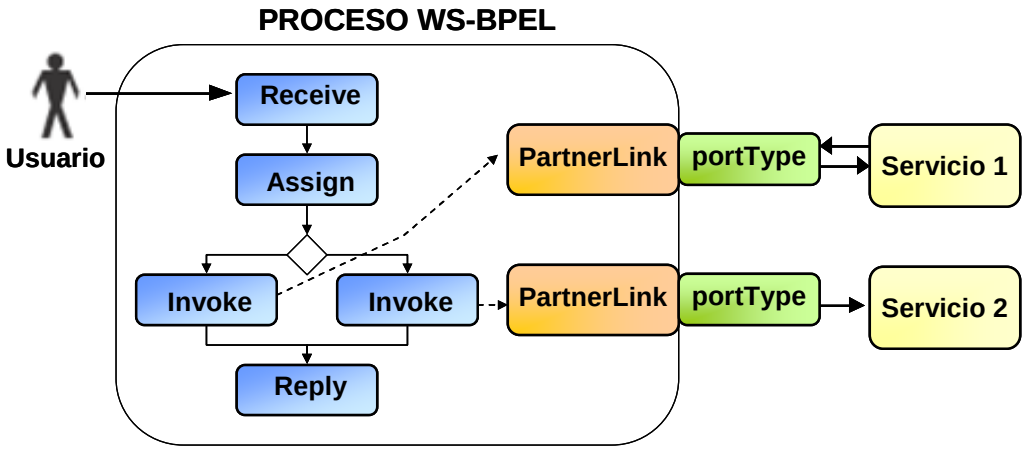


Figura 3-5: Proceso WS-BPEL

WS-BPEL distingue entre procesos abstractos y procesos ejecutables. Los procesos abstractos se utilizan para describir los protocolos del negocio o protocolos de intercambio de mensajes, los cuales describen las interacciones visibles externamente sin necesidad de definir la lógica interna del negocio. En cambio, los procesos ejecutables se encargan de definir la lógica del negocio del servicio asociado que se encuentra detrás de los protocolos externos. Esta distinción entre procesos abstractos y procesos ejecutables

permite que WS-BPEL pueda ser utilizado tanto para la coreografía de servicios, definiendo los mensajes públicos a intercambiar entre dos o más procesos abstractos, como para la orquestación de servicios, definiendo las llamadas a servicios y la especificación de cómo se llevan a cabo a través del uso de procesos ejecutables. Los dos tipos de procesos utilizan el mismo lenguaje de construcción, utilizando construcciones del lenguaje específicas según el tipo de proceso.

En la actualidad WS-BPEL tiene todavía algunas carencias relacionadas sobre todo con la manipulación de datos, que es bastante compleja de utilizar y la interoperabilidad, ya que muchos fabricantes añaden soluciones propietarias a la especificación con el fin de "completar" los vacíos de la especificación de WS-BPEL.

OWL-S

OWL-S también llamado *Semantic Markup for Web Services* [OWL-S 2004] es una iniciativa conjunta de *BBN Technologies*, la *Universidad de Carnegie Mellon*, *Nokia*, la *Universidad de Stanford*, *SRI International* y la *Universidad de Yale* que surgió con el fin de definir una ontología para la descripción de servicios web. Es la sucesora de la especificación DAML-S, siendo publicada la primera versión del OWL-S en el año 2003. La última versión disponible es la 1.1, publicada en Noviembre del año 2004 por el grupo de trabajo dentro de la *W3C*.

El objetivo de la especificación OWL-S es ayudar en la búsqueda, localización, invocación, composición y monitorización de los servicios web describiendo los servicios de tal modo que sean interpretables por las máquinas. Para ello, OWL-S permite modelar el concepto de servicio web, sus operaciones y su modelo de proceso así como establecer relaciones entre ellos de manera que un motor de inferencia sea capaz de razonar sobre esta ontología y lograr, así, automatizar el descubrimiento, invocación, composición, y monitorización de la ejecución de los servicios web.

Para definir esta ontología se utilizó como base todo el trabajo realizado en la estandarización de un lenguaje para definición de ontologías llamado OWL (Web Ontology Language) [OWL 2004] que está apoyado como una recomendación por la organización W3C desde Febrero del año 2004. OWL-S permite representar los servicios

semánticamente. Al basarse en OWL, soporta razonamientos sobre inclusión en taxonomías de conceptos y permite la relación entre conceptos.

En la Figura 3-6 se muestra la estructura de la ontología OWL-S. Como se puede observar se parte de un servicio. La especificación OWL-S define como servicio "*cualquier sitio accesible por web que no proporcione simplemente información estática sino que permita realizar alguna acción o cambio sobre el mundo, tales como la venta de un producto o el control de un dispositivo físico*" [OWL-S 2004].

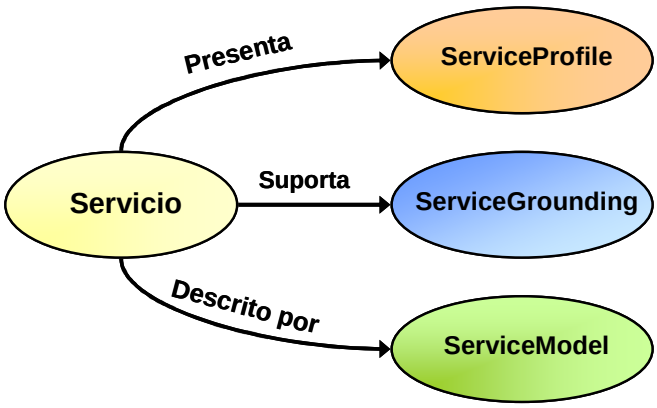


Figura 3-6: Visión general de OWL-S

Para describir dicho servicio se utilizan tres ontologías: ServiceProfile, ServiceModel y ServiceGrounding.

- **ServiceProfile**: Esta ontología describe qué es lo que hace el servicio. Es decir, permite describir el servicio ofrecido desde tres puntos de vista, la información del proveedor, la descripción funcional del mismo y los rasgos que especifican las características del servicio. Toda esta información es utilizada para publicar y posteriormente localizar un servicio web requerido por el usuario y proporcionado por un proveedor.

  Esta ontología define una relación bidireccional entre un servicio y un perfil, de manera que un servicio puede estar relacionado con un perfil y un perfil a un servicio.

- **ServiceModel**: Esta ontología define cómo funciona el servicio. Es decir, permite definir el protocolo de negocio que permitirá al cliente interactuar de una forma correcta con el servicio web así como los distintos procesos que intervienen durante la ejecución del servicio y sus precondiciones y efectos.

El objetivo de esta ontología es facilitar al cliente un mayor entendimiento sobre el funcionamiento del servicio de tal modo que el cliente pueda interactuar con dicho servicio de forma correcta y sencilla.

- **ServiceGrounding**: Permite describir cómo se accede al servicio, indicando también cómo utilizarlo.

  Las clases ServiceProfile y ServiceModel se consideran representaciones abstractas porque no proporcionan detalles concretos sobre el servicio, como el formato de los mensajes, los protocolos usados o la dirección URL de acceso al servicio. Sin embargo, el objetivo de la ontología ServiceGrounding es proporcionar información específica sobre cómo concretar en mensajes y operaciones las entradas y salidas abstractas asociadas a un proceso.

  Esta tarea para especificar los mensajes concretos ya la realiza la especificación WSDL [WSDL 2001] a través del mecanismo conocido con el nombre de asociación (en inglés *binding*). Por ello, OWL-S aprovecha el trabajo realizado en la especificación WSDL y define nuevos puntos de extensión. De este modo, OWL-S también toma ventaja de la relación de WSDL con SOAP y UDDI, permitiendo una extensión sencilla, para añadir contenido semántico a los servicios web tradicionales.

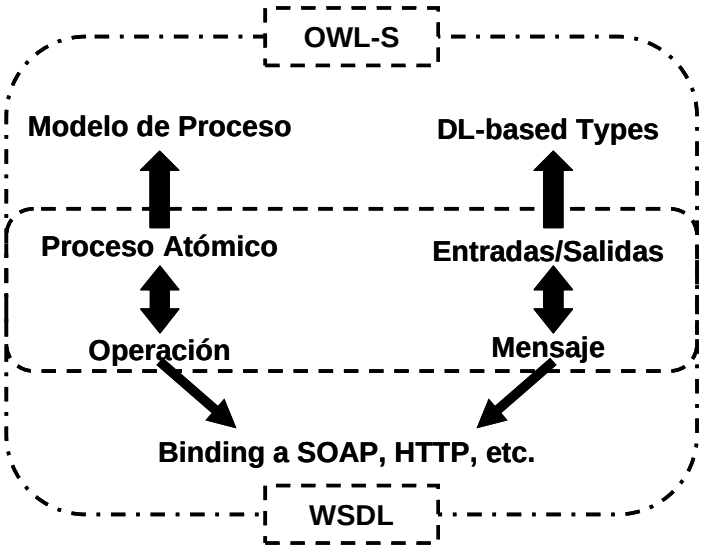


Figura 3-7: Relación entre OWL-S y WSDL

Así, las especificaciones de OWL-S y WSDL son complementarias ya que OWL-S no sustituye a WSDL. Ambas especificaciones se solapan en lo que en WSDL se denominan tipos abstractos, que definen las entradas y las salidas de un servicio. Mientras OWL -S proporciona capacidad semántica a dichas definiciones abstractas, WSDL define la información de asociación. La Figura 3-7 muestra la relación entre las especificaciones OWL-S y WSDL.

eFlow

Hewlett Packard ha definido un enfoque denominado eFlow [Casati 2000], [Casati 2001] que permite especificar, representar y monitorizar servicios compuestos en tiempo de ejecución. En eFlow, un servicio compuesto es descrito como un esquema del proceso que compone otros servicios básicos o compuestos, utilizando para ello un lenguaje propietario. Este esquema es modelado gráficamente para definir el orden de ejecución de los nodos que forman el proceso. Este grafo puede incluir nodos de servicio, nodos de decisión y nodos de evento. Las especificaciones de estos nodos se realizan en XML cuya sintaxis está determinada por un documento DTD [DTD 2000].

Una instancia de un proceso de servicio es una representación de un esquema de proceso. El mismo proceso de servicio puede ser instanciado múltiples veces y varias de dichas instancias pueden ser ejecutadas concurrentemente.

Como se ha mencionado anteriormente, son tres las clases de nodos principales definidos en el enfoque eFlow:

- **Nodos de servicio**: Representan la invocación a uno o más servicios. eFlow define tres tipos diferentes de nodos de servicio: nodos básicos, nodos multiservicio y nodos genéricos.

- **Nodos de decisión**: eFlow define tres tipos diferentes de nodos de decisión: nodos de división (*split*), nodos de unión (*join*) y nodos de enrutado genéricos.

- **Nodos de evento**: eFlow permite capturar y gestionar eventos de flujo de trabajo, eventos de datos, eventos de específicos de aplicaciones y eventos temporales. Para definir los eventos que pueden ser enviados o recibidos dentro de una instancia de

proceso se utilizan los nodos de evento. Se distinguen entre dos tipos de nodos de evento: nodos de notificación y nodos de petición.

El enfoque eFlow proporciona un modelo de proceso configurable que permite que las aplicaciones que se ejecutan en dispositivos con poca capacidad de almacenamiento y de procesamiento sean capaces de trabajar con una versión reducida de eFlow que sólo incluye aquellos aspectos necesarios para dicho entorno.

Así, esta capacidad de configuración permite utilizar una versión básica de eFlow que sólo define procesos secuenciales y lineales, sin incluir nodos de decisión o evento y sólo proporciona nodos de servicio básicos. Además, está versión básica tampoco proporciona aspectos de transacciones ni registro de auditoría. La versión básica de eFlow puede ser extendida de manera progresiva añadiendo más funcionalidades y haciendo un sistema más complejo teniendo en cuenta las capacidades del dispositivo que va a utilizar este enfoque para componer servicios.

La Figura 3-8 muestra los diferentes bloques de control para el modelo eFlow y cómo se pueden añadir aspectos más complejos a las configuraciones más básicas de eFlow.

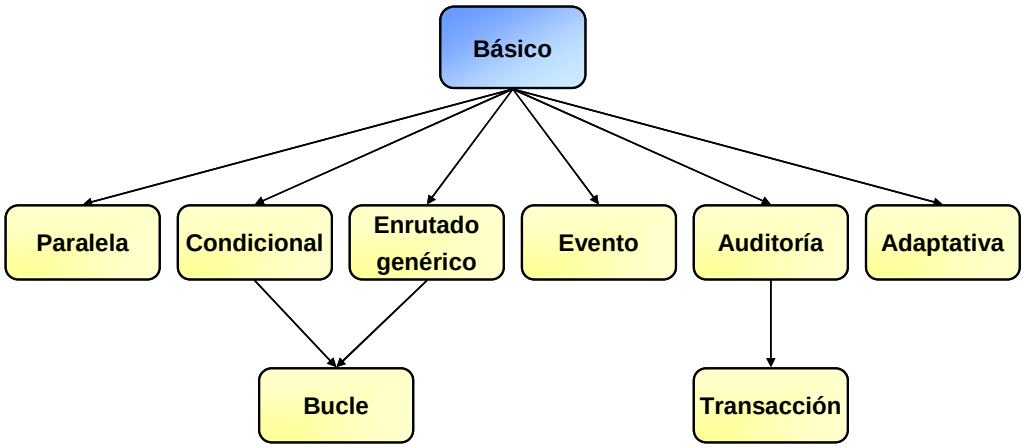


Figura 3-8: Bloques de control del modelo eFlow

Cada caja representa una capa y las flechas que relacionan dos capas indican que la configuración de la capa destino requiere como mínimo la característica proporcionada por la configuración de la capa origen.

eFlow también permite realizar extensiones al sistema a través del uso de componentes personalizados. Así, aunque eFlow proporciona componentes por defecto para definir, monitorizar, migrar instancias de procesos y seleccionar servicios a invocar, todos estos componentes pueden ser reemplazados por componentes definidos por los usuarios con el fin de extender las características del sistema y permitir el uso de diferentes estrategias y políticas de proveedores de servicios.

3.2.2 Enfoques Académicos

Durante los últimos años, también se han desarrollado diferentes iniciativas académicas en torno a la composición de servicios ([Lazcano et al. 2001], [Ludwig et al. 1999], [Mecella et al. 2002], [Pires et al. 2002], [Schuster et al. 2000], [Sun et al. 2003], [Yang et al. 2002], [Narayanan et al. 2002]). Dos de los enfoques considerados como los más relevantes son:

SELF-SERV

En SELF-SERV [Benatallah et al. 2003] [Sheng 2002] se utiliza un enfoque declarativo para describir los servicios compuestos que, posteriormente, serán ejecutados en un entorno punto a punto dinámico. SELF-SERV define tres tipos de servicios: servicios elementales, servicios compuestos y comunidades de servicios.

- Un **servicio elemental** es una aplicación web individual y accesible, por ejemplo un programa Java, que de manera explícita no hace uso de otro servicio web.

- Un **servicio compuesto** agrega múltiples servicios web que son referenciados como los componentes de dicho servicio compuesto. SELF-SERV define un lenguaje declarativo para definir servicios compuestos que está basado en diagramas UML de transición de estados, los cuales describen los estados, los parámetros de entrada y salida y los eventos que generan las transiciones. La información relativa al estado del servicio compuesto se obtiene del diagrama de estados. Una operación de un servicio compuesto tiene parámetros de entrada y de salida y también consume y genera eventos, y es el diagrama de estados donde se aglutina toda esta información. Este enfoque expresa la lógica de negocio de una operación de servicio compuesto como un diagrama de estado que define el flujo de invocaciones a las operaciones de los servicios que componen el servicio compuesto.

- Las **comunidades de servicios** es un concepto utilizado por el enfoque SELF-SERV para referenciar la composición de un gran número de potenciales servicios web dinámicos. Las comunidades de servicios son básicamente contenedores de servicios alternativos que proporcionan la descripción de aquellos servicios deseados (por ejemplo, servicios de reserva de vuelos) sin hacer referencia a ningún proveedor. Cuando una comunidad de servicios recibe en tiempo de ejecución una petición de ejecución para cierta operación, dicha petición se delega a uno de los miembros de la comunidad. La elección sobre a qué servicio se delega dicha petición depende de diversos aspectos como los parámetros de la petición, las características de los miembros de la comunidad, el historial de anteriores ejecuciones y el estado de las ejecuciones que estén ocurriendo en el momento dado.

SELF-SERV define otro concepto denominado **contenedor** de servicios, el cual facilita la composición de un conjunto de servicios potencialmente grande y cambiante. Un contenedor es un servicio que agrega otros servicios sustituibles los cuales proporcionan una capacidad común a través del mismo conjunto de operaciones bien sea directamente o por medio de una mapeo intermedio.

La ejecución de un servicio compuesto en SELF-SERV es coordinado por diferentes componentes software P2P llamados **coordinadores**. Cada estado de un servicio compuesto tiene asociado un coordinador. Cada uno de estos coordinadores es responsable de iniciar, controlar y monitorizar sus estados asociados y además, también se encarga de colaborar con los otros coordinadores para gestionar la ejecución del servicio.

El conocimiento adquirido en tiempo de ejecución por cada uno de los coordinadores involucrados en un servicio compuesto se extrae de manera estática del diagrama de estados del servicio y se representa en forma tabular en lo que se conoce como **tablas de enrutamiento**. Las tablas de enrutamiento contienen precondiciones y post-procesamientos. Las precondiciones son utilizadas para determinar cuando debería de ser ejecutado un servicio; mientras que los post-procesamientos se utilizan para determinar qué debería de ser hecho después de terminar la ejecución de un servicio. De este modo, los coordinadores no necesitan implementar ningún algoritmo complejo.

SELF-SERV adopta una arquitectura por capas con el fin de proporcionar soporte a la localización, creación, composición y despliegue de servicios web. La Figura 3-9 muestra los elementos de dicha arquitectura, los cuales se agrupan en cinco capas:

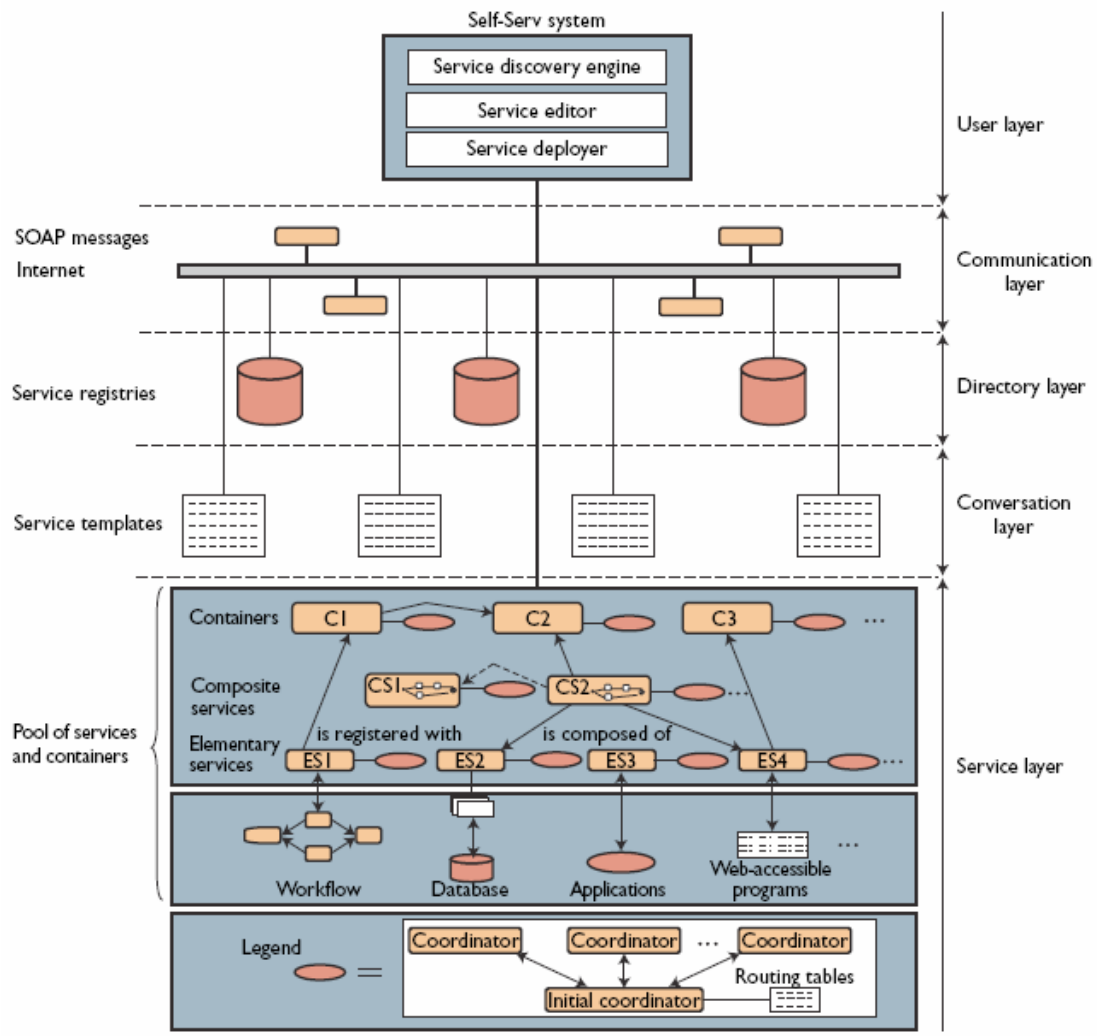


Figura 3-9: Arquitectura de SELF-SERV [Benatallah et al. 2003]

- **Capa de Servicio**: Esta capa consiste en una colección de servicios compuestos y de contenedores. Dentro de esta capa se gestiona la invocación a las operaciones de los servicios y, por otro lado, se recogen los resultados generados por cada invocación.

- **Capa de conversación**: Esta capa proporciona soporte a las interacciones estandarizadas entre los diferentes servicios. Para ello, define un conjunto de plantillas de servicio predeterminadas enfocadas en diferentes estándar B2B, como por ejemplo Electronic Data Interchange [EDI 2002], RosettaNet [RosettaNet 1998], o cXML [cXML 2007].

- **Capa de directorios**: Esta capa consiste en un conjunto de directorios que almacenan metadatos referentes a servicios y contenedores. Los directorios de metadatos contienen información que describe el significado, la categoría, las propiedades, las capacidades, la localización y la dirección de acceso de los servicios disponibles. La capa de usuario de SELF-SERV utiliza esos metadatos para localizar servicios y contenedores. Los proveedores de servicios pueden publicar estos metadatos en directorios o y registros de servicios como UDDI [UDDI 2005] o ebXML [ebXML 2001]. La implementación actual de SELF-SERV utiliza un registro UDDI centralizado.

- **Capa de comunicación**: Esta capa se encarga de gestionar la comunicación entre las diferentes capas basándose en el estándar de mensajes Simple Object Access Protocol [SOAP 2007].

- **Capa de usuario**: Esta capa proporciona acceso al entorno de composición de servicios a través de tres componentes principales: el motor de búsqueda de servicios, el editor de servicios y el gestor de despliegue de servicios.

Traverso y Pistore

En [Traverso et al. 2004], se propone un enfoque para realizar composiciones automáticas de servicios web. En este enfoque, se entiende por composición automática la tarea de generar automáticamente un nuevo servicio web que cumpla un objetivo dado a través de la interacción con otros servicios disponibles.

Este enfoque utiliza como punto de partida la especificación OWL-S para describir los servicios con el objetivo de facilitar la generación automática de los procesos ejecutables descritos con el lenguaje BPEL4WS.

A partir de un conjunto de servicios disponibles, este enfoque transforma los modelos de los procesos descritos con OWL-S en sistemas de transición de estados no determinísticos que describen las interacciones dinámicas con los servicios externos. Cuando se dice que los sistemas de transición de estados son no determinísticos, se está haciendo referencia a que tomar la misma acción en el mismo estado puede dar resultados diferentes. Esto se modela definiendo para cada transición un conjunto de estados diferentes.

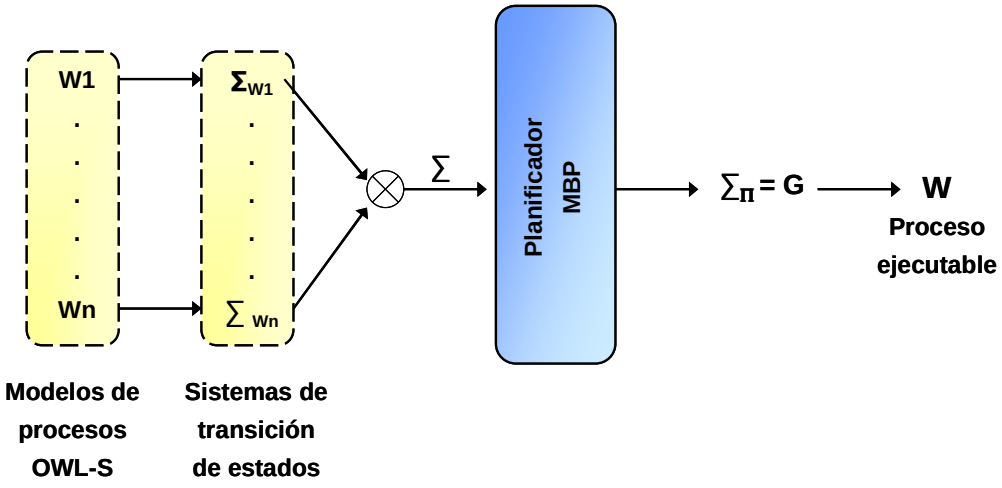


Figura 3-10: Arquitectura del enfoque de Traverso y Pistore

Tal y como se refleja en la Figura 3-10, dadas las descripciones de modelos de proceso de n servicios disponibles, (W_1,...,W_n), cada una de dichas descripciones se transforman en un sistema de transición de estados (ΣW_1,..., ΣW_n). Los sistemas de transición de estados proporcionan capacidad para añadir semántica operacional a los modelos de proceso OWL-S. Cada uno de esos sistemas describe un servicio web como un sistema dinámico basado en estados que puede evolucionar y además, puede ser parcialmente controlado y monitorizado por agentes externos. De esta manera, se describe un protocolo que define cómo los agentes externos al sistema pueden interactuar con el servicio.

Desde el punto de vista de un nuevo servicio compuesto que tiene que ser desarrollado, por ejemplo el servicio W, los sistemas de transición de estados $\sum W_1, \ldots, \sum W_n$ constituyen el entorno en el cual el servicio W tiene que no sólo operar sino también recibir y enviar peticiones de servicio. Estos sistemas de transición de estados se pueden considerar como un dominio de planificación $\sum$ que combina $\sum W_1, \ldots, \sum W_n$. Formalmente, esta combinación es un producto síncrono que permite a los n servicios evolucionar de manera independiente y en paralelo. $\sum$ representa todos los posibles comportamientos y evoluciones del dominio de planificación, sin que el servicio que va a ser desarrollado tenga ningún control sobre dicho dominio.

Una **función de transición** describe cómo la ejecución de una acción puede llevar desde un estado a otros posibles estados diferentes. Las evoluciones de los sistemas pueden ser monitorizadas a través de observaciones que describen la parte visible del estado del

sistema. Una **función de observación** define la observación asociada a cada estado del dominio.

El **objetivo de composición** G define algunos requisitos del comportamiento deseado del dominio de planificación. Dado ∑ y G, el planificador genera un plan π que controla el dominio de planificación, es decir, interacciona con los servicios externos $W_1, ..., W_n$ de tal manera que las evoluciones del dominio satisfacen el objetivo G. El plan π codifica el nuevo servicio W que tiene que ser desarrollado, el cual dinámicamente recibe y envía notificaciones de/a servicios externos $W_1, ..., W_n$, observa sus comportamientos y evoluciona dependiendo de las respuestas recibidas de dichos servicios externos. El plan π debe tener la capacidad de definir construcciones de programación comunes, como por ejemplo condiciones o bucles. Después, dicho plan π puede ser transformado en un proceso ejecutable utilizando un lenguaje de composición ejecutable como WS-BPEL.

3.2.3 Comparativa de los Diferentes Enfoques

El objetivo de esta sección es proporcionar una comparativa de los principales enfoques sobre composición de servicios descritos, e identificar las fortalezas y debilidades de cada uno ellos. Para ello, en [Dustdar et al. 2005] se identifican un conjunto de requisitos de vital importancia que deben tenerse en cuenta durante el diseño y desarrollo de servicios compuestos. Dichos requisitos son cinco:

- **Tipo de composición**: Hace referencia al momento en el que se realiza la asociación entre los servicios abstractos y los servicios concretos. Si ocurre en tiempo de diseño la composición es estática, mientras que si ocurre en tiempo de ejecución la composición es dinámica.

- **Composición automática**: La generación del flujo ejecutable de la composición puede ser un paso manual o automático.

  — Dicho paso es considerado manual o estático si se realiza antes de la ejecución y no después de ser requerido por un usuario.

  — En cambio, la composición se considera automática o dinámica si es realizada *on the fly*, es decir, justo después de la petición de un usuario pero antes de la ejecución y siempre en un tiempo razonable. O también si la generación de la

composición se realiza durante la ejecución, en el caso de que la composición necesite ser re-planificada por cambios en el flujo debidos por ejemplo, a servicios no disponibles.

- **Escalabilidad**: Es la capacidad de crecimiento de un servicio o sistema frente al aumento del número de servicios involucrados, con el objetivo de satisfacer todas las demandas realizadas. La escalabilidad es un factor importante ya que las composiciones dinámicas requieren técnicas escalables y flexibles. Componer dos servicios no es lo mismo que componer 20 o 100. Sin embargo, en la práctica, tanto los usuarios como las empresas consumen servicios compuestos que involucran a su vez otras cadenas de servicios.

- **Calidad de Servicio**: Hace referencia a la especificación de aspectos de calidad tales como rendimiento o seguridad.

- **Validación y Verificación**: Es necesario validar y verificar que el servicio compuesto es correcto y confiable. En otras palabras, hay que garantizar no sólo que el servicio compuesto cumplen con los requisitos establecidos, sino que los resultados ofrecidos por dicho servicio son los esperados.

La siguiente Tabla 3-1 recoge una comparativa de los cinco enfoques de composición descritos previamente frente a estos requisitos.

	WS-BPEL	OWL-S	eFlow	SELF-SERV	Traverso y Pistore
TIPO DE COMPOSICIÓN	Estática	Estática	Semi-dinámica	Semi-dinámica	Estática
COMPOSICIÓN AUTOMÁTICA	Manual	Manual	Semi-automática	Manual	Semi-automática
ESCALABILIDAD	Media	Media	Baja	Alta	Baja
QOS	No	Si	No	No	Si
VALIDACIÓN	No	No	Si	No	No

Tabla 3-1: Comparativa de los principales enfoques de composición

Analizando estos principales enfoques en el campo de la composición de servicios *desde un punto de vista técnico*, se extrae que la mayor parte del esfuerzo dedicado a nivel industrial se ha enfocado en la creación y definición de lenguajes ejecutables para formalizar las composiciones de servicios, como son WS-BPEL o OWL-S, sin definir métodos o guías que ayuden al diseño y desarrollo de dichas composiciones de servicios.

Además, la mayor parte de las soluciones actuales sólo permiten definir composiciones de servicios ya pre-planificadas, pre-especificadas y muy difíciles de extender y especializar. Sólo eFlow y SELF-SERV proponen soluciones más enfocadas a composiciones dinámicas, mientras que WS-BPEL, OWL-S o Traverso y Pistore sólo dan soporte a la composición estática.

La escalabilidad es un factor importante ya que las composiciones dinámicas requieren técnicas escalables y flexibles. La mayoría de las propuestas utilizan un enfoque centralizado, lo cual dificulta la escalabilidad de la composición; en cambio, SELF-SERV facilita la escalabilidad ya que utiliza un enfoque P2P. Además, ni WS-BPEL ni OWL-S dan soporte a la validación de la composición.

Por otro lado, los enfoques formales, a pesar de proporcionar mecanismos de verificación y validación, son difíciles de aplicar en un entorno empresarial real y presentan problemas de escalabilidad.

Como conclusión, eFlow y SELF-SERV pueden ser vistos como un primer intento de crear plataformas de composición de servicios teniendo en cuenta aspectos dinámicos y localización dinámica de servicios. Sin embargo, *desde un punto de vista metodológico*, los métodos de trabajos que definen están enfocados en proporcionar soporte a la aplicación de los sistemas propietarios que ofrecen, lo cual hace muy complejo y difícil generalizar el uso y aplicación de dichos métodos de trabajo a la composición de servicios. Desde un punto de vista técnico, eFlow ofrece un entorno comercial que utiliza un lenguaje de composición propietario y no se basa en estándares como WSDL [WSDL 2001] o UDDI [UDDI 2005]; mientras que SELF-SERV no es capaz de definir estrategias de selección de posibles servicios candidatos a utilizar fuera de las comunidades de servicios predefinidas en tiempo de diseño. Además, ambos enfoques son trabajos que no han evolucionado en los últimos cuatro años.

3.3 Ingeniería de Líneas de Producto

La industrialización ha generado un cambio significativo en el proceso de producción, pasando de la generación de los productos de un modo artesano, manual y dirigido a clientes particulares, a un proceso más automatizado y general. Por industrialización se entiende la aplicación de métodos y procesos en la industria con el objetivo final de permitir la automatización de la producción reduciendo al máximo la utilización de métodos artesanales.



Figura 3-11: Familia de productos de teléfonos móviles

Hace 100 años, en el dominio automovilístico Ford creó el concepto de *línea de productos*, el cual orientaba la producción hacia un mercado de masas mucho más barato que la creación de productos individuales y artesanales. [Parnas 1979] define el concepto de línea de productos como "*una colección de sistemas que comparten unas características comunes como una familia de sistemas*". Es decir, una línea de productos es un grupo o familia de productos que presentan ciertas características comunes y que satisfacen las necesidades específicas de un segmento de mercado concreto o un área de actividad específica. Sin embargo, la aplicación de la línea de producción reducía notablemente las posibilidades de diversificación.

A continuación se explican los principales conceptos relacionados con la variabilidad definidos en el enfoque de la Ingeniería de Líneas de Producto. Son varios los aspectos de variabilidad que se encuentran dentro del contexto de una línea de producto. A la representación de cada uno de estos aspectos de variabilidad se conoce como **Punto de Variación** (en inglés, *Variation Point*). [Pohl et al. 2005] define un Punto de Variación como

la representación de un aspecto de variabilidad a través de un artefacto del dominio enriquecido con información sobre el contexto.

Cada aspecto de variabilidad está compuesto por diferentes objetos de variabilidad, los cuales se denominan **Variantes** (en inglés, *Variants*). [Pohl et al. 2005] define una Variante como la representación de un objeto de variabilidad con un artefacto del dominio.

Tomemos como ejemplo el aspecto de variabilidad 'color', el cual identifica una propiedad de los objetos del mundo real. 'Color' tiene asociados diferentes objetos de variabilidad, tales como 'rojo', 'blanco', 'azul', 'verde', etc. Si una compañía quiere fabricar teléfonos móviles de diferentes colores, entonces definiría un punto de variación denominado 'Color del móvil'.

Tómese en consideración el caso de que dicha compañía quisiera fabricar teléfonos rojos y verdes. Entonces sólo habría definidas dos Variantes asociadas a dicho Punto de Variación del dominio de los móviles, que serían 'rojo' y 'verde'.

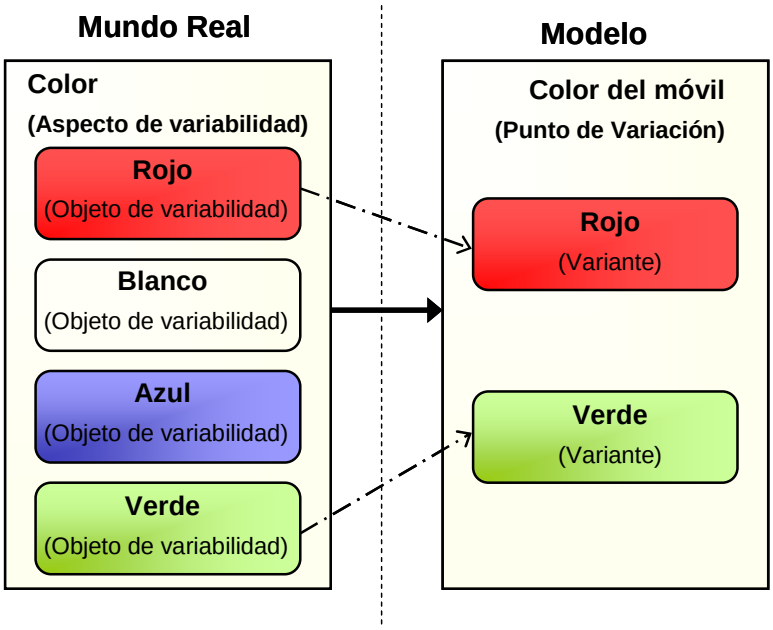


Figura 3-12: Representación de la Variabilidad en el mundo real

En la Figura 3-12 se recoge la relación entre la variabilidad que se presenta en el mundo real y la variabilidad definida en un modelo sobre dicha realidad.

3.3.1 La Ingeniería de Líneas de Producto Aplicada al Software

El proceso de desarrollo de software ha pasado de ser una tarea manual a convertirse en un proceso industrial. Bajo esta premisa, y haciendo referencia a otros procesos industriales que han ocurrido en diversos sectores (automoción, aeronáutica, etc.), la tecnología denominada Ingeniería de Líneas de Producto [Northrop 2002] se presenta como un elemento clave en este proceso de industrialización de la producción de software.

Supongamos que proporcionamos el mismo documento de especificación de requisitos a un grupo formado por cinco analistas de sistemas y cinco programadores dentro de la misma organización. Lo que se obtiene al final son cinco diseños diferentes y, por supuesto, cinco implementaciones o cuerpos de código fuente también distintos. La pregunta que surge es por qué en el mundo del software los analistas y desarrolladores siguen invirtiendo una ingente cantidad de tiempo y dinero en escribir diferentes líneas de código para aplicaciones similares con diseños y requisitos similares.

Así, nace el término de Ingeniería de Líneas de Producto Software [SEI 2007] como un nuevo paradigma para el desarrollo de aplicaciones software (orientado a sistemas de software intensivo y productos software), que permite a las empresas mejorar la calidad del software que producen así como reducir los costes y tiempos de lanzamiento.

En el campo de la Ingeniería del Software, la tecnología de las líneas de producto hace referencia a la generación automática de aplicaciones software y sistemas que presentan ciertas similitudes dentro de una misma familia pero, al mismo tiempo, difieren las unas de las otras en diversos aspectos bien definidos. En [Pohl et al. 2005] se distingue entre los conceptos variabilidad y comunalidad dentro del ámbito de la Ingeniería del Software.

La **variabilidad** de una línea de producto software es aquella variabilidad que puede ser modelada con el objetivo de facilitar el desarrollo de aplicaciones personalizadas a través de la reutilización de artefactos predefinidos y ajustables. Es decir, la variabilidad de una línea de producto software distingue entre distintas aplicaciones de una línea de producto. Por ejemplo, los clientes de un sistema de gestión administrativo tienen la posibilidad de elegir el lenguaje del interfaz de usuario antes de que el sistema sea

instalado. Es más, como característica adicional, pueden obtener una versión multilingüe que les permite cambiar el lenguaje de la aplicación en cualquier momento.

La comunalidad se refiere a aquellas características que son parte de cada aplicación exactamente de la misma forma. Por ejemplo, el interfaz de usuario de un programa de gestión ofrece a los usuarios la posibilidad de elegir el lenguaje de la aplicación. Esta característica es parte de cada uno de los sistemas de gestión vendidos a diferentes clientes.

Resumiendo, en la Ingeniería de Líneas de Producto Software es posible decidir si una característica es variable para cierta línea de producto (variabilidad), o si es una característica común a todas las aplicaciones software lo cual se conoce como **comunalidad**.

3.3.2 Beneficios de la Ingeniería de Líneas de Producto

El principal objetivo que persigue la Ingeniería de Líneas de Producto Software es proporcionar productos software personalizados a un coste razonable.

A continuación se describen algunos de los diferentes motivos y ventajas que presenta este paradigma al aplicarlo en el desarrollo de software [Pohl et al. 2005], [van de Linden et al. 2007].

- Reducción de los costes de desarrollo. Gracias a la reutilización del código, se produce una reducción en los costes en el desarrollo de nuevas aplicaciones y software similares. Hay que tener en cuenta que para poder aplicar la idea de reutilización, es necesario que previamente se realicen ciertas inversiones de tiempo y esfuerzo.

- Aumento de la calidad. El hecho de reutilizar partes de código en diferentes sistemas y aplicaciones permite que dicho código sea revisado y probado en diversos entornos, lo cual ofrece una mayor posibilidad de detectar errores y corregirlos.

- Reducción del tiempo de mercado. Un factor crítico de éxito para cualquier producto, incluso el software, es el tiempo que se invierte desde que se inicia el desarrollo de dicho producto hasta que sale al mercado. Desde el punto de vista de la Ingeniería de Líneas de Producto, en las primeras etapas de su implantación el tiempo de mercado

es realmente alto, pero este tiempo se ve reducido de forma considerable cuanto más se reutilice para crear nuevos productos.

Gracias a todas estas ventajas de la aplicación del paradigma de Ingeniería de Líneas de Producto en el ámbito del software, los clientes pueden obtener productos adaptados a sus necesidades y deseos a un precio razonable y con ciertas garantías de calidad.

3.4 Modelado Formal de Metodologías

Las metodologías de desarrollo de software normalmente se describen pensando en los desarrolladores. Así, se habla de qué tareas y qué técnicas se deben utilizar, de cuál es el ciclo de vida más apropiado a aplicar (por ejemplo, en cascada o lineal) o cómo se organizan las diferentes tareas y se asignan a las personas adecuadas para llevarlas a cabo. Para describir estas metodologías de desarrollo de software han ido surgiendo diferentes metamodelos que permiten especificar las metodologías de manera formal, más allá de hacerlo en texto puro.

Los metamodelos permiten especificar los diferentes conceptos, reglas y relaciones utilizados para definir un conjunto de metodologías que pertenecen a la misma familia, como por ejemplo las metodologías aplicadas al campo del software. Aunque es posible describir una metodología sin utilizar un metamodelo explícito, el uso de un metamodelo permite formalizar las ideas de la metodología; de tal modo que se facilitan las posibles extensiones o modificaciones que se vayan a realizar en la metodología e, incluso, comprobar la consistencia de la misma.

Para que un metamodelo sea completo es necesario que cubra cada uno de los diferentes aspectos de una metodología; por ejemplo, debe permitir definir qué procesos hay que seguir, qué productos de trabajo se van a obtener o quiénes son los responsables de realizar las diferentes tareas.

Los metamodelos son utilizados por los metodólogos para construir o modificar metodologías que después, serán utilizadas por los desarrolladores de software en sus proyectos de software.

Son dos los metamodelos más relevantes en el campo de las metodologías de software: *Software Process Engineering Metamodel* (*SPEM*) [SPEM 2007] definido dentro del organismo de estandarización denominado *Object Management Group* (*OMG*) y el estándar ISO/IEC 24744 *Software Engineering-Metamodel for Development Methodologies* definido dentro del organismo de estandarización denominado *International Organization for Standardization* (*ISO*) [ISO/IEC 24744 2007].

El enfoque que hemos adoptado para especificar la metodología de composición de servicios propuesta en el capítulo 5 consiste en definir aquellas fases, procesos, tareas, roles, productos de trabajo y responsabilidades específicas de la definición y desarrollo de composiciones de servicios, tanto estáticas como dinámicas y sensibles al contexto de ejecución. Y para modelar de manera formal toda esta información hemos seleccionado la notación propuesta por el metamodelo SPEM v2.0.

Después de examinar de manera crítica ambos metamodelos, las principales motivaciones para utilizar SPEM v2.0 como lenguaje de modelado de la metodología de composición han sido las siguientes:

- SPEM ha sido desarrollado por el organismo *OMG*, lo cual ha favorecido una amplia aceptación en el mundo industrial. El metamodelo ISO/IEC 24744 es un estándar de reciente creación (Febrero 2007), por lo que aún es poco conocido y aceptado. Además, el metamodelo ISO/IEC 24744 ha sido desarrollado con un enfoque académico, al contrario que el metamodelo SPEM que ha nacido del mundo industrial.

- SPEM es un metamodelo diseñado para describir procesos y sus componentes, siguiendo un enfoque de modelado orientado a objetos con base en UML [UML 1997]. UML es el lenguaje de modelado aceptado entre las organizaciones de desarrollo de software, lo que puede facilitar la asimilación de SPEM como lenguaje de modelado de procesos en este tipo de organizaciones. El metamodelo ISO/IEC 24744 también sigue un enfoque de modelado basado en UML, pero añade nuevos conceptos (como los *clabjects*) que dificultan su comprensión, implementación y aplicación.

- SPEM proporciona una notación gráfica estándar bien definida, mientras que el metamodelo ISO/IEC 24744 no define, hasta el momento, ninguna notación gráfica.

- Finalmente, para modelar procesos con SPEM es posible usar cualquier herramienta que permita el uso de perfiles de UML. Dado el extendido uso de UML, existen muchas herramientas disponibles con esta característica. Incluso, entornos ampliamente utilizados como plataformas de desarrollo de software, comienzan a proporcionar facilidades para modelar procesos con base en SPEM, tal es el caso del proyecto *Eclipse Process Framework* [EPF 2007] o de su proyecto homólogo dirigido por IBM denominado *Rational Method Composer* [RMC 2007]. En la actualidad no existe ninguna herramienta que de soporte al modelado de procesos con ISO/IEC 24744.

3.4.1 Descripción de SPEM 2.0

Para describir formalmente la metodología propuesta se ha utilizado el lenguaje de modelado SPEM versión 2. En esta sección se describen las principales características de este lenguaje de modelado y los principales elementos que hemos utilizado en la formalización de la metodología.

SPEM 2.0 se utiliza para definir procesos de desarrollo de software y sistemas así como los componentes que forman dichos sistemas. El alcance de SPEM está limitado a definir los elementos mínimos necesarios para definir procesos de desarrollo software y de sistemas, sin añadir aspectos específicos de dominios de desarrollo particulares.

El metamodelo de SPEM 2.0 está dividido en siete paquetes de metamodelos, estructurando el modelo en unidades lógicas. Cada unidad extiende aquellas otras unidades de las que depende, proporcionando estructuras y capacidades adicionales a los elementos definidos.

Estos siete paquetes o unidades lógicas son las siguientes:

- ***Core*** (Núcleo): Contiene aquellas clases y abstracciones del metamodelo que forman la base para construir los elementos de cada uno de los seis paquetes de metamodelos restantes.

- ***Process Structure*** (Estructura del proceso): Define la base para todos los modelos de procesos. Este metamodelo proporciona soporte a la creación de modelos de procesos simples y flexibles. Su estructura de datos principal se basa en un desglose de

actividades anidadas donde cada actividad mantiene listas de referencias a clases de tipo Rol y de tipo Producto de Trabajo.

- ***Process Behavior*** (Comportamiento del proceso): Representa un proceso como una estructura estática que define las dependencias entre las distintas actividades. Este metamodelo permite extender la estructura del proceso con modelos de comportamiento.

- ***Managed Content*** (Contenido gestionado): En la mayoría de las ocasiones, hay información asociada a un proceso que define buenas prácticas, pero que no puede ser formalizada en modelos y tiene que ser expresada en lenguaje natural. Este metamodelo introduce conceptos para representar y gestionar el contenido textual de dichas descripciones y documentaciones relevantes para la comprensión del proceso.

- ***Method Content*** (Contenido del método): Proporciona los conceptos necesarios para que los usuarios de SPEM 2.0 puedan definir una base de conocimiento de desarrollo que sea independiente de cualquier proceso o proyecto de desarrollo específico. Este metamodelo define todas las explicaciones textuales relacionadas con los pasos a seguir por el proceso, describiendo cómo los objetivos son alcanzados por los roles a través del uso de diversos recursos y resultados. Todo ello de manera independiente de cómo haya que realizar dichos pasos dentro de un ciclo de desarrollo específico.

- ***Process With Methods*** (Proceso con métodos): Define nuevas estructuras y redefine estructuras existentes para permitir la integración entre los procesos definidos con el metamodelo *ProcessStructure* y las instancias de los conceptos del metamodelo *MethodContent*.

- ***Method Plugin*** (Plugin del método): Introduce conceptos para diseñar y gestionar librerías o repositorios de procesos y contenidos de métodos reutilizables, complejos y configurables.

Los elementos que hemos utilizado para definir y formalizar la metodología de composición de servicios son los definidos principalmente en los paquetes de metamodelo *ManagedContent*, *MethodContent*, *ProcessWithMethods* y *MethodPlugin*.

Con el objetivo de facilitar la comprensión de la aplicación de SPEM 2.0 a la metodología, en la siguiente tabla se describen aquellos conceptos y elementos del metamodelo de SPEM 2.0 y sus correspondientes iconos que han sido utilizados para modelar la metodología propuesta en el capítulo 5.

Dentro del metamodelo *ManagedContent* los elementos utilizados son:

ELEMENTOS	DESCRIPCIÓN	ICONO
Guidance (Guía)	Proporciona información adicional en forma de guía sobre una estructura específica o un tipo de contenido.	

Dentro del metamodelo *MethodContent* los elementos utilizados son:

ELEMENTOS	DESCRIPCIÓN	ICONO
RoleDefinition (Rol)	Define un papel o rol involucrado en el proceso caracterizándolo con un conjunto de habilidades, competencias y responsabilidades. Los roles realizan tareas y son responsables de uno o varios productos de trabajo. Un rol puede estar involucrado en una tarea de modo obligatorio, recomendado u opcional.	
Step (Paso)	Describe una parte de todo el trabajo que hay que realizar en una tarea. Una tarea está compuesta de un conjunto de pasos que permiten alcanzar el objetivo de dicha tarea.	
TaskDefinition (Tarea)	Define una tarea o unidad de trabajo asignable. Cada tarea es asignada a uno o varios roles y referencia también a uno o varios productos de trabajos.	
WorkProduct Definition (Producto de trabajo)	Describe los productos de trabajo, en la mayoría de los casos tangibles, que son consumidos, producidos o modificados por las tareas. Estos productos de trabajo pueden servir como base para definir activos reutilizables. Los roles utilizan productos de trabajo para realizar tareas y producen productos de trabajo en el transcurso de la ejecución de las tareas.	

Dentro del metamodelo *ProcessWithMethods* los elementos utilizados son:

ELEMENTOS	DESCRIPCIÓN	ICONO
Activity (Actividad)	Representa un grupo de elementos asociados, como por ejemplo, instancias de otras actividades, fases, procesos, usos de tareas, usos de roles, etc.	
RoleUse (Uso de rol)	Representa la utilización o involucración de un rol en el contexto de una actividad específica, como por ejemplo, una tarea.	
TaskUse (Uso de tarea)	Representa la utilización o involucración de una tarea en el contexto de una actividad específica, como por ejemplo, un proceso.	
WorkProduct Use (Uso de producto de trabajo)	Representa la utilización o involucración de un producto de trabajo en el contexto de una actividad específica, como por ejemplo, una tarea.	

Dentro del metamodelo *MethodPlugin* los elementos utilizados son:

ELEMENTOS	DESCRIPCIÓN	ICONO
Concept (Concepto)	Tipo específico de guía que define las principales ideas asociadas con el elemento al que referencia.	
Template (Plantilla)	Tipo específico de guía que predefine un formato estandarizado y una tabla de contenido a seguir para definir un producto de trabajo.	
Toolmentor (Herramienta)	Tipo específico de guía que define cómo usar una herramienta para alcanzar un objetivo de una tarea o una actividad.	
Artifact (Artefacto)	Tipo específico de producto de trabajo que describe y define un producto de trabajo tangible.	
Deliverable (Entregable)	Tipo específico de producto de trabajo que representa un resultado obtenido de un proceso y que tiene cierto valor para el cliente o usuario.	

Outcome (Resultado)	Tipo específico de producto de trabajo que describe y define un producto de trabajo no tangible.	
Phase (Fase)	Tipo específico de actividad que representa un período significativo de un proyecto.	
Process (Proceso)	Tipo específico de actividad que describe una estructura de ciertas clases de proyectos de desarrollo o partes de ellas. Los procesos de desarrollo se componen de secuencias de fases que definen el ciclo de vida del producto a desarrollar.	

3.5 Marco de Trabajo NIMSAD

Para la evaluación de la metodología de composición de servicios propuesta se ha seleccionado el marco de trabajo denominado "*Normative Information Model-based System Analysis and Design*", también conocido con el acrónimo NIMSAD [Jayaratna 1994], entre los diferentes enfoques existentes de evaluación de metodologías ([Avison y Fitzgerald 1995], [Kitchenham 1996]), porque NIMSAD es considerado el enfoque más apropiado a utilizar en la evaluación de metodologías orientadas a objetos para el desarrollo de Sistemas de Información. El enfoque NIMSAD tiene en cuenta aspectos "duros" y "suaves" de la metodología[1] de una forma compensada, habiendo sido aplicado con éxito en múltiples casos, y gozando de una amplia aceptación en la comunidad de sistemas de información [Avison y Fitzgerald 1995] [Berki y Georgiadou 1998].

El entorno de trabajo NIMSAD tiene tres objetivos principales:

- Servir como medio para comprender, de manera general, el contexto del problema a solucionar;
- Ayudar a evaluar metodologías, analizando su estructura, pasos, forma y naturaleza; y

[1] Un aspecto "suave" de una metodología hace referencia principalmente a factores humanos; mientras que un aspecto "duro" está orientado a características más técnicas.

- Ayudar a obtener conclusiones.

NIMSAD es un marco de trabajo sistemático e independiente de cualquier metodología que pueda usarse para entender y evaluar metodologías; y cuyo objetivo es promocionar la comprensión de los procesos resolutorios de problemas en general. El enfoque NIMSAD está basado en los modelos y la epistemología de los sistemas pensantes [Checkland 1981].

De acuerdo con el enfoque NIMSAD, se considera que una metodología es una forma explícita de estructurar los pensamientos y acciones. Las metodologías contienen modelos y reflejan perspectivas particulares de la realidad basadas en paradigmas filosóficos. Una metodología debería especificar ***qué*** pasos son necesarios, ***cómo*** realizar esos pasos, y lo más importante, ***por qué*** dichos pasos deben realizarse en un orden particular. Por lo tanto, una metodología puede ser definida como un conjunto de pasos y actividades explícitamente ordenados con una justificación convincente y un conjunto de buenas prácticas sobre cómo realizar dichos pasos y actividades para conseguir una cierta transformación.

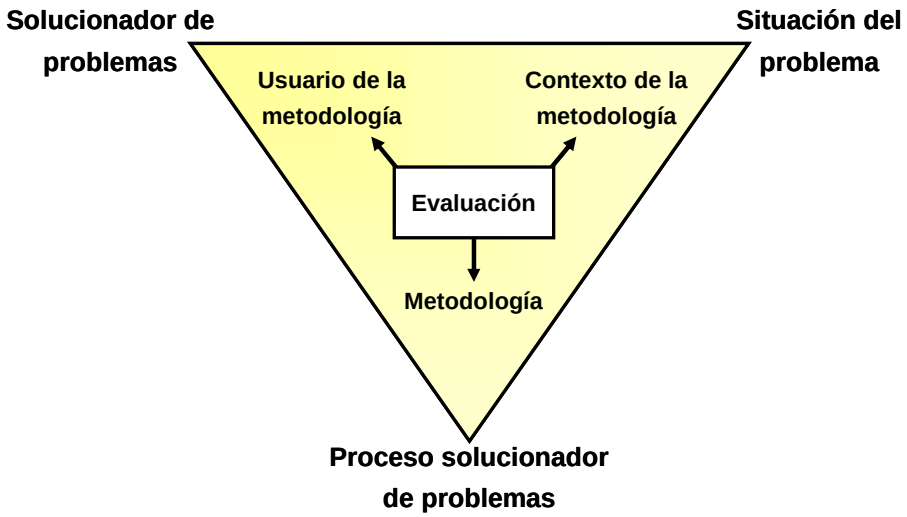


Figura 3-13: Elementos del marco de trabajo NIMSAD

Como puede verse en la Figura 3-13 el marco de trabajo NIMSAD consiste en tres elementos esenciales, los cuales constituyen la principal perspectiva para la evaluación de una metodología.

Estos tres elementos son:

- **La situación del problema**: Este elemento hace referencia al tipo de problemas que una metodología puede resolver y a los propietarios de dichos problemas. Es decir, la situación del problema se enfoca en el contexto de la metodología.

- **El solucionador de problemas**: Este elemento hace referencia a los usuarios de la metodología. Se enfoca en las personas que aplican una metodología a una situación de un problema determinado. Estos solucionadores de problemas seleccionan aquellos elementos de la situación del problema que son relevantes. Esta selección, aunque a veces pueda estar indicada por la metodología utilizada, está influenciada de gran manera por la construcción mental del usuario de la metodología. Por construcción mental se entiende la estructura mental de uno mismo y consiste en un número de partes útiles a saber, proceso perceptivo, valores, ética, motivos y prejuicios, habilidad de razonamiento, experiencias, habilidades y conjuntos de conocimientos, procesos de estructuración, modelos, entornos de trabajo y roles.

- **El proceso solucionador de problemas**: Este elemento hace referencia a la forma en la que se realiza la resolución del problema, es decir, al proceso de la metodología. El proceso solucionador de problemas se descompone a su vez en tres fases esenciales, las cuales se expanden para formar ocho pasos detallados que son aplicables a cualquier proceso de resolución de problemas. Estas fases y sus respectivos pasos se detallan a continuación:

  — **Fase 1: Formulación del problema**. Esta fase se compone de cinco pasos:
    - Comprensión de la "situación de preocupación": La "situación de preocupación" es una situación en la que varias personas perciben problemas. El alcance de esta situación problemática determina el enfoque de la investigación y establece la "situación de preocupación" real.
    - Realización del diagnóstico: El diagnóstico es la proyección explicita o la expresión del entendimiento que se obtiene tras investigar la "situación de preocupación". Por lo tanto, este paso intenta responder a la pregunta de dónde estamos ahora.

- Definición del resultado pronosticado: El resultado de un pronóstico es la expresión de una situación deseada. Este paso se enfoca en la definición del estado deseado para la "situación de preocupación" actual e intenta responder a la pregunta de dónde queremos estar y por qué.
- Definición de problemas: Los problemas impiden alcanzar el estado deseado. Este paso se enfoca en qué es lo que impide que el estado actual cambie al estado deseado.
- Derivación de sistemas teóricos: Los sistemas teóricos son construcciones teóricas que se consideran importantes para resolver los problemas de la "situación de preocupación". Se espera que sus partes se conviertan en la base para superar los problemas y por lo tanto para transformar la "situación de preocupación".

— **Fase 2: Diseño de la solución**. Esta fase se compone de dos pasos:

- Realización del diseño conceptual/lógico: El resultado de este paso es la producción de la especificación de un diseño lógico acordado y aceptable que establece la naturaleza y la función de las partes lógicas del estado deseado de la "situación de preocupación". Por "lógica" se entiende la argumentación de dichas partes como útiles y esenciales para la realización de los sistemas teóricos y del estado deseado de la "situación de preocupación".
- Realización del diseño físico: El diseño físico puede ser considerado como la selección de un conjunto de formas y maneras para realizar el diseño lógico con un número determinado de recursos, instalaciones y otros factores del entorno.

— **Fase 3: Implementación del diseño**. Esta fase se compone de un único paso:

- Implementación de los diseños: La implementación está relacionada con la realización de los sistemas teóricos dentro del contexto de la "situación de preocupación".

El entorno de trabajo NIMSAD está diseñado para realizar preguntas a las metodologías sobre qué elementos del marco de trabajo cubren, y en qué orden lo hacen. Por lo tanto el rol del entorno de trabajo es ayudar a cuestionar las metodologías sobre qué intentan transformar, por qué intentan transformarlo y cómo facilitan a los usuarios de la

metodología llevar a cabo dicha transformación, facilitando de esta manera la forma de debatir la justificación tanto de los pasos de la metodología como de su estructura. Esta forma de evaluación mide la efectividad del proceso solucionador de problemas y al solucionador en la "situación problemática", pero hay que tener en cuenta que su aplicación requiere considerar conscientemente la base de la justificación y las relaciones de las actividades sugeridas por los pasos de la metodología que va a ser evaluada.

De acuerdo con estas observaciones, la metodología de composición de servicios se va a evaluar extrayendo de la definición del enfoque NIMSAD un número de preguntas relacionadas con cada uno de los elementos definidos en NIMSAD, e intentando responder dichas preguntas teniendo en cuenta las propiedades, las fases, los pasos, los modelos y la estructura de la metodología propuesta tal y como se describe en el capítulo 5. Estas preguntas corresponden a las características que la metodología propuesta debería tener de acuerdo con el enfoque NIMSAD.

3.6 Valoración del Estado del Arte

La mayoría de los actuales esfuerzos realizados en torno a la composición de servicios se centran en la composición estática de servicios. Algunos de los enfoques o iniciativas más destacables, como eFlow y SELF-SERV, pueden ser vistos como un primer intento de crear plataformas de composición de servicios teniendo en cuenta aspectos dinámicos y localización dinámica de servicios. Sin embargo, desde un punto de vista metodológico, los métodos de trabajos que definen están enfocados en proporcionar soporte a la aplicación de los sistemas propietarios que ofrecen, lo cual hace muy complejo y difícil la aplicación de dichos métodos de trabajo de manera general al diseño y desarrollo de composiciones de servicios.

Para poder definir y gestionar de manera eficiente las composiciones de servicios, es necesario proporcionar especificaciones, lenguajes y estándares que cubran los diferentes requisitos de las composiciones de servicios, así como aplicaciones, herramientas y metodologías que permitan ponerlas en práctica.

Por ello, a pesar de los diferentes esfuerzos realizados en torno a la composición de servicios, existe una carencia de soportes metodológicos para el desarrollo de

composiciones de servicios tanto estáticas como dinámicas. Futuras iniciativas deberán ampliar el campo de investigación de la composición hacia un enfoque más dinámico, facilitando la asociación dinámica de servicios concretos a la composición durante su ejecución y permitiendo que la composición se adapte dinámicamente a las características del contexto de ejecución.

Durante los últimos años, se ha estado trabajando en el desarrollo de un entorno que permita componer servicios no sólo estáticos sino también dinámicos y sensibles al contexto de ejecución. Esta tesis doctoral presenta los resultados obtenidos para definir una metodología de composición de servicios que aplica la Ingeniería de Líneas de Producto para definir dichas composiciones de servicios dinámicas y flexibles según el contexto de ejecución. Esta metodología de composición ha sido formalizada utilizando la notación definida por metamodelo SPEM 2.0.

Finalmente, para evaluar la metodología de composición de servicios se va a utilizar el marco de trabajo NIMSAD, ya que es considerado el enfoque más apropiado a utilizar en la evaluación de metodologías orientadas a objetos para el desarrollo de sistemas de información frente a otros enfoques existentes de evaluación de metodologías. Además, NIMSAD ha sido aplicado con éxito en múltiples casos dentro de la comunidad de Sistemas de Información.

La verdadera ciencia enseña, por encima de todo, a dudar y a ser ignorante.

Miguel de Unamuno

CAPITULO 4

Enfoque de la Tesis

En este capítulo se describe el enfoque que hemos seguido para desarrollar la solución propuesta en esta tesis doctoral.

4.1 Soporte a la Composición de Servicios

Un problema fundamental en el desarrollo de composiciones de servicios, tanto estáticas como dinámicas, es la carencia de técnicas y metodologías que faciliten el diseño, desarrollo y ejecución de servicios compuestos capaces de proporcionar la funcionalidad más adecuada y una calidad óptima según las condiciones del contexto de ejecución en cada momento.

El principal reto en la composición de servicios es proporcionar una solución completa, formada por metodologías y herramientas, que de soporte a la composición de servicios en todo su ciclo de vida desde la planificación, definición, implementación hasta la ejecución. Además, también es necesario facilitar la definición de composiciones de servicios que sean capaces de adaptarse al contexto de ejecución de una manera rápida y en la mayor brevedad de tiempo posible, ya que, en caso contrario, la composición dinámica no sería práctica.

En el evento de *Microsoft Tech-Ed 2007* organizado en Orlando (U.S.A.) en Junio del año 2007, el sueco Ivar Jacobson, conocido como una de las mayores autoridades en el campo de la Ingeniería del Software, predijo que "*en el futuro no va a haber más metodologías prescriptivas como lo fueron UP, procesos estilo CMMI o procesos ágiles estilo XP, SCRUM y otros. En cambio, lo que va a haber es una paleta de prácticas. Las prácticas van a venir primero y las metodologías van a ser meras colecciones de prácticas que las organizaciones van a escoger de la paleta*".

Así, el objetivo de la presente tesis doctoral no es definir otra metodología de desarrollo de software más, sino resaltar las características más importantes en el ámbito del diseño y desarrollo de las composiciones de servicios dinámicas y sensibles al contexto de ejecución. Por ello, el enfoque que hemos adoptado para especificar la metodología de composición de servicios ha consistido en definir aquellas fases, procesos, tareas, pasos, roles, productos de trabajo y responsabilidades específicas de la definición y desarrollo de composiciones de servicios que, posteriormente, pueden ser incorporadas a una metodología de desarrollo de software existente, como por ejemplo *Rational Unified Process* [RUP 2002], *Extreme Programming* [Beck 2000], o *Feature Driven Development* [Palmer y Felsing 2002]. En el capítulo 6 de este documento se incluyen dos ejemplos sobre cómo aplicar la metodología de composición de servicios propuesta en una organización que ya tiene implantada una metodología general de desarrollo de software. Estos ejemplos están enfocados en *Rational Unified Process* y *Extreme Programming*.

Esta metodología o compendio de buenas prácticas enfoca la composición de servicios desde una perspectiva completa e independiente de cualquier implementación, aplicando un enfoque *top-down*. Esto es, a partir de una visión general del sistema se van refinando cada una de las partes de dicho sistema en sucesivos pasos hasta obtener la composición final. El resultado que se obtiene es un modelo de composición que hay que implementar, utilizando un lenguaje como por ejemplo WS-BPEL, para que pueda ser interpretado por un entorno de ejecución y así se pueda ejecutar la composición de servicios.

Para diseñar y configurar la dinamicidad de una composición de servicios se han aplicado los mecanismos para la gestión de la variabilidad definidos en el enfoque de Ingeniería de Líneas de Producto (PLE). La aplicación del enfoque PLE se explica con más detalle en la sección 4.2.

Con el fin de facilitar la aplicación de la metodología, también se ha desarrollado una herramienta que proporciona soporte a los diferentes procesos y tareas definidos por la metodología propuesta. Dicha herramienta está explicada en detalle en la sección 5.2.

Una vez definida la composición de servicios hay que preparar un entorno de ejecución que sea capaz de ejecutar dicha composición y también de interpretar la variabilidad de la composición identificada en tiempo de diseño. Para ello, CEFRIEL (*ICT Center of Excellence for Research, Innovation, Education and industrial Labs partnership*) y la Universidad de Unisannio en Italia, con quienes estamos colaborando estrechamente, han desarrollado un entorno de ejecución capaz de ejecutar estas composiciones dinámicas. En la sección 5.3 se describe más detalladamente cómo se implementan estas composiciones dinámicas y también las características del entorno de ejecución desarrollado para dar soporte a la ejecución de las composiciones de servicios dinámicas y flexibles.

Finalmente, para poder validar la solución propuesta hemos definido, en conjunto con varias empresas europeas del ámbito de la automoción y las telecomunicaciones, un escenario industrial real donde se ha aplicado nuestra solución. Los resultados de dicha evaluación se recogen en el capítulo 7.

4.2 Ingeniería de Líneas de Producto Software aplicada a la Composición de Servicios

La aplicación de los mecanismos de variabilidad definidos en el enfoque de la Ingeniería de Líneas de Producto a la composición de servicios tiene como objetivo identificar y especificar la variabilidad o dinamicidad de dicha composición (en el caso de que la hubiera). De este modo, describiendo la variabilidad en tiempo de diseño y proporcionando soporte a su interpretación durante la ejecución de la composición de servicios, se facilita que la composición sea capaz de adaptarse y autoconfigurarse dinámicamente durante su ejecución, teniendo en cuenta las condiciones y las características del contexto de ejecución.

Los conceptos sobre variabilidad necesarios para definir la dinamicidad de una composición de servicios son dos: **Punto de Variación** (*Variation Point*) y **Variante** (*Variant*). La Figura 4-1 describe los atributos que definen a estos dos conceptos así como las relaciones entre los mismos.

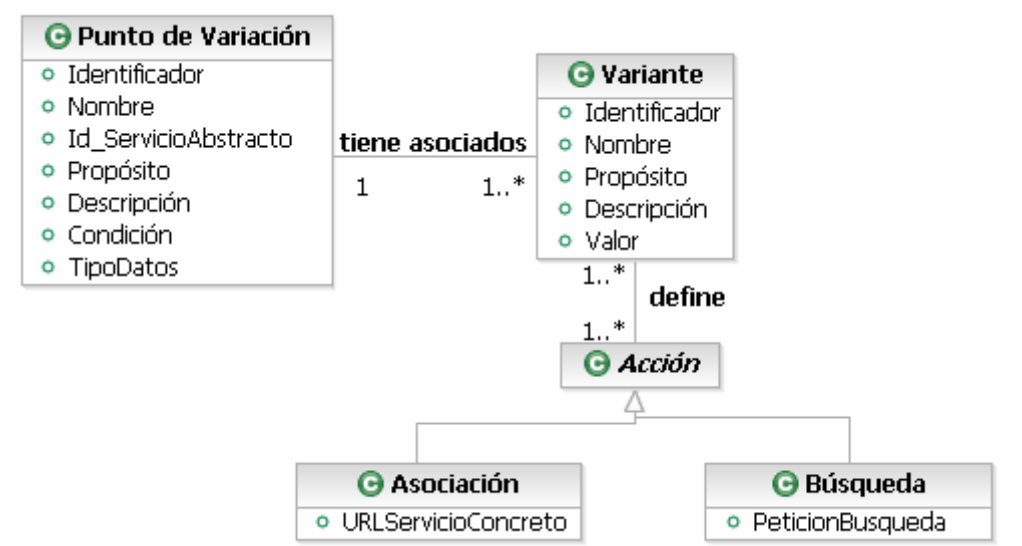


Figura 4-1: Modelado de la variabilidad

Un ***Punto de Variación*** identifica una parte del flujo de la composición que puede cambiar su comportamiento según el valor de una decisión que depende de las características del contexto de ejecución. Dicho punto de variación debe recoger la siguiente información:

- Un identificador único y un nombre que identifiquen el punto de variación dentro de la composición.
- Una referencia al identificador de la especificación o servicio abstracto a la cual está asociado dicho punto de variación.
- Un propósito que defina cuál es la funcionalidad que el punto de variación añade al sistema.
- Una descripción que proporcione la información necesaria para dar soporte al proceso de toma de decisión. También puede incluir información sobre aspectos no funcionales o deseables que afecten al punto de variación.
- Una condición que defina la decisión que debe ser evaluada para seleccionar de entre las variantes asociadas aquella que sea la más adecuada. Esta decisión puede hacer referencia a información contextual, como por ejemplo el nombre del país del usuario del servicio, o puede ser una expresión lógica.

- El tipo de datos del valor obtenido tras evaluar la condición del punto de variación, el cual puede ser, por ejemplo, un valor numérico o una cadena de caracteres.

Una ***Variante*** define una de las diferentes alternativas asociadas a un punto de variación. Aunque un punto de variación puede tener asociadas diversas variantes, sólo puede haber una variante activa en cada momento por cada punto de variación.

Por cada variante o alternativa se debe definir la siguiente información:

- Un identificador único y un nombre que identifiquen la variante dentro del punto de variación al que está asociado.
- Un propósito que defina cuál es la funcionalidad que la variante añade al sistema.
- Una descripción que proporcione la información necesaria sobre dicha alternativa.
- Un valor que indica, según la decisión a tomar, cuando será seleccionada la variante.
- Una acción o lista de acciones que se ejecutarán sólo si la variante es seleccionada. Una acción puede ser de dos tipos: la definición de un servicio concreto a asociar a la composición para satisfacer el servicio abstracto requerido; o una petición de búsqueda dinámica en tiempo de ejecución de posibles servicios concretos que satisfagan un conjunto de requisitos.

Resumiendo, a través de la definición de puntos de variación y sus correspondientes variantes o alternativas, es posible identificar qué partes de una composición podrían variar dependiendo del contexto de ejecución y, así, poder definir una o varias acciones a realizar en cada caso. Estas acciones pueden especificar el uso de un determinado servicio concreto o la búsqueda dinámica de posibles servicios concretos candidatos a satisfacer una funcionalidad requerida y otros requisitos como, por ejemplo, el consumo del servicio más barato entre todos los existentes en un momento dado.

Una vez que se ha modelado y especificado la variabilidad de la composición, hay que definir cómo se va a tratar dicha variabilidad durante la ejecución. Los detalles sobre cómo añadir dicha información de variabilidad a la composición se definen detalladamente en la sección 5.3.

El sabio no dice nunca todo lo que piensa, pero siempre piensa todo lo que dice.

Aristóteles

CAPITULO 5

Método de Diseño para Composiciones de Servicios Dinámicas

En este capítulo se presenta la metodología o compendio de buenas prácticas sobre la composición de servicios que hemos desarrollado con el propósito de dar soporte a dicha actividad, desde el análisis, el diseño, la implementación y la validación y verificación hasta su puesta en ejecución [Bastida et al. 2008a][Bastida y Nieto 2007][Bastida y Nieto 2006][Nieto et al. 2005]. Para facilitar la aplicación de esta metodología hemos desarrollado también una herramienta que sirve de apoyo a los usuarios que van a utilizarla.

En este capítulo también se incluye una breve descripción de cómo dar soporte en tiempo de ejecución a las composiciones dinámicas definidas utilizando la metodología de composición de servicios que proponemos en esta tesis.

5.1 Metodología de Composición de Servicios

Una metodología orientada a dar soporte al diseño y desarrollo de composiciones de servicios no sólo debe facilitar la definición de composiciones estáticas sino también de composiciones dinámicas y sensibles al contexto de ejecución. Por ello, la identificación y descripción de la variabilidad de una composición necesita ser cubierta durante la fase de diseño.

Dentro de este ciclo de vida que la metodología cubre se distinguen cuatro tipos de fases: Análisis, Diseño, Implementación y Pruebas. Aunque el punto de partida de esta metodología son los requisitos del sistema, no hemos incluido la Ingeniería de Requisitos como un proceso más ya que lo consideramos una disciplina tradicional y autónoma ([Robertson y Robertson 2006], [Kotonya y Sommerville 2001], [Sommerville y Sawyer 1997], [Wieringa 1996], [Davis 1993]). A continuación describimos el objetivo de cada una de estas fases de la metodología propuesta.

- La fase de **Análisis** tiene como objetivo "*el estudio del sistema actual de la empresa y la definición de la necesidades y prioridades de los usuarios para la construcción de un nuevo sistema de información*" [Whitten et al. 1996].

- La fase de **Diseño** tiene como objetivo "*evaluar las soluciones alternativas y especificar una solución detallada de tipo informático*" [Whitten et al. 1996].

- La fase de **Implementación** "*es la construcción del nuevo sistema y el paso de dicho sistema a producción (funcionamiento diario)*" [Whitten et al. 1996].

- La fase de **Pruebas** tiene como objetivo conocer en realidad los resultados del sistema. "*Todo componente del sistema debe ser individualmente probado; y todo el sistema completo debe ser también probado para asegurar que funciona correctamente y satisface los requisitos y expectativas del usuario*" [Whitten et al. 2004].

La transición entre estas fases no debe ser un proceso secuencial de un único paso, ya que el proceso tiende a ser iterativo e incremental por naturaleza y debería ser lo suficientemente ágil para adecuarse a revisiones en aquellas situaciones donde el alcance no esté completamente definido de antemano.

Para ayudar en el modelado de la composición de servicios hemos utilizado el Lenguaje Unificado de Modelado (UML) [UML 1997], que es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, además de estar respaldado por el organismo *OMG*. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

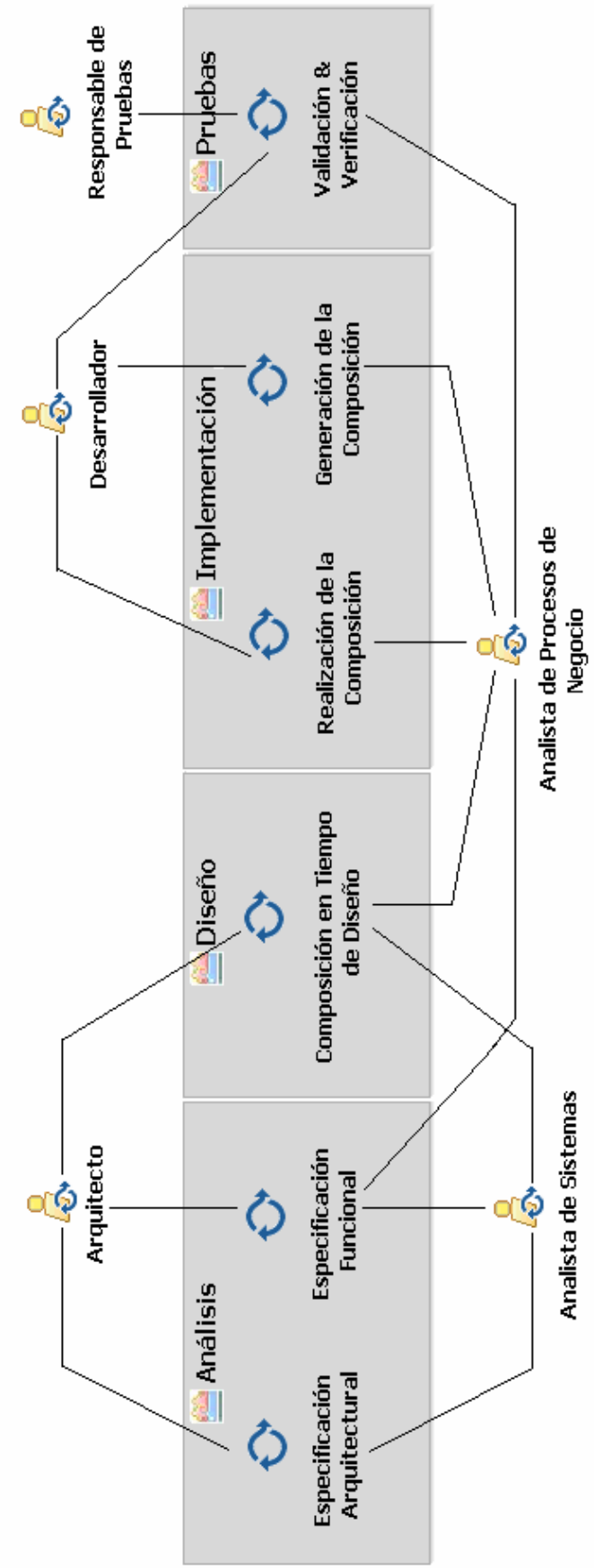


Figura 5-1: Metodología para diseñar composiciones de servicios

En la Figura 5-1 se recogen las fases, los procesos y los roles definidos dentro de la metodología de composición para diseñar tanto composiciones estáticas como dinámicas aplicando los conceptos de variabilidad definidos en el enfoque de la Ingeniería de Líneas de Producto. En total se han identificado seis tipos de procesos: Especificación Arquitectural, Especificación Funcional, Composición en Tiempo de Diseño, Gestión de la Variabilidad, Realización de la Composición y Validación & Verificación.

A continuación se explican con más detalle los diferentes procesos, roles y productos de trabajo de la metodología de composición de servicios.

5.1.1 Procesos de la Metodología de Composición

ESPECIFICACIÓN ARQUITECTURAL

PROPÓSITO
El propósito de este proceso es identificar los requisitos arquitectónicos del sistema y definir las políticas arquitectónicas que deberán tenerse en cuenta a la hora de tomar decisiones a nivel de la arquitectura del sistema.

DESCRIPCIÓN	
	Para alcanzar este objetivo, es importante recoger y mantener la información relativa a otros proyectos realizados dentro de la organización con el fin de poder analizar los resultados obtenidos en otras soluciones similares. Haciendo uso de esta experiencia se pueden definir unas políticas arquitectónicas que describan las restricciones y posibles arquitecturas candidatas para la composición que se va a realizar, facilitando de esta manera la toma de decisión sobre qué arquitectura es la más adecuada para la composición después de analizar las ventajas y desventajas de cada una de ellas. Este proceso está incluido dentro de la fase de Análisis.

A continuación se describen cada una de las tareas incluidas en el proceso de Especificación Arquitectural (Figura 5-2):

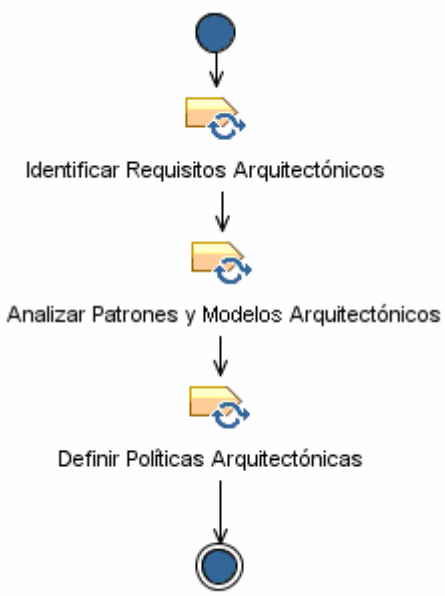


Figura 5-2: Especificación Arquitectural

Identificar Requisitos Arquitectónicos

PROPÓSITO
El propósito de esta tarea es identificar qué requisitos tienen un impacto directo en la arquitectura del sistema ya que delimitarán las decisiones arquitectónicas a tomar.

DESCRIPCIÓN	
	Esta tarea recoge y analiza aquellos requisitos del sistema que restringirán la arquitectura de la composición de servicios. Es decir, establece los primeros pasos a dar para definir la arquitectura de alto nivel del sistema. Esta tarea se basa en la experiencia obtenida en el desarrollo de sistemas o dominios de problema similares para delimitar y enfocar la arquitectura.

RELACIONES	
Roles	Obligatorio: • Arquitecto
	Recomendado: • Analista del Sistema

Productos de Trabajo	Lee: • Requisitos del Sistema
	Crea: • Requisitos Arquitectónicos

PASOS	
	**Extraer los requisitos arquitectónicos** Identificar qué requisitos del sistema impactan significativamente en la arquitectura.
	Analizar cómo impactan los requisitos arquitectónicos Explorar y refinar los requisitos arquitectónicos para ver cómo impactan en la arquitectura del sistema. Este análisis servirá para seleccionar aquella arquitectura que mejor se adecue al sistema y cumpla todos o la mayor parte de los requisitos arquitectónicos identificados.

Analizar Patrones y Modelos Arquitectónicos

PROPÓSITO
El propósito de esta tarea es, a partir de experiencias previas de la organización, identificar aquellos patrones y modelos arquitectónicos aplicables a la composición de servicios a desarrollar analizando las ventajas y desventajas de cada uno de dichos patrones y estrategias.

DESCRIPCIÓN	
	El análisis de las diferentes alternativas arquitectónicas aplicables a una solución es una tarea importante para ayudar a los arquitectos a seleccionar la arquitectura más adecuada a aplicar al problema a solucionar. Es importante destacar que el éxito de tarea depende en su mayor parte de la experiencia de la empresa y de los miembros del equipo de trabajo, ya que es a partir de dichas experiencias de donde se van a identificar las ventajas y desventajas de la puesta en práctica de los diferentes patrones y modelos arquitectónicos.

RELACIONES	
Roles	Obligatorio: • Arquitecto
	Recomendado: • Analista del Sistema
Productos de Trabajo	Lee: • Requisitos Arquitectónicos • Repositorio Organizacional de Modelos
	Crea: • Patrones y Modelos Arquitectónicos

PASOS	
	**Identificar los patrones y modelos arquitectónicos a aplicar** Teniendo en cuenta los requisitos arquitectónicos previamente identificados y el repositorio de experiencias de la organización, decidir qué patrones y modelos arquitectónicos son capaces de satisfacer dichos requisitos.
	Definir los criterios para medir la viabilidad Definir los diferentes criterios que se van a utilizar para medir la viabilidad de cada patrón y modelo arquitectónico identificado como aplicable al sistema. Es necesario especificar qué se va a medir y cómo se va a medir.
	Analizar los patrones y modelos arquitectónicos según dichos criterios A partir de los criterios definidos analizar cada patrón y modelo arquitectónico.
	Recoger los resultados obtenidos Realizar un informe con los resultados del análisis de viabilidad.

Definir Políticas Arquitectónicas

PROPÓSITO
El propósito de esta tarea es establecer los principios arquitectónicos que van a delimitar la arquitectura del sistema a desarrollar.

DESCRIPCIÓN	
	Una vez que se ha realizado el estudio de viabilidad sobre los diferentes patrones y estrategias arquitectónicas aplicables al sistema, esta tarea recoge y analiza los resultados obtenidos así como los requisitos arquitectónicos previamente identificados para ayudar a la hora de seleccionar la arquitectura más adecuada para desarrollar la composición.

RELACIONES	
Roles	Obligatorio: • Arquitecto
	Recomendado: • Analista del Sistema
Productos de Trabajo	Lee: • Requisitos Arquitectónicos • Patrones y Modelos Arquitectónicos
	Crea: • Políticas Arquitectónicas

PASOS	
	**Analizar la viabilidad de cada patrón y modelo arquitectónico** A partir del informe que recoge los resultados obtenidos tras el análisis de viabilidad, analizar cada uno de los patrones y modelos arquitectónicos definidos. Si alguno de los patrones o modelos identificados obtienen unos resultados muy bajos o diferentes a los esperados, es recomendable eliminarlo de la lista de alternativas.
	Priorizar la lista de patrones y modelos arquitectónicos Priorizar la lista de patrones y modelos arquitectónicos teniendo en cuenta cuáles son los más adecuados para aplicar al sistema.
	Definir las políticas arquitectónicas Definir el conjunto de principios, objetivos, restricciones y máximas que delimitarán las decisiones relativas a la arquitectura, diseño, implementación y despliegue de la solución.

ESPECIFICACIÓN FUNCIONAL

PROPÓSITO
El propósito de este proceso es identificar y especificar los servicios abstractos requeridos para cubrir la funcionalidad de la composición.

DESCRIPCIÓN	
	Este proceso se encarga de describir las especificaciones de los servicios que forman parte de la composición sin proporcionar ningún detalle específico sobre aspectos de implementación. Estas especificaciones de servicios se denominan abstractas ya que sólo definen la funcionalidad de cada servicio requerido y, posteriormente, serán utilizadas para encontrar servicios concretos que satisfagan dichos requisitos. En el caso de que alguna de estas especificaciones no sea cubierta por algún servicio existente, sería necesario desarrollar un servicio que la implementase. Este proceso hace uso del lenguaje UML para modelar las funcionalidades de la composición. Este proceso está incluido dentro de la fase de Análisis.

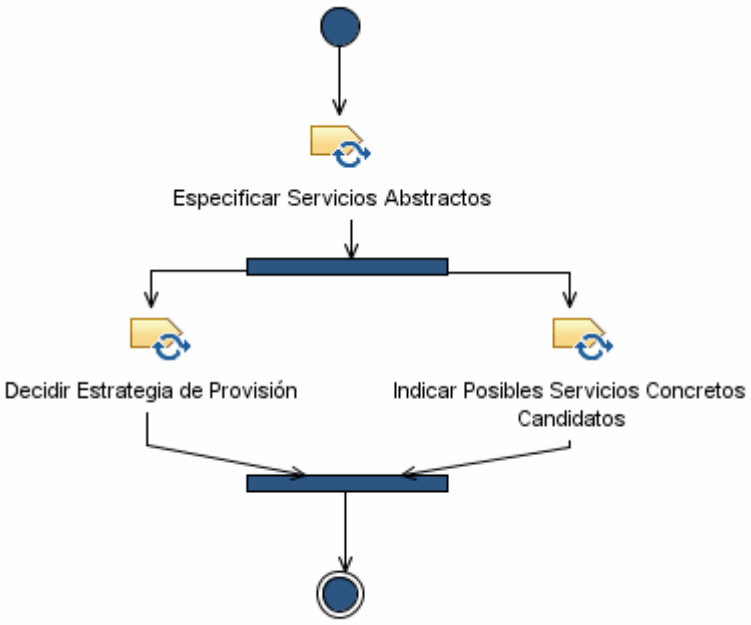


Figura 5-3: Especificación Funcional

A continuación se describen cada una de las tareas incluidas en este proceso (Figura 5-3):

Especificar Servicios Abstractos

PROPÓSITO
El propósito de esta tarea es identificar y proporcionar una descripción completa de las diferentes funcionalidades que forman la composición de servicios.

DESCRIPCIÓN	
	Esta tarea identifica las diferentes funcionalidades de la composición y las describe como servicios abstractos independientes de cualquier implementación. Por cada servicio abstracto se especifica información relevante como el propósito, una breve descripción, los requisitos no funcionales o posibles servicios concretos a utilizar.

RELACIONES	
Roles	Obligatorio: • Analista del Sistema
	Recomendado: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Requisitos del Sistema • Patrones y Modelos Arquitectónicos • Políticas Arquitectónicas
	Crea: • Modelo de Especificación Funcional

PASOS	
	**Identificar los servicios abstractos** A partir de los Requisitos del Sistema, extraer la lista de funcionalidades que componen la composición de servicios. Estas funcionalidades se modelarán a través de un diagrama UML de casos de uso, permitiendo definir los límites del sistema.

	Completar la información sobre cada servicio abstracto Es importante recoger toda la información relevante sobre cada servicio abstracto para poder buscar aquellos servicios concretos más satisfactorios y comprender el papel de cada servicio dentro de la composición. Por cada servicio abstracto se deberá especificar, como mínimo, la siguiente información: un identificador, el propósito, una breve descripción del servicio enfocándose en los aspectos funcionales de la composición, otros requisitos relevantes tales como requisitos no funcionales o restricciones, actores involucrados y dependencias con otros servicios abstractos.
	Analizar la granularidad de los servicios abstractos Analizar si alguno de los servicios abstractos puede descomponerse en sub-funcionalidades más pequeñas. Esto implicaría que dicho servicio abstracto se podría implementar a su vez como una composición. Este análisis es necesario porque se puede dar el caso de que no existan servicios concretos que cubran toda la funcionalidad definida en un servicio abstracto pero es posible que, al descomponerlo en sub-funcionalidades más pequeñas se pueda definir una composición de servicios que satisfaga el servicio abstracto completo. Esta granularidad de un servicio abstracto se representará a través de un diagrama UML de casos de uso.

Decidir Estrategia de Provisión

PROPÓSITO
El propósito de esta tarea es definir qué estrategia se va a seguir durante la realización de los servicios abstractos, teniendo en cuenta que o bien se pueden consumir servicios proporcionados por terceros, desarrollar los servicios dentro de la propia organización o bien reutilizar y adaptar los sistemas ya existentes dentro de la organización.

DESCRIPCIÓN	
	Depende del tipo de organización que esté desarrollando la composición se adoptarán diferentes estrategias de provisión. Si la organización no se dedica a desarrollar servicios o no quiere invertir esfuerzo en dichos desarrollos, probablemente su estrategia sea consumir servicios de terceros. En cambio, si la organización lo que desea es integrar sus sistemas dentro de la composición, dicha organización optará por reutilizar y adaptar dichos sistemas heredados y ofrecerlos como servicios para integrarlos en la composición. En el caso de que la organización decidiera desarrollar alguno de los servicios abstractos requeridos, sería necesario especificar tanto el lenguaje de programación como el entorno de desarrollo a utilizar.

RELACIONES	
Roles	Obligatorio: • Analista del Sistema
	Recomendado: • Arquitecto
Productos de Trabajo	Lee: • Políticas Arquitectónicas
	Modifica: • Modelo de Especificación Funcional

PASOS	
	**Consensuar los puntos de vista de todo el equipo de trabajo y la organización** Es importante a la hora de decidir la estrategia de provisión a seguir que se tengan en cuenta los diferentes puntos de vista y opiniones, no sólo a nivel de negocio y de la organización, sino también del equipo de trabajo involucrado en el diseño y desarrollo de la composición de servicios.

	Tomar la decisión final sobre la estrategia de provisión A partir de las opiniones obtenidas tanto a nivel de negocio de la organización como de las personas involucradas en el equipo de trabajo, hay que decidir la estrategia de provisión que se va a seguir.

Indicar Posibles Servicios Concretos Candidatos

PROPÓSITO
El propósito de esta tarea es definir posibles implementaciones candidatas que satisfagan uno o varios de los servicios abstractos que forman la composición. La selección final de los servicios concretos a asociar a cada servicio abstracto se realizará en el proceso de Realización de la Composición. Esta tarea es opcional.

DESCRIPCIÓN	
	Para localizar estas posibles implementaciones candidatas a utilizar, la organización hará uso de su experiencia previa en el uso y consumo de servicios tanto de terceros como a nivel interno. También es posible realizar búsquedas de servicios existentes en el mercado que satisfagan los requisitos especificados para un servicio abstracto.

RELACIONES	
Roles	Obligatorio: • Analista del Sistema
	Recomendado: • Arquitecto
Productos de Trabajo	Lee: • Patrones y Modelos Arquitectónicos • Políticas Arquitectónicas • Requisitos del Sistema
	Crea: • Lista de Servicios Disponibles para ciertos Requisitos (Opcional)

	Modifica: • Modelo de Especificación Funcional

PASOS	
	**Identificar los servicios concretos candidatos a nivel organizacional** A partir de las Políticas Arquitectónicas que recogen restricciones sobre la composición, de los Requisitos del Sistema donde el cliente puede haber solicitado utilizar ciertos servicios existentes, y de la experiencia tanto de la organización y como del equipo de trabajo, hay que identificar posibles servicios concretos candidatos a cubrir uno o varios servicios abstractos.
	Buscar servicios concretos candidatos a partir de ciertos requisitos Para ello se utilizará como criterio de búsqueda el conjunto de requisitos definido para cada servicio abstracto. Este paso es opcional por lo que en caso de que no se realice, no se creará una lista de servicios disponibles para dichos requisitos.
	Verificar los servicios concretos En ocasiones puede ocurrir que la organización tenga algún acuerdo con proveedores de servicios por lo que también habría que verificar que las condiciones previamente establecidas siguen estando vigentes.

HERRAMIENTAS	
	**Buscador de Servicios a partir de Requisitos** Este buscador de servicios es una herramienta que permite localizar en tiempo de diseño aquellos servicios concretos disponibles que satisfacen unos requisitos específicos utilizados como criterio de búsqueda.

COMPOSICIÓN EN TIEMPO DE DISEÑO

PROPÓSITO
El propósito de este proceso es definir el flujo de la composición y especificar otros requisitos relevantes de la composición como los que se refieren a la gestión de errores, restricciones de seguridad o gestión de transacciones.

DESCRIPCIÓN	
	Este proceso se encarga de identificar las diferentes funcionalidades de la composición de servicios y recogerlas en un modelo abstracto, denominado Modelo de Especificación de la Composición, donde se describe en detalle cada funcionalidad y se define tanto el flujo de trabajo como otros requisitos relevantes para la composición. Este proceso está incluido dentro de la fase de Diseño.

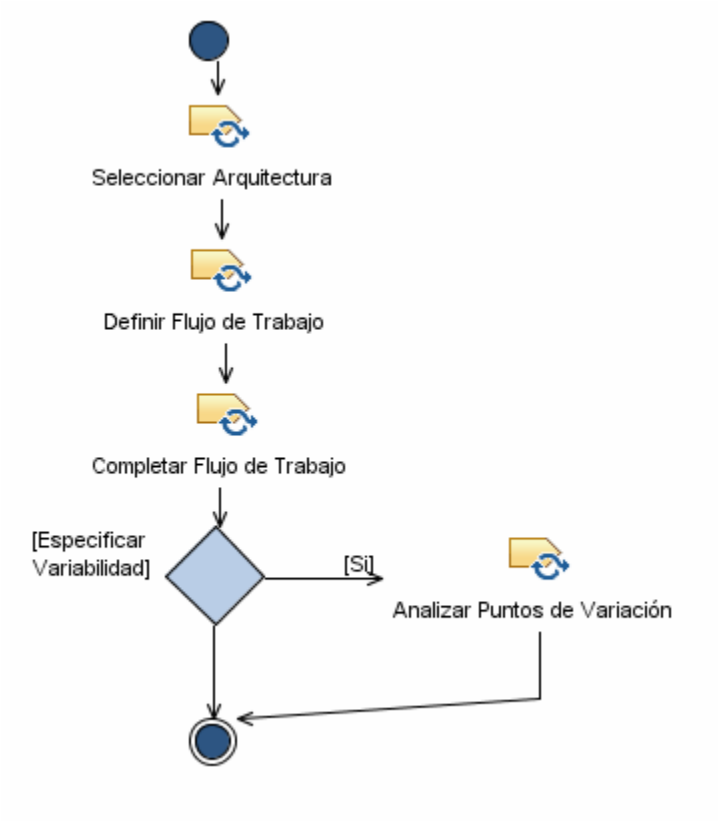


Figura 5-4: Composición en Tiempo de Diseño

A continuación se describen cada una de las tareas incluidas en el proceso de Composición en Tiempo de Diseño (Figura 5-4):

Seleccionar Arquitectura

PROPÓSITO
El propósito de esta tarea es seleccionar el modelo arquitectónico que mejor satisface las necesidades y requisitos del sistema.

DESCRIPCIÓN	
	Esta tarea identifica si los modelos arquitectónicos identificados en el documento Patrones y Modelos Arquitectónicos siguen siendo válidos, teniendo en cuenta que la composición ha sido especificada con mayor detalle y han podido surgir nuevos requisitos que condicionen la arquitectura a adoptar.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
	Recomendado: • Arquitecto
Productos de Trabajo	Lee: • Patrones y Modelos Arquitectónicos • Modelo de Especificación Funcional
	Modifica: • Políticas Arquitectónicas

PASOS	
	**Analizar los patrones y modelos arquitectónicos propuestos** Teniendo en cuenta la Especificación Funcional del sistema y las Políticas Arquitectónicas definidas, hay que extraer nuevas restricciones arquitectónicas no identificadas hasta el momento y verificar si los patrones y modelos arquitectónicos propuestos siguen siendo válidos.
	Seleccionar el modelo arquitectónico más adecuado Con la información del análisis realizado, hay que seleccionar aquel modelo arquitectónico más adecuado para aplicar a la composición.

Definir Flujo de Trabajo

PROPÓSITO
El objetivo de esta tarea es describir el enfoque de la solución a nivel de proceso de negocio, especificando el flujo de trabajo de la composición de servicios.

DESCRIPCIÓN	
	Esta tarea describe el flujo de trabajo donde se integran los diferentes servicios abstractos identificados previamente para satisfacer los requisitos de la composición de servicios.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Modelo de Especificación Funcional • Requisitos del Sistema
	Crea: • Modelo de Especificación de la Composición

PASOS	
	Identificar los grupos funcionales Hay que dividir las funcionalidades de la composición previamente identificadas en grupos funcionales basándose en los requisitos del sistema, con el fin de poder identificar nuevas partes funcionales que no se identificaron a partir de los requisitos del sistema pero que son necesarias para satisfacer ciertos requisitos.
	**Analizar las dependencias entre los grupos funcionales** El análisis de las dependencias entre los grupos funcionales ayuda a definir el orden de las funcionalidades identificadas dentro del flujo de trabajo de la composición.
	Definir el flujo de trabajo Teniendo en cuenta las dependencias entre los diferentes grupos funcionales, hay que definir el flujo de trabajo de la composición de servicios identificando, por ejemplo, qué partes pueden realizarse en paralelo o dónde van a tener lugar las iteraciones. El flujo de trabajo del sistema se modelará a través de un diagrama UML de actividad.

Completar Flujo de Trabajo

PROPÓSITO
El propósito de esta tarea es añadir a la descripción del flujo de trabajo detalles sobre requisitos no funcionales.

DESCRIPCIÓN	
	Esta tarea identifica aquellos requisitos no funcionales que afectan al flujo de trabajo de la composición y que condicionan una correcta ejecución de la composición y la garantía de satisfacer las expectativas de los usuarios.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Requisitos Arquitectónicos • Requisitos del Sistema • Políticas Arquitectónicas
	Modifica: • Modelo de Especificación de la Composición

PASOS	
	**Describir con mayor detalle el comportamiento del sistema** Es posible proporcionar más detalles sobre el comportamiento del sistema haciendo uso de un diagrama UML de secuencia que defina la ordenación temporal de los mensajes a intercambiar. Este paso es opcional.
	Identificar los requisitos no funcionales Es importante poner atención a aquellos requisitos no funcionales derivados del uso de transacciones o de aspectos de seguridad. Se puede extraer información sobre posibles requisitos no funcionales a partir de los Requisitos del Sistema, el Modelo de Especificación Funcional e incluso las Políticas Arquitectónicas.

	Analizar el impacto de dichos requisitos no funcionales Analizar el impacto de cada uno de los requisitos no funcionales identificados previamente y ver qué grupos funcionales del flujo de trabajo se verían afectados.
	Completar la descripción del flujo de trabajo Tras analizar el impacto de los requisitos no funcionales, es posible que sea necesario realizar algunas modificaciones en el flujo de trabajo para satisfacer dichos requisitos. Además, también habrá que completar la información del flujo de trabajo para incluir más detalles, como por ejemplo, restricciones o gestión de errores.

Analizar Puntos de Variación

PROPÓSITO
El propósito de esta tarea es analizar el flujo de trabajo e identificar qué partes de la composición son variables dependiendo del entorno de ejecución.

DESCRIPCIÓN	
	Esta tarea identifica en tiempo de diseño qué partes de la composición podrían variar dependiendo del entorno de ejecución y define un conjunto de requisitos que especifican dicha variabilidad para que pueda ser ejecutada durante la ejecución de la composición. Es importante que se mantenga la coherencia con el objetivo final de la composición.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
	Recomendado: • Analista del Sistema

Productos de Trabajo	Lee: • Modelo de Especificación Funcional • Modelo de Especificación de la Composición • Requisitos del Sistema
	Crea: • Modelo de Especificación de los Puntos de Variación

PASOS	
	**Identificar los puntos de variación** Identificar qué partes del flujo de trabajo de la composición podrían variar en tiempo de ejecución.
	Describir los puntos de variación Por cada punto de variación identificado, se define el propósito, la descripción y la condición que define la decisión que debe ser evaluada para seleccionar la alternativa adecuada. También hay que definir el tipo de datos obtenido de dicha decisión.
	Definir las alternativas o variantes asociadas Definir las diferentes alternativas o variantes asociadas a cada punto de variación especificando su propósito, una breve descripción y los diferentes valores que determinarán la selección de una u otra alternativa. Las acciones a realizar por cada una de estas alternativas se definirán en la fase de Implementación.

REALIZACIÓN DE LA COMPOSICIÓN

PROPÓSITO
El propósito de este proceso es asociar cada uno de los servicios abstractos definidos en la composición con un servicio concreto y así obtener la funcionalidad deseada tras ejecutar la composición final.

DESCRIPCIÓN	
	Una vez que se ha definido el modelo abstracto de la composición de servicios, es necesario seleccionar aquellos servicios concretos que implementan la composición abstracta. En esta actividad se buscan y analizan servicios concretos candidatos por cada servicio abstracto. Haciendo uso del Modelo de Especificación Funcional y del Modelo de Especificación de Composición, se seleccionan aquellos servicios concretos más adecuados y que mejor satisfagan los requisitos establecidos. Todo esto se recoge en lo que se denomina el Modelo de Realización de la Composición, que también define cómo interactúan entre sí los servicios y qué nivel de calidad de servicio (QoS) se ofrece. Este proceso está incluido dentro de la fase de Implementación.

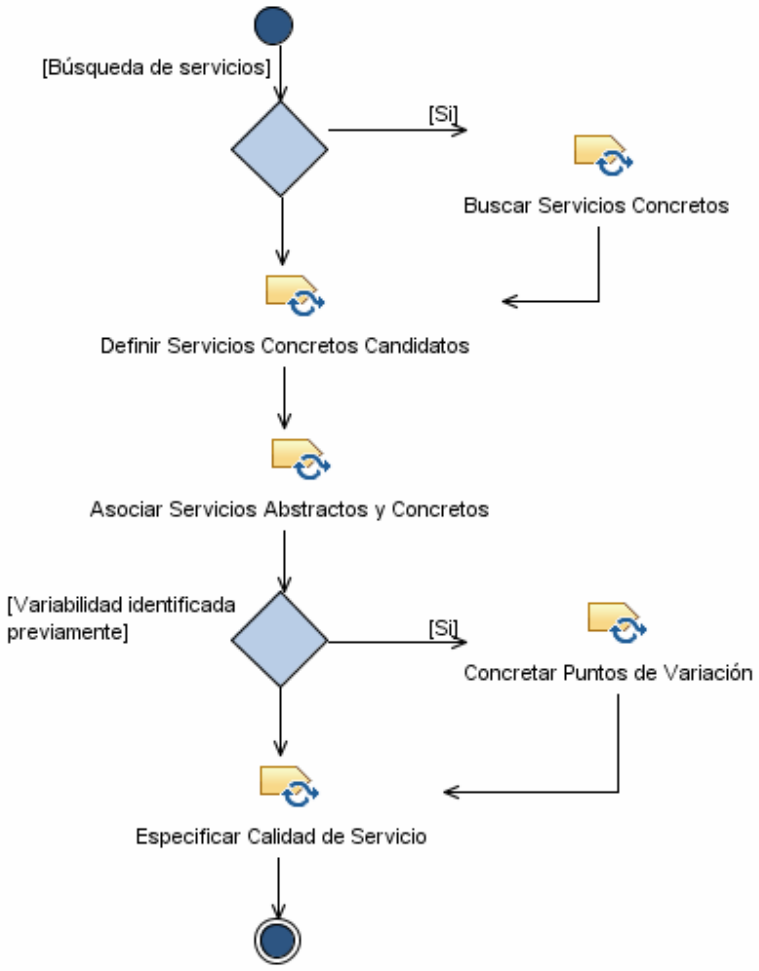


Figura 5-5: Realización de la Composición

A continuación se describen cada una de las tareas incluidas en el proceso de Realización de la Composición (Figura 5-5):

Buscar Servicios Concretos

PROPÓSITO
El propósito de esta tarea es buscar servicios concretos que son potenciales candidatos a cubrir una especificación dada, tanto a nivel funcional como a nivel de requisitos no funcionales.

DESCRIPCIÓN	
	Esta tarea de búsqueda de servicios concretos candidatos puede considerarse opcional ya que previamente a esta tarea ya se han podido realizar búsquedas, o tener de antemano unos servicios acordados y pre-seleccionados.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
	Recomendado: • Desarrollador
Productos de Trabajo	Lee: • Políticas Arquitectónicas • Modelo de Especificación de la Composición • Requisitos del Sistema
	Crea: • Lista de Servicios Disponibles para una Especificación

PASOS	
	**Definir las peticiones de búsqueda** Definir los criterios de búsqueda que se utilizarán para localizar los potenciales servicios concretos candidatos a cubrir los diferentes servicios abstractos.

	Información relevante que deberían incluir esas peticiones son, por ejemplo, una descripción de las operaciones, los parámetros de dichas operaciones, el objetivo del servicio o el nivel de calidad de servicio.
	Realizar las búsquedas de los servicios concretos candidatos En este paso es muy útil hacer uso de herramientas o motores de búsqueda que procesen esas peticiones y devuelvan una lista de potenciales servicios concretos. También es importante, si es posible, limitar la búsqueda a servicios ofrecidos por proveedores de confianza.
	Recoger los resultados de las búsquedas Estos resultados obtenidos de las diferentes búsquedas se utilizarán en las siguientes tareas.

HERRAMIENTAS	
	**Buscador de Servicios a partir de Especificaciones** Este buscador de servicios es una herramienta que permite localizar en tiempo de diseño aquellos servicios concretos disponibles que satisfacen una especificación dada como criterio de búsqueda.

Definir Servicios Concretos Candidatos

PROPÓSITO
El propósito de esta tarea es definir y priorizar los potenciales servicios concretos candidatos a asociarse con cada servicio abstracto.

DESCRIPCIÓN	
	De entre todos los potenciales servicios concretos candidatos hay que seleccionar aquellos que mejor satisfacen los requisitos de los diferentes servicios abstractos y priorizar el orden de uso de dichos servicios concretos. También hay que tener en cuenta que el uso de alguno de dichos servicios concretos implique ciertas restricciones o que no se satisfagan todos los requisitos establecidos.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
	Recomendado: • Desarrollador
Productos de Trabajo	Lee: • Lista de Servicios Disponibles para ciertos Requisitos (Opcional) • Lista de Servicios Disponibles para una Especificación (Opcional) • Modelo de Especificación de la Composición • Modelo de Especificación Funcional • Políticas Arquitectónicas
	Crea: • Lista de Servicios Concretos Candidatos

PASOS	
	Listar todos los servicios concretos candidatos Crear una lista completa de todos los servicios concretos candidatos para cada servicio abstracto. Esta lista incluirá: • Servicios concretos descubiertos en la tarea previa. • Servicios concretos desarrollados por la organización. • Servicios concretos utilizados en otros proyectos. • Servicios concretos pre-seleccionados durante la fase de Análisis.
	**Analizar el listado de los servicios concretos candidatos** Si no se encuentra un servicio concreto para cubrir un servicio abstracto, habría dos posibles opciones: desarrollar un servicio concreto que cubra dicha funcionalidad, o dividir el servicio abstracto en sub-funcionalidades. Esta segunda opción implicaría actualizar el Modelo de Especificación Funcional y el Modelo de Especificación de la Composición. Sería conveniente volver a realizar de nuevo los procesos de Especificación Funcional y Composición en Tiempo de Diseño.

	Definir las medidas de evaluación Definir aquellas medidas o criterios para evaluar los diferentes servicios concretos candidatos, no sólo a nivel funcional sino también frente a otros requisitos como, por ejemplo, el coste del servicio, el proveedor u otros aspectos adicionales del servicio. Estas medidas ayudarán también a definir qué requisitos son más críticos y prioritarios.
	Evaluar los servicios concretos candidatos Evaluar cada servicio concreto candidato según las medidas definidas previamente y a partir de dichas evaluaciones, se puntuará cada servicio concreto. Un ejemplo de método a aplicar para realizar estas puntuaciones podría ser el utilizado por el enfoque SELF-SERV (en la sección 6.1.2 se describe y aplica este método).
	Descartar los servicios concretos candidatos con valoración baja Tras evaluar los diferentes servicios concretos candidatos, hay que descartar aquellos que hayan obtenido una puntuación demasiado baja.
	Priorizar los servicios concretos candidatos restantes Priorizar los diferentes servicios concretos candidatos a partir de la puntuación obtenida tras la evaluación. El primer servicio concreto en cada lista asociada a cada servicio abstracto será la mejor alternativa.

Asociar Servicios Abstractos y Concretos

PROPÓSITO
El propósito de esta tarea es seleccionar los servicios concretos a asociar a cada servicio abstracto y garantizar la integridad de la composición.

DESCRIPCIÓN	
	Después de definir la lista de potenciales servicios concretos candidatos a cubrir los diferentes servicios abstractos, hay que seleccionar aquel servicio concreto más adecuado para cada servicio abstracto y definir sus

	operaciones y sus parámetros más en detalle. Es interesante mantener ese listado de servicios concretos asociados a cada servicio abstracto para poder usarlo posteriormente en tiempo de ejecución para seleccionar otro servicio concreto cuando el que está asociado por defecto no funciona o no está disponible.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Lista de Servicios Concretos Candidatos • Modelo de Especificación de la Composición • Modelo de Especificación Funcional • Políticas Arquitectónicas
	Crea: • Modelo de Realización de la Composición

PASOS	
	**Seleccionar los servicios concretos candidatos por cada servicio abstracto** Se deberán tener en cuenta las puntuaciones obtenidas tras evaluar los servicios concretos candidatos y otros aspectos relevantes, como el coste del servicio o el proveedor que ofrece el servicio. También es posible asociar una lista de servicios concretos candidatos al servicio abstracto para usarla, por ejemplo, en el caso de que surgiera algún problema con el servicio concreto asociado por defecto.
	Detallar las operaciones y parámetros del servicio concreto Detallar qué operaciones van a ser consumidas de cada servicio concreto. También es necesario especificar los parámetros de entrada y de salida para establecer las relaciones entre los diferentes servicios integrados en el flujo de trabajo.

	Verificar la integridad de la composición Hay que tener en cuenta los servicios concretos que se van a consumir, ya que su uso puede implicar restricciones al sistema.

Concretar Puntos de Variación

PROPÓSITO
El propósito de esta tarea es concretar la información relativa a las acciones asociadas a los Puntos de Variación de la composición.

DESCRIPCIÓN	
	Las acciones asociadas a las diferentes alternativas de cada Punto de Variación pueden consistir en, o bien asociar un servicio concreto específico que se ejecutará cuando se cumplan las condiciones de la variante; o bien especificar que en tiempo de ejecución se localice aquel servicio concreto que mejor satisface la funcionalidad requerida así como un conjunto de requisitos dependientes del contexto.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Lista de Servicios Concretos Candidatos • Modelo de Especificación de la Composición • Modelo de Especificación de los Puntos de Variación
	Modifica: • Modelo de Realización de la Composición

PASOS	
	**Identificar las acciones a realizar** Identificar qué tipo de acciones habría que realizar por cada variante especificada en cada Punto de Variación.

	Analizar la lista de servicios concretos candidatos Analizar cuál de los servicios concretos candidatos a cubrir los diferentes servicios abstractos identificados como Puntos de Variación satisface mejor las condiciones de cada una de las alternativas asociadas a los Puntos de Variación.
	Definir las acciones asociadas a cada Puntos de Variación Las acciones a realizar pueden ser la asociación de un servicio concreto o una petición de búsqueda en tiempo de ejecución acorde a un conjunto de requisitos.

Especificar Calidad de Servicio

PROPÓSITO
El propósito de esta tarea es especificar la calidad de servicio que se garantiza a los diferentes usuarios que vayan a consumir la composición de servicios.

DESCRIPCIÓN	
	Una vez que se han seleccionado los servicios concretos a utilizar, hay que analizar la calidad de servicio que va a ofrecer la composición. Esta calidad final dependerá de la calidad proporcionada por cada servicio concreto integrado en el flujo de trabajo de la composición.

RELACIONES	
Roles	Obligatorio: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Lista de Servicios Concretos Candidatos • Modelo de Especificación de la Composición • Modelo de Especificación Funcional
	Modifica: • Modelo de Realización de la Composición

PASOS	
	**Definir los aspectos de calidad de servicio a satisfacer** Definir los aspectos de calidad de servicios a satisfacer e indicar cómo se van a medir para calcular la calidad final de la composición.
	Analizar el impacto de los servicios concretos en la calidad de servicio Es necesario analizar cómo impacta cada uno de los servicios concretos integrados en la composición en la calidad de servicio final.
	Especificar la calidad de servicio final de la composición Después de analizar los diferentes servicios concretos de la composición, hay que especificar cuál es la calidad de servicio final proporcionada por la composición.

GENERACIÓN DE LA COMPOSICIÓN EJECUTABLE

PROPÓSITO
El propósito de este proceso es desarrollar el modelo ejecutable de la composición.

DESCRIPCIÓN	
	Para desarrollar el modelo ejecutable de la composición, hay que definir previamente en qué lenguaje se va a realizar, como por ejemplo WS-BPEL. Este proceso está incluido dentro de la fase de Implementación.

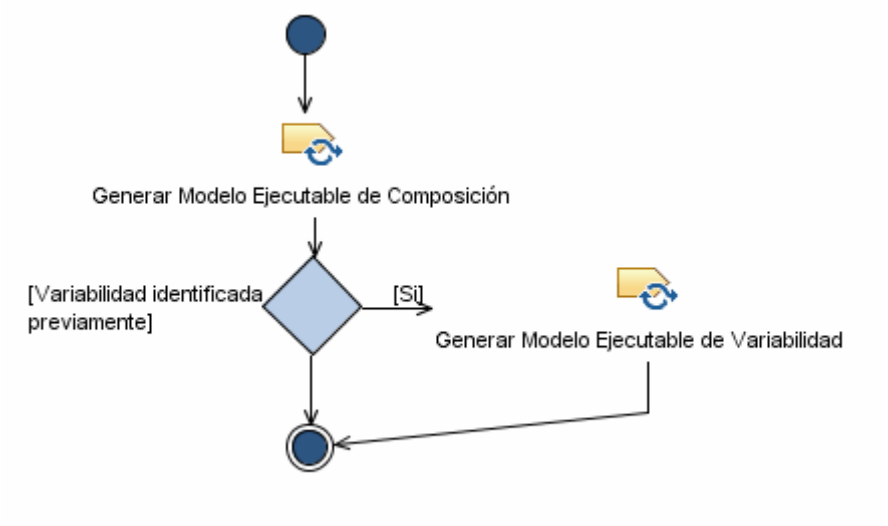


Figura 5-6: Generación de la Composición Ejecutable

A continuación se describen cada una de las tareas incluidas en el proceso de Generación de la Composición Ejecutable (Figura 5-6):

Generar Modelo Ejecutable de Composición

PROPÓSITO
El propósito de esta tarea es generar un modelo ejecutable de la composición definido en un lenguaje concreto y listo para ser ejecutado por un motor de ejecución.

DESCRIPCIÓN	
	Una vez que el diseño de la composición se ha terminado y también se han definido los diferentes servicios concretos a consumir, hay que transformar dicha composición en un modelo ejecutable. Las Políticas Arquitectónicas y/o la experiencia de la organización pueden influenciar en el lenguaje de ejecución que se elija para implementar la composición.

RELACIONES	
Roles	Obligatorio: • Desarrollador
	Recomendado: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Modelo de Especificación Funcional • Modelo de Especificación de la Composición • Modelo de Realización de la Composición • Políticas Arquitectónicas • Lista de Servicios Concretos Candidatos
	Crea: • Modelo Ejecutable de la Composición

PASOS	
	**Desarrollar el modelo ejecutable de la composición** Definir la parte funcional de la composición.

	Añadir requisitos y aspectos no funcionales al modelo ejecutable Completar el modelo ejecutable de la composición con aspectos no funcionales como, por ejemplo, los aspectos relacionados con la seguridad técnica, gestión de errores o transacciones.

Generar Modelo Ejecutable de Variabilidad

PROPÓSITO
El propósito de esta tarea es proporcionar un modelo ejecutable de la variabilidad especificada en la composición para introducir esa dinamicidad definida a través de los puntos de variación.

DESCRIPCIÓN	
	En el caso de que se hayan identificado y especificado algún punto de variación en la composición, es necesario generar un modelo ejecutable de dicha variabilidad que estará asociado al Modelo Ejecutable de la Composición. El motor de ejecución tendrá que ser capaz de interpretar este Modelo Ejecutable de Variabilidad para añadir esa dinamicidad o variabilidad a la composición durante su ejecución. Este Modelo Ejecutable de Variabilidad puede ser definido utilizando la notación de las reglas ECA (Evento-Condición-Acción).

RELACIONES	
Roles	Obligatorio: • Desarrollador
	Recomendado: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Modelo de Especificación de los Puntos de Variación • Modelo de Realización de la Composición • Políticas Arquitectónicas • Lista de Servicios Concretos Candidatos

	Crea: • Modelo Ejecutable de la Variabilidad
	Modifica: • Modelo Ejecutable de la Composición

PASOS	
	**Desarrollar el modelo ejecutable de los Puntos de Variación** Definir el modelo ejecutable de la variabilidad de la composición a partir de los puntos de variación identificados en el Modelo Especificación de Puntos de Variación.
	Adaptar el modelo ejecutable de la composición Si se ha identificado variabilidad en la composición, hay que adaptar el Modelo Ejecutable de la Composición de tal manera que aquellas partes del flujo de la composición identificados como puntos de variación hagan referencia al Modelo Ejecutable de la Variabilidad para que pueda ejecutarse dicha variabilidad durante la ejecución de la composición.

VALIDACIÓN Y VERIFICACIÓN

PROPÓSITO
El propósito de este proceso es validar la composición con el fin de asegurar que los servicios concretos seleccionados proporcionan la funcionalidad y la calidad apropiadas.

DESCRIPCIÓN	
	Esta actividad se encarga de validar y verificar la composición realizando diversas pruebas durante la ejecución de la composición de servicios obtenida. Se pueden aplicar las técnicas tradicionales de validación y verificación de software como las pruebas unitarias y las pruebas de integración. Este proceso está incluido dentro de la fase de Pruebas.

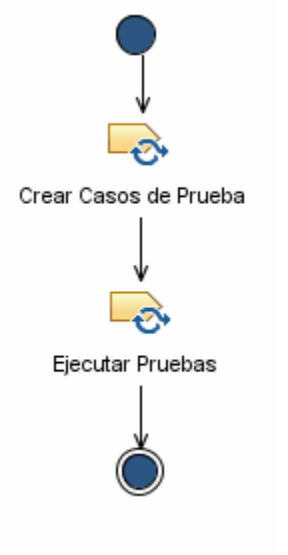


Figura 5-7: Validación y Verificación

A continuación se describen cada una de las tareas incluidas en el proceso de Validación y Verificación (Figura 5-7):

Crear Casos de Prueba

PROPÓSITO
El propósito de esta tarea es especificar los casos de prueba a aplicar para demostrar que la composición de servicios satisface los requisitos definidos.

DESCRIPCIÓN	
	Estos casos de prueba pueden constar tanto de pruebas de caja blanca como de pruebas de caja negra.

RELACIONES	
Roles	Obligatorio: • Responsable de pruebas
	Recomendado: • Analista de Procesos de Negocio • Desarrollador

Productos de Trabajo	Lee: • Requisitos del Sistema • Requisitos Arquitectónicos • Modelo Ejecutable de la Composición • Modelo Ejecutable de la Variabilidad
	Crea: • Casos de Prueba • Guiones de Prueba

PASOS	
	**Identificar los casos de prueba** Identificar y describir un conjunto de Casos de Prueba que serán utilizados para verificar y validar el sistema.
	Definir los casos de prueba Definir la secuencia de pasos necesarios para realizar las diferentes pruebas enfocándose en cada Caso de Prueba.
	Especificar los datos concretos para cada caso de prueba Completar cada caso de prueba añadiendo los valores de datos concretos que se utilizarán como Guiones de Prueba para validar y verificar los resultados obtenidos tras su ejecución.

Ejecutar Pruebas

PROPÓSITO
El propósito de esta tarea es verificar y validar cómo satisface la composición de servicios desarrollada los requisitos establecidos.

DESCRIPCIÓN	
	La ejecución de los Casos de Prueba y los Guiones de Prueba definidos en la tarea anterior permitirán verificar y validar la composición de servicios. Es importante realizar estas pruebas en cada iteración del sistema.

RELACIONES	
Roles	Obligatorio: • Responsable de pruebas
	Recomendado: • Analista de Procesos de Negocio
Productos de Trabajo	Lee: • Casos de Prueba • Guiones de Prueba • Modelo Ejecutable de la Composición • Modelo Ejecutable de la Variabilidad
	Crea: • Registro de Pruebas

PASOS	
	**Ejecutar las pruebas por cada caso de prueba** Ejecutar las pruebas siguiendo los Guiones de Prueba definidos. Los resultados obtenidos de dichas pruebas se recogerán en un Registro.
	Analizar los resultados de la prueba Analizar los resultados recogidos en el Registro de Pruebas con el fin de identificar los posibles problemas.
	Comunicar los resultados al equipo de trabajo Resumir y comunicar los resultados de las pruebas realizadas al sistema al resto del equipo de trabajo con el fin de tomar las acciones necesarias si procediese.

5.1.2 Roles de la Metodología de Composición

ARQUITECTO

PROPÓSITO
Este rol es el responsable de definir y establecer la estructura total de la arquitectura de la solución, lo cual incluye especificar y tomar aquellas decisiones técnicas que delimitarán el diseño y la implementación del sistema final.

DESCRIPCIÓN	
	Este rol coordina el diseño técnico del sistema y tiene la responsabilidad de tomar las decisiones técnicas relacionadas con la arquitectura del sistema. Para ello, debe definir los patrones y modelos arquitecturales que servirán como guía para el diseño, implementación y despliegue de la composición de servicios. La persona que realice este rol deberá estar involucrada desde el comienzo del proyecto hasta el final del mismo.

HABILIDADES
Un Arquitecto debe tener el siguiente conocimiento así como las siguientes habilidades y capacidades: • Capacidad para tener una visión general de la solución. • Amplia experiencia para comprender rápidamente los principales aspectos y realizar decisiones a pesar de no tener una información completa sobre el sistema. • Conocimiento sobre diferentes modelos y patrones arquitectónicos existentes.

RELACIONES	
Realiza	Rol obligatorio en: • Identificar Requisitos Arquitectónicos • Analizar Patrones y Modelos Arquitectónicos • Definir Políticas Arquitectónicas
	Rol adicional en: • Decidir Estrategia de Provisión • Indicar Posibles Servicios Candidatos • Seleccionar Arquitectura

ANALISTA DEL SISTEMA

PROPÓSITO
La persona que realiza este rol es la encargada de capturar, priorizar y coordinar los requisitos del cliente y de los usuarios finales con el fin de comprender el problema a solucionar y delimitar la funcionalidad del sistema.

DESCRIPCIÓN	
	Este rol está principalmente ligado con aquellos procesos relacionados con los requisitos del cliente y/o usuario final. El rol de Analista del Sistema está principalmente involucrado al comienzo del desarrollo de la composición de servicios ya que es el encargado de recoger, analizar y coordinar aquellos requisitos que la composición debe cumplir y satisfacer. También es el encargado de delimitar la funcionalidad del sistema estableciendo qué actores y casos de uso existen y como interactúan.

HABILIDADES
Un Analista del Sistema debe tener el siguiente conocimiento así como las siguientes habilidades y capacidades: • Experiencia en identificar y comprender problemas y oportunidades. • Capacidad de expresar con claridad las necesidades asociadas al problema a resolver o a la oportunidad identificada. • Buena capacidad de comunicación, tanto verbal como escrita. • Conocimiento del dominio de negocio y de la tecnología o, en su defecto, capacidad para entender rápidamente toda esa información.

RELACIONES	
Realiza	Rol obligatorio en:: • Especificar Servicios Abstractos • Decidir Estrategia de Provisión • Indicar Posibles Servicios Candidatos
	Rol adicional en: • Identificar Requisitos Arquitectónicos • Analizar Patrones y Modelos Arquitectónicos • Definir Políticas Arquitectónicas • Analizar Puntos de Variación

ANALISTA DE PROCESOS DE NEGOCIO

PROPÓSITO
La persona que realiza este rol es la encargada de definir el flujo de trabajo de las diferentes funcionalidades del sistema y de detallar la especificación del sistema.

DESCRIPCIÓN	
	Este rol está principalmente involucrado en el diseño del flujo de trabajo del sistema y en la especificación de aspectos no funcionales tales como seguridad, transacciones o gestión de errores. El rol de Analista de Procesos de Negocio también es el encargado de seleccionar los servicios concretos más adecuados y de definir la variabilidad del sistema.

HABILIDADES
Un Analista de Procesos de Negocio debe tener ciertas habilidades y capacidades: • Capacidad de liderazgo y de gestión de equipos. • Experiencia en las diferentes fases del ciclo de vida del desarrollo de software para dar soporte y guiar al resto de equipo. • Capacidad para resolver posibles conflictos y/o problemas. • Facilidad para la comunicación y negociación.

RELACIONES	
Realiza	Rol obligatorio en: • Seleccionar Arquitectura • Definir Flujo de Trabajo • Completar Flujo de Trabajo • Analizar Puntos de Variación • Buscar Servicios Concretos • Definir Servicios Concretos Candidatos • Asociar Servicios Concretos y Abstractos • Concretar Puntos de Variación • Especificar Calidad de Servicio

	Rol adicional en: • Especificar Servicios Abstractos • Generar Modelo Ejecutable de Composición • Generar Modelo Ejecutable de Variabilidad • Crear Casos de prueba • Ejecutar Pruebas

DESARROLLADOR

PROPÓSITO
Este rol es el encargado de implementar e integrar los diferentes componentes que forman parte del sistema final.

DESCRIPCIÓN	
	Este rol es el responsable de implementar la composición de servicios definiendo el modelo ejecutable final que será ejecutado por un motor de ejecución de composiciones. En el caso de que se vayan a realizar desarrollos, el Desarrollador también es el encargado de desarrollar dichas implementaciones de servicios que, posteriormente, se integrarán en la composición final. El rol de Desarrollador también está involucrado en la fase de pruebas para dar soporte durante las pruebas al sistema.

HABILIDADES
Un Desarrollador debe tener el siguiente conocimiento así como las siguientes habilidades y capacidades: • Comprender la arquitectura del sistema. • Experiencia en determinadas plataformas de desarrollo, lenguajes de programación y/o áreas de negocio. • Dar soporte en la definición de pruebas para verificar y validar el comportamiento requerido de los componentes técnicos.

RELACIONES	
Realiza	Rol obligatorio en: • Generar Modelo Ejecutable de Composición • Generar Modelo Ejecutable de Variabilidad
	Rol adicional en: • Buscar Servicios Concretos • Definir Servicios Concretos Candidatos • Crear Casos de prueba • Ejecutar Pruebas

RESPONSABLE DE PRUEBAS

PROPÓSITO
Este rol es el responsable de las actividades básicas de ejecución de pruebas sobre el sistema y también del registro y análisis de los resultados de dichas pruebas.

DESCRIPCIÓN	
	Este rol es el encargado de las siguientes tareas: • Definir las pruebas que se van a realizar. • Identificar la implementación más apropiada para cada prueba. • Implementar pruebas individuales. • Ejecutar las pruebas. • Registrar los resultados así como verificar que las pruebas han sido realizadas. • Analizar y definir acciones para solucionar los errores de ejecución. • Comunicar los resultados al equipo de trabajo, en especial al Desarrollador.

HABILIDADES
Un Responsable de Pruebas debe tener el siguiente conocimiento así como las siguientes habilidades y capacidades: • Conocimiento de enfoques y técnicas de testing.

• Capacidad de realizar diagnósticos y de resolver problemas. • Sería deseable que tuviera conocimientos sobre el sistema a probar.

RELACIONES	
Realiza	Rol obligatorio en: • Crear Casos de prueba • Ejecutar Pruebas

5.1.3 Productos de Trabajo de la Metodología de Composición

REQUISITOS DEL SISTEMA

PROPÓSITO
El propósito de los Requisitos del Sistema es capturar los requisitos que debe satisfacer y cumplir el sistema no sólo a nivel de funcionalidad, sino también las interacciones con los usuarios finales, las entidades del sistema, las interfaces con otros sistemas y los requisitos no funcionales.

DESCRIPCIÓN	
	Este producto de trabajo es de tipo entregable y, además de definir los requisitos obtenidos de los clientes y usuarios finales, también puede incluir información sobre el modelo de negocio y posibles escenarios a cubrir por el sistema a desarrollar. Este producto de trabajo es de origen externo, es decir, no se ha incluido un proceso en la metodología que se encargue de capturar y definir estos requisitos ya que se considera la Gestión de Requisitos como un proceso bien definido y estándar dentro de cualquier desarrollo de software.

REQUISITOS ARQUITECTÓNICOS

PROPÓSITO
El propósito de los Requisitos Arquitectónicos es identificar a partir de los Requisitos del Sistema aquellos requisitos que definirán la arquitectura del sistema.

DESCRIPCIÓN	
	Este producto de trabajo de tipo entregable identifica y analiza aquellos requisitos del sistema que añaden restricciones a la arquitectura de la composición de servicios a desarrollar. Los aspectos relativos al dominio de aplicación y a la naturaleza de la composición son también aspectos importantes que pueden delimitar la arquitectura del sistema.

REPOSITORIO ORGANIZACIONAL DE MODELOS

PROPÓSITO
El propósito de este Repositorio Organizacional de Modelos es facilitar el análisis e implementación arquitectónica y fomentar la flexibilidad y el rendimiento de las soluciones.

DESCRIPCIÓN	
	Este producto de trabajo de tipo artefacto proporciona un repositorio que recoge toda la experiencia de la organización en torno a los modelos arquitectónicos incluyendo información sobre los beneficios y desventajas de usar cada uno de dichos modelos. Este producto de trabajo es de origen externo.

PATRONES Y MODELOS ARQUITECTÓNICOS

PROPÓSITO
El propósito de los Patrones y Modelos Arquitectónicos es ayudar a elegir la mejor arquitectura para la composición de servicios.

DESCRIPCIÓN	
	La lista de posibles patrones y modelos arquitectónicos es de tipo artefacto y sirve como base para definir el esqueleto de la composición de servicios. Además, también puede incluir información adicional sobre cómo y cuándo utilizar cada patrón y modelo, un análisis de viabilidad, o ejemplos de aplicación.

POLÍTICAS ARQUITECTÓNICAS

PROPÓSITO
El propósito de las Políticas Arquitectónicas es definir un conjunto de principios, objetivos, restricciones y máximas que delimitarán las decisiones relativas a la arquitectura, diseño, implementación y despliegue de la solución.

DESCRIPCIÓN	
	Las Políticas Arquitectónicas son instrucciones prácticas para llevar a cabo principios arquitectónicas definidos a nivel de la organización. Estas políticas también pueden especificar cuáles serían las mejores alternativas para desarrollar la composición de servicios, así como una guía de cómo podría evolucionar la arquitectura para facilitar la adaptación de la composición a los cambios en el contexto de ejecución.

MODELO DE ESPECIFICACIÓN FUNCIONAL

PROPÓSITO
El propósito de este producto de trabajo es especificar las diferentes funcionalidades que compondrán la composición final.

DESCRIPCIÓN	
	El Modelo de Especificación Funcional es de tipo entregable y describe cada funcionalidad de la composición como un servicio abstracto. Por cada servicio abstracto se define su propósito, una breve descripción y un conjunto de requisitos no funcionales a cubrir.

GUÍAS	
	Este producto de trabajo tiene asociada una plantilla recogida en el Anexo 1.

LISTA DE SERVICIOS DISPONIBLES PARA CIERTOS REQUISITOS

PROPÓSITO
El propósito de este producto de trabajo es proporcionar una lista con las diferentes alternativas para cubrir una funcionalidad a partir de un conjunto de requisitos.

DESCRIPCIÓN	
	Este producto de trabajo de tipo artefacto ofrece una lista de servicios concretos candidatos a cubrir un conjunto de requisitos dados. Por cada servicio se incluye como información su especificación e implementación que será utilizada para analizar cada servicio candidato y seleccionar el más adecuado. Esta lista es un producto de trabajo opcional.

MODELO DE ESPECIFICACIÓN DE LA COMPOSICIÓN

PROPÓSITO
El propósito de este producto de trabajo es proporcionar una representación abstracta de la composición sin tener en cuenta aspectos de la implementación.

DESCRIPCIÓN	
	El Modelo de Especificación de la Composición es de tipo entregable y define el comportamiento de la composición, cómo se combinan las diferentes funcionalidades a través del flujo de trabajo y otros aspectos relacionados con requisitos no funcionales.

GUÍAS	
	Este producto de trabajo tiene asociada una plantilla recogida en el Anexo 1.

MODELO DE ESPECIFICACIÓN DE LOS PUNTOS DE VARIACIÓN

PROPÓSITO
El propósito de este producto de trabajo es especificar los puntos de variación de la composición para facilitar su dinamicidad y flexibilidad durante su ejecución.

DESCRIPCIÓN	
	El Modelo de Especificación de los Puntos de Variación es de tipo entregable e identifica qué partes del flujo de trabajo de la composición pueden variar durante la ejecución de la composición. Esta variabilidad de la composición puede deberse a que dichas partes del flujo de trabajo; o bien dependen de información recogida del contexto de ejecución; o bien se desea ofrecer una composición más dinámica que se adapte a cada entorno de ejecución satisfaciendo un conjunto de requisitos.

GUÍAS	
	Este producto de trabajo tiene asociada una plantilla recogida en el Anexo 1.

LISTA DE SERVICIOS DISPONIBLES PARA UNA ESPECIFICACIÓN

PROPÓSITO
El propósito de este producto de trabajo es proporcionar una lista de servicios concretos candidatos a cubrir la implementación de cierta especificación de servicio.

DESCRIPCIÓN	
	Este producto de trabajo de tipo artefacto ofrece una lista de servicios concretos candidatos a cubrir una especificación dada. Por cada servicio se incluye como información su especificación e implementación que será utilizada para analizar cada servicio candidato y seleccionar el más adecuado. Esta lista es un producto de trabajo opcional.

LISTA DE SERVICIOS CONCRETOS CANDIDATOS

PROPÓSITO
El propósito de este producto de trabajo es proporcionar un listado donde por cada servicio abstracto se define una lista de servicios concretos candidatos.

DESCRIPCIÓN	
	Por cada servicio concreto candidato se incluye como información su propósito, una descripción, el proveedor del servicio, las operaciones proporcionadas, los parámetros de entrada y salida, las restricciones asociadas y otros datos como los costes de uso o la calidad de servicio.

MODELO DE REALIZACIÓN DE LA COMPOSICIÓN

PROPÓSITO
El propósito de este producto de trabajo es proporcionar una visión de la composición a nivel de implementación.

DESCRIPCIÓN	
	El Modelo de Realización de la Composición es de tipo entregable y define los diferentes servicios concretos asociados a cada servicio abstracto de la composición. En el caso de que previamente se haya identificado y especificado variabilidad en la composición, este modelo también definirá las diferentes acciones a realizar asociadas las diferentes variantes definidas por cada punto de variación descrito en el Modelo de Especificación de Puntos de Variación.

GUÍAS	
	Este producto de trabajo tiene asociada una plantilla recogida en el Anexo 1.

MODELO EJECUTABLE DE LA COMPOSICIÓN

PROPÓSITO
El propósito de este producto de trabajo es proporcionar un modelo ejecutable de la composición que será utilizado por un motor de ejecución.

DESCRIPCIÓN	
	El Modelo Ejecutable de la Composición es de tipo entregable y define un guión listo para ser ejecutado por un motor de ejecución que sea capaz de entender el lenguaje de implementación utilizado. Este producto de trabajo se define a partir de los Modelos de Especificación de la Composición y de Realización de la Composición.

MODELO EJECUTABLE DE LA VARIABILIDAD

PROPÓSITO
El propósito de este producto de trabajo es proporcionar un modelo ejecutable de la variabilidad especificado en la composición que será utilizado junto al Modelo Ejecutable de la Composición para introducir esa dinamicidad definida a través de los puntos de variación.

DESCRIPCIÓN	
	El Modelo Ejecutable de la Variabilidad es de tipo entregable y define un modelo listo para ser ejecutado que recoge la variabilidad asociada a la composición. Este producto de trabajo se define a partir de los Modelos de Especificación de la Variabilidad y de Realización de la Composición.

CASOS DE PRUEBA

PROPÓSITO
El propósito de este producto de trabajo es proporcionar un modo de capturar los parámetros, condiciones y resultados esperados de la composición de modo que si se alcanzan los resultados esperados se verifica que el sistema cumple los requisitos.

DESCRIPCIÓN	
	Este producto es de tipo artefacto y especifica las condiciones que son necesarias validar para verificar ciertos aspectos de un sistema bajo prueba.

GUIONES DE PRUEBA

PROPÓSITO
El propósito de este producto de trabajo es definir un guión sobre qué pasos hay que seguir para realizar las pruebas necesarias para verificar y validar el sistema.

DESCRIPCIÓN	
	Los Guiones de Pruebas son de tipo artefacto y pueden tener ser un conjunto de instrucciones textuales que hay que ejecutar manualmente, o un conjunto de instrucciones comprensibles por un ordenador de tal manera que se puede ejecutar las pruebas de un modo automático.

REGISTRO DE PRUEBAS

PROPÓSITO
El propósito de este producto de trabajo es verificar que se han realizado las pruebas y proporcionar información sobre los resultados obtenidos de dichas pruebas.

DESCRIPCIÓN	
	El Registro de Pruebas es de tipo artefacto y permite crear un registro preciso de las pruebas llevadas a cabo y de los resultados obtenidos facilitando el posterior análisis de dicha información.

5.2 Herramienta de Soporte a la Metodología

Para dar soporte a la aplicación y al uso de la metodología de composición de servicios propuesta, hemos desarrollado una herramienta cuyo objetivo es facilitar al usuario de la metodología el seguimiento de las diferentes tareas a realizar para diseñar y desarrollar una composición de servicios, y también facilitar la integración y comunicación del trabajo realizado por los diferentes roles involucrados.

Esta herramienta está diseñada para actuar como un asistente al diseño y desarrollo de composiciones de servicios siguiendo la metodología propuesta. Para ello, permite crear nuevas composiciones de servicios y abrir composiciones previamente guardadas. Esta herramienta va indicando al usuario qué tareas debe realizar, proporcionando una

descripción del objetivo de cada tarea, los productos de trabajo que se necesitan, los productos de trabajo que se van a generar y qué información debe proporcionar el usuario para completar dicha tarea.

El resultado generado por esta herramienta de soporte a la metodología es el conjunto de productos de trabajo definidos en la metodología siguiendo, en algunos casos, las plantillas asociadas. Además, este asistente también permite exportar la composición de servicios como un documento XML para que el usuario pueda personalizar la presentación de la información generada.

En el Anexo 2 se proporciona información más detallada acerca del funcionamiento y del interfaz de esta herramienta.

5.3 Soporte a la Ejecución de Composiciones Dinámicas

La Figura 5-8 muestra una visión global del proceso de diseño, desarrollo y ejecución de las composiciones de servicios dinámicas utilizando el enfoque propuesto en esta tesis doctoral. A partir de los requisitos del sistema y aplicando la metodología de composición de servicios, bien sea directamente o a través de la herramienta de soporte desarrollada, se genera la composición de servicios, compuesta del modelo ejecutable de la composición y del modelo ejecutable de la variabilidad.

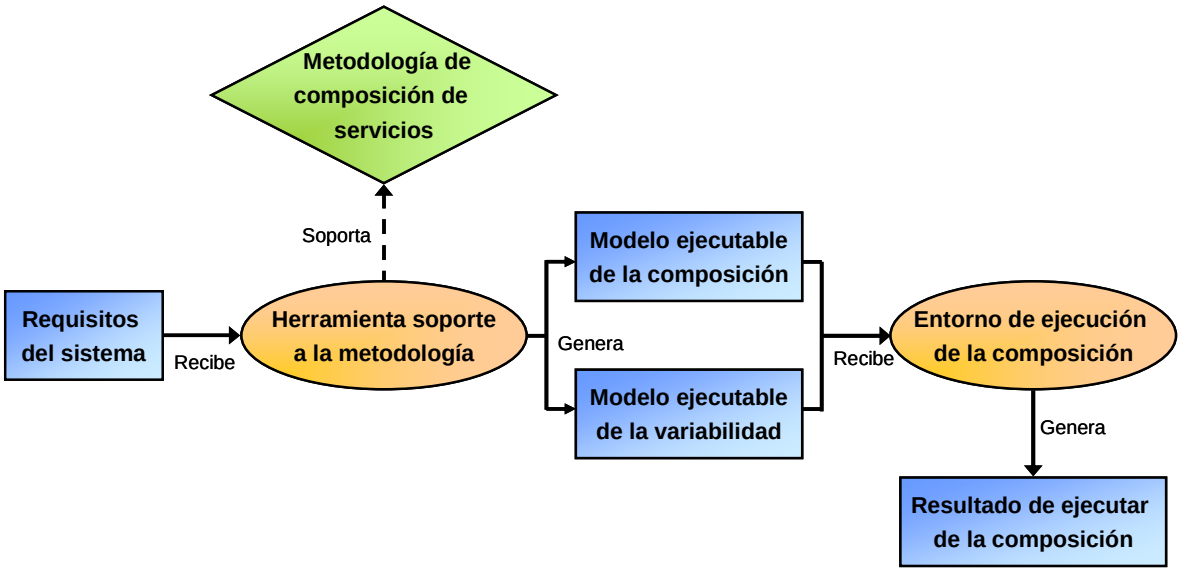


Figura 5-8: Visión global del diseño, desarrollo y ejecución de las composiciones de servicios

Una vez que se han generado los modelos ejecutables de la composición de servicios, la cuestión a resolver es cómo se va a dar soporte en tiempo de ejecución a la dinamicidad de la composición definida durante el tiempo de diseño. Son varios los enfoques a adoptar para dar solución a esta pregunta. Una de las soluciones más conocidas que permiten dan soporte a la asociación dinámica de servicios en tiempo de ejecución es el uso de motores de reglas o de ficheros de configuración para recoger la información requerida, combinado con un middleware que, simplemente, redirige la invocación al servicio concreto cuando es necesario [Butler 2006].

En nuestro caso, para implementar la variabilidad de una composición se ha optado por el uso de reglas ECA (Evento-Condición-Acción). El uso de estas reglas nos permite lanzar eventos para poder realizar asociaciones dinámicas de servicios concretos e incluso búsquedas de posibles servicios concretos durante la ejecución de la composición. De esta forma, cuando se ha definido un punto de variación en el flujo de trabajo de la composición, se evalúa la condición de la regla, según el valor obtenido se selecciona la variante apropiada y se realizan las acciones definidas. El uso de reglas también permite la definición de pre-condiciones y post-condiciones asociadas a la composición para validar, por ejemplo, que los resultados obtenidos tras ejecutar un servicio concreto son los esperados, y en caso negativo, para que se tomen las acciones oportunas. Además, es importante resaltar que existen varias herramientas en el mercado que dan soporte a la ejecución de reglas ECA.

El ámbito del soporte a la composición dinámica de servicios en tiempo de ejecución queda fuera del alcance de esta tesis. Sin embargo, hemos estado colaborando de manera estrecha con otras organizaciones y empresas dentro de un proyecto europeo llamado SeCSE (*Service Centric System Engineering*) [SeCSE 2008] en esta línea de trabajo.

El motivo de explicar brevemente en esta sección el trabajo llevado a cabo para dar soporte a la composición dinámica de servicios en tiempo de ejecución es que nos permite validar y verificar que las composiciones realizadas utilizando la metodología de composición de servicios descrita en esta tesis son capaces de adaptarse dinámicamente en tiempo de ejecución a las condiciones del contexto de ejecución.

Así, dentro del proyecto SeCSE se está desarrollando un entorno que proporciona soporte a la ejecución de las composiciones dinámicas basadas en el enfoque propuesto. Los dos elementos principales de este entorno son una herramienta para diseñar composiciones de acuerdo al enfoque propuesto y un middleware para la ejecución de dichas composiciones. El middleware de ejecución (Figura 5-9) está compuesto principalmente por dos componentes, que son un motor de procesos, Active BPEL, y un motor de reglas llamado Drools, que se encargan de la ejecución de la composición definida. Estos componentes están situados sobre una infraestructura de eventos que actúa como elemento integrador y da soporte a la comunicación *multicast* tanto síncrona como asíncrona entre dichos componentes.

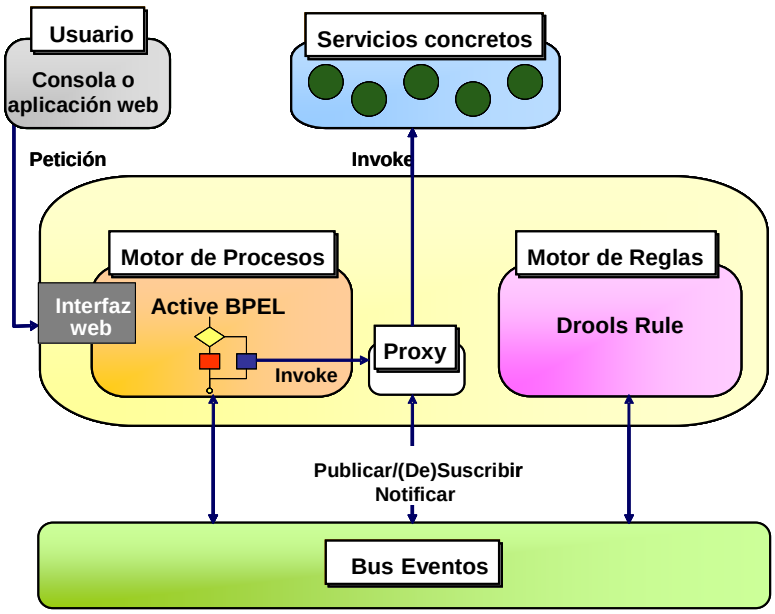


Figura 5-9: Entorno de ejecución de composiciones dinámicas

El procedimiento de ejecución de una composición de servicios es el siguiente: cuando en el flujo de trabajo de la composición se detecta una llamada a un servicio concreto, la plataforma de ejecución invoca al componente proxy. Si esa llamada a un servicio externo no está asociada a un servicio concreto o bien hace referencia a un servicio concreto que no se comporta como se esperaba, el proxy lanza un evento de variabilidad que recibe el motor de reglas. En este momento, es cuando se ejecutan aquellas reglas asociadas al flujo de trabajo de la composición en el punto donde se detectó la llamada a un servicio externo.

Después de que la regla o reglas asociadas son ejecutadas y las acciones definidas completadas, existe un nuevo servicio concreto listo para asociarlo dentro del flujo de la composición. En este punto el proxy es notificado y se prepara para invocar al nuevo servicio concreto. [Colombo et al. 2006] y [SeCSE 2008] proporcionan información más detallada sobre este entorno.

Las ciencias aplicadas no existen, sólo las aplicaciones de la ciencia.

Louis Pasteur

CAPITULO 6

Ejemplos de Aplicación de la Metodología Propuesta

La mayor parte de las organizaciones están familiarizadas con metodologías ya establecidas por lo que es complicado esperar que las dejen de lado para aprender una nueva metodología desde cero. Por este motivo, la mejor alternativa es influenciar en las metodologías disponibles y personalizarlas con la incorporación de características específicas de la composición de servicios, así como los roles y los productos de trabajo correspondientes. De esta forma se mantiene la confianza de las organizaciones y de los usuarios porque que no tienen que abandonar sus metodologías, y se reduce el tiempo de adaptación y de aprendizaje necesario para la adopción del nuevo modelo de trabajo.

En este capítulo se proporcionan dos ejemplos de aplicación de la metodología de composición de servicios. Para estos ejemplos se han escogido dos metodologías de software con una amplia acogida por la industria: *Rational Unified Process* (*RUP*) y *Extreme Programming* (*XP*). No es un objetivo de esta tesis recomendar el uso de ciertas metodologías ágiles de software, sino demostrar que la metodología de composición de servicios es lo suficientemente genérica como para incorporarla a cualquier metodología ágil de software.

6.1 Rational Unified Process

El *Rational Unified Process*, también conocido con el acrónimo RUP, es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado [UML 1997], constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Aunque este ejemplo intenta ser lo más completo posible, RUP no es un sistema con pasos firmemente establecidos, por lo que dependiendo de la naturaleza de cada proyecto será necesario adaptar el proceso al contexto y necesidades de cada organización.

6.1.1 Introducción a RUP

La arquitectura de RUP tiene dos dimensiones o ejes [RUP 2002]. El eje horizontal representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Figura 6-1 que RUP consta de cuatro fases: Concepción, Elaboración, Construcción y Transición. Cada una de estas fases se subdivide a la vez en iteraciones.

RUP consta de cuatro fases: Concepción, Elaboración, Construcción y Transición. Cada una de estas fases se subdivide a su vez en iteraciones.

- **Concepción**: El objetivo es generar un plan de fases y de identificar los principales casos de uso y los potenciales riesgos.

- **Elaboración**: Se encarga de hacer un plan de proyecto, analizar el dominio del problema y de modelar el negocio y los requisitos a partir de los casos de uso identificados previamente.

- **Construcción**: El objetivo es enfocarse en la elaboración de un producto totalmente operativo y eficiente y también de un manual de usuario.

- **Transición**: El objetivo es instalar el producto en el cliente y dar formación a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

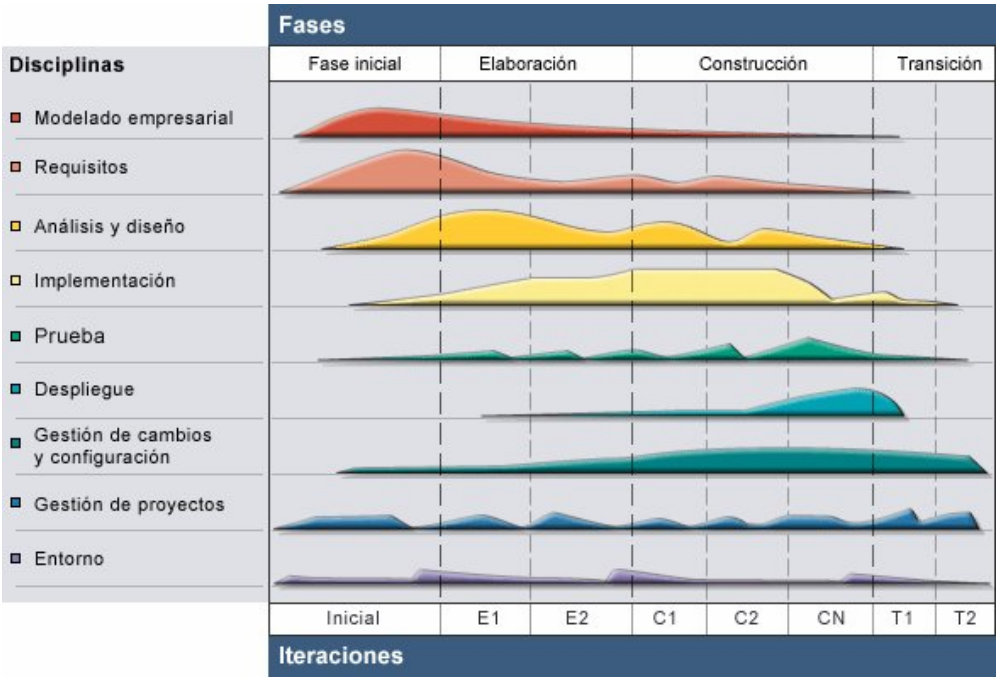


Figura 6-1: Rational Unified Process (Fuente: IBM Rational Software)

Como se observa en la Figura 6-1, las primeras iteraciones (en las fases de Concepción y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una *baseline* de la arquitectura.

Durante la fase de Concepción las iteraciones ponen mayor énfasis en actividades de modelado del negocio y de requisitos. En la fase de Elaboración, las iteraciones se orientan al desarrollo de la *baseline* de la arquitectura, abarcan además los flujos de trabajo de requisitos, el modelo de negocios (refinamiento), el análisis, el diseño y una parte de implementación orientado a la línea base de la arquitectura.

En la fase de Construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se seleccionan algunos casos de uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se producen el número de iteraciones necesarias hasta terminar la implementación de la nueva versión del producto. En la fase de Transición se pretende garantizar que el producto está preparado para su entrega a la comunidad de usuarios.

El eje vertical de la Figura 6-1 representa los aspectos estáticos del proceso. Para ello, describe el proceso en términos de disciplinas, definiendo los componentes del proceso, actividades, flujos de trabajo, artefactos y roles. En cada fase participan todas las disciplinas, pero dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

RUP distingue nueve tipos de disciplinas.

- **Modelado del negocio**: Esta disciplina tiene el objetivo de entender las necesidades del negocio para poder definir lo que el producto software debe hacer. Para ello se apoya en documentos de requisitos generales y de alto nivel, reglas del negocio o glosarios de términos.

- **Gestión de Requisitos**: Esta disciplina traduce las necesidades del negocio a requisitos dentro de la arquitectura software. Estos requisitos poseen un carácter más técnico y se suelen representar como casos de uso UML. El objetivo de esta disciplina es obtener un mayor entendimiento del modelo de negocio por parte de los integrantes del equipo de desarrollo.

- **Análisis y diseño**: Esta disciplina define, a partir de los requisitos, la arquitectura del sistema más adecuada y el diseño detallado necesario previo a la disciplina de implementación.

- **Implementación**. Esta disciplina incluye todas aquellas actividades de codificación del sistema siguiendo el diseño previamente definido y siempre cumpliendo con los requisitos del sistema.

- **Pruebas**: Esta disciplina se encarga de verificar y validar que todos los elementos que se producen, bien sean documentos, diseños o códigos, cumplen con los requisitos y con los estándares de calidad definidos para el proyecto.

- **Despliegue**: Esta disciplina define el conjunto de actividades que permiten tener el sistema instalado en los entornos en que finalmente va a ser explotado.

- **Gestión de configuración**: Esta disciplina se encarga de la gestión de los cambios y todos los elementos que intervienen en el proceso de construcción

- **Gestión del proyecto**: Esta disciplina gestiona el proceso de desarrollo del proyecto enfocándose en los planes, recursos, seguimiento y control y gestión de riesgos.

- **Entorno de desarrollo**: Esta disciplina tiene como objetivo proporcionar todos los recursos hardware y software necesarios para facilitar la puesta en marcha y mantenimiento de los distintos entornos de desarrollo y pruebas del sistema.

6.1.2 Aplicación de la Metodología de Composición

La Tabla 6-1 muestra esa correspondencia de terminologías entre los términos utilizados en la metodología de composición de servicios y los términos utilizados en RUP.

METODOLOGÍA DE COMPOSICIÓN	RUP
Ciclo de vida	Fases
Fases	Disciplinas
Productos de trabajo	Artefactos
Roles	Roles

Tabla 6-1: Correspondencia general entre la metodología propuesta y RUP

La aplicación de la metodología de composición a la metodología RUP se va a realizar desde tres perspectivas: las fases, los roles y los productos de trabajo definidos en la metodología de composición.

La metodología de composición de servicios está orientada a recoger los aspectos de diseño y desarrollo de composiciones de servicios por lo que, de entre las nueve disciplinas definidas en RUP, sólo nos enfocaremos en tres de ellas: Análisis y Diseño, Implementación y Pruebas.

Por cada una de estas disciplinas se definirá el flujo de trabajo principal que será extraído directamente de la metodología RUP. La mayoría de las actividades definidas en cada disciplina no van a necesitar ser adaptadas por lo que sólo se hará referencia a aquellas actividades donde haya que integrar los aspectos específicos de la composición de servicios recogidos en la metodología presentada en esta tesis. También es posible que haya que añadir nuevas actividades al flujo de trabajo de alguna de las disciplinas de RUP ya que están más orientadas al desarrollo de software y no al desarrollo de servicios.

Disciplina: Análisis y Diseño

El flujo de trabajo de la disciplina de Análisis y Diseño es el siguiente:

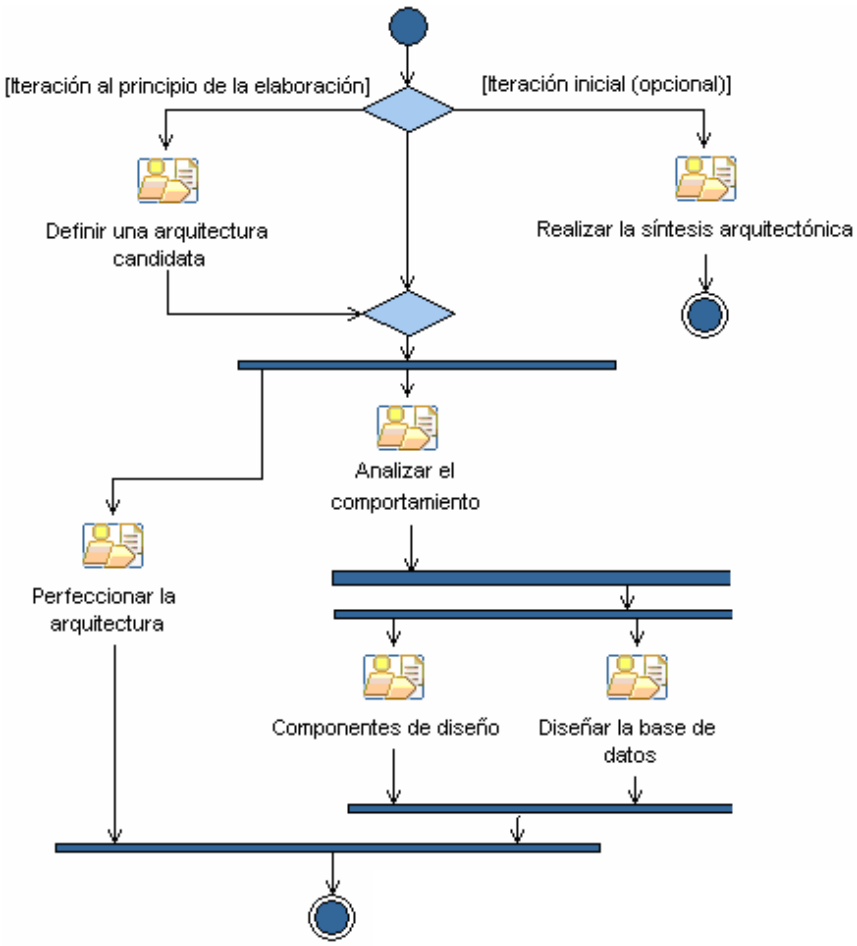


Figura 6-2: Flujo de trabajo de la disciplina Análisis y Diseño (Fuente: IBM Rational Method Composer v7.0)

En esta disciplina habría que integrar las actividades incluidas en las fases de Análisis y de Diseño de la metodología de composición de servicios. Tras analizar cada una de las actividades definidas en RUP dentro de esta disciplina, la integración se realizaría de la siguiente manera:

- El objetivo de la actividad *Especificación Arquitectural* de la metodología de composición coincide con el objetivo de la actividad *Definir una arquitectura candidata* de RUP, por lo tanto habría que integrar ambas actividades con el fin de obtener los productos de trabajo requeridos por la metodología de composición.

- Las tareas definidas en las actividades *Especificación Funcional* y *Composición en Tiempo de Diseño* de la metodología de composición no encajan en ninguna de las actividades descritas en RUP ya que son específicas de los servicios; por lo tanto, se puede extender el flujo de trabajo de RUP incluyendo estas dos actividades (Figura 6-3).

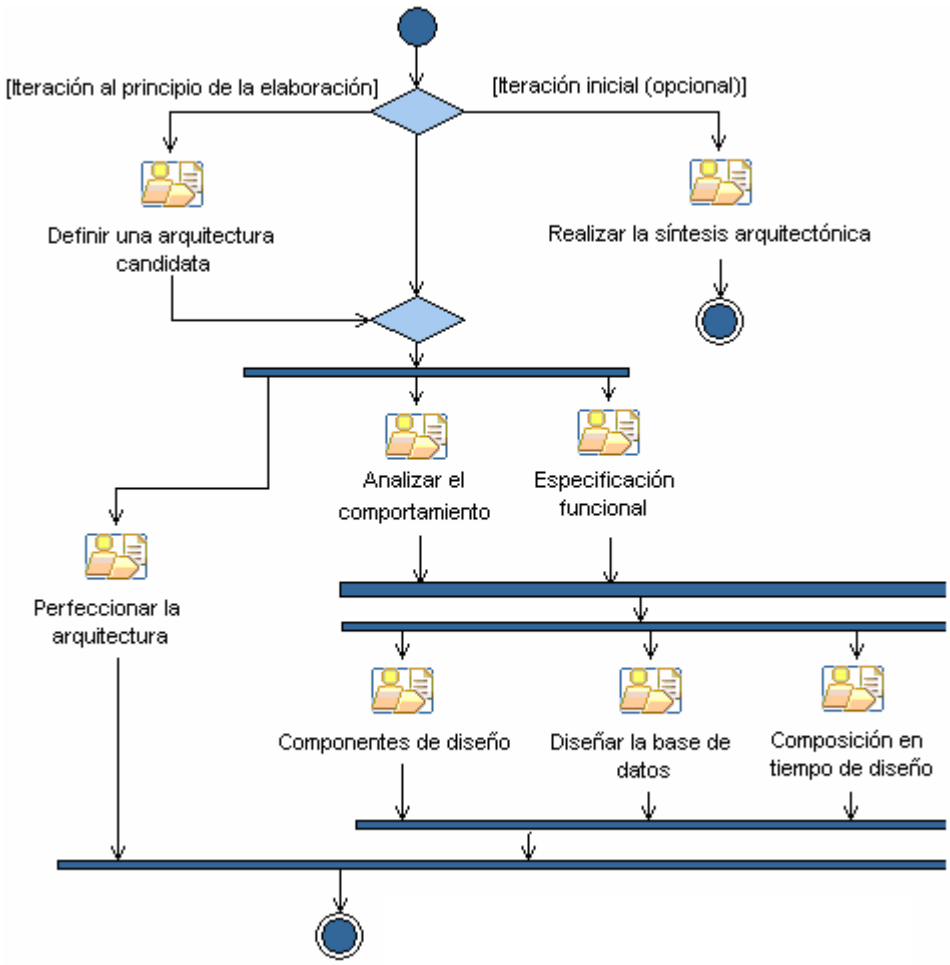


Figura 6-3: Flujo de trabajo integrado con RUP para el Análisis y Diseño (Fuente: IBM Rational Method Composer v7.0)

La actividad *Especificación Funcional* se realizaría en paralelo con la actividad *Analizar el comportamiento* de RUP. Y la actividad *Composición en Tiempo de Diseño* se realizaría en paralelo con las actividades *Componentes de Diseño* y *Diseñar la base de datos*.

Disciplina: Implementación

El flujo de trabajo de la disciplina de Implementación está reflejado en la Figura 6-4.

En esta disciplina habría que integrar las actividades incluidas en la fase de Implementación de la metodología de composición de servicios.

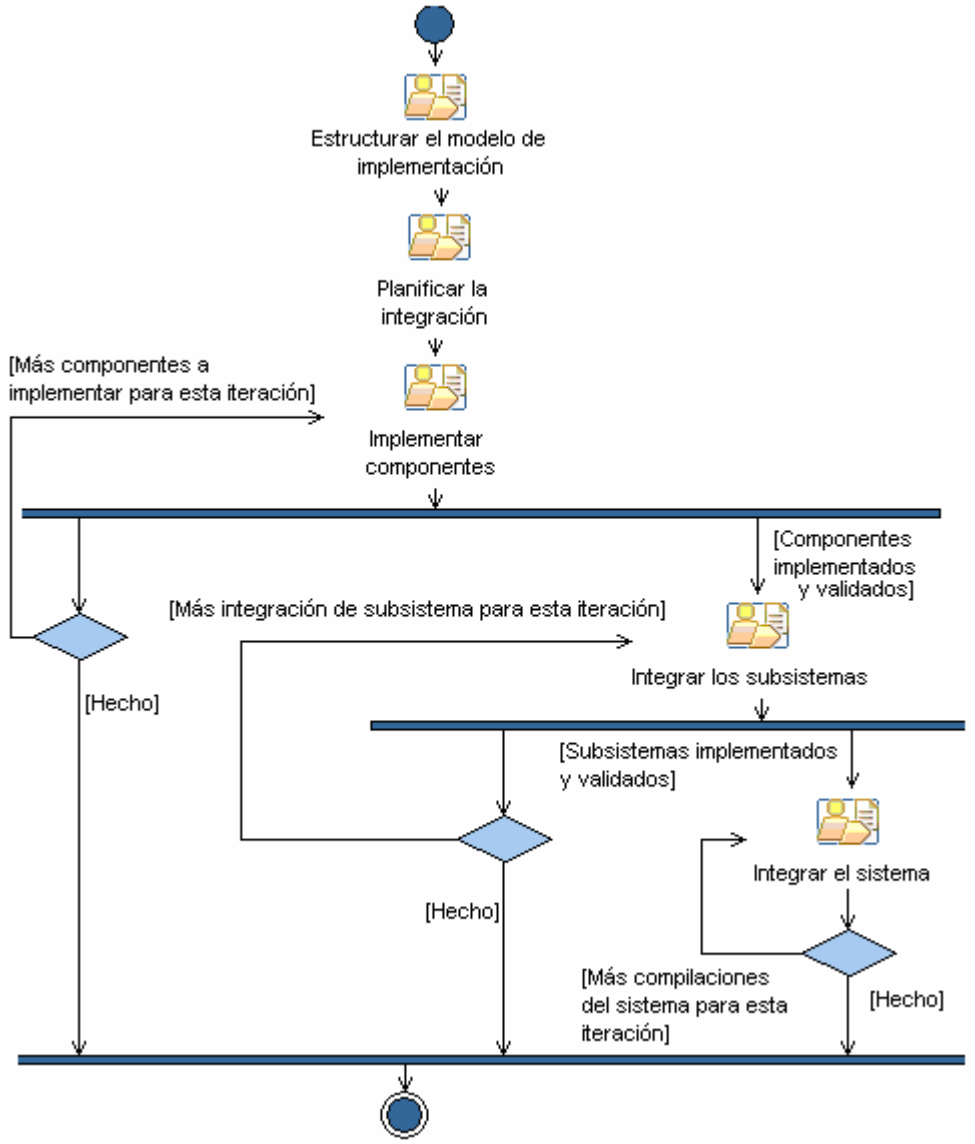


Figura 6-4: Flujo de trabajo de la disciplina Implementación (Fuente: IBM Rational Method Composer v7.0)

Tras analizar cada una de las actividades definidas en RUP dentro de esta disciplina, la integración se realizaría de la siguiente manera:

- Las tareas definidas en las actividades *Realización de la Composición* y *Generación de la Composición Ejecutable* de la metodología de composición no encajan en ninguna de las actividades descritas en RUP ya que están más enfocadas al desarrollo de componentes software y no de composiciones de servicios.

  Aunque en el caso de que se vaya a desarrollar un servicio, éste se podría tratar como un componente software a ser integrado en la composición. Por lo tanto, se puede

extender el flujo de trabajo de RUP para integrar estas dos actividades de la metodología de composición de servicios (Figura 6-5).

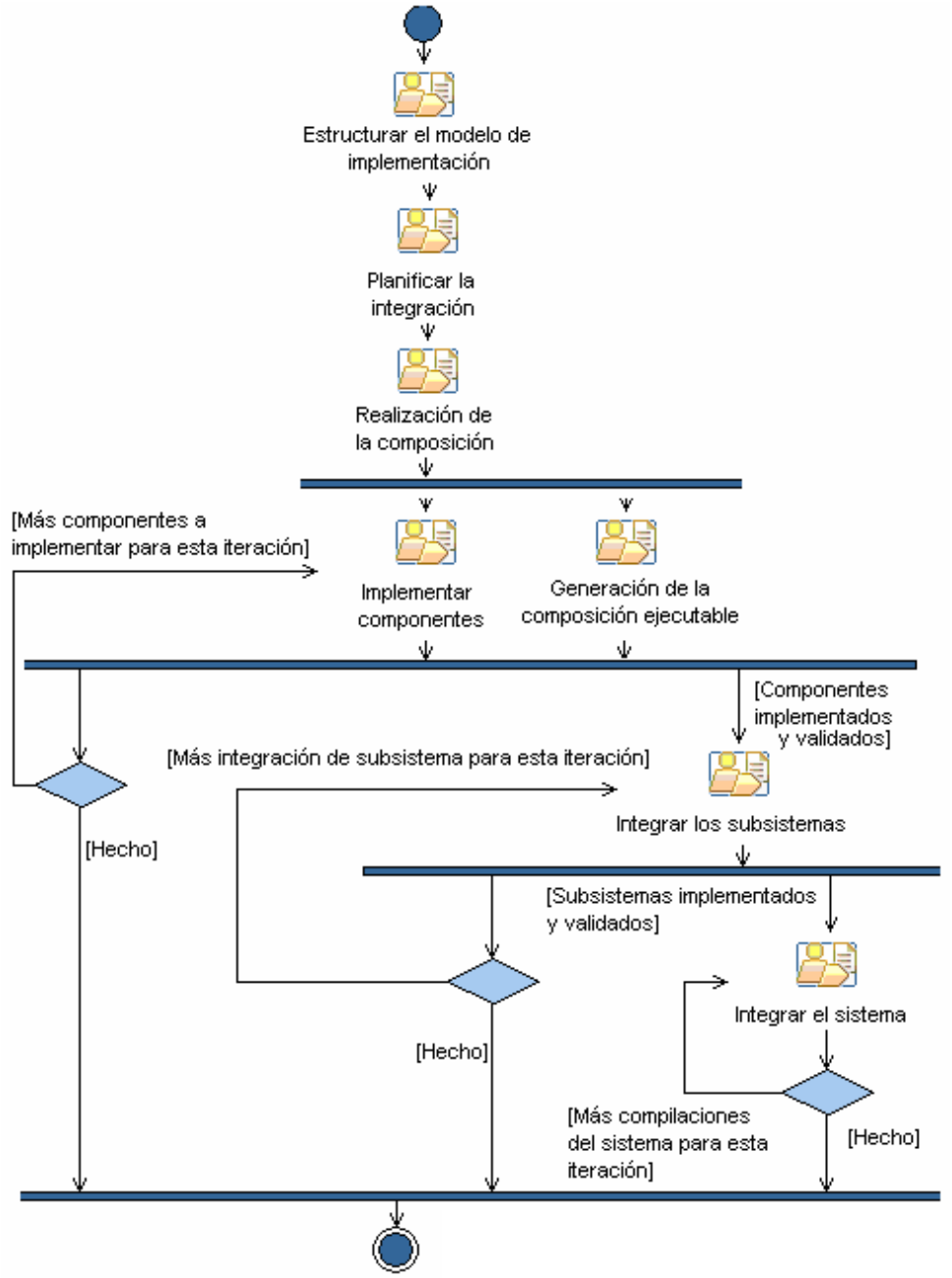


Figura 6-5: Flujo de trabajo integrado con RUP para la Implementación (Fuente: IBM Rational Method Composer v7.0)

La actividad *Realización de la Composición* se realizaría antes de empezar la implementación de los componentes. Y la actividad *Generación de la Composición Ejecutable* se realizaría en paralelo con la actividad *Implementación de componentes*.

Disciplina: Pruebas

El flujo de trabajo de la disciplina de Pruebas es el siguiente:

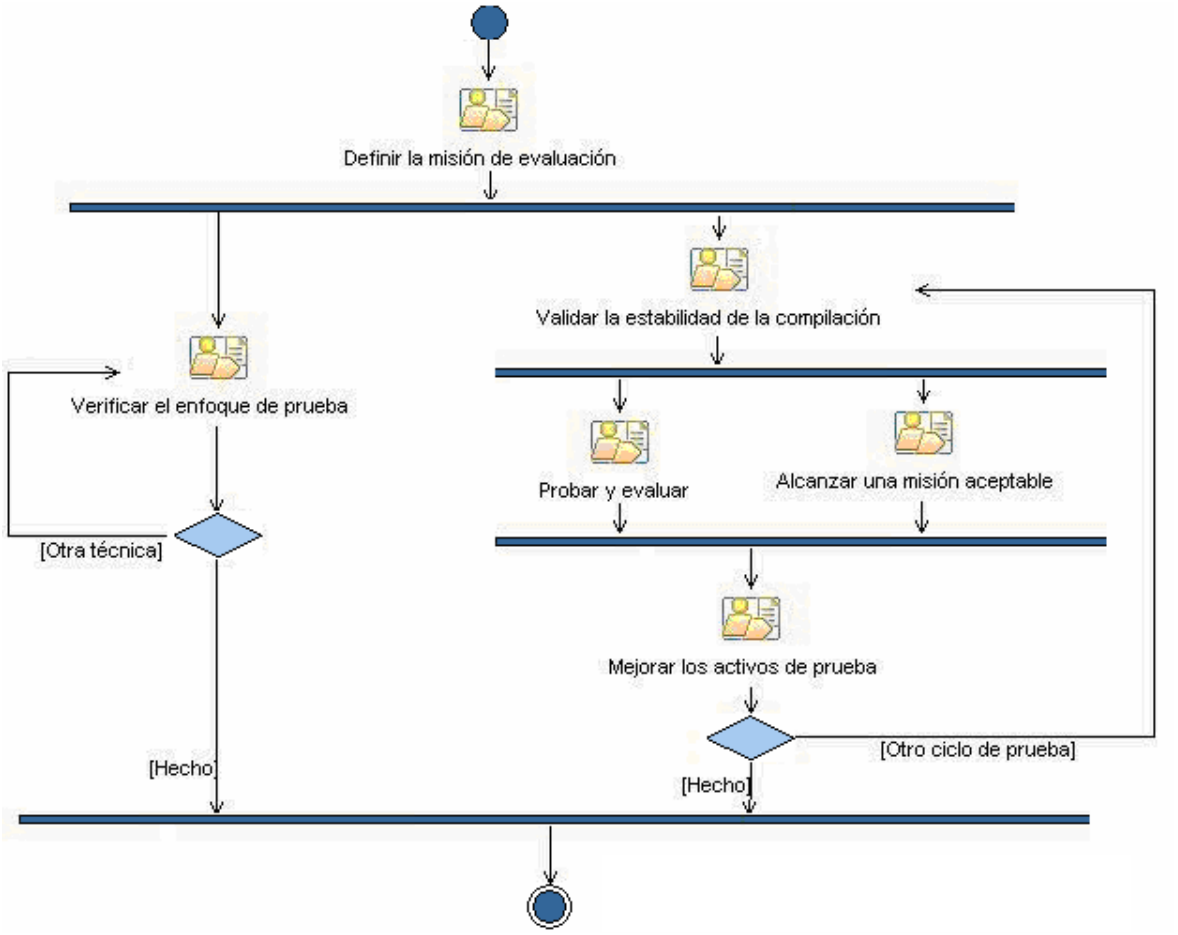


Figura 6-6: Flujo de trabajo de la disciplina Pruebas (Fuente: IBM Rational Method Composer v7.0)

En esta disciplina habría que integrar las actividades incluidas en la fase de Pruebas de la metodología de composición de servicios. La disciplina de Pruebas de RUP recoge un completo conjunto de actividades para realizar la validación y verificación del sistema. En el caso de la metodología de composición que proponemos en esta tesis, sólo se incluyen dos tareas, *Crear Casos de Prueba* y *Ejecutar Pruebas*, dentro de la fase de Pruebas.

Los pasos definidos en estas dos tareas de la metodología de composición, así como los resultados obtenidos, son los mismos que están descritos en la actividad *Probar y evaluar* definida en RUP por lo que ya estaría integrada esta fase en la metodología de RUP.

La Tabla 6-2 muestra la correspondencia entre los roles definidos en la metodología de composición de servicios y RUP.

METODOLOGÍA DE COMPOSICIÓN	RUP
Analista del Sistema	Analista de Sistemas
Analista de procesos de negocio	Diseñador del negocio
Arquitecto	Arquitecto
Desarrollador	Implementador
Responsable de pruebas	Dependiendo de la actividad se divide en: • Analista de pruebas • Diseñador de pruebas • Verificador

Tabla 6-2: Correspondencia de roles entre la metodología de composición y RUP

En cuanto a los productos de trabajo definidos en la metodología de composición de servicios, una parte de dichos productos de trabajo tienen una correspondencia con otro producto de trabajo definido en RUP (Tabla 6-3).

METODOLOGÍA DE COMPOSICIÓN	RUP
Caso de Prueba	Caso de Prueba
Guiones de prueba	Script de prueba
Modelo de Especificación de la Composición Modelo de Especificación de los Puntos de Variación	Modelo de Diseño
Modelo de Especificación Funcional	Modelo de Análisis
Modelo de Realización de la Composición	Modelo de Implementación
Modelo Ejecutable de la Composición Modelo Ejecutable de la Variabilidad	Compilación
Patrones y modelos arquitectónicos	Arquitectura de referencia
Políticas Arquitectónicas	Documento de arquitectura del software
Registro de Prueba	Registro de Prueba
Requisitos arquitectónicos	Arquitectura de prueba de concepto

Tabla 6-3: Correspondencia de productos de trabajo entre la metodología propuesta y RUP

Pero también hay varios productos de trabajo nuevos que habría que incluir en el repertorio de algunas de las disciplinas de RUP, como son (Tabla 6-4):

DISCIPLINA DE RUP	METODOLOGÍA DE COMPOSICIÓN
Análisis y Diseño	Repositorio Organizacional de Modelos Lista de Servicios Disponibles para ciertos Requisitos
Implementación	Lista de Servicios Disponibles para una Especificación Lista de Servicios Concretos Candidatos

Tabla 6-4: Nuevos productos de trabajo a incluir en RUP

6.2 Extreme Programming

La metodología *Extreme Programming* [Beck 2000], también conocida con el acrónimo *XP*, es un enfoque de la Ingeniería de Software formulado por primera vez por Kent Beck en el año 1999. El *Extreme Programming* surgió como respuesta y posible solución a los problemas derivados del cambio en los requisitos, siendo hoy en día uno de los procesos ágiles de desarrollo de software más destacados. En resumen, la metodología XP combina las mejores prácticas para desarrollar software, y las lleva al extremo.

6.2.1 Introducción a XP

El *Extreme Programming* es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Es decir, XP está orientado hacia quien produce y usa el software, enfocándose en una realimentación continua entre cliente y desarrollador.

Las características esenciales de esta metodología son las cuatro siguientes:

- **Comunicación**: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

- **Simplicidad**: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.

- **Realimentación**: Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.

- **Coraje**: Se debe tener coraje o valentía para cumplir los tres puntos anteriores. Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador. Hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y, por último, hay que tener valor y confiar en que la realimentación sea efectiva.

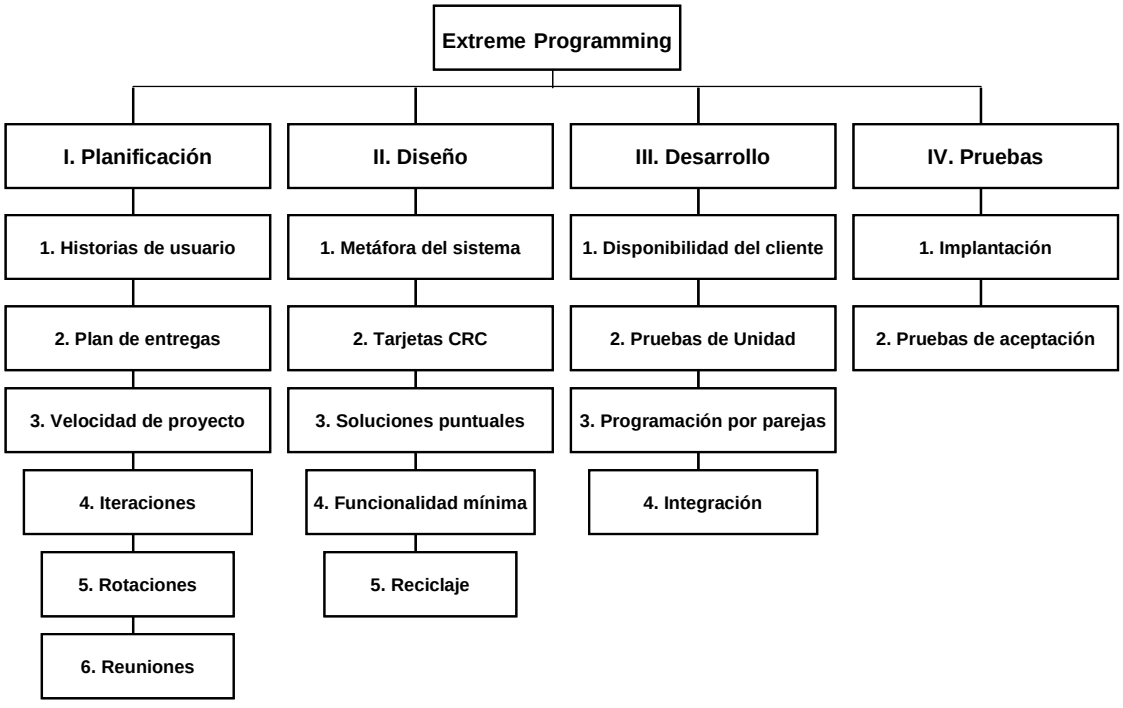


Figura 6-7: Visión general de la metodología XP

La Figura 6-7 muestra un conjunto de reglas y buenas prácticas para cada una de las fases de la metodología XP, que son Planificación, Diseño, Desarrollo y Pruebas.

XP define un conjunto de prácticas que de forma aislada no tienen mucho sentido, pero en conjunto, unas prácticas compensan las carencias que las otras puedan tener.

- **Juego de la Planificación** (*Planning Game*): El objetivo del juego es maximizar el valor del software producido. La estrategia es poner en producción las características más importantes lo antes posible. Las piezas clave son las historias de usuario. Los jugadores son los desarrolladores y el cliente, y los posibles movimientos a realizar son la exploración, la selección y la actualización.

- **Versiones cada poco tiempo** (*Short Releases*): Un sistema simple se pone rápidamente en producción. Periódicamente, se producen nuevas versiones agregando en cada iteración aquellas funciones que se consideran que dan valor añadido al sistema de cara a satisfacer los requisitos del cliente.

- **Metáfora del Sistema** (*Metaphor*): Cada proyecto es guiado por una historia simple de cómo funciona el sistema en general, reemplaza a la arquitectura y debe usar un lenguaje común y entendible tanto para el cliente como para los desarrolladores.

- **Diseño Simple** (*Simple Designs*): El sistema se diseña con la máxima simplicidad posible. Se plasma el diseño en tarjetas CRC (Clase–Responsabilidad–Colaboración) y no se implementan aquellas características que no son necesarias. De este modo, las clases descubiertas durante el análisis pueden ser filtradas para determinar qué clases son realmente necesarias para el sistema.

- **Pruebas Continuas** (*Testing*): Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

- **Refactorización** (*Refactoring*): Es posible reestructurar el sistema sin cambiar su comportamiento, por ejemplo eliminando el código duplicado o simplificando funciones. Si el código se va complicando se debería modificar el diseño y volver a uno más simple.

- **Programación por Parejas** (*Pair Programming*): El código es escrito por dos personas trabajando en el mismo ordenador.

- **Posesión Colectiva del Código** (*Collective Code Ownership*): Nadie es dueño de un módulo. Cualquier programador puede cambiar cualquier parte del sistema en cualquier momento, siempre que se utilicen estándares. Las pruebas siempre deben funcionar al 100 por ciento para realizar integraciones con todo el código permanentemente.

- **Integración Continua** (*Continuous Integration*): Los cambios se integran en el código base varias veces por día. Todos lo casos de prueba se deben pasar antes y después de la integración. Es necesario disponer de una máquina para la integración y realizar pruebas funcionales en donde participa el cliente.

- **Semana Laboral de 40 horas** (*40-Hour Week*): Cada trabajador debe trabajar como máximo 40 horas por semana. Si fuera necesario hacer horas extra, esto no debería hacerse dos semanas consecutivas. De esta manera se reduce la rotación del personal y se mejora la calidad del producto.

- **Disponibilidad del Cliente** (*On Site Customer*): El equipo de desarrollo tiene acceso todo el tiempo al cliente, el cual está disponible para responder posibles preguntas o fijar prioridades.

- **Estándares de Codificación** (*Coding Standard*): Todo el código debe estar escrito de acuerdo a un estándar de codificación.

6.2.2 Aplicación de la Metodología de Composición

La aplicación de la metodología de composición de servicios a una organización que ya está utilizando la metodología XP es muy sencilla ya que XP sólo propone un conjunto de reglas y prácticas a seguir en cada fase, sin identificar las tareas a realizar o los productos de trabajo que hay que obtener.

Por ello, para poder integrar la metodología de composición de servicios con XP sólo es necesario tener en cuenta esas prácticas en cada una de las fases definidas en la metodología de composición e involucrar en la medida de lo posible a los clientes en todo

el proceso de desarrollo de la composición de servicios desde el análisis hasta la puesta en marcha.

Fase: Planificación

En esta fase, con ayuda del cliente se redactan las historias de usuario que definen cómo debe funcionar el sistema y el equipo de trabajo se encarga de analizar dichos requisitos y traducirlo a una especificación técnica, identificando los bloques funcionales de tal modo que sea comprensible por el programador. Para realizar este análisis, en esta fase de Planificación se incluirían todos los procesos definidos en fase de Análisis de la metodología de composición de servicios (*Especificación Arquitectural* y *Especificación Funcional*).

Fase: Diseño

A partir de las historias de usuario definidas en la fase anterior y de los bloques funcionales identificados, en esta fase se refinan dichos bloques funcionales y se definen sus relaciones y dependencias. En esta fase de Diseño se incluirían todos los procesos definidos en fase de Diseño de la metodología de composición de servicios (*Composición en Tiempo de Diseño*).

Fase: Codificación

En la metodología XP se pone mucho énfasis en la programación de tal modo que se desarrolle código óptimo y modular. La fase de Implementación de la metodología de composición de servicios se enfoca en reutilizar, en la medida de lo posible, servicios ya existentes e integrarlos en la composición por lo que la codificación se reduciría al desarrollo de ciertos servicios si fueran necesarios y a la generación de los modelos ejecutables de la composición. Por lo tanto, en esta fase de Codificación se incluirían todos los procesos definidos en fase de Implementación de la metodología de composición de servicios (*Realización de la Composición* y *Generación de la Composición Ejecutable*).

Fase: Pruebas

XP recomienda que todas las pruebas, o al menos la mayoría de ellas, sean automatizadas de tal modo que sea más fácil y rápido reaccionar ante un cambio de los requisitos. En esta fase se incluirían todos los procesos definidos en fase de Pruebas de la metodología de composición de servicios (*Validación y Verificación*).

La Tabla 6-5 muestra esa correspondencia de terminologías entre los términos utilizados en la metodología de composición de servicios que hemos definido en el presente documento y los términos generales utilizados en XP.

METODOLOGÍA DE COMPOSICIÓN	XP
Análisis	Planificación
Diseño	Diseño
Implementación	Desarrollo
Pruebas	Pruebas
Roles	Roles

Tabla 6-5: Correspondencia general entre la metodología propuesta y XP

La Tabla 6-6 muestra la correspondencia entre los roles definidos en la metodología de composición de servicios y los roles definidos en XP.

METODOLOGÍA DE COMPOSICIÓN	XP
Analista del Sistema Analista de procesos de negocio Arquitecto Desarrollador	Programador
Responsable de pruebas	Encargado de pruebas

Tabla 6-6: Correspondencia de roles entre la metodología propuesta y XP

En cuanto a los productos de trabajo, la metodología XP no define artefactos para cada proceso, por lo tanto se pueden usar directamente los productos de trabajo definidos en la metodología de composición. Sin embargo, es importante resaltar que los requisitos del sistema en la metodología XP se describen en forma de historias de usuario. Las historias de usuario tienen el mismo objetivo que los casos de uso pero no son lo mismo, ya que lo escriben los propios clientes por lo que las descripciones son cortas y no suelen emplear un lenguaje técnico.

Las ciencias tienen las raíces amargas, pero muy dulces los frutos.

Aristóteles

CAPITULO 7

Validación y Evaluación de la Metodología Propuesta

El objetivo de este capítulo consiste en validar y evaluar la metodología de composición de servicios propuesta en el capítulo 5 utilizando como criterios de validación las características generales fijadas para la metodología en la sección 1.2.2. Estas características son cuatro: el valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de dicha metodología.

Para ello, la sección 7.1 valida la metodología de composición describiendo de un modo conciso el desarrollo de un servicio compuesto dentro de un escenario industrial real, utilizando los pasos definidos en la metodología de composición propuesta. Es importante resaltar que la metodología presentada en el capítulo 5 es ya la versión refinada y final, la cual ha sufrido cambios y mejoras extraídas de las diferentes validaciones y evaluaciones realizadas sobre cada versión de la metodología de composición de servicios.

Tras validar la metodología, en la sección 7.2 se realiza una evaluación de dicha metodología de composición de acuerdo al marco *Normative Information Model-based Systems Analysis and Design* (NIMSAD). Finalmente, en la sección 7.3 se resaltan los resultados de la validación y evaluación obtenidos.

7.1 Validación de la Metodología de Composición

Como se ha mencionado anteriormente, la metodología de composición de servicios propuesta en esta tesis requiere de una adecuada validación. Una práctica habitual para validar metodologías enfocadas al desarrollo de sistemas de información consiste en aplicar dichas metodologías a un escenario industrial real donde se desarrollan diferentes sistemas típicos. De este modo, se muestran los resultados que se generan, se describen las decisiones que han de ser tomadas, se proporcionan guías y recomendaciones cuando se requieren y se resaltan posibles soluciones a ciertos problemas típicos que podrían surgir.

La utilización de casos de estudio industriales y reales para validar metodologías es también una práctica muy común en el campo de la ingeniería de servicios ([SeCSE 2008], [SODIUM 2007], [ATHENA 2007]) utilizada de manera satisfactoria para demostrar y validar el diseño, desarrollo, despliegue y gestión de servicios en diferentes ámbitos, como por ejemplo, el sector de las telecomunicaciones, la automoción o la e-Salud. Por este motivo, hemos seleccionado este enfoque de la "validación por ejemplo" para validar la metodología de composición de servicios a partir de la definición de un escenario industrial real; y, de esta manera, confirmar que dicha metodología está caracterizada por su valor práctico, facilidad de uso, compatibilidad y aplicabilidad.

Para validar la metodología de composición de servicios bajo unas condiciones que fueran lo más realistas posible y, al mismo tiempo, poder determinar el valor práctico y la aplicabilidad de la metodología propuesta, hemos desarrollado un servicio compuesto dentro de un escenario industrial real que combina los sectores de la automoción y de las telecomunicaciones [Bastida et al. 2008b] siguiendo los pasos definidos en la metodología. El servicio compuesto desarrollado, denominado XTrip, ha sido identificado como un nuevo servicio que va a proporcionar un valor añadido a los sectores de la automoción y de las telecomunicaciones. Se ha seleccionado este servicio compuesto por su complejidad y porque integra varias funcionalidades de diferentes ámbitos.

Esta validación a partir de un escenario industrial real va a proporcionar información valiosa sobre el proceso propuesto en la metodología de composición, en términos de fases, tareas y pasos; además de revelar deficiencias existentes o posibles mejoras.

El uso de un ejemplo para validar la metodología también puede ser considerado como un caso de aplicación de la metodología de composición orientado a todas aquellas personas que deseen desarrollar servicios compuestos, facilitando el entendimiento y comprensión sobre dicha metodología propuesta.

7.1.1 Definición del Escenario Industrial Real

Esta sección describe un escenario industrial real [Bastida et al. 2008b] donde se resaltan las ventajas que podrían proporcionar ciertos servicios avanzados en el campo de la automoción y las telecomunicaciones. Este escenario define una breve historia en la que se mezclan aspectos del mundo de la automoción y las telecomunicaciones con el objetivo de proporcionar servicios avanzados e interesantes para los conductores de vehículos, como por ejemplo, un servicio que permite gestionar de manera remota las agendas personales de los conductores y planificar sus viajes.

El escenario ha sido definido por un consorcio formado por socios industriales dentro del contexto del proyecto SeCSE. Los socios involucrados han sido el *Centro Ricerche Fiat* y *Telecom Italia Lab* de Italia, *Daimler Chrysler* de Alemania, y *Telefónica I+D* y *Atos Origin* de España. Hemos utilizado este escenario para validar la metodología de composición de servicios y analizar su aplicación a áreas de acción específicas, como son el área de la automoción y el área de las telecomunicaciones.

La descripción del escenario está dividida desde dos perspectivas:

- **Perspectiva de los proveedores de servicios para vehículos**, que describe cómo los proveedores de servicios buscan ofrecer nuevos servicios que proporcionen valor añadido a los vehículos y satisfagan potenciales necesidades de todos o de algún sector concreto de los conductores.

- **Perspectiva del usuario**, que describe las ventajas de consumir los servicios proporcionados a través de un dispositivo incorporado en el vehículo. En este caso, el

usuario es un conductor de coches que utiliza dicho dispositivo especial incorporado al vehículo y un asistente digital personal (*PDA*).

Perspectiva de los proveedores de servicios para vehículos

XService es una compañía que se encarga de desarrollar y mantener servicios de valor añadido proporcionados a los conductores de vehículos a través de un sistema de suscripción, y consumidos por medio de un dispositivo especial incorporado al vehículo. El director de XService quiere mejorar la funcionalidad ofrecida en los vehículos de gama alta con nuevos servicios, para lo cual pide al equipo de XService que prepare un informe sobre potenciales necesidades y nuevas funcionalidades interesantes para los conductores.

Durante el desarrollo de dicho análisis, el equipo de XService recibe un anuncio de una compañía proveedora de telecomunicaciones sobre la existencia de un nuevo servicio llamado kRecall, el cual es capaz de establecer una conexión automática entre los puntos remotos, evitando la necesidad de marcar manualmente el número de teléfono al que se desea llamar.

Después de realizar un análisis de mercado y de conocer la posibilidad de poder consumir el servicio kRecall, los expertos de XService deciden que una importante mejora para los vehículos de gama alta sería proporcionar un servicio que permitiera al conductor gestionar de manera remota sus viajes, garantizando que no hay conflictos en su agenda personal y, en el caso de que los hubiere, se realizaría una llamada a un número de teléfono especificado, por ejemplo el de su secretaria, para intentar solventar dichos conflictos. Este nuevo servicio se va a llamar XTrip.

En el informe sobre el análisis de mercado se identifica como potenciales consumidores a los profesionales, como por ejemplo los trabajadores autónomos o los directivos de compañías, que necesitan planear y gestionar sus viajes y reuniones de una manera sencilla y rápida. Este servicio también podría ser utilizado por conductores regulares que quieren planificar sus vacaciones o sus viajes. El informe concluye que la explotación del servicio es factible y conveniente, por lo cual se toma como decisión final integrar este nuevo servicio dentro de la cartera de servicios proporcionados por XService.

Según la estrategia de la compañía XService, todos los servicios que se proporcionen deben de cumplir ciertos requisitos de calidad con el fin de satisfacer siempre a los

usuarios. Entre estos requisitos no funcionales se incluyen aspectos relacionados con la disponibilidad, el bajo tiempo de respuesta, la usabilidad, la privacidad de datos y la seguridad.

Perspectiva del usuario

Roberto y Lara trabajan para AirRepair, una compañía especializada en reparación de equipos de aire acondicionado. Roberto es uno de los técnicos de la compañía y Lara trabaja como su secretaria. Todos los días Roberto necesita utilizar uno de los coches de la compañía para poder desplazarse a los domicilios de los clientes y reparar los equipos con problemas. La rutina diaria de Roberto comienza en el aparcamiento de coches de la compañía AirRepair. Una vez que Roberto está dentro del coche, arranca el motor y enciende el dispositivo de abordo para obtener su plan de trabajo para el día.

Lo primero que tiene que hacer Roberto es autenticarse en el sistema para que el dispositivo de abordo le muestre a Roberto su plan de trabajo. Una vez que Roberto tiene su plan de trabajo, selecciona la primera reparación de la lista para obtener toda la información que se disponga, como por ejemplo los datos del cliente, la causa del problema con el equipo de aire acondicionado o los datos del equipo en cuestión. A continuación, Roberto le indica al dispositivo de abordo que le muestre la mejor ruta para llegar a la dirección del cliente haciendo uso del sistema GPS incorporado en el coche y teniendo en cuenta información como problemas de tráfico o cortes de carreteras.

Esta petición de Roberto para calcular la mejor ruta implica que se invoque el servicio XTrip, el cual se encarga de comprobar que no hay ningún conflicto entre el viaje que se ha planificado (en términos de duración en tiempo) con otras reparaciones o reuniones que Roberto tenga planificadas en su agenda.

En caso de que se detecte un conflicto, el servicio XTrip realiza una llamada automática a un número específico de teléfono, en este caso a Lara, con el fin de cambiar la fecha de la reunión planificada y así resolver el conflicto. Para realizar esta llamada telefónica se selecciona el servicio proporcionado por aquella compañía proveedora de telecomunicaciones que, dependiendo del prefijo del número de teléfono que se marque, proporcione la tarifa más barata.

7.1.2 Aplicación de la Metodología Propuesta al Escenario

Esta sección explica detalladamente el desarrollo del servicio compuesto XTrip siguiendo la metodología de composición de servicios propuesta en esta tesis. Sin embargo, esta validación no es sólo importante para validar el valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de la metodología propuesta; sino que también permite ofrecer una visión general sobre cómo se aplica la metodología de composición para diseñar y desarrollar un servicio compuesto, en este caso denominado XTrip, y para ofrecer al lector un plan de trabajo capaz de guiarle a la hora de utilizar dicha metodología de composición.

Finalmente es importante remarcar que, por razones de simplicidad, no se ha tenido en cuenta el ciclo de vida que se ha utilizado para el desarrollo del servicio compuesto XTrip, presentando este caso de estudio como si hubiera sido desarrollado en un único ciclo de desarrollo.

Especificación Arquitectural

La primera tarea en el diseño y desarrollo del servicio compuesto XTrip es analizar el sistema desde el punto de vista de los requisitos arquitectónicos. Para ello, a partir de los requisitos del sistema, el arquitecto de sistemas del grupo XService identifica dos requisitos arquitectónicos, los cuales son clasificados haciendo uso del sistema de clasificación de requisitos FURPS [Grady 1992]:

- Requisito funcional: El sistema deberá proveer control de acceso.
- Requisito de confiabilidad: El sistema tendrá una disponibilidad del 99,95 por ciento.

A continuación, el arquitecto de sistemas analiza los diferentes patrones y modelos arquitectónicos aplicables a la solución final, definiendo sus ventajas y desventajas. El objetivo de este análisis es servir como base para seleccionar la arquitectura más adecuada para la solución a plantear. Los patrones y modelos arquitectónicos identificados son el modelo "pizarra" (*blackboard*), el modelo por capas, el modelo cliente-servidor y el modelo en estrella (*hub-and-spoke*).

Los criterios que se van a seguir para medir la viabilidad de cada patrón y modelo arquitectónico son nueve [Buschmann et. al 1996]:

- **Rendimiento** (*Performance*): Este criterio se puede medir en términos de capacidad de procesamiento, latencia, tiempo de ejecución y tiempo de transacción.

- **Confiabilidad** (*Reliability*): Este criterio representa "*la habilidad de un sistema o componente para realizar las funciones requeridas bajo ciertas condiciones durante un período de tiempo específico*" [IEEE 1990b].

- **Disponibilidad** (*Availability*): Este criterio define el grado en el cual una composición de servicios es operacional y accesible siempre que sea requerido su uso [IEEE 1990a].

- **Seguridad** (*Security*): Este criterio define la habilidad de dar soporte a la autenticación, autorización, integridad, confidencialidad y no-repudio.

- **Escalabilidad** (*Scalability*): Este criterio representa la habilidad de un servicio compuesto de gestionar un incremento en las peticiones de usuario recibidas, las operaciones o las transacciones en un intervalo de tiempo sin impactar drásticamente en su rendimiento, disponibilidad y confiabilidad.

- **Interoperabilidad** (*Interoperability*): Este criterio identifica la capacidad de un patrón de composición para facilitar la interoperabilidad entre servicios localizados en diferentes entornos de desarrollo, sin tener en cuenta aspectos tales como lenguajes de programación o sistemas operativos soportados.

- **Adaptabilidad** (*Adaptability*): Este criterio define la facilidad con la cual una composición de servicios satisface las restricciones del sistema y las necesidades de los usuarios [Evans y Marciniak 1987].

- **Usabilidad** (*Usability*): Este criterio define la facilidad de uso, de integración y de gestión de una composición de servicios.

- **Facilidad de análisis** (*Analyzability*): Este criterio representa el grado de dificultad que una composición de servicios pone a la hora de ser analizada, comprendida o explicada. Pensando en patrones de flujo de trabajo, esta complejidad podría ser caracterizada en términos de número de interfaces, transiciones, ramas condicionales y paralelas, existencia de bucles, roles, categorías de actividades y tipos de estructuras de datos [Cardoso et al. 2005].

A partir de los criterios de viabilidad ya definidos, se analizan cada uno de los patrones y modelos arquitectónicos para identificar sus ventajas e inconvenientes, recogiendo los resultados finales en un informe.

- El **modelo "Pizarra"** (*blackboard*) define una colección de servicios independientes que cooperan en base a una estructura de datos común. Cada servicio está especializado para resolver una parte de la tarea completa y todos los servicios trabajan juntos para alcanzar la solución.

  Este patrón o modelo arquitectónico tiene una influencia positiva en los criterios de calidad [Buschmann et. al 1996] de adaptabilidad, facilidad de análisis y confiabilidad. En cambio, influye negativamente en el criterio de rendimiento.

- El **modelo por capas** permite descomponer las aplicaciones en grupos de sub-tareas con un nivel de abstracción diferente. Este modelo por capas soporta la abstracción y la modularización de la solución.

  Este patrón o modelo arquitectónico tiene una influencia positiva en el criterio de calidad [Buschmann et. al 1996] de adaptabilidad, pero influye negativamente en los criterios de rendimiento y escalabilidad.

- El **modelo cliente-servidor** estructura una aplicación basada en servicios en un servidor y los servicios clientes. Los servicios interactúan con los clientes requiriendo la funcionalidad desde los servidores. Los clientes y servidores pueden residir en procesos separados y en máquinas separadas, comunicándose a través de mensajes.

  Este patrón o modelo arquitectónico tiene una influencia positiva en los criterios de calidad [Buschmann et. al 1996] de adaptabilidad, interoperabilidad, escalabilidad y seguridad. Sin embargo, influye negativamente en los criterios de rendimiento y disponibilidad.

- El **modelo en estrella** (*hub-and-spoke*) define un componente software mediador, denominado *hub*, que actúa como coordinador activo de diferentes componentes software externos, a los cuales llama. El mediador combina la funcionalidad de otros servicios y proporciona un interfaz a los clientes.

Este patrón o modelo arquitectónico tiene una influencia positiva en los criterios de calidad [Buschmann et. al 1996] de confiabilidad, disponibilidad, seguridad, escalabilidad, adaptabilidad y facilidad de uso. Sin embargo, influye negativamente en la robustez de la solución.

Teniendo en cuenta los requisitos arquitectónicos previamente identificados, el repositorio de experiencias de la organización y el análisis de los patrones y modelos arquitectónicos, el arquitecto de sistemas identifica que el modelo cliente-servidor no es adecuado para la solución requerida ya que influye negativamente en la disponibilidad, que ha sido identificada como un requisito arquitectónico a cumplir por la composición de servicios. El resto de patrones y modelos arquitectónicos son priorizados en el siguiente orden: modelo en estrella, modelo "pizarra" y modelo por capas. El arquitecto de sistemas identifica el modelo en estrella como el más apropiado a aplicar a la solución a desarrollar.

Una vez hecho esto, el arquitecto de sistema tiene que definir las políticas arquitectónicas que delimitarán las decisiones relativas a la arquitectura, diseño, implementación y despliegue de la solución. Dentro de estas políticas arquitectónicas se recogen no sólo las políticas referentes a los requisitos arquitectónicos, sino también políticas generales de la organización.

- ***Política 1***: La disponibilidad se consigue a través de una replicación. En este caso, dichas réplicas serán de estado persistente, el control de réplicas se realizará de manera pesimista y la consistencia será descentralizada.

- ***Política 2***: Para asegurar el control de acceso, hay que gestionar la autenticación y autorización del usuario al sistema. En cuanto a la autenticación se hará uso de una contraseña personal; y en cuanto a la autorización será individual.

- ***Política 3***: A nivel de organización está definido que todos los interfaces deben de ser compatibles con WS-I [WS-I 2007].

- ***Política 4***: A nivel de organización está definido que todas las llamadas a servicios externos deben utilizar WS-Security [WS-SEC 2006].

Especificación Funcional

Tras haber realizado la especificación arquitectural de la composición de servicios, los dos analistas del sistema del grupo XService empiezan con el proceso de la especificación funcional. Para ello, usando como base los requisitos del sistema, los analistas del sistema comienzan el análisis definiendo qué información necesita recibir el servicio compuesto y cuál es la funcionalidad de dicho servicio compuesto formalizada a través de un diagrama UML de casos de uso.

El servicio compuesto XTrip necesita recibir como parámetros de entrada:

- Los datos del destino al que se desea llegar, en términos de longitud y latitud o con el nombre de la ciudad; y los datos de la posición actual del usuario, los cuales pueden ser extraídos del sistema GPS del coche en términos de longitud y latitud.
- El sistema de gestión de agenda que utiliza el usuario, por ejemplo Microsoft Outlook o Google Docs.
- El número de teléfono del usuario y el número del teléfono con el cual hay que establecer una llamada en el caso de que se detecte un conflicto entre un viaje del usuario y otras reuniones planificadas en su agenda.

En cuanto a la funcionalidad del servicio compuesto, el diagrama de casos de uso Figura 7-1 identifica cuatro casos de uso, que se corresponden con los cuatro servicios abstractos de XTrip: 'Calcular ruta', 'Comprobar agenda', 'Actualizar agenda', 'Realizar llamada'.

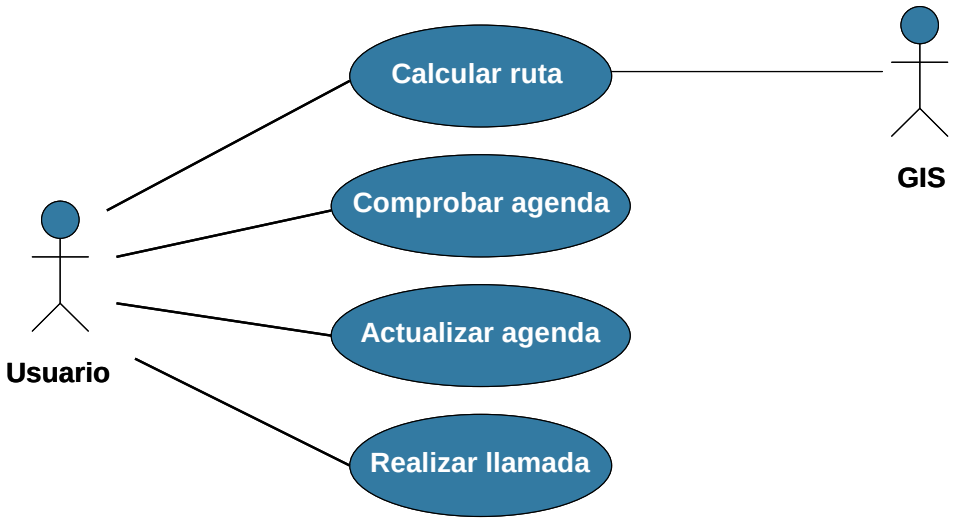
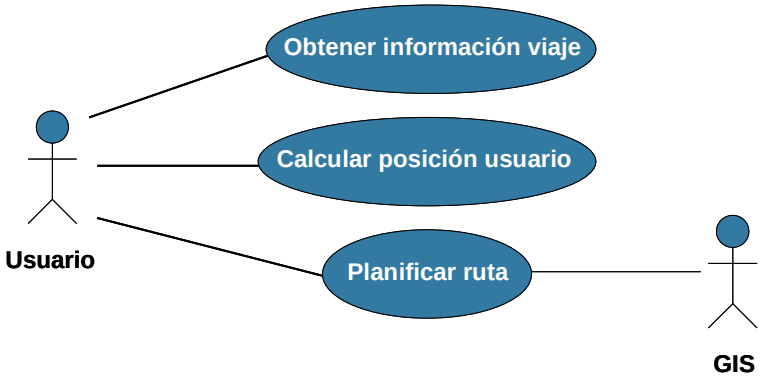
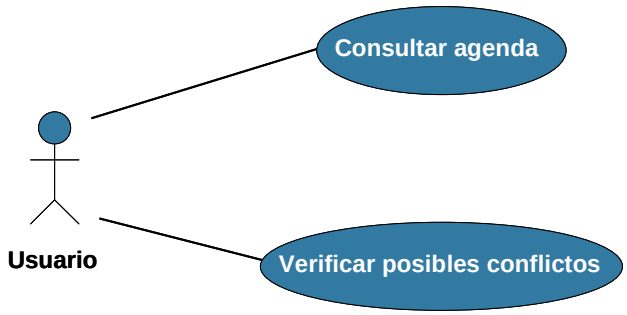


Figura 7-1: Diagrama de casos de uso del servicio compuesto XTrip

Por cada servicio abstracto los analistas del sistema proporcionan la siguiente información:

Calcular ruta	
Identificador	SA01
Propósito	Calcular la mejor ruta entre dos lugares.
Descripción	La funcionalidad proporcionada por 'Calcular ruta' se puede dividir en varias sub-funcionalidades, las cuales ayudan a comprender mejor su comportamiento. En primer lugar, es necesario recopilar toda la información necesaria sobre el destino al que hay que viajar. A continuación, se calcula la posición del usuario a través del sistema GPS incorporado en el vehículo. Con toda esta información, se planifica el viaje identificando la mejor ruta para llegar desde el punto de inicio al destino a través de un Sistema de Información Geográfica (GIS) externo.
Otros requisitos	Tiempo de respuesta bajo (menor a 5 milisegundos) y alta disponibilidad (mínimo 99,95%).
Actores involucrados	Usuario, GIS.
Dependencias	Ninguna.
Caso de uso de nivel inferior	Este caso de uso describe con mayor detalle las sub-funcionalidades que componen el servicio abstracto.  Figura 7-2: Diagrama de casos de uso de 'Calcular ruta'

Comprobar agenda	
Identificador	SA02
Propósito	Comprobar la agenda del usuario para verificar que no va a existir ningún conflicto entre las reuniones planificadas y el viaje a realizar (en términos de duración del viaje).
Descripción	La funcionalidad proporcionada por 'Comprobar agenda' se puede dividir en varias sub-funcionalidades, las cuales ayudan a comprender mejor su comportamiento. En primer lugar, es necesario consultar la agenda del usuario para ver todas las reuniones planificadas. A continuación, se verifica que no haya conflictos entre la duración del viaje y dichas reuniones ya planificadas. En caso de que haya uno o varios conflictos, se le notifican al usuario.
Otros requisitos	Control de acceso a través de autenticación del usuario y garantía de privacidad de los datos. En este caso hay una dependencia con el tipo de sistema que el usuario utiliza para gestionar su agenda personal.
Actores involucrados	Usuario.
Dependencias	Ninguna.
Caso de uso de nivel inferior	Este caso de uso describe con mayor detalle las sub-funcionalidades que componen el servicio abstracto 'Comprobar agenda'.  Figura 7-3: Diagrama de casos de uso de 'Comprobar agenda'

Actualizar agenda	
Identificador	SA03
Propósito	Actualizar la información sobre una reunión en la agenda del usuario.
Descripción	La funcionalidad proporcionada por 'Actualizar agenda' consiste únicamente en acceder a la agenda del usuario y actualizar la información relacionada con la duración del viaje a realizar. Durante dicho tiempo el usuario no estará disponible para realizar ninguna otra reunión.
Otros requisitos	Control de acceso a través de autenticación del usuario y garantía de privacidad de los datos. En este caso hay una dependencia con el tipo de sistema que el usuario utiliza para gestionar su agenda personal.
Actores involucrados	Usuario.
Dependencias	Ninguna.
Caso de uso de nivel inferior	No hay.

Realizar llamada	
Identificador	SA04
Propósito	Establecer una llamada entre dos puntos.
Descripción	Esta funcionalidad establece una conexión entre dos puntos tan pronto como el punto destino esté disponible.
Otros requisitos	Se utilizará aquel servicio que sea más barato.
Actores involucrados	Usuario.
Dependencias	Ninguna.
Caso de uso de nivel inferior	No hay.

Una vez completada la información relacionada con cada servicio abstracto, el grupo XService se reúne con el personal a nivel de negocio de la organización para decidir qué estrategia se va a seguir durante la realización de los servicios abstractos. La decisión final que se toma es que siempre que se pueda se van a consumir servicios proporcionados por terceras partes. Pero en el caso de que no se encuentren servicios concretos candidatos que implementen las especificaciones que se necesitan, se desarrollarán dichas implementaciones dentro de la propia organización.

Al mismo tiempo que se está decidiendo qué estrategia de provisión seguir, los analistas del sistema identifican el servicio web **kRecall**, proporcionado por el proveedor italiano TELCO con el que la empresa ya ha establecido ciertas condiciones sobre el mismo, ya que es un servicio concreto candidato a nivel organizacional para cubrir el servicio abstracto 'Realizar llamada'. Este servicio web **kRecall** permite establecer una llamada con un teléfono especificado tan pronto como dicho teléfono esté disponible. Dichos analistas del sistema también deciden que todavía no se van a buscar servicios concretos candidatos para los servicios abstractos hasta que se haya definido mejor el flujo de trabajo.

Composición en tiempo de diseño

El rol del analista de procesos de negocio lo desempeña uno de los analistas del sistema del grupo XService. Este analista, junto con el arquitecto del sistema decide que la mejor arquitectura a aplicar a la solución es el modelo en estrella (*hub-and-spoke*) ya que no se han identificado nuevas restricciones arquitectónicas.

Una vez definido el modelo arquitectónico a seguir, el analista de procesos clasifica las funcionalidades o servicios abstractos previamente definidos en tres grupos funcionales:

- El grupo funcional 'Planificar viaje' incluye el servicio abstracto 'Calcular ruta'.
- El grupo funcional 'Gestionar agenda' incluye los servicios abstractos 'Comprobar agenda' y 'Actualizar agenda'.
- El grupo funcional 'Gestionar llamada' incluye el servicio abstracto 'Realizar llamada'.

El analista identifica las siguientes dependencias entre dichos grupos funcionales:

- 'Planificar viaje' depende de los datos que reciba del usuario del servicio compuesto.

- 'Gestionar agenda' tiene dependencia del grupo funcional 'Planificar viaje', ya que necesita información relativa a la planificación del viaje.

- 'Realizar llamada' sólo se llevará a cabo si se detecta un conflicto en la agenda; por lo tanto, depende del grupo funcional 'Gestionar agenda'.

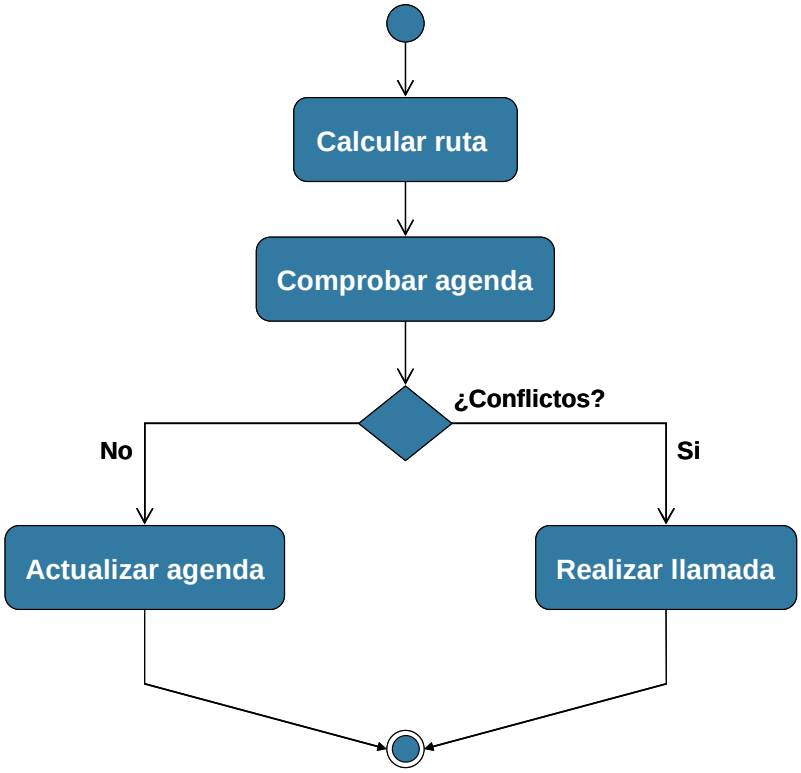


Figura 7-4: Diagrama de actividad del servicio compuesto XTrip

Teniendo en cuenta estas dependencias entre los diferentes grupos funcionales, el analista de procesos define el flujo de trabajo de la composición de servicios a través de un diagrama UML de actividad Figura 7-4.

Para finalizar la descripción del flujo de trabajo, el analista de procesos especifica varios requisitos no funcionales a nivel general de la composición y a nivel de los grupos funcionales.

- A nivel de la composición, la disponibilidad debe ser como mínimo de un 99,95 por ciento.

- A nivel de la composición, se requiere un control de acceso que será realizado a través de la autorización y autenticación del usuario.

- A nivel del grupo funcional 'Gestionar agenda', se requiere una autenticación del usuario para acceder a su servicio de agenda personal.

Después de definir el flujo de trabajo de la composición de servicios, hay que identificar si algunas partes de dicho flujo de trabajo podrían variar en tiempo de ejecución. En este caso el analista de procesos identifica tres puntos de variación y define la siguiente información relacionada:

- El grupo funcional 'Gestionar agenda' depende del sistema de gestión de agenda personal que utilice el usuario que consuma el servicio. Por ello, los servicios abstractos 'Comprobar agenda' y 'Actualizar agenda' tienen asociado un punto de variación respectivamente.

Consultar Agenda Personal	
Identificador	PV01
Identificador servicio abstracto	SA02
Propósito	Permite acceder a la agenda personal del usuario para consultar la información guardada.
Descripción	Este punto de variación seleccionará aquel servicio concreto que permita acceder a la agenda personal del usuario para consultar la información almacenada. Son varios los sistemas de gestión de agendas personales existentes en el mercado. En esta primera aproximación del servicio XTrip, el equipo XService sólo se va a enfocar en dos de los sistemas más utilizados: Microsoft Outlook y Google Docs. En un futuro, se podrían añadir más sistemas incluyendo más alternativas a este punto de variación.
Condición	Nombre del sistema de gestión de agenda personal.
Tipo datos	String (cadena de caracteres).

Alternativas	

PV01-Microsoft	
Identificador	PV01-A01
Propósito	Permite consultar la agenda personal del usuario gestionada a través de Microsoft Outlook.
Descripción	Se asocia el servicio concreto que permite acceder a agendas personales de Microsoft Outlook.
Valor	Microsoft Outlook

PV01-Google	
Identificador	PV01-A02
Propósito	Permite consultar la agenda personal del usuario gestionada a través de Google Docs.
Descripción	Se asocia el servicio concreto que permite acceder a agendas personales de Google Docs.
Valor	Google Docs

Modificar Agenda Personal	
Identificador	PV02
Identificador servicio abstracto	SA02
Propósito	Permite acceder a la agenda personal del usuario para modificar la información guardada añadiendo nuevas reuniones o modificando reuniones ya planificadas.
Descripción	Este punto de variación seleccionará aquel servicio concreto que permita acceder a la agenda personal del usuario para

<table>
<tr><td></td><td>modificar la información almacenada o añadir nuevas reuniones. Al igual que en el punto de variación PV01, el equipo XService sólo se va a enfocar en dos de los sistemas más utilizados: Microsoft Outlook y Google Docs.</td></tr>
<tr><td>Condición</td><td>Nombre del sistema de gestión de agenda personal.</td></tr>
<tr><td>Tipo datos</td><td>String (cadena de caracteres).</td></tr>
<tr><td>Alternativas</td><td>
<table>
<tr><th colspan="2">PV02-Microsoft</th></tr>
<tr><td>Identificador</td><td>PV02-A01</td></tr>
<tr><td>Propósito</td><td>Permite modificar los datos de la agenda personal del usuario gestionada a través de Microsoft Outlook.</td></tr>
<tr><td>Descripción</td><td>Se asocia el servicio concreto que permite acceder a agendas personales de Microsoft Outlook.</td></tr>
<tr><td>Valor</td><td>Microsoft Outlook</td></tr>
</table>
<table>
<tr><th colspan="2">PV02-Google</th></tr>
<tr><td>Identificador</td><td>PV02-A02</td></tr>
<tr><td>Propósito</td><td>Permite modificar los datos de la agenda personal del usuario gestionada a través de Google Docs.</td></tr>
<tr><td>Descripción</td><td>Se asocia el servicio concreto que permite acceder a agendas personales de Google Docs.</td></tr>
<tr><td>Valor</td><td>Google Docs</td></tr>
</table>
</td></tr>
</table>

- El servicio abstracto ‘Realizar llamada’ tiene asociado un requisito no funcional que define que hay que usar siempre el servicio concreto que ofrezca el precio más barato.

Proveedor Llamadas	
Identificador	PV03
Identificador servicio abstracto	SA04
Propósito	Establece una conexión entre dos puntos siempre utilizando el servicio concreto con el precio más barato.
Descripción	Este punto de variación seleccionará el servicio concreto proporcionado por un proveedor de telecomunicaciones que sea más barato. Para ello, utilizará como referencia el prefijo del número de teléfono al que hay que llamar.
Condición	Valor del prefijo del número a llamar.
Tipo datos	Integer.
Alternativas	

PV03-kRecall	
Identificador	PV03-A01
Propósito	Utiliza el servicio concreto kRecall.
Descripción	Si el prefijo es 39, asocia el servicio concreto kRecall.
Valor	39

PV03-BuscarBarato	
Identificador	PV03-A02
Propósito	Busca el servicio concreto más barato disponible.
Descripción	Busca dinámicamente el servicio concreto que satisfaga la funcionalidad 'Realizar llamada' y sea el más barato.
Valor	Distinto de 39

Realización de la composición

En este punto, una vez diseñado el flujo de trabajo y los diferentes puntos de variación identificados, hay que comenzar a buscar servicios concretos candidatos a cubrir las funcionalidades definidas por los servicios abstractos previamente descritos. Así, el analista de procesos, apoyado por el grupo de desarrolladores de XService realizan varias búsquedas en diferentes buscadores de servicios, registros y a través de contactos directos con proveedores de servicios. Con los resultados obtenidos, se crea una lista completa de todos los servicios concretos candidatos por cada servicio abstracto, incluyendo también los servicios concretos candidatos desarrollados por la organización y los pre-seleccionados durante la fase de Análisis.

- 'Calcular ruta': *RouteService* proporcionado por *Mappoint Web Services* de *Microsoft*, *RoutingFinder* ofrecido por *ArcWeb* y *RouteCalculation* ofrecido por *ViaMichelin*.

- 'Comprobar agenda': No existen servicios concretos que permitan acceder a las funcionalidades de las agendas personales de Microsoft Outlook o Google Docs. Sin embargo, existen dos APIs (*Exchange Server 2007 Web Service API* y *Google Docs API*) que facilitan el desarrollo de servicios web para acceder a dichos sistemas de gestión de agendas personales respectivamente. Es decir, el equipo XService tiene que desarrollar un servicio concreto que implemente el servicio abstracto 'Comprobar agenda'.

- 'Actualizar agenda': Ocurre lo mismo que con el servicio abstracto 'Comprobar agenda'. El equipo XService tiene que desarrollar un servicio concreto que implemente el servicio abstracto 'Actualizar agenda' para los sistemas de gestión de agendas personales de Microsoft Outlook y Google Docs.

- 'Realizar llamada': Servicio concreto kRecall proporcionado por TELCO.

Antes de definir las siguientes medidas, hay un requisito que todo servicio concreto candidato debe cumplir y es que proporcione una disponibilidad mayor o igual al 99,95 por ciento. Por lo tanto, todos aquellos servicios concretos que no lo cumplan deben de ser descartados de inmediato. En este caso, todos los servicios concretos candidatos lo cumplen.

Ahora, el analista de procesos tiene que definir el conjunto de medidas que se usará para evaluar los diferentes servicios concretos candidatos no sólo a nivel funcional sino también teniendo en cuenta aspectos o requisitos no funcionales. En este caso, sólo es necesario evaluar los servicios concretos candidatos del servicio abstracto 'Calcular ruta'; ya que los servicios abstractos 'Comprobar agenda' y 'Actualizar agenda' implican un desarrollo, y el servicio abstracto 'Realizar llamada' ya tiene asociado un servicio concreto candidato predefinido que ya satisface los requisitos establecidos.

Como el método utilizado para puntuar los diferentes servicios concretos candidatos es el sistema de puntuación definido en SELF-SERV, que consiste en una selección en función de varios atributos teniendo en cuenta la siguiente fórmula:

$$U(s) = \sum_{i \in SA} w_i \bullet Score_i(s)$$

donde:

- U(s) indica el valor de la función de utilidad para el servicio s,
- $Score_i(s)$ es una función que, dado el valor de un atributo i del servicio s devuelve un numero entero positivo. SA indica el conjunto de atributos seleccionados, como por ejemplo el precio, el tiempo de ejecución o la fiabilidad,
- wi indica el peso w asignado al atributo i.

MEDIDA	PESO	FUNCIÓN DE PUNTUACIÓN
Tiempo de respuesta	40	Si tiempo es R (s), entonces Score(R (s)) = 1/R (s), donde a mayor tiempo, más baja puntuación.
Países sobre los que proporcionan mapas	25	Si países es M (s), entonces Score(M (s)) = M (s), donde a mayor número de países, mayor puntuación.
Precio	25	Si precio es P (s), entonces Score(P (s)) = 1/P (s), donde a mayor precio, más baja puntuación.
Información en tiempo real sobre el tráfico	10	Si tráfico es T (s), entonces si se proporciona información de tráfico Score(T (s)) = 1, sino Score(T (s)) = 0.

Tabla 7-1: Medidas de puntuación para los servicios concretos candidatos

Así es posible establecer la puntuación de cada servicio y seleccionar aquel servicio con mayor puntuación obtenida en la función de utilidad multi-atributo. En la Tabla 7-1, el analista de procesos recoge las medidas definidas así como el peso y la función de puntuación asociadas a cada medida.

Tras aplicar la fórmula para calcular la puntuación final, las puntuaciones obtenidas por los tres servicios concretos candidatos del servicio abstracto 'Calcular ruta' ordenadas de mayor a menor son las siguientes:

- *RouteService* de *Mappoint Web Services Microsoft*: **1335**
- *RouteCalculation* de *ViaMichelin*: **1155**
- *RoutingFinder* de *ArcWeb*: **672**

Una vez puntuados los servicios concretos candidatos, los desarrolladores del equipo XService se encargan de desarrollar los cuatro servicios concretos que van a implementar las especificaciones 'Comprobar agenda' y 'Actualizar agenda' utilizando las APIs de Microsoft Outlook y Google Docs.

Tras terminar el desarrollo de estos servicios concretos, el analista de procesos comienza a definir las asociaciones entre los servicios abstractos y los servicios concretos y a detallar sus correspondientes operaciones y parámetros.

- 'Calcular ruta': Como se puede observar, el servicio concreto ofrecido por *ArcWeb* ha obtenido una puntuación muy baja en comparación con los otros servicios concretos candidatos por lo que el analista de procesos decide eliminarlo de la lista.

  Así, el servicio concreto que se va a asociar al servicio abstracto 'Calcular ruta' es el servicio ofrecido por Microsoft. Sin embargo, el servicio concreto ofrecido por ViaMichelin se usará como respaldo en el caso de que el servicio proporcionado por Microsoft falle.

R01-Servicio Candidato 1	
Nombre	RouteService
Proveedor	Microsoft
Propósito	El objetivo de este servicio web es proporcionar una ruta entre dos puntos proporcionados por el usuario. Además de la ruta, este servicio web también proporciona un conjunto de indicaciones y el tiempo que duraría el viaje.
Dirección	http://staging.mappoint.net/standard-30/mappoint.wsdl
Operación	CalculateRoute
Parámetros de entrada	El parámetro de entrada es de tipo *RouteSpecification*, el cual está compuesto a su vez de cuatro parámetros: - *Segments*: Define la localización a donde se quiere llegar y si se quiere obtener la ruta más rápida o más corta. - *ResultMask*: Describe si se quiere obtener simplemente el itinerario o también la ruta calculada en un mapa. - *DataSourceName*: Define el punto de partida. - *DriverProfile*: Especifica las fechas de salida y llegada deseadas, así como los caminos preferentes.
Parámetros de salida	El parámetro de salida es de tipo *Route*, el cual está compuesto a su vez de tres parámetros: - *RouteSpecification*: Recoge toda la información de entrada proporcionada al llamar al servicio web. - *Itinerary*: Describe el itinerario de la ruta, incluyendo otros aspectos importante como la duración del viaje y la distancia. - *CalculatedRepresentation*: Proporciona la representación gráfica de la ruta.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,99 por ciento y un tiempo de respuesta de un milisegundo.

R01-Servicio Candidato 2	
Nombre	RouteCalculation
Proveedor	ViaMichelin
Propósito	El objetivo de este servicio web es calcular una ruta y/o una representación gráfica de dicha ruta.
Dirección	http://webservices.viamichelin.com/ws2/services/RouteCalculation?wsdl
Operación	getRoute
Parámetros de entrada	El parámetro de entrada está compuesto de dos parámetros: - *Request*: Contiene la especificación de los datos de la ruta a calcular, que serán utilizados para especificar el itinerario y la indicaciones oportunas. - *authenticationParams*: Especifica un nombre de usuario y contraseña requeridos para acceder al servicio web.
Parámetros de salida	El parámetro de salida es de tipo *ItineraryResult*, el cual contiene una lista de instrucciones asociadas a una ruta, detalles sobre el viaje como la duración y la distancia; y también puede incluir una representación gráfica de dicha ruta.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,97 por ciento y un tiempo de respuesta de entre uno y dos milisegundos.

- 'Comprobar agenda': Los datos de los dos servicios concretos desarrollados son:

R02-Servicio Candidato 1	
Nombre	XAgendaMicrosoft
Proveedor	XService
Propósito	El objetivo de este servicio es facilitar la gestión de la agenda personal del usuario gestionada a través de Microsoft

	Outlook.
Dirección	http://localhost:8080/axis/services/ XAgendaMicrosoft
Operación	checkConflict
Parámetros de entrada	El parámetro de entrada está compuesto de tres parámetros: - *Usuario*: Nombre del usuario de la agenda. - *Request*: Contiene la fecha y hora de inicio del viaje. - *Duration*: Recoge la duración del viaje.
Parámetros de salida	El parámetro de salida es de tipo *string* y define si hay o no uno o varios conflictos en la agenda.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,95 por ciento y un tiempo de respuesta de entre uno y dos segundos.

R02-Servicio Candidato 2	
Nombre	XAgendaGoogle
Proveedor	XService
Propósito	El objetivo de este servicio es facilitar la gestión de la agenda personal del usuario gestionada a través de Google Docs.
Dirección	http://localhost:8080/axis/services/XAgendaGoogle
Operación	checkConflict
Parámetros de entrada	El parámetro de entrada está compuesto de tres parámetros: - *Usuario*: Nombre del usuario de la agenda. - *Request*: Contiene la fecha y hora de inicio del viaje. - *Duration*: Recoge la duración del viaje.
Parámetros de salida	El parámetro de salida es de tipo *string* y define si hay o no algún conflicto en la agenda.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,95 por ciento y un tiempo de respuesta de entre uno y dos segundos.

- 'Actualizar agenda': Los datos de los dos servicios concretos desarrollados son los siguientes:

R03-Servicio Candidato 1	
Nombre	XAgendaMicrosoft
Proveedor	XService
Propósito	El objetivo de este servicio es facilitar la gestión de la agenda personal del usuario gestionada a través de Microsoft Outlook.
Dirección	http://localhost:8080/axis/services/ XAgendaMicrosoft
Operación	confirmCommitment
Parámetros de entrada	El parámetro de entrada está compuesto de tres parámetros: - *Usuario*: Nombre del usuario de la agenda. - *Start*: Contiene la fecha y hora de inicio del viaje. - *End*: Contiene la fecha y hora de fin del viaje.
Parámetros de salida	El parámetro de salida es de tipo *booleano* y define si la actualización de la agenda con el nuevo viaje ha sido realizada correctamente.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,95 por ciento y un tiempo de respuesta de entre uno y dos segundos.

R03-Servicio Candidato 2	
Nombre	XAgendaGoogle
Proveedor	XService
Propósito	El objetivo de este servicio es facilitar la gestión de la agenda personal del usuario gestionada a través de Google Docs.
Dirección	http://localhost:8080/axis/services/XAgendaGoogle
Operación	confirmCommitment

Parámetros de entrada	El parámetro de entrada está compuesto de tres parámetros: - *Usuario*: Nombre del usuario de la agenda. - *Start*: Contiene la fecha y hora de inicio del viaje. - *End*: Contiene la fecha y hora de fin del viaje.
Parámetros de salida	El parámetro de salida es de tipo *booleano* y define si la actualización de la agenda con el nuevo viaje ha sido realizada correctamente.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,95 por ciento y un tiempo de respuesta de entre uno y dos segundos.

- 'Realizar llamada': Los datos del servicio concreto kRecall proporcionado por TELCO son:

R04-Servicio Candidato 1	
Nombre	kRecall
Proveedor	TELCO
Propósito	El objetivo de este servicio es realizar llamadas a través de redes de telecomunicaciones usando un interfaz SIP.
Dirección	http://163.162.173.70:8001/WsGateway/services/ppxSipTpc
Operación	makeCall
Parámetros de entrada	El parámetro de entrada está compuesto de dos parámetros: - *CallingParty*: Contiene la información sobre el usuario que desea realizar la llamada. - *CalledParty*: Contiene la información sobre el usuario que al que se desea llamar.
Parámetros de salida	El parámetro de salida es de tipo *string* y define un identificador de llamada.
Comentarios	Este servicio concreto garantiza una disponibilidad del 99,97 por ciento y un tiempo de respuesta mínimo de un segundo.

	El tiempo máximo no está especificado porque este servicio concreto es un servicio de prueba desarrollado para este escenario.

Como se ha definido variabilidad en este servicio compuesto, hay que concretar las acciones asociadas a las diferentes variantes de cada punto de variación especificado.

- 'PV01-Consultar agenda personal' tiene asociadas dos variantes. PV01-Microsoft define como acción asociar el servicio concreto '**R02-Servicio Candidato 1'** a la composición. PV01-Google define como acción asociar el servicio concreto '**R02-Servicio Candidato 2'** a la composición.

- 'PV02-Modificar agenda personal' tiene asociadas dos alternativas. PV02-Microsoft define como acción asociar el servicio concreto '**R03-Servicio Candidato 1'** a la composición. PV02-Google define como acción asociar el servicio concreto '**R03-Servicio Candidato 2'** a la composición.

- 'PV03-Proveedor llamadas' tiene asociadas dos alternativas. PV03-kRecall define como acción asociar el servicio concreto '**R04-Servicio Candidato 1'** a la composición. PV03-BuscarBarato define como acción una petición de búsqueda dinámica de aquel servicio concreto que satisfaga la funcionalidad 'Realizar llamada' y además ofrezca el precio más barato.

PV03-Buscar Barato - Petición de Búsqueda	
Descripción	Petición de búsqueda del servicio concreto de realizar llamadas más barato.
Petición	Buscar dinámicamente servicios concretos que: - Satisfagan la funcionalidad 'Realizar llamada' - Ofrezca el precio más bajo - Tenga como parámetros de entrada el teléfono del usuario y el teléfono al que se desee llamar.
Comentarios	Ninguno.

Finalmente, hay que definir la calidad de servicio de la composición de servicios. Para ello, el analista de procesos utiliza el enfoque de *Web Services Flow Graph* [Hu 2003] para calcular el nivel de calidad ofrecido. Suponiendo un ratio de visita relativo igual para todos los nodos y una latencia de la red sobre cero, el equipo XService garantiza que, como mínimo, y a modo general, el servicio XTrip proporciona una disponibilidad del 99,95 por ciento y un tiempo de respuesta de cinco segundos.

Generación de la composición ejecutable

Una vez terminado el diseño de la composición, el grupo de desarrolladores se encargan de generar los modelos ejecutables tanto de la composición como de la variabilidad definida. En el Anexo 3 está recogido el código generado de dichos modelos ejecutables de la composición y de la variabilidad.

Validación y Verificación

Por último, el responsable de pruebas se encarga de validar y verificar el servicio compuesto XTrip con el fin de asegurar que proporciona la funcionalidad deseada y que la comunicación entre los servicios concretos utilizados no genera ningún resultado inesperado. Para ello, se realiza una validación y verificación del servicio compuesto desde dos perspectivas.

Desde un punto de vista de la funcionalidad y rendimiento del sistema, se va a validar y verificar el servicio compuesto definiendo casos de prueba enfocados, principalmente, en simular accesos múltiples simultáneos al sistema y poder observar cómo responde el sistema a peticiones que requieren gran cantidad de información y, así, comprobar si la integridad de los datos se podría ver afectada. En esta validación y verificación del servicio compuesto XTrip han participado también usuarios finales que han consumido el servicio.

Desde un punto de vista de la dinamicidad del servicio compuesto, se va a validar y verificar el servicio compuesto definiendo cinco casos de prueba cuyo objetivo es comprobar que el servicio compuesto es capaz de adaptarse a las condiciones del entorno de ejecución.

Estos casos de prueba realizados y los resultados obtenidos son los siguientes:

- **Caso de Prueba 1: No hay conflictos en la agenda.**

  Los parámetros de entrada de este caso de prueba son:

  - Origen: Milán (45.25, 9.12)
  - Destino: Orbassano (45.01, 7.68)
  - Gestor de agenda personal: Microsoft
  - Teléfono origen: +390123456789
  - Teléfono destino: +391234567890

  Durante la ejecución del servicio compuesto XTrip, y teniendo en cuenta que el gestor de la agenda del usuario es Microsoft, en los puntos de variación definidos en los servicios abstractos 'Comprobar_Agenda' y 'Actualizar_Agenda' se selecciona la primera alternativa, que asocia el servicio concreto *XAgendaMicrosoft*.

  El resultado obtenido es una actualización en la agenda personal del usuario, ya que no se detecta ningún conflicto. En la Figura 7-5 se muestra un resumen de las asociaciones que se han realizado entre los servicios abstractos y los servicios concretos.

Current association activity-concrete service

Activity name	Concrete service endpoint	Bind Date
Calcular_Ruta	http://localhost:8080/axis/services/mappoint	2008-03-20 11:04:28
Comprobar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:04:29
Actualizar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:04:31

Figura 7-5: Asociación de servicios abstractos-concretos para el caso de prueba 1

- **Caso de Prueba 2: Hay conflictos en la agenda.**

  Los parámetros de entrada de este caso de prueba son:

  - Origen: Milán (45.25, 9.12)
  - Destino: Niza (43.39, 7.12)
  - Gestor de agenda personal: Microsoft
  - Teléfono origen: +390123456789

- Teléfono destino: +391234567890

Durante la ejecución del servicio compuesto XTrip, y teniendo en cuenta que el gestor de la agenda del usuario es Microsoft, en el punto de variación definido en el servicio abstracto 'Comprobar_Agenda' se selecciona la primera alternativa, que asocia el servicio concreto *XAgendaMicrosoft*. Al haber un conflicto con reuniones planificadas en la agenda personal del usuario, hay que realizar la llamada al número indicado. Dicho número comienza por 39, por lo que del punto de variación definido para el servicio abstracto 'Realizar_Llamada' se selecciona la primera alternativa, asociando el servicio concreto *kRecall*.

El resultado obtenido es la realización de una llamada al número indicado. Un resumen de las asociaciones que se han realizado entre los servicios abstractos y los servicios concretos es mostrado en la Figura 7-6.

Current association activity-concrete service

Activity name	Concrete service endpoint	Bind Date
Calcular_Ruta	http://localhost:8080/axis/services/mappoint	2008-03-20 11:08:01
Comprobar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:08:02
Realizar_Llamada	http://163.162.173.70/WsGateway/services/ppxSipTpc	2008-03-20 11:08:04

Figura 7-6: Asociación de servicios abstractos-concretos para el caso de prueba 2

- **Caso de Prueba 3: Hay conflictos en la agenda.**

Los parámetros de entrada de este caso de prueba son:

- Origen: Milán (45.25, 9.12)
- Destino: Niza (43.39, 7.12)
- Gestor de agenda personal: Microsoft
- Teléfono origen: +390123456789
- Teléfono destino: +341234567890

Durante la ejecución del servicio compuesto XTrip, y teniendo en cuenta que el gestor de la agenda del usuario es Microsoft, en el punto de variación definido en el servicio abstracto 'Comprobar_Agenda' se selecciona la primera alternativa, que asocia el servicio concreto *XAgendaMicrosoft*. Al haber un conflicto con reuniones planificadas

en la agenda personal del usuario, hay que realizar la llamada al número indicado. Dicho número comienza por 34, por lo que del punto de variación definido para el servicio abstracto 'Realizar_Llamada' se selecciona la segunda alternativa, que consiste en buscar dinámicamente aquel servicio concreto más barato. En este caso se asocia un servicio concreto llamado *ThirdPartyCall1*.

El resultado obtenido es la realización de una llamada al número indicado. Un resumen de las asociaciones que se han realizado entre los servicios abstractos y los servicios concretos se muestra en la Figura 7-7.

Current association activity-concrete service		
Activity name	Concrete service endpoint	Bind Date
Calcular_Ruta	http://localhost:8080/axis/services/mappoint	2008-03-20 11:12:08
Comprobar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:12:09
Realizar_Llamada	http://localhost:8080/axis/services/ThirdPartyCall1	2008-03-20 11:12:11

Figura 7-7: Asociación de servicios abstractos-concretos para el caso de prueba 3

- **Caso de Prueba 4: No hay conflictos en la agenda.**

  Los parámetros de entrada de este caso de prueba son:

  - Origen: Niza (43.39, 7.12)
  - Destino: Turin (45.07, 7.67)
  - Gestor de agenda personal: Google
  - Teléfono origen: +390123456789
  - Teléfono destino: +391234567890

  Durante la ejecución del servicio compuesto XTrip, al invocar al servicio concreto de *mappoint* el resultado obtenido es erróneo (en este caso, menor a cero) por lo que el sistema lanza un evento de comportamiento erróneo, clasifica dicho servicio concreto dentro de la lista negra de servicios concretos a no utilizar y selecciona al servicio concreto de *viamichelin* para asociarlo a 'Calcular_Ruta'.

  Además, teniendo en cuenta que el gestor de la agenda del usuario es Google, en los puntos de variación definidos en los servicios abstractos 'Comprobar_Agenda' y

'Actualizar_Agenda' se selecciona la segunda alternativa, que asocia el servicio concreto *XAgendaGoogle*.

El resultado obtenido es una actualización en la agenda personal del usuario, ya que no se detecta ningún conflicto. Un resumen de las asociaciones que se han realizado entre los servicios abstractos y los servicios concretos es mostrado en la Figura 7-8.



Current association activity-concrete service

Activity name	Concrete service endpoint	Bind Date
Calcular_Ruta	http://localhost:8080/axis/services/viamichelin	2008-03-20 11:19:56
Comprobar_Agenda	http://localhost:8080/axis/services/XAgendaGoogle	2008-03-20 11:19:57
Actualizar_Agenda	http://localhost:8080/axis/services/XAgendaGoogle	2008-03-20 11:19:59

Figura 7-8: Asociación de servicios abstractos-concretos para el caso de prueba 4

- **Caso de Prueba 5: No hay conflictos en la agenda**

  Los parámetros de entrada de este caso de prueba son:

  - Origen: Milán (45.25, 9.12)
  - Destino: Orbassano (45.01, 7.68)
  - Gestor de agenda personal: Microsoft
  - Teléfono origen: +390123456789
  - Teléfono destino: +391234567890

  Durante la ejecución del servicio compuesto XTrip, como el servicio concreto de *mappoint* está clasificado en la lista negra, se asocia el servicio concreto proporcionado por *viamichelin* al servicio abstracto 'Calcular_Ruta'. Además, como el gestor de la agenda del usuario es Microsoft, en los puntos de variación definidos en los servicios abstractos 'Comprobar_Agenda' y 'Actualizar_Agenda' se selecciona la primera alternativa, que asocia el servicio concreto *XAgendaMicrosoft*.

  El resultado obtenido es una actualización en la agenda personal del usuario, ya que no se detecta ningún conflicto. Un resumen de las asociaciones que se han realizado entre los servicios abstractos y los servicios concretos es mostrado en la Figura 7-9.

Current association activity-concrete service		
Activity name	Concrete service endpoint	Bind Date
Calcular_Ruta	http://localhost:8080/axis/services/viamichelin	2008-03-20 11:23:40
Comprobar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:23:41
Actualizar_Agenda	http://localhost:8080/axis/services/XAgendaMicrosoft	2008-03-20 11:23:42

Figura 7-9: Asociación de servicios abstractos-concretos para el caso de prueba 5

7.1.3 Resultado de la Validación de la Metodología de Composición

La validación recogida en la sección anterior es sólo una de las diferentes validaciones que hemos llevado a cabo junto a diversos usuarios finales sobre la aplicación de la metodología de composición de servicios en escenarios industriales reales. Es importante resaltar que estas validaciones que hemos ido realizando durante el desarrollo de la metodología nos han permitido refinarla y mejorarla hasta llegar a la versión final explicada en el capítulo 5. Para recoger las valoraciones de los usuarios finales, se les proporcionó un cuestionario cuyo objetivo era evaluar el valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de la metodología en los diferentes escenarios industriales reales.

Para definir estas valoraciones, se ha aplicado la escala Likert [Likert 1932], donde 1 equivale a 'terrible', 2 a 'malo', 3 a 'medio', 4 a 'bueno' y 5 a 'óptimo'; con el objetivo de establecer una guía común a la hora de que los usuarios finales proporcionasen sus valoraciones. Los resultados medios obtenidos en la última validación, realizada en Enero del año 2008, se recogen en la Tabla 7-2. Los resultados medios obtenidos varían en un rango entre 4,5 y 4,9 por lo que podemos extraer que la media de la evaluación es una puntuación de 4,77 que según la escala de Likert corresponde a '**óptimo**'.

Como conclusión, los resultados de las validaciones realizadas a través de aplicaciones en escenarios industriales reales revelan y proporcionan una clara evidencia de que la metodología de composición de servicios propuesta puede ser usada para el diseño y desarrollo de composiciones de servicios dinámicas y sensibles al contexto de ejecución, obteniendo un resultado satisfactorio que cumple las expectativas deseadas y se adapta a las características y a la información del contexto de ejecución.

CRITERIO	PREGUNTA	RESULTADO
Valor práctico	La metodología permite diseñar y desarrollar composiciones de servicios dinámicas y sensibles al contexto de ejecución.	4,9
Facilidad de uso	La metodología es intuitiva y fácil de usar.	4,7
	La metodología incluye una descripción detallada de los procesos y tareas definidos.	4,8
	La metodología incluye algún tipo de soporte a su aplicación, como plantillas, buenas prácticas o herramientas.	4,9
Compatibilidad	Es posible utilizar la metodología para mejorar composiciones ya existentes e incluso añadirles variabilidad.	4,5
Aplicabilidad	Se pueden diseñar y desarrollar composiciones de servicios estáticas.	4,9
	Se pueden diseñar y desarrollar composiciones dinámicas, capaces de adaptarse al contexto de ejecución.	4,9

Tabla 7-2: Cuestionario de valoración de la metodología

Además, el ejemplo de aplicación de la metodología recogido en esta sección también ilustra la madurez y la aplicabilidad de la metodología, y clarifica el modo en que dicha metodología puede ser aplicada para el diseño y desarrollo de composiciones de servicios.

7.2 Evaluación de la Metodología Propuesta

El valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de la metodología de composición de servicios propuesta en esta tesis han sido examinadas mediante la validación a través de diferentes escenarios industriales reales, uno de los cuales ha sido descrito en la sección 7.1.

Sin embargo, sigue siendo necesario evaluar la metodología propuesta para comprobar que es una metodología completa y determinar si cubre todos los casos que una metodología enfocada en el desarrollo de sistemas debería cubrir.

Es importante resaltar que dicha evaluación también va a ayudar a mejorar la comprensión crítica de la metodología propuesta.

7.2.1 Evaluación

Las tablas 7-2, 7-3, 7-4, 7-5 y 7-6 presentan preguntas y respuestas relacionadas con los tres elementos esenciales definidos por NIMSAD: la situación del problema, el solucionador del problema y el proceso solucionador del problema.

Estas preguntas y respuestas constituyen colectivamente la evaluación de la metodología de composición de servicios de acuerdo al marco de trabajo NIMSAD.

PREGUNTAS Y RESPUESTAS SOBRE "LA SITUACIÓN DEL PROBLEMA"
P: ¿Quiénes son los clientes? R: La información sobre los clientes se debe recoger el documento de Requisitos del Sistema, definido en la fase de Captura de Requisitos.
P: ¿Cómo se conduce a los clientes a través del proceso de resolución del problema? R: A través de la adopción de un proceso iterativo e incremental.
P: ¿Cuáles son las necesidades de los clientes en relación a la situación del problema? R: Estas necesidades se recogen el documento de Requisitos del Sistema, definido en la fase de Captura de Requisitos.
P: ¿A qué tipos de situación del problema se enfrentan los usuarios de la metodología? R: Diseño y desarrollo de composiciones de servicios tanto estáticas como dinámicas.
P: ¿Cuál es la situación del problema para la que la metodología dice ser apropiada? R: Diseño y desarrollo de composiciones de servicios tanto estáticas como dinámicas.
P: ¿Qué evidencias y explicaciones se ofrecen? R: Guías específicas, aplicación del estándar UML para el modelado y diseño de la solución, validación y verificación del servicio compuesto generado.

P: ¿Cuáles son las características de la situación del problema? R: Se recogen en el documento de Requisitos del Sistema.
P: ¿Qué tipo de requisitos especiales pueden afectar a la metodología? R: Políticas propias del usuario, incluidas en el proceso Especificación Arquitectural.
P: ¿Qué tipo de 'realidad' percibe el cliente? R: La que está descrita en el documento de Requisitos del Sistema.

Tabla 7-3: Preguntas y respuestas sobre "la Situación del Problema"

PREGUNTAS Y RESPUESTAS SOBRE "EL SOLUCIONADOR DE PROBLEMAS"
P: ¿Se facilita la participación real de los usuarios de la metodología? R: Sí, ofreciendo unas guías específicas, discutiendo los procesos propuestos, y explicando las decisiones importantes.
P: ¿Se guía a los usuarios de la metodología para que encuentren más información sobre asuntos específicos? R: Sí. Se ofrecen referencias a través de toda la metodología.
P: ¿Es posible para los usuarios de la metodología el uso de herramientas de software apropiadas durante su trabajo? R: Sí. Pueden utilizar cualquier herramienta de modelado UML, aunque la herramienta de soporte de la metodología necesita que dicha herramienta de modelado UML pueda exportar documentos XMI versión 2.1.
P: ¿Informa la metodología a sus potenciales usuarios de la naturaleza y el tipo de los conceptos con los que deben familiarizarse antes de aplicar la metodología? R: Sí. Es una metodología de desarrollo de composición de servicios, tanto estática como dinámica y flexible. Seria beneficioso estar familiarizado con UML.
P: ¿Qué nivel de pensamiento abstracto y técnico, y qué conjuntos de conocimientos y habilidades demanda la metodología de sus usuarios antes de que puedan ponerla en práctica? R: Deberían ser desarrolladores de servicios o integradores de servicios. Seria beneficioso estar familiarizado con UML.

P: ¿Es posible para los usuarios de la metodología personalizar la metodología? R: Sí, en el capítulo 6 se definen dos ejemplos sobre cómo personalizar e incorporar la metodología de composición en el método seguido por el cliente.
P: ¿Es necesario que los usuarios de la metodología tengan experiencia en el dominio de la "situación del problema"? R: Sí. Deberían ser desarrolladores de servicios o integradores de servicios. Es una metodología especializada para la composición de servicios.

Tabla 7-4: Preguntas y respuestas sobre "el Solucionador del Problema"

PREGUNTAS Y RESPUESTAS SOBRE EL PROCESO SOLUCIONADOR DE PROBLEMAS -FORMULACIÓN DEL PROBLEMA-
Comprensión de la "situación de preocupación":
P: ¿Qué soporte ofrece la metodología para identificar a los actores independientes involucrados en la "situación del problema"? R: La identificación de los actores involucrados en la "situación del problema" se realiza en la tarea 'Especificar servicios abstractos' del proceso 'Especificación Funcional'.
P: ¿Avisa la metodología a sus usuarios de los límites del problema? R: Sí. Es un paso que hay que definir en la tarea 'Especificar servicios abstractos' del proceso 'Especificación Funcional'.
P: ¿Qué soporte ofrece la metodología para definir los límites del problema? R: La definición de los límites se realiza a través de un diagrama UML de casos de uso.
P: ¿Cuál es la posición del cliente con respecto a la metodología? R: Los clientes participan desde el comienzo, adoptando un enfoque iterativo e incremental.
P: ¿Ayuda la metodología a sus usuarios a identificar las fuentes de información? R: La identificación de las fuentes de información se realiza en la fase de Captura de Requisitos, la cual queda fuera de la metodología por ser considerada una disciplina tradicional y autónoma.
Realización del diagnóstico:

P: ¿Es necesario actualizar de forma regular las características de la "situación del problema"? R: No, se asume que éstas son estables. Sin embargo, es posible que se realicen cambios debido a la naturaleza iterativa e incremental de la metodología.
P: ¿Avisa la metodología a sus usuarios sobre la importancia de describir con precisión la "situación del problema"? R: Sí, y por ello se especifica la información mínima qué se debe proporcionar para describir con precisión la "situación del problema".
***Definición del resultado pronosticado**:*
P: ¿Qué ayuda ofrece la metodología a la definición del estado deseado con relación a la "situación del problema"? R: Una descripción precisa de la "situación del problema" y conceptos adecuados.
P: ¿Qué nociones de modelado y técnicas ofrece la metodología para expresar el estado deseado? R: El modelado del sistema se hace utilizando diagramas UML y los pasos a realizar están definidos en los procesos 'Especificación Arquitectural' y 'Especificación Funcional'.
P: ¿Cuáles son las implicaciones de malinterpretar el estado deseado? R: La validez de los requisitos del sistema se examina en los siguientes ciclos del desarrollo de la composición de servicios. Por lo tanto, cualquier problema será eventualmente descubierto y corregido.
Definición de problemas:
P: ¿Explica la metodología que es lo que está evitando alcanzar el estado deseado? R: Sí, mediante una precisa explicación del estado deseado.
P: ¿Cómo ayuda a sus usuarios a realizar tal explicación? R: A través de los procesos 'Especificación Arquitectural' y 'Especificación Funcional' de la fase de Análisis.
P: ¿Se consideran de igual importancia todos los problemas definidos? R: Sí, ya que es necesario que dichos problemas se solucionen para proporcionar la funcionalidad final del servicio compuesto.

P: ¿Es necesario definir todos los problemas desde el principio? R: No, debido a la naturaleza incremental e iterativa de la metodología. Los casos de uso se pueden planificar con los ciclos de desarrollo de la composición de servicios.
Derivación de sistemas teóricos:
P: ¿La metodología deriva sistemas teóricos a partir de los problemas identificados? R: Sí, a través de los procesos 'Especificación Funcional' y 'Composición en tiempo de diseño'.
P: ¿Ofrece alguna guía para formular sistemas teóricos? R: Sí, a través de los procesos 'Especificación Funcional' y 'Composición en tiempo de diseño'.
P: ¿La metodología considera los efectos de las características de la "situación del problema" en la formulación de los sistemas teóricos? R: Sí, ofrece guías específicas y considera los modelos UML.
P: ¿Es posible modificar los sistemas teóricos? R: Si, debido a la naturaleza iterativa e incremental de la metodología.
P: ¿La metodología aclara el propósito de los sistemas teóricos? R: Sí, a través de los procesos 'Especificación Funcional' y 'Composición en tiempo de diseño'.

Tabla 7-5: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Formulación del Problema"

PREGUNTAS Y RESPUESTAS SOBRE EL PROCESO SOLUCIONADOR DE PROBLEMAS -DISEÑO DE LA SOLUCIÓN-
Realización del diseño conceptual / lógico:
P: ¿La metodología acepta los sistemas teóricos como el punto de inicio del diseño lógico? R: Sí. Dichos sistemas teóricos serán refinados durante el diseño.
P: ¿Depende la metodología de la experiencia o el conocimiento de sus usuarios para el diseño de las soluciones? R: Ofrece una guía detallada pero se requiere habilidad de diseño.

P: ¿Qué técnicas sugiere la metodología para la formulación de las soluciones? R: Las técnicas están definidas en el proceso 'Composición en tiempo de diseño' de la fase de Diseño.
P: ¿Se explican de forma suficiente estas técnicas a lo largo de una serie de pasos? R: Sí.
P: ¿Cómo asegura la metodología que el cliente estará de acuerdo con el diseño lógico? R: A través de la adopción de un enfoque iterativo e incremental.
P: ¿Hay algunos aspectos de la "situación del problema" que la metodología excluye en su apoyo al diseño? R: No.
P: ¿La metodología considera o apoya el modelado de asuntos de comportamiento? R: Sí, a través de la definición de diagramas UML de actividad y de secuencia y, también, de descripciones explicativas.
Realización del diseño físico:
P: ¿La metodología se convierte en un instrumento a través del cual los usuarios pueden aplicar lo último en técnicas de gestión y de trabajo? R: Sí, la metodología recoge qué aspectos hay que tener en cuenta en el diseño y desarrollo de la composición de servicios dinámica y sensible al contexto de ejecución. También define buenas prácticas para llevar a cabo esta composición de servicios pero permite aplicar otras técnicas o prácticas de trabajo.
P: ¿La metodología avisa a sus usuarios de la importancia de realizar con cuidado las decisiones sobre tecnologías? R: Sí, en el proceso 'Especificación Funcional', de la fase de Análisis.
P: ¿Ofrece apoyo para considerar ambos diseños, lógico y físico, de una forma integra? R: Sí, en el proceso 'Realización de la composición', de la fase de Implementación.
P: ¿Es el diseño lógico independiente de las tecnologías específicas? R: Sí.

Tabla 7-6: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Diseño de la Solución"

PREGUNTAS Y RESPUESTAS SOBRE EL PROCESO SOLUCIONADOR DE PROBLEMAS -IMPLEMENTACIÓN DEL DISEÑO-
Implementación de los diseños:
P: ¿Qué ayuda ofrece la metodología a la implementación de los diseños de la solución? R: Detalla los pasos a seguir en el proceso 'Generación de la composición' de la fase de Implementación del servicio compuesto.
P: ¿La metodología soporta actividades de testeo? R: Sí. Validación y verificación de la composición de servicios.
P: ¿La metodología avisa a sus usuarios de la posibilidad de cambios en el diseño lógico debido al descubrimiento de problemas durante la implementación? R: Sí. En el proceso 'Realización de la composición' de la fase de Implementación.
P: ¿La metodología considera los problemas con la interfaz de usuario? R: Sí. A través del proceso 'Realización de la composición' de la fase de Implementación.

Tabla 7-7: Preguntas y respuestas sobre "el Proceso Solucionador de Problemas-Implementación del Diseño"

7.2.2 Conclusiones de la Evaluación

De las tablas 7-2, 7-3, 7-4, 7-5 y 7-6 se extrae como conclusión que la metodología de composición de servicios propuesta en el capítulo 5 satisface los requisitos impuestos por el marco de trabajo NIMSAD, y además, posee todas las características deseadas sobre los tres elementos definidos por NIMSAD; aunque hay que tener en cuenta que las características relacionadas con el proceso solucionador de problemas son las más significativas. De este modo, se asegura la integridad y la calidad general de la metodología de composición de servicios, y se incrementa la confianza en su aplicabilidad y su valor práctico para una variedad de actividades de creación de servicios compuestos, tanto estáticos como dinámicos.

Más aún, el objetivo, las características, la naturaleza y la filosofía de la metodología propuesta se entienden mejor ahora, ofreciendo a los desarrolladores e integradores de servicios una visión más amplia de sus posibilidades antes de aplicarla, con lo que se aprecian mejor los puntos fuertes de la metodología de composición de servicios, y les

permite personalizarla de la mejor manera posible de acuerdo con los requisitos de una circunstancia particular.

Finalmente, la evaluación de la metodología de composición de servicios resalta la necesidad de que los desarrolladores e integradores de servicios usen su capacidad de razonamiento intelectual y apliquen la metodología de una manera crítica, ya que el éxito de cualquier metodología se ve afectado de forma muy significativa por la "construcción mental" del usuario de la metodología y su verdadero entendimiento de las guías y la estructura de la metodología.

Todos somos muy ignorantes; lo que ocurre es que no todos ignoramos las mismas cosas.

Albert Einstein

CAPITULO 8

Conclusiones

Tras revisar el estado del arte acerca de la composición de servicios, poniendo especial énfasis en la composición dinámica, llegamos a la conclusión de que esta tecnología no ha avanzado tanto como el conjunto de oportunidades que presenta. De hecho, el desarrollo de servicios compuestos todavía es una tarea ad-hoc que consume mucho tiempo y requiere un considerable esfuerzo de programación a bajo nivel. El hecho de utilizar soluciones ad-hoc implica confiar en soluciones propietarias, lo cual dificulta la coordinación e interoperabilidad entre las empresas y los servicios ofrecidos, y también la evolución hacia una economía orientada a servicios.

Hoy en día se está evolucionando hacia un entorno más dinámico donde las soluciones software ofrecidas deben ser sensibles al contexto de ejecución, proporcionando la funcionalidad más adecuada y la calidad óptima en cada momento. Sin embargo, la mayoría de las iniciativas realizadas en torno a la composición de servicios se centra en las composiciones estáticas de servicios.

Aunque hay algunos enfoques, como eFlow o SELF-SERV, que se pueden considerar como un primer intento de creación de plataformas de soporte para composiciones dinámicas, desde un punto de vista metodológico estos enfoques están muy centrados en soluciones propietarias, lo cual dificulta su generalización y aplicación al diseño y desarrollo de cualquier composición de servicios, bien sea estática o dinámica.

En consecuencia, nos encontramos con una carencia de mecanismos metodológicos que den soporte al diseño y desarrollo tanto de composiciones de servicios estáticas como dinámicas.

Esta ausencia de un marco adecuado para la composición de servicios queda solventada con la aportación de la presente investigación: un nuevo método de diseño que mejora la capacidad de composición e integración de servicios que la sociedad actual y del mañana requiere para hacer realidad la evolución hacia una economía orientada a servicios. Ese método está centrado en el desarrollo de composiciones de servicios no sólo estáticas sino también dinámicas, auto-configurables y sensibles al contexto de ejecución.

Su principal innovación frente a las alternativas existentes es que permite desarrollar composiciones de servicios que satisfagan a un sector del mercado amplio, sin tener que adaptarse a cada caso específico en tiempo de diseño, lo que implica menos tiempo de desarrollo y costes asociados menores.

La descripción de dicho método se hace a través de una metodología o compendio de buenas prácticas que enfoca la composición de servicios desde una perspectiva completa e independiente de cualquier implementación tecnológica. Por un lado, se tienen en cuenta los modelos de negocio y patrones arquitecturales más adecuados para aplicar en el diseño de la composición; y por otro, se ayuda a identificar en tiempo de diseño la variabilidad de la composición de tal modo que la composición de servicios pueda adaptarse a las condiciones del contexto de ejecución. Para diseñar y configurar la dinamicidad existente en una composición de servicios, aplicamos las técnicas de gestión de variabilidad definidas en el enfoque de Ingeniería de Líneas de Producto.

Para modelar de manera formal la metodología de composición aportada hemos seleccionado la notación propuesta por el organismo de estandarización *OMG* llamada *SPEM* v2.0, frente al metamodelo ISO/IEC 24744 propuesto por el organismo de estandarización ISO.

Entre los principales motivos para escoger el metamodelo SPEM v2.0 destacan su amplia aceptación en el mundo industrial, que se basa en el lenguaje de modelado UML y que proporciona una notación gráfica estándar.

Es importante remarcar que el objetivo de esta investigación no era definir otra metodología más de desarrollo de software, sino una que resaltase las características más importantes en el ámbito del diseño y desarrollo de las composiciones de servicios dinámicas y sensibles al contexto de ejecución. Por ello, el enfoque adoptado para especificar la metodología de composición de servicios ha consistido en definir aquellas fases, procesos, tareas, pasos, roles, productos de trabajo y responsabilidades específicas de la definición y desarrollo de composiciones de servicios que, posteriormente, pueden ser incorporadas a una metodología de desarrollo de software existente. Con el fin de ilustrar el uso de la metodología hemos incluido dos ejemplos sobre su aplicación en una organización que ya tiene implantado *RUP* y *XP* como metodología general de desarrollo de software.

Por otra parte, para facilitar la aplicación y el uso de la metodología de composición de servicios propuesta, hemos desarrollado una herramienta que guía al usuario de la metodología en el seguimiento de las diferentes tareas a realizar para diseñar y desarrollar una composición de servicios, y también facilitar la integración y comunicación del trabajo realizado por los diferentes roles involucrados.

Como hemos indicado anteriormente, el ámbito del soporte a la composición dinámica de servicios en tiempo de ejecución queda fuera del alcance de esta tesis. Sin embargo, para verificar y validar que las composiciones desarrolladas siguiendo la metodología propuesta son capaces de auto-configurarse y adaptarse al contexto de ejecución, hemos presentado brevemente un entorno de ejecución que se está desarrollando dentro del proyecto europeo SeCSE para dar soporte a la ejecución de las composiciones dinámicas basadas en el enfoque propuesto. Dicho entorno de ejecución, que ha sido utilizado en el proceso de validación de la metodología de composición, implementa la variabilidad de una composición como reglas ECA (Evento-Condición-Acción). El uso de estas reglas permite lanzar eventos para poder realizar asociaciones dinámicas de servicios concretos e incluso búsquedas de posibles servicios concretos durante la ejecución de la composición.

Finalmente, la validación de esta metodología la hemos realizado en el contexto de diferentes escenarios industriales reales, centrándonos principalmente en el área de la automoción y de las telecomunicaciones.

Los criterios de validación utilizados han sido cuatro: el valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de dicha metodología. Los resultados obtenidos de dichas experimentaciones revelan la satisfacción de los usuarios finales y proporcionan una clara evidencia de que la metodología de composición de servicios propuesta es un mecanismo adecuado para diseñar y desarrollar composiciones de servicios dinámicas y sensibles al contexto de ejecución.

Además, con el fin de evaluar si la metodología propuesta cubre todos los casos que una metodología completa enfocada en el desarrollo de sistemas debería cubrir, también hemos realizado una evaluación de esta metodología con referencia al marco *Normative Information Model-based Systems Analysis and Design* (NIMSAD), obteniendo también unos resultados que así lo confirman.

Así, todo ello nos permite concluir que la hipótesis planteada como punto de partida de esta investigación, "***Si se proporciona soporte a la composición de servicios integrando las prácticas de la Ingeniería de Líneas de Producto, se posibilitará el diseño y desarrollo de composiciones de servicios flexibles, dinámicas y sensibles al contexto de ejecución, capaces de autoconfigurarse según las condiciones del entorno de ejecución,***" es cierta, ya que la metodología propuesta integra las prácticas de variabilidad de la Ingeniería de Líneas de Producto para definir la dinamicidad de las composiciones de servicios y dicha metodología es vista por los potenciales usuarios como una guía clara y precisa para diseñar y desarrollar composiciones de servicios capaces de autoconfigurarse y adaptarse al contexto de ejecución.

8.1 Consecución de los Objetivos Operacionales

En esta sección, a modo de resumen, exponemos los logros alcanzados con el cumplimiento tanto de los objetivos operacionales, como de los objetivos específicos y, por tanto, del objetivo general.

En el capítulo 1 definimos cinco objetivos operacionales. En la Tabla 8-1 se muestra en qué secciones de la presente memoria han sido abordados cada uno de ellos.

OBJETIVOS OPERACIONALES	SECCIÓN
OO1 - Definir un modelo de diseño de la variabilidad	4.2
OO2 - Definir una metodología de composición	5.1, 6
OO3 - Construir una herramienta de soporte	5.2,Anexo2
OO4 - Realizar pruebas de uso de la metodología en escenarios industriales reales	7.1
OO5 - Evaluar la metodología desde un punto de vista del desarrollo de sistemas	7.2

Tabla 8-1: Objetivos operacionales más relevantes y sección donde se abordan

El objetivo operacional OO1 puede darse por satisfecho gracias a la definición de un modelo de diseño de la dinamicidad de una composición obtenido a partir de las técnicas de variabilidad definidas en el enfoque de la Ingeniería de Líneas de Producto.

Dentro del objetivo operacional OO2, definimos la metodología de composición de servicios basado en el modelo de diseño de la variabilidad para poder diseñar y desarrollar composiciones de servicios dinámicas y flexibles. Dentro de este objetivo destaca que definimos una metodología basada en un compendio de buenas prácticas, describiendo en detalle los procesos, tareas, roles y responsabilidades involucrados en la composición de servicios.

Con respecto al objetivo operacional OO3, definimos una herramienta de soporte diseñada para actuar como un asistente al diseño y desarrollo de composiciones de servicios siguiendo la metodología propuesta. En el Anexo 2 proporcionamos información más detallada acerca del funcionamiento de esta herramienta.

Los objetivos operacionales OO4 y OO5 fueron tratados a lo largo del capítulo 7, donde diseñamos y realizamos distintas pruebas o experimentos basados en escenarios industriales reales con el fin de comprobar el comportamiento de la solución planteada bajo diversas circunstancias y examinar el valor práctico, la facilidad de uso, la compatibilidad y la aplicabilidad de la metodología de composición de servicios propuesta. También realizamos una evaluación de nuestra metodología para determinar si era completa y si cubría todos los casos que una metodología enfocada en el desarrollo de sistemas debería cubrir. Los resultados obtenidos en las pruebas fueron favorables, permitiendo confirmar las bondades perseguidas por nuestra solución.

Tal y como se muestra en la Tabla 8-2, y a la vista de la consecución de los cinco objetivos operacionales planteados, los objetivos específicos pueden darse igualmente por alcanzados con el desarrollo de la solución planteada, así como nuestro objetivo general.

OBJETIVOS ESPECÍFICOS	OBJETIVOS OPERACIONALES.
OE1 - Definir la solución metodológica basada en los principios de variabilidad del enfoque PLE	OO1, OO2
OE2 - Construir una herramienta de soporte a la metodología de composición	OO3
OE3 - Verificar los resultados del uso de la metodología	OE4, OO5

Tabla 8-2: Relación entre los objetivos específicos y los objetivos operacionales

8.2 Futuras líneas de investigación

Son varias las potenciales futuras líneas de investigación que podrían ser abordadas como continuación del trabajo expuesto en esta memoria, con el objeto de ofrecer una solución completa y global para las composiciones de servicios, poniendo especial énfasis en las composiciones de servicios flexibles, dinámicas y sensibles al contexto de ejecución.

Una primera línea de investigación abierta es proporcionar un entorno integrado de trabajo compuesto por metodologías y herramientas que den soporte a todo el ciclo de vida de los servicios: definición, búsqueda, composición, desarrollo, despliegue, negociación y monitorización.

Una segunda línea de investigación se enfocaría en la re-negociación de los acuerdos de nivel de servicio (SLA), que también es un punto clave en las composiciones de servicios dinámicas, ya que la identificación de los servicios concretos a asociar puede requerir algún tipo de negociación de los parámetros de calidad de servicio. La negociación automática de dichos parámetros constituye una línea de investigación abierta que ayudará a mejorar la calidad ofrecida por las composiciones de servicios dinámicas y sensibles al contexto de ejecución.

Como tercera línea de investigación aparece lo que hoy en día se denomina composición "***on-the-fly***" [Pistore et al. 2005]. Este nuevo término dentro del área de las composiciones de servicios dinámicas hace referencia a la creación automática de composiciones de servicios en tiempo de ejecución a partir de ciertos requisitos. El objetivo es proveer un sistema capaz de generar automática y dinámicamente una composición de servicios a partir de un conjunto de requisitos del usuario. Dicho sistema debería seguir unas pautas para poder diseñar, desarrollar, implementar y ejecutar el servicio compuesto requerido.

También es importante destacar que debido a la evolución que está sufriendo el mercado y las organizaciones hacia un entorno orientado a servicios, está surgiendo la necesidad de integrar los sistemas heredados de las organizaciones con las arquitecturas orientadas a servicios. Esto implica la creación de composiciones de sistemas denominadas **híbridas**, donde se combinan servicios con funcionalidades ofrecidas por sistemas heredados. Por este motivo, sería aconsejable proporcionar un soporte tanto metodológico como tecnológico al diseño y desarrollo de estos sistemas híbridos. Esta sería otra posible línea de investigación abierta.

Por último, consideramos que las aportaciones de esta tesis, siendo hoy en día innovadoras, sientan las bases de lo que será práctica habitual en el desarrollo del software y de las futuras líneas de investigación sobre la composición de servicios.

Para terminar, dando apoyo a esta última idea, unas palabras de Anne-Marie Sassen y Charles Macmillan, responsables del área de investigación de Ingeniería de Servicios de la Comisión Europea, extraídas del informe '*The service engineering area: an overview of its current state and a vision of its future*':

> "*In order to develop value networks of services it is necessary to 'compose' or put together the services. Currently this can be done manually by a tedious process based on the syntactical descriptions of the service interfaces. Usually, only services developed within one company are composed. The risk of relying on services provided by a third party is in general still too high. The real potential of software services will be available when the step from this static intra-company composition to dynamic, inter-company composition is made. Only then will it be possible to create an on the fly value network of services that can respond to very personal needs of an individual or organisation*" [Sassen y Macmillan 2006].

La ciencia se compone de errores, que a su vez son los pasos hacia la verdad.

Julio Verne

CAPITULO 9

Bibliografía

[ATHENA 2007] *ATHENA (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications), Integrated Project*, IST-507849, V Programa Marco, 2004-2007. http://www.athena-ip.org/

[Avison y Fitzgerald 1995] Avison, D.E.; Fitzgerald, G.: *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill, 1995.

[Avison et al. 1999] Avison, D.E.; Lau, F.; Myers, M.; Nielsen, A.: "Action research", *Communications of the ACM Press*, vol. 42, pp.94-97, Nueva York, USA, 1999.

[Basili 2000] Basili, V.R.: "Using Experiments to Build a Body of Knowledge", *7th European Workshop on Software Process Technology (EWSPT) keynote*, Salzburg, Austria, 2000.

[Baskerville 1997] Baskerville, R.: "Distinguishing Action Research from Participative Case Studies", *Journal of Systems and Information Technology*, vol. 1, pp. 25-45, 1997.

[Baskerville 1999] Baskerville, R.: "Investigating Information Systems with Action Research", *Communications of the Association for Information Systems*, vol. 2, 1999, artículo 19.

[Baskerville y Wood-Harper 1996] Baskerville, R.; Wood-Harper, A.T.: "A Critical Perspective on Action Research as a Method for Information Systems", *Research Journal of Information Technology*, Vol. 11, pp. 235-246, 1996.

[Bastida y Benguría 2005] Bastida, L.; Benguría, G.: "Business Process Definition Languages vs Tradicional Methods Towards Interoperability", *COTS-Based Software Systems: 4th International Conference*, vol. 3412/2005, Editores Franch, X.; Port, D.; ICCBSS 2005, Bilbao, España, Febrero 7-11, 2005

[Bastida et al. 2008a] Bastida, L.: "A Methodology for Dynamic Service Composition", *ICCBSS 2008 Conference Proceedings*, Madrid, España, Febrero 25-28, 2008.

[Bastida 2008] Bastida, L.; Berreteaga, A.; Cañadas, I.: "Adopting Service Oriented Architectures Made Simple", *I-ESA 2008 Conference Proceedings*, Berlín, Alemania, Marzo 26-28, 2008.

[Bastida et al. 2008b] Bastida, L.; Nieto, F.J.; Tola, R.: "Context-Aware Service Composition: A Methodology and a Case Study", *SDSOA 2008 Worshop*, ICSE Conference Proceedings, Leipzig, Alemania, Mayo 11, 2008.

[Bastida y Berreteaga 2007] Bastida, L.; Berreteaga, A.: "Cómo Adoptar el Enfoque de las Arquitecturas Orientadas a Servicios en una Organización", *JSWEB 2007 Conference Proceedings*, Zaragoza, España, Septiembre 12-13, 2007.

[Bastida y Nieto 2007] Bastida, L.; Nieto, F.J.: "Context-Aware Service Compositions: A Way to Facilitate Interoperability", *I-ESA 2007 Conference Proceedings*, Madeira, Portugal, Marzo 29-31, 2007.

[Bastida y Nieto 2006] Bastida, L.; Nieto, F.J.: "Aplicación del Enfoque de Líneas de Producto para Componer Servicios Dinámicos y Sensibles al Contexto", *JSWEB 2006*, España, Noviembre 16-18, 2006.

[BEA 2007] BEA: "BEA Delivers Foundation for WorkSpace 360° at BEAWorld San Francisco", *Nota de prensa*, Septiembre 2007. http://www.bea.com/framework.jsp?CNT=pr01853.htm&FP=/content/news_events/press_releases/2007

[Beck 2000] Beck, K: *Extreme Programming Explained. Embrace Change*, Addison-Wesley, Boston, 2000.

[Benatallah et al. 2003] Benatallah, B.; Sheng, Q.; Dumas, M.: "The Self-Serv Environment for Web Services Composition", *IEEE Internet Computing*, pp. 40-48, 2003.

[Berki y Georgiadou 1998] Berki, E.; Georgiadou, E.: "A Comparison of Qualitative Framework for Information Systems Development Methodologies", *Proceedings de 12th International Conference for Quality*, Jerusalén-Israel, Diciembre 1998.

[Bharti 2005] Bharti, N.: "Enthusiasm, caution guide SOA adoption", *News Editor in SearchWebServices.com*, 2005. http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1079520,00.html

[Buecker et al. 2007] Buecker, A.; Ashley, P.; Borrett, M.; Lu, M.; Muppidi, S.; Readshaw, N.: *Understanding SOA Security: Design and Implementation*, IBM RedBooks, Noviembre 2007.

[Buschmann et. al 1996] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Shal, M.: *Pattern-oriented software architecture: A systems of patterns*, Vol. 1, Wiley, 1996.

[Butler 2006] Butler J.: "Applying Product Line Techniques to SOA", *CBDI Journal*, 2006.

[Cardoso et al. 2005] Cardoso, J.: "Evaluating Workflows and Web Process Complexity", *Workflow handbook*, Ed. Layna Fiacher, USA, Future Strategies Inc., pp. 284, 2005.

[Casati 2000] Casati, F.; Ilnicki, S.; Jin, L.; Shan M.C.: "An Open, Flexible and Configurable System for E-Service Composition", *Informe de Hewlett Packard,* HLP-2000-41, Marzo 2001.

[Casati 2001] Casati, F.: "Dynamic and adaptive composition of e-services", *Information Systems*, vol. 26, no. 3, pp. 143-163, 2001.

[Checkland 1981] Checkland, P.B.: *Systems Thinking, Systems Practice*, Wiley, Chichester-UK, 1981.

[Clements et al. 2001] Clements, P.; Northrop, L.: "Software Product Lines-Practices and Patters", *SEI series in Software Engineering*, Editorial Addison-Wesley, 2001.

[CMM 1995] Software Engineering Institute (SEI): *The Capability Maturity Model: Guidelines for Improving the Software Process*, 1995.

[Colombo et al. 2006] Colombo, M.; Di Nitto, E.; Mauri, M.; "SCENE: a service composition execution environment supporting dynamic changes disciplined through rules", *proceedings de ICSOC'06*, Chicago U.S., 2006.

[Chein et al. 1948] Chein, I.; Cook, S.; Harding, J.: "The Field of Action Research", *The American Psychologist*, vol. 3, pp. 42-50, 1948.

[cXML 2007] "Commerce XML Resources, cXML version 1.2.017", Abril 2007. http://cxml.org/

[Davis 1993] Davis, A.: *Software Requirements*, Prentice Hall, 1993.

[DTD 2000] W3C: "XML Specification (XMLspec) DTD, Version 2.1", W3C, Febrero 2000. http://www.w3.org/XML/1998/06/xmlspec-report-v21.htm

[Dustdar et al. 2005] Dustdar, S.; Schreiner, W.: "A survey on web services composition", *International Journal on Web and Grid Services*, vol. 1, pp. 1-30, 2005.

[ebXML 2001] OASIS y UN/CEFACT: *ebXML Technical Architecture Specification v 1.0.43*, Equipo de trabajo ebXML Technical Architecture, Febrero 2001.

[EDI 2002] ISO 9735-2002: *Electronic data interchange for administration, commerce and transport (EDIFACT) -- Application level syntax rules*, estándar ISO, Julio 2002.

[Emerson 1990] Emerson, E. A.: "Temporal and modal logic", Editor J. van Leeuwen, Handbook of Theoretical Computer Science, vol. B: Formal Models and Semantics, Elsevier, 1990.

[EPF 2007] Eclipse Process Framework Project (EPF), versión 1.2, Julio 2007. http://www.eclipse.org/epf/

[Estay y Pastor 2000] Estay, C.; Pastor, J.: "Improving Action Research in Information Systems with Project Management", *Americas Conference on Information Systems (AMCIS)*, Long Beach, California, 2000.

[Estay y Pastor 2000b] Estay, C.; Pastor, J.: "Towards a project structure for Action-Research in Information Systems", *10th Annual Business and Information Technology Conference (BTI)*, Manchester, UK, 2000.

[Estay y Pastor 2001] Estay, C.; Pastor, J.: "Un Modelo de Madurez para Investigación-Acción en Sistemas de Información", *VI Jornadas de Ingeniería del Software y Bases de Datos*, Ciudad Real, España, 2001.

[Evans y Marciniak 1987] Evans, M.; Marciniak, J.: *Software Quality Assurance and Management*, Wiley & Sons, Nueva York, 1987.

[French y Bell 1996] French, W.L.; Bell, C.H.: *Organisational Development: Behavioral Science Interventions for Organization Improvement*, 6ª ed., Prentice-Hall, Londres.

[Gartner 2007] Thompson, J.; Barnes, M.; Iijima, K.; Lheureux, B.J.; Malinverno, P.; Natis, Y.V.; Pezzini, M.; Schulte, R.W. y Hayward, S.: "Magic Quadrant for Application Infrastructure for Back-End Application Integration Projects, 2007", *Gartner RAS Core Research Note G00147400*, June 2007. http://mediaproducts.gartner.com/reprints/microsoft/vol1/article1/article1.html

[Grady 1992] Grady, R.: *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, 1992.

[Glass et al. 2002] Glass, R.; Vessey, I.; Ramesh, V.: "Research in software engineering: an analysis to the literature", *Information and Software Technology*, vol. 44, pp. 491-506, 2002.

[Gonsalves 2007] Gonsalves, A: "Analyst Firm Predicts Rapid Growth In BPM Software Market", *Intelligent Enterprise*, Septiembre 2007. http://www.intelligententerprise.com/showArticle.jhtml?articleID=201804780

[Halmans y Pohl 2002] Halmans, G.; Pohl, K.: "Considering Product Family Assets when Defining Customer Requirements", *Proceedings del Internacional Workshop on Product Line Engineering (PLEE)*, pp. 37–42, Erfurt, Fraunhofer IESE, 2001.

[Henderson-Sellers y Gonzalez-Perez 2005] Henderson-Sellers, B. y Gonzalez-Perez, C.: "The Rationale of Powertype-Based Metamodelling", *2nd Asia-Pacific Conference on Conceptual Modelling*, Australian Computer Science Communications vol. 7, nº. 6, pp 7-16, 2005.

[Hu 2003] Hu, M.: "Web Services Composition, Partition, and Quality of Service in Distributed System Integration and Re-engineering", *Proceedings of XML*, 2003.

[Hurwitz et al. 2007] Hurwitz, J.; Bloor, R.; Baroudi. C.; Kaufman, M.: *Service Oriented Architecture for Dummies* ®, Ed. Wiley Publishing, Indiana, 2007.

[IDC 2007a] IDC Italia: "Enterprise Integration e SOA Conference 2007", Evento local, Milán-Italia, 18 Abril 2007.

[IDC 2007b] IDC: "Worldwide SOA-Driven Software 2007-2011 Forecast: A Changing IT Lifestyle," #*207080*, Mayo 2007.

[IDC 2007c] IDC: "Worldwide SOA-Based Services 2007-2011 Forecast and Analysis: A Maturing SOA Market Fuels New and Different Demands for Professional Services," #*206554*, Abril 2007.

[IEEE 1990a] IEEE Standard Computer Dictionary: *A compilation of IEEE Standard Computer Glosaries*, New York, 1990.

[IEEE 1990b] *IEEE Standard Glosary of Software Engineering Terminology*, IEEE Std.610.12-1990, 1990.

[ISO/IEC 24744 2007] ISO/IEC 24744:2007 "*Software Engineering -- Metamodel for Development Methodologies*", estándar internacional, Febrero 2007.

[Kitchenham 1996] Kitchenham, B.: "DESMET: A method for evaluating Software Engineering methods and tools", *Technical Report*, Universidad de Keele-UK, Agosto 1996.

[Koch y Lau 2001] Koch, N.; Lau, F.; *Information Systems Action Research: Serving Two Demanding Masters. Information Technology & People* (special issue on Action Research in Information Systems), 2001.

[Kotonya y Sommerville 2001] Kotonya, G; Sommerville, I.: *Requirements Engineering. Processes and Techniques*, John Wiley & Sons, Marzo 2001.

[Jayaratna 1994] Jayaratna, J.: *Understanding and Evaluating Methodologies, NIMSAD: A Systemic Framework*, McGraw-Hill, 1994.

[Lazcano et al. 2001] Lazcano, A.; Schuldt, H.; Alonso, G.; Schek, H. J.: "WISE: Process based E-Commerce", *IEEE Data Engineering Bulletin*, vol. 24, pp. 46-51, 2001.

[Likert 1932] Likert, R.: "A Technique for the Measurement of Attitudes", *Archives of Psychology 140,* pp. 1-55, 1932.

[Ludwig et al. 1999] Ludwig, H.; Hoffner, Y.: "Contract-based Cross-Organisational Workflows - The CrossFlow Project.", *Proceedings del International Joint Conference on Work Activities Coordination and Collaboration*, pp. 1-6, San Francisco-California, USA, 1999.

[MDA 2001] OMG: *Model Driven Architecture (MDA)* ® , 2001. http://www.omg.org/mda/

[McTaggart 1991] Mc Taggart, R.: "Principles of Participatory Action Research", *Adult Education Quaterly*, vol. 41, 1991.

[Mecella et al. 2002] Mecella, M.; Pernici, B.: "Building Flexible and Cooperate Applications based on e-Services", Dipartamento di Informatica e Sistemistica, Universidad de Roma 'La Sapienza', *Technical report 21-2002*, Roma-Italia, 2002.

[Microsoft 2007] Microsoft: "Microsoft Unveils Vision and Road Map to Simplify SOA, Bridge Software Plus Services, and Take Composite Applications Mainstream", *Nota de prensa*, Octubre 2007. http://www.microsoft.com/presspass/press/2007/oct07/10-30OsloPR.mspx

[Moody 2000] Moody, D.: "Building links between IS research and professional practice: improving the relevance and impact of IS research. 21", *Proceedings del International Conference on Information Systems*, Brisbane-Australia, Diciembre 2000.

[Myers 1997] Myers, M.: "Qualitative Research in Information Systems", *MISQ Discovery*, vol. 21, pp. 241-242, 1997.

[Narayanan et al. 2002] Narayanan, S.; McIlraith, S.: "Simulation, Verification and Automated Composition of Web Services", *Proceedings del International World Wide Web Conference* (ICWS), pp. 77-88, Honolulu- Hawaii, USA, 2002. http://www.icsi.berkeley.edu/~snarayan/nar-mci-www11.pdf

[Nieto et al. 2005] Nieto, F.J.; Bastida, L.; Escalante, M.; Gortazar, A.; "Development of Dynamic Composed Services based on the

Context", *I-ESA 2006 Conference Proceedings*, Bordeaux, Francia, Marzo 22-24, 2006.

[Northrop 2002] Northrop, L.: *SEI's Software Product Line Tenets*, IEEE Software 19-4, 32-40pp, Julio/Agosto 2002.

[OASIS 2006] OASIS Service Oriented Architecture TC: "*OASIS Reference Model for Service Oriented Architecture*", versión 1.0, Espcificación aprobada por el Comité Oficial, Agosto 2006.

[ORACLE 2006] "Oracle Expands its Oracle(R) Fusion Middleware SOA Offering", *Nota de prensa*, Octubre 2006. http://www.oracle.com/corporate/press/2006_oct/openworldsf06-04.html

[OWL 2004] W3C Web Ontology Working Group: "OWL Web Ontology Language Overview", Recomendación *W3C*, 10 Febrero 2004. http://www.w3.org/TR/owl-features/

[OWL-S 2004] France Telecom, NIST, Network Inference, Nokia, SRI International, Toshiba y las Universidades de Maryland, Stanford y Southampton: "OWL-S: Semantic Markup for Web Services", *Propuesta a la W3C*, 22 Noviembre 2004. http://www.w3.org/Submission/OWL-S/

[Padak y Padak 2005] Padak, N.; Padak, G.: "*Guidelines for Planning Action Research Projects*", última actualización: Noviembre 2005. http://literacy.kent.edu/Oasis/Pubs/0200-08.htm

[Palmer y Felsing 2002] Palmer, S.R.; Felsing, J.M.: *A Practical Guide to Feature-Driven Development*, Prentice Hall, 2002.

[Parnas 1979] Parnas D.: "Designing Software for Ease of Extension and Contraction", *IEEE Transactions on Software Engineering*, March 1979.

[Philips 1998] Philips, P.A.: "Disseminating and Applying the Best Evidence", *Medical Journal of Australia (MJA*, vol. 168, pp. 260-261, 1998.

[Pires et al. 2002] Pires, P. F.; Benevides, M. R. F.; Mattoso, M.: "Building Reliable Web Services Compositions", *Proceedings del Web Databases and Web Services*, Springer-Verlag, LNCS 2593, pp. 59–72, Berlin Heidelberg, 2002.

[Pistore et al. 2005] Pistore, M.; Roberti, P.; Traverso, P.: "Process-Level Composition of Executable Web Services: 'On-the-fly' Versus 'Once-for-all' ", *The Semantic Web: Research and Applications*, Springer-Verlag, LNCS 3532, pp. 62–77, Berlin Heidelberg, Mayo 2005.

[PMB 2000] PMBOK: *A Guide to the Project Management Body of Knowledg*", USA. 2000.

[Pohl et al. 2005] Pohl, K.; Böckle G.; van der Linden, F.: *·Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer-Verlag, Berlin Heidelberg, 2005.

[RMC 2007] IBM Rational Method Composer (RMC), versión 7.2, Septiembre 2007.
http://www-306.ibm.com/software/awdtools/rmc/

[Robertson y Robertson 2006] Robertson, S. y Roberston, J.: *Mastering the Requirements Process*, 2ª Edición, Addison-wesley Professional, 2006.

[RosettaNet 1998] RosettaNet, Silicon Valley, USA, 1998.
www.rosettanet.org

[RUP 2002] Rational Software Corporation: Rational Unified Process (RUP), IBM, 2002.

[Sassen y Macmillan 2006] Sassen, A. M. y Macmillan, C.: "The service engineering area: An overview of its current state and a vision of its future", *Informe Técnico*, Comisión Europea, Enero 2006.

[Schuster et al. 2000] Schuster, H.; Baker, D.; Cichocki, A.; Georgakopoulos, D.; Rusinkiewicz, M.: "The Collaboration Management Infrastructure", *Proceddings del IEEE International Conference on Data Engineering*, pp. 485-487, San Jose-California, USA, 2000.

[Seaman 1999] Seaman, C.B.: "Qualitative Methods in Empirical Studies of Software Engineering", *IEEE Transactions on Software Engineering*, vol. 25, pp. 557-572, 1999.

[SeCSE 2008] SeCSE *(Service Centric System Engineering) Integrate Project*, IST-511680, VI Programa Marco, 2005-2008.
http://secse.eng.it

[SEI 2007] Northrop, L. M.; Clements, P. C.; con Bachmann, F.; Bergey, J.; Chastek, G.; Cohen, S.; Donohoe, P.; Jones, L.; Krut, R.; Little, R.; McGregor, J.; O'Brien, L.: *A Framework for Software Product Line Practice, Version 5.0*, July 2007

[Sheng 2002] Sheng, Q.; Benatallah, B.; Dumas, M.; Oi-Yan Mak, E.: "SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment", *Proceedings de la 28th Conferencia VLDB,* Hong Kong, China, 2002.

[SOAP 2007] MIT, Ercim y Keio: *SOAP Version 1.2 Part 1: Messaging Framework*, 2ª Edición, Recomendación de la W3C, Abril 2007
http://www.w3.org/TR/soap/

[SODIUM 2007] SODIUM (*Service-Oriented Development In a Unified framework*), STREP Project, IST-004559, VI Programa Marco, 2004-2007.
http://www.atc.gr/sodium/

[Sommerville y Sawyer 1997] Sommerville, I.; Sawyer, P.: *Requirements Engineering: A good practice guide*, Wiley, 1997.

[SPEM 2005] Alcatel, DRM, Fujitsu, IBM, Q-Labs, Rational Software Corp., Softeam y Unisys Corp.: *Software Process Engineering Metamodel*

Specification, version 1.1, OMG, Adopted Final Specification, Enero 2005.

[SPEM 2007] Adaptive Ltd., Fujitsu, Fundación ESI, IBM, OMG y Softeam: *Software Process Engineering Metamodel Specification*, version 2, OMG, Adopted Specification, Marzo 2007.

[Srivastava et al. 2003] Srivastava, B.; Koehler, J.: "Web Service Composition - Current Solutions and Open Problems", *Proceedings del ICAPS*, Workshop on Planning for Web Services, pp. 28-35, 2003. http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf

[Stansberry 2005] Stansberry, M.: "Yankee Group-SOA everywhere by 2006.", *SearchdataCenter.com*, Septiembre 2005. http://searchdatacenter.techtarget.com/originalContent/0,289142,sid80_gci1127784,00.html

[Sun et al. 2003] Sun, H.; Wang, X.; Zhou, B.; Zou, P.: "Research and Implementation of Dynamic Web Services Composition", *Proceedings del Advanced Parallel Programming Technologies Workshop (APPT)*, Springer-Verlag, LNCS 2834, pp. 457–466, Berlin Heidelberg, 2003.

[Svahnberg et al. 2001] Svahnberg, M., van Gurp, J., y Bosch, J., "On the Notion of Variability in Software Product Lines", *proceeding de Working IEEE/IFIP Conference on Software Architecture*, IEEE Computer Society Press, 2001. pp: 45-54.

[TIBCO 2006] TIBCO: "TIBCO ActiveMatrix", *Folleto SOA*, 2006. http://www.tibco.com/resources/software/soa/tibco_soa_br_lr.pdf

[Traverso et al. 2004] Traverso, P.; Pistore, M.: "Automated Composition of Semantic Web Services into Executable Processes", *Proceedings del Third International Semantic Web Conference (ISWC)*, Springer-Verlag GmbH, pp. 380, Hiroshima- Japón, Noviembre 2004.

http://sra.itc.it/people/traverso/papers/iswc04.pdf

[UDDI 2005] OASIS: *Universal Description, Discovery and Integration (UDDI)*, versión 3, estándar OASIS, Febrero 2005. http://www.oasis-open.org/committees/uddi-spec/

[UML 1997] OMG: *Unified Modeling Language (UML)*, versión 1.1 OMG, Adopted Specification, Enero 1997. http://www.uml.org/

[UML2 2007] Adaptive Ltd., Alcatel, Borland, CA, Ericsson, Fujitsu, HP, I-Logix, IBM, IONA, Kabira, MEGA Int., Motorola, OMG, Oracle, Softeam, Telelogic AB, Unisys y X-Change Techologies: *Unified Modeling Language (UML)*, versión 2.1.1 OMG, Adopted Specification, Febrero 2007.

[van de Linden et al. 2007] van der Linden, F. J.; Schmid, K.; Rommes, E.: *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Ed. Springer, Julio 2007

[Wadsworth 1998] Wadsworth, Y.: "What is participatory Action Research?", *Action Research International*, 1998. http://www.scu.edu.au/schools/gcm/ar/ari/p-ywadsworth98.html [Consulta: 10 Enero 2006]

[Waters 2005] Waters, R.: "Plugging together software may soon be painless", *Financial Times*, Mayo 2005. http://news.ft.com

[Whitten et al. 1996] Whitten, J.L; Bentley, L.D. y Barlow V.: *Análisis y Diseño de Sistemas de Información*, 3ª Edición, McGraw Hill, Irwin, Boston, 1996.

[Whitten et al. 2004] Whitten, J.L; Bentley, L.D. y Dittman K.C.: *Systems Analysis and Design Methods*, 6º Edición, McGraw Hill, Irwin, Boston, 2004.

[Wieringa 1996] Wieringa, R.J.; *Requirements Engineering: A Framework for understanding,* John Wiley&Sons, 1996

[Wood-Harper 1985] Wood-Harper, A.T.: "Research Methods in Information Systems: Using Action Research", *Research Methods in Information Systems*, pp. 169-191, Amsterdam, 1985.

[WSA 2003] W3C: *Web Services Architecture*, Borrador de trabajo, W3C, 8 Agosto 2003.
http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[WSA 2004] W3C: *Web Services Architecture*, Nota de trabajo, 11 Febrero 2004.
http://www.w3.org/TR/ws-arch/

[WS-BPEL 2007] Comité Técnico de WS-BPEL de OASIS: *Web Services Business Process Execution Language (WS-BPEL)*, versión 2.0, OASIS estándar, Abril 2007.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[WSDL 2001] Ariba, IBM y Microsoft: *Web Services Description Language (WSDL) 1.1*, Nota de la W3C, Marzo 2001.
http://www.w3.org/TR/wsdl/

[WS-I 2007] Web Services Interoperability Organisation (WS-I): *WS-I Basic Profile v1.2*, Borrador aprobado, Octubre 2007.
http://www.ws-i.org/

[WS-SEC 2006] Grupo de Trabajo de WS-Security (WSS): *Web Services Security (WS-Security) v1.1*, OASIS, Febrero 2006.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[Yang et al. 2002] Yang, J.; Papazoglou, P.: "Web Component: A Substrate for Web Service Reuse and Composition", *Proceedings del CAISE*, Springer-Verlag, LNCS 2348, pp. 21–36, Berlin-Heidelberg, 2002.

En el punto donde se detiene la ciencia, empieza la imaginación.

Jules Gaultier

CAPITULO 10

Glosario

ACID: El término ACID representa un conjunto de propiedades que toda transacción debe cumplir para ofrecer una mínima garantía de calidad y éxito. ACID es el acrónimo de Atomicidad, Consistencia, Aislamiento y Durabilidad. Por Atomicidad se hace referencia a que todas las operaciones que componen una transacción sean tratadas como una única unidad, es decir, o bien todas se ejecutan con éxito, o bien ninguna se ejecuta. Consistencia significa que, si los datos afectados por la transacción son válidos antes de la misma, han de ser válidos al finalizar. Por Aislamiento se entiende que cualquier operación que tenga lugar simultáneamente con la transacción no debe recibir datos en estados intermedios. Y por Durabilidad se entiende la garantía de que los efectos de una transacción finalizada con éxito no se perderán, sino que permanecerán.

Ad-hoc: Es un término latino que significa "para un fin determinado, para este propósito particular". Generalmente indica que una solución ha sido desarrollada de acuerdo a las condiciones y requisitos establecidos según un propósito específico. Una solución ad-hoc es todo lo contrario a una solución de propósito general.

Arquitectura Orientada a Servicios: La Arquitectura Orientada a Servicios, también conocida con el acrónimo SOA (*Service Oriented Architecture*), es un enfoque o patrón arquitectural para sistemas distribuidos en el que los agentes que intervienen son denominados 'servicios'.

B2B: Business-to-Business es un concepto de negocio utilizado para describir el comercio electrónico que surge entre dos entidades de negocio, en contraposición al comercio que tiene lugar entre una entidad de negocio y un individuo particular.

Binding: El término binding se utiliza para hacer referencia a la asociación entre la especificación de un servicio y una implementación concreta de dicho servicio.

Contexto: El contexto es la información utilizada por los servicios para adaptarse al momento y proveer al usuario del servicio, bien sea un sistema o una persona, un comportamiento personalizado. Puede contener diversa información como, por ejemplo, los datos del cliente o del dispositivo que se está usando para acceder al servicio.

CORBA: La Arquitectura Común de Intermediarios en Peticiones a Objetos (en inglés, *Common Object Request Broker Architecture*) es un estándar de la OMG que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

Coreografía de servicios: Define cómo interactúan entre sí varios servicios desde el punto de vista de la colaboración. La coreografía está enfocada hacia las composiciones dinámicas ya que los servicios implicados en la composición están débilmente acoplados.

DCOM: El Modelo de Objetos de Componentes Distribuidos (en inglés, *Distributed Component Object Model*) es una tecnología propietaria de Microsoft enfocada al desarrollo de componentes software distribuidos en varios ordenadores y que se comunican entre sí. DCOM ha sido abandonado en favor de la tecnología .NET de Microsoft.

DTD: Siglas de *Document Type Definition* es una definición, en un documento XML, que especifica restricciones en la estructura y sintaxis del mismo. Un documento DTD se puede incluir dentro del archivo del documento, pero normalmente se almacena en un fichero ASCII de texto separado.

FURPS: Es el acrónimo de un modelo de clasificación de atributos de calidad software. El modelo FURPS establece cinco características como factores de calidad que son los que le dan nombre: *Functionality* (Funcionalidad), *Usability* (Usabilidad), *Reliability* (Confiabilidad), *Perfomance* (Rendimiento) y *Supportability* (Soporte). El modelo FURPS

también incluye, además de los factores de calidad y los atributos, restricciones de diseño y requisitos de implementación, físicos y de interfaz.

IDE: Siglas de *Integrated Development Environment*, conocido también en castellano como entorno de desarrollo integrado, es un entorno de programación empaquetado como un programa de aplicación con el fin de ofrecer un conjunto de herramientas útiles para los programadores. Un entorno de desarrollo integrado puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

IA: El método Investigación-Acción (IA) es un método de investigación cualitativo. Es una forma de investigar de carácter colaborativo que busca unir la teoría y la práctica entre diferentes grupos de investigadores y practicantes mediante un proceso de naturaleza cíclica. La principal característica de este método es que el investigador forma parte integral del equipo que investiga.

Ingeniería de Líneas de Producto: El enfoque de la Ingeniería de Líneas de Producto (PLE) permite definir una línea de productos como un conjunto de sistemas de software intensivos que comparten un conjunto de características comunes y satisfacen las necesidades específicas de un segmento particular del mercado. El enfoque PLE define diversos mecanismos para modificar, instanciar y generar las líneas de productos.

Orquestación de Servicios: Es el mecanismo para definir cómo son las interacciones en la composición desde el punto de vista de su ejecución, conectando varios servicios que, al trabajar juntos, compondrán el servicio final de alto nivel.

OASIS: Siglas de Organization for the Advancement of Structured Information Standards es un consorcio internacional sin ánimo de lucro dedicado a la definición, desarrollo y adopción de estándares en el ámbito del comercio electrónico. Este consorcio produce más especificaciones y estándares sobre servicios web que cualquier otra organización. Fundado en 1993, OASIS cuenta con más de 5000 participantes representando a cerca de 600 organizaciones y miembros individuales en unos 100 países.

On-the-fly: El término on-the-fly hace referencia a algo que es creado, modificado o utilizado en el momento que es necesario y sin preparación previa. En el ámbito de la

composición de servicios, se refiere a la creación del proceso de la composición, en tiempo de ejecución a partir de un conjunto de requisitos.

Peer-to-Peer: El enfoque peer-to-peer (P2P), también denominado punto-a-punto, hace referencia a un esquema de comunicaciones basado en diferentes tecnologías que permite que una computadora en red pueda interactuar con otra sin necesidad de la intervención de un servidor central. Así, cada parte involucrada tiene las mismas capacidades y puede iniciar una sesión de comunicación en cualquier momento.

QoS: El término QoS (*Quality of Service*) aplicado al ámbito de los sistemas orientados a servicios hace referencia a los aspectos relativos al nivel de calidad que ofrece y cumple un servicio. No sólo incluye aquellos aspectos de calidad relativos a las características de la red y de las comunicaciones, sino que también abarca otras propiedades relevantes de los servicios como, por ejemplo, las relativas a la seguridad.

RPC: La Llamada a Procedimiento Remote (en inglés, *Remote Procedure Call*) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin necesidad de preocuparse por las comunicaciones entre ambos. Este protocolo es muy utilizado en el paradigma cliente-servidor.

RUP: El *Rational Unified Process*, también conocido con el acrónimo RUP, es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Servicio abstracto: Es una especificación de un servicio que no tiene asociada una implementación concreta.

Servicio atómico: Se denomina servicio atómico a todo servicio individual que no está formado por otros.

Servicio concreto: Es un servicio cuya especificación está asociada a una implementación concreta.

Servicio: Es una aplicación software modular residente en la red y accesible a través de protocolos estandarizados y que ofrece cierta funcionalidad. Dicha funcionalidad puede

ser automáticamente encontrada y después puede ser consumida directamente o integrada en otras aplicaciones para formar servicios más complejos. Un servicio está compuesto de un servicio abstracto y de uno o varios servicios concretos.

Servicio web: En inglés se lo conoce como Web Service. Según el organismo de estandarización W3C, "un servicio web es un sistema software diseñado para soportar interacciones interoperables de máquina a máquina sobre una red. Dicho servicio tiene un interfaz descrito en un formato procesable por máquinas (específicamente WSDL). Otros sistemas interaccionan con el servicio web de una manera descrita por su descripción utilizando mensajes SOAP" [WSA 2004].

SIP: Siglas de *Session Initiation Protocol*, es un protocolo basado en texto desarrollado por el IETF y propuesto como estándar RFC 3261 para la iniciación, modificación y finalización de sesiones interactivas de usuario. Dichas sesiones pueden incluir elementos multimedia como el video, voz, mensajería instantánea, juegos online y realidad virtual. SIP es similar a HTTP y SMTP. donde intervienen. En Noviembre del año 2000, SIP fue aceptado como el protocolo de señalización de 3GPP y elemento permanente de la arquitectura IMS (*IP Multimedia Subsystem*). SIP es uno de los protocolos de señalización para voz sobre IP

SOAP: Siglas de *Simple Object Access Protocol*, es un estándar del World Wide Web Consortium (W3C). SOAP es un protocolo ligero para el intercambio de información en un ambiente descentralizado y distribuido, es un protocolo basado en XML que consiste de tres partes: un sistema de trabajo para describir qué es lo que hay en un mensaje y cómo procesarlo; un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos para aplicaciones; y una convención para representar llamados a procedimientos remotos y respuestas.

UML: Siglas de *Unified Modeling Language*, es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar sistemas de software. UML permite entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML permite captar la información relativa a un sistema, tanto sobre la estructura estática como sobre su comportamiento dinámico.

W3C: Fundado en 1994 para desarrollar estándares comunes para Internet, la *World Wide Web Consortium* (W3C) es un consorcio internacional de la industria dirigido por el *Massachusetts Institute of Technology Laboratory for Computer Science* (MIT/LCS) en EE.UU., por el *Institut National de Recherche en Informatique et en Automatique* (INRIA) en Europa y por el *Keio University Shonan Fujisawa Campus* en Asia.

XML: Siglas de *eXtensible Markup Language*, es el formato universal para documentos y datos estructurados en la Web. XML es un protocolo de estándar industrial administrado por la organización W3C.

XP: El *Extreme Programming* (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo

ANEXO 1

Plantillas de la Metodología

A1.1 Modelo de Especificación Funcional

Modelo de Especificación Funcional

Esta es una plantilla de propósito general, por lo que deberá ser adaptada a cada caso de aplicación.

Este documento describe las principales funcionalidades de la composición de servicios. Cada funcionalidad consta de una breve descripción general, así como información más detallada de sus características, restricciones de requisitos y otros aspectos relevantes. También se puede especificar la existencia de uno o varios servicios concretos que satisfacen alguna de dichas funcionalidades.

Diagrama de caso de uso general

En esta sección se identifican las principales funcionalidades del sistema (también llamadas servicios abstractos) y se describen a través de un diagrama de casos de uso. Aquí se puede introducir una breve descripción del sistema completo, incluyendo cuál es la estrategia de provisión adoptada.

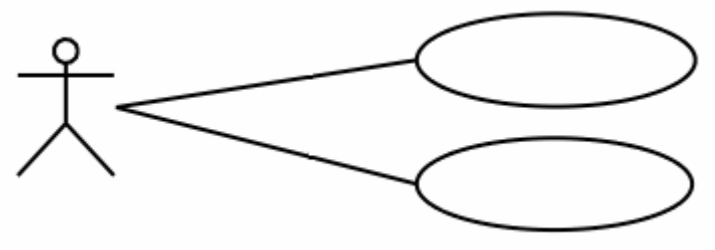


Figura 1: Representación gráfica del diagrama de casos de uso

Los servicios abstractos que han sido identificados son:

Nombre de servicio abstracto 1	
Identificador	Identificador único para cada servicio abstracto.
Propósito	¿Cuál es la funcionalidad que este servicio abstracto añade al sistema?
Descripción	Descripción del servicio abstracto (qué operaciones realiza, cómo las realiza, etc.). Descripción de las sub-funcionalidades que componen este servicio abstracto. Estas sub-funcionalidades se recogerán en el caso de uso de nivel inferior.
Otros requisitos	Requisitos no funcionales, requisitos deseables, etc.
Actores involucrados	Todos los actores (externos e internos), sistemas, etc. que contribuyen a la realización de la funcionalidad.
Dependencias	Dependencias con otras funcionalidades.
Caso de uso de nivel inferior	Este caso de uso describe con mayor detalle las sub-funcionalidades que componen el servicio abstracto. Se puede incluir un diagrama UML de casos de uso.
Servicios concretos candidatos	Descripción de posibles servicios concretos candidatos a utilizar.

A1.2 Modelo de Especificación de la Composición

Modelo de Especificación de la Composición

Esta es una plantilla de propósito general, por lo que deberá ser adaptada a cada caso de aplicación.

Este documento describe la composición de servicios a nivel de proceso de negocio, es decir, este modelo es una representación abstracta de la composición. Para ello incluye la descripción del flujo de trabajo, el comportamiento del sistema, las posibles transacciones y aspectos de seguridad de la composición y otros requisitos relevantes.

Diagrama de actividad

En esta sección se define un diagrama de actividad que, basándose en los patrones y modelos arquitectónicos existentes y en la descripción de la composición de la especificación del servicio, representa el flujo de trabajo de la composición. Este diagrama también puede incluir notas y restricciones para describir requisitos no funcionales.

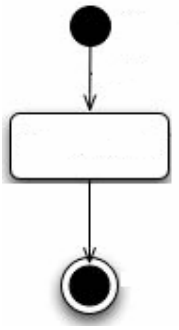


Figura 1: Representación gráfica del diagrama de actividad

Descripción

Introducir aquí una descripción sobre la composición del servicio abstracto que satisface un objetivo final mediante la combinación de diferentes funcionalidades.

Diagrama de secuencia (opcional)

Aquí se puede introducir un diagrama de secuencia más elaborado con el fin de proporcionar una mejor comprensión del conjunto del sistema.

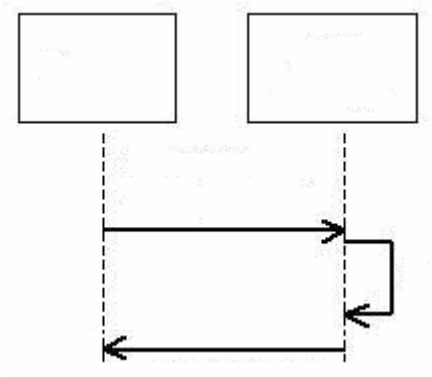


Figura 2: Representación gráfica del diagrama de secuencia

Requisitos no funcionales

Introducir una descripción de otros aspectos no funcionales que la composición de servicios debe satisfacer, tales como transacciones, gestión de errores o gestión de la seguridad.

A1.3 Modelo de Especificación de los Puntos de Variación

Modelo de Especificación de los Puntos de Variación

Esta es una plantilla de propósito general, por lo que deberá ser adaptada a cada caso de aplicación.

Este documento describe los puntos de variación identificados en la composición. Cada punto de variación consta de una breve descripción general, así como información más detallada como la condición y las diferentes alternativas asociadas a dicho punto de variación.

Descripción de los puntos de variación

En esta sección se identifican los puntos de variación de la composición de servicios y se describen sus condiciones y alternativas asociadas.

Los puntos de variación que han sido identificados son:

Nombre de punto de variación 1	
Identificador	Identificador único para cada punto de variación.
Identificador servicio abstracto	Identificador de la especificación o servicio abstracto al cual está asociado dicho punto de variación.
Propósito	¿Cuál es la funcionalidad que este punto de variación añade al sistema? ¿Cuáles son los motivos para incluirlo en la composición de servicios?
Descripción	Descripción del punto de variación. Pueden incluirse otros requisitos no funcionales o requisitos deseables a tener en cuenta.
Condición	Descripción informal de la condición del punto de variación según la cual se seleccionará alguna de las alternativas asociadas.
Tipo de datos	El tipo de datos del valor obtenido tras evaluar la condición del punto de variación. Puede ser, por ejemplo, un valor numérico o una cadena de caracteres.
Alternativas	Descripción de las diferentes alternativas asociadas a dicho punto de variación. Cada alternativa consta de un propósito, una breve descripción general así como de cuál debe ser el valor de la condición para que dicha alternativa sea seleccionada.

A1.4 Modelo de Realización de la Composición

Modelo de Realización de la Composición

Esta es una plantilla de propósito general, por lo que deberá ser adaptada a cada caso de aplicación.

Este documento describe la composición de servicios a nivel de proceso de negocio, es decir, este modelo es una representación abstracta de la composición.

Para ello incluye la descripción del flujo de trabajo, el comportamiento del sistema, las posibles transacciones y aspectos de seguridad de la composición y otros requisitos relevantes.

Este documento contiene información sobre todos los servicios concretos involucrados en la composición, así como la forma en la que cubren la funcionalidad requerida.

Además, por cada servicio abstracto es posible incluir una lista con otros posibles servicios concretos candidatos. También se describe cuál es el nivel de calidad de servicio proporcionado por la composición.

Descripción de los servicios

En esta sección se describen todos los servicios abstractos involucrados en la composición y por cada uno de ellos, se identifica un conjunto de servicios concretos candidatos.

La lista de los servicios abstractos involucrados en la composición es la siguiente:

<table>
<tr><th colspan="3">Descripción de los servicios</th></tr>
<tr><td>Nombre Servicio Abstracto1</td><td colspan="2">Por cada servicio concreto candidato:
<table><tr><th colspan="2">Servicio Concreto Candidato 1</th></tr><tr><td>Nombre</td><td>Nombre del servicio candidato.</td></tr><tr><td>Proveedor</td><td>Nombre del proveedor.</td></tr><tr><td>Propósito</td><td>Descripción del servicio candidato</td></tr><tr><td>Dirección</td><td>Dirección URL del acceso al servicio candidato.</td></tr><tr><td>Operación</td><td>Qué operación ofrecida por el servicio candidato va a ser invocada.</td></tr><tr><td>Parámetros de entrada</td><td>Descripción detallada de los parámetros de entrada de la operación a invocar.</td></tr></table></td></tr>
</table>

	Parámetros de salida — Descripción detallada de los parámetros de salida de la operación a invocar. **Comentarios** — Comentarios o clarificaciones sobre el servicio candidato. **Servicio Concreto Candidato 2**
Nombre Servicio Abstracto2	Por cada servicio concreto candidato: **Servicio Concreto Candidato 1**
…	

Descripción de los puntos de variación

En esta sección se describen todos los puntos de variación identificados en la composición y por cada uno de ellos, se definen las acciones asociadas. La lista de los puntos de variación identificados en la composición es la siguiente:

Descripción de los puntos de variación	
Nombre Punto de Variación 1	Por cada variante asociada se define la lista de acciones a realizar: **Servicio Concreto** **Nombre** — Nombre del servicio candidato. **Proveedor** — Nombre del proveedor. **Propósito** — Descripción del servicio candidato **Dirección** — Dirección URL del acceso al servicio candidato.

<table>
<tr><td></td><td>
<table>
<tr><td>Operación</td><td>Qué operación ofrecida por el servicio candidato va a ser invocada.</td></tr>
<tr><td>Parámetros de entrada</td><td>Descripción detallada de los parámetros de entrada de la operación a invocar.</td></tr>
<tr><td>Parámetros de salida</td><td>Descripción detallada de los parámetros de salida de la operación a invocar.</td></tr>
<tr><td>Comentarios</td><td>Comentarios o clarificaciones sobre el servicio candidato.</td></tr>
</table>
o, en el caso de que sea una petición de búsqueda:
<table>
<tr><th colspan="2">Petición de Búsqueda</th></tr>
<tr><td>Descripción</td><td>Descripción de la petición de búsqueda.</td></tr>
<tr><td>Petición</td><td>Especificación no formal de petición de búsqueda acorde a un conjunto de requisitos.</td></tr>
<tr><td>Comentarios</td><td>Comentarios o clarificaciones sobre la petición.</td></tr>
</table>
</td></tr>
<tr><td>Nombre Punto de Variación2</td><td>Por cada variante asociada se define la lista de acciones a realizar:

…</td></tr>
<tr><td>…</td><td></td></tr>
</table>

Nivel de calidad de servicio de la composición

En esta sección se especifica cuál es el nivel final de calidad de servicio que garantiza la composición de servicios.

ANEXO 2

Descripción de la Herramienta de Soporte a la Metodología Propuesta

Este anexo describe el funcionamiento y el interfaz de la herramienta de soporte a la metodología de composición que hemos desarrollado.

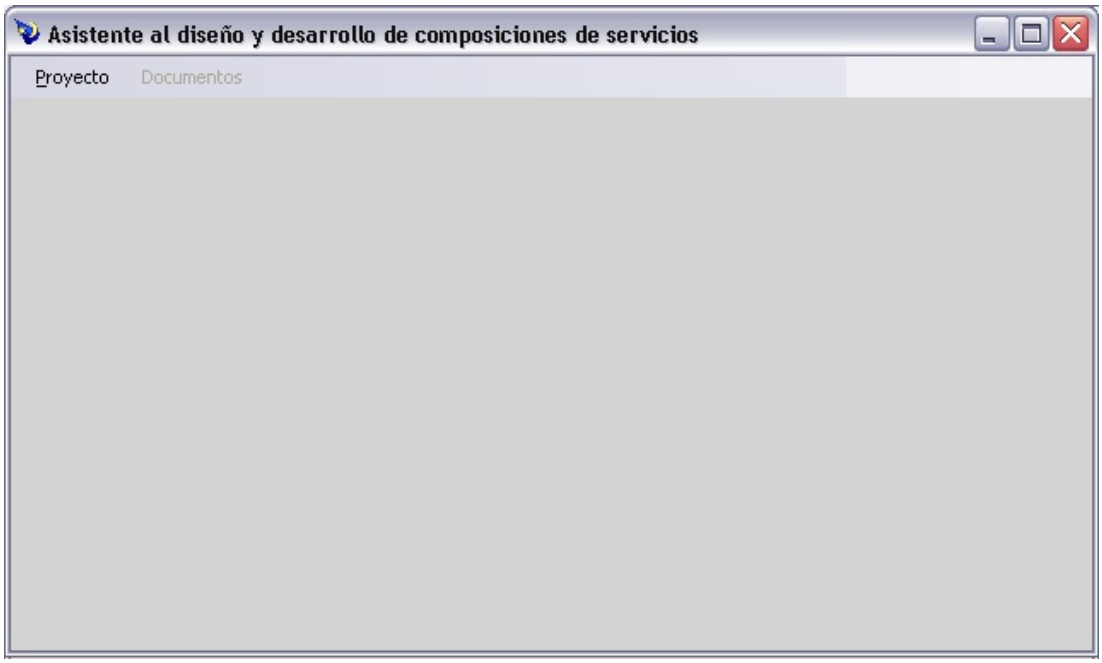


Figura A2-1: Pantalla principal de la herramienta de soporte

La Figura A2-1 muestra la pantalla principal de esta herramienta, donde se muestra un menú, dividido a su vez en dos sub-menús, con las principales acciones proporcionadas por dicha herramienta. El sub-menú denominado '*Proyecto*' Figura A2-2, a través de la acción '*Nuevo*', permite crear un proyecto de composición de servicios siguiendo la metodología propuesta. Cada proyecto tiene asociado una carpeta con el mismo nombre donde se almacena toda la información generada. Si se desea cargar un proyecto de composición previamente creado se puede hacer a través de la acción '*Abrir*', permitiendo continuar el diseño de la composición a partir del punto donde se guardó. Una vez que el usuario ha terminado de ejecutar todos los pasos, a través de la acción '*Exportar*' se podrá exportar un documento XML con toda la información sobre la composición de servicios definida, facilitando la interacción con otras aplicaciones.



Figura A2-2: Sub-menú 'Proyecto' de la herramienta

El sub-menú '*Documentos*' Figura A2-3 permite modificar la información relativa al documento de requisitos y además, permite consultar los diferentes productos de trabajo que se han ido generando durante el diseño de la composición de servicios sobre la que se está trabajando. La herramienta se encarga de guardar de manera automática la información proporcionada en alguna de las tareas después de pulsar el botón '*Siguiente*'.

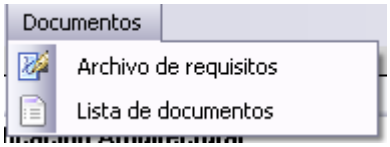


Figura A2-3: Sub-menú 'Documentos' de la herramienta

Una vez que se ha creado o abierto un proyecto de composición, la pantalla de la herramienta se divide en dos partes principales:

- La parte izquierda del interfaz muestra los diferentes procesos y sus correspondientes tareas. Para indicar el estado y el tipo de una tarea se utilizan una serie de símbolos descritos en la siguiente tabla:

SÍMBOLO	SIGNIFICADO
	Tarea sin realizar.
	Tarea realizada.
	Siguiente tarea a realizar.
	Tarea opcional.
	Tarea opcional que no hay sido realizada.

Tabla A2-1: Simbología utilizada en la herramienta

- La parte derecha del interfaz muestra la información asociada a la tarea seleccionada en ese momento. Por un lado se muestra una descripción de la tarea así como los productos de trabajo necesarios para realizar la tarea y los productos de trabajo generados Figura A2-4. También se puede acceder a esta información pulsando F1.

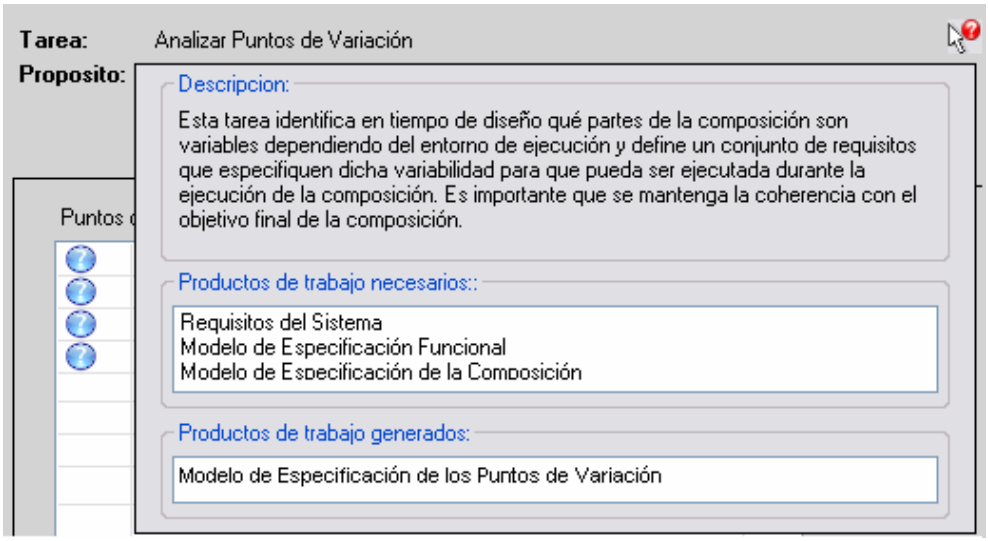


Figura A2-4: Información de la tarea 'Analizar Puntos de Variación'

Por otro lado, también se muestra qué información es requerida para completar dicha tarea. Esta información dependerá de la tarea que se esté realizando, pudiendo ser, por ejemplo, la inclusión del documento que describe los requisitos arquitectónicos Figura A2-5 o bien la descripción de los puntos de variación identificados Figura A2-6.

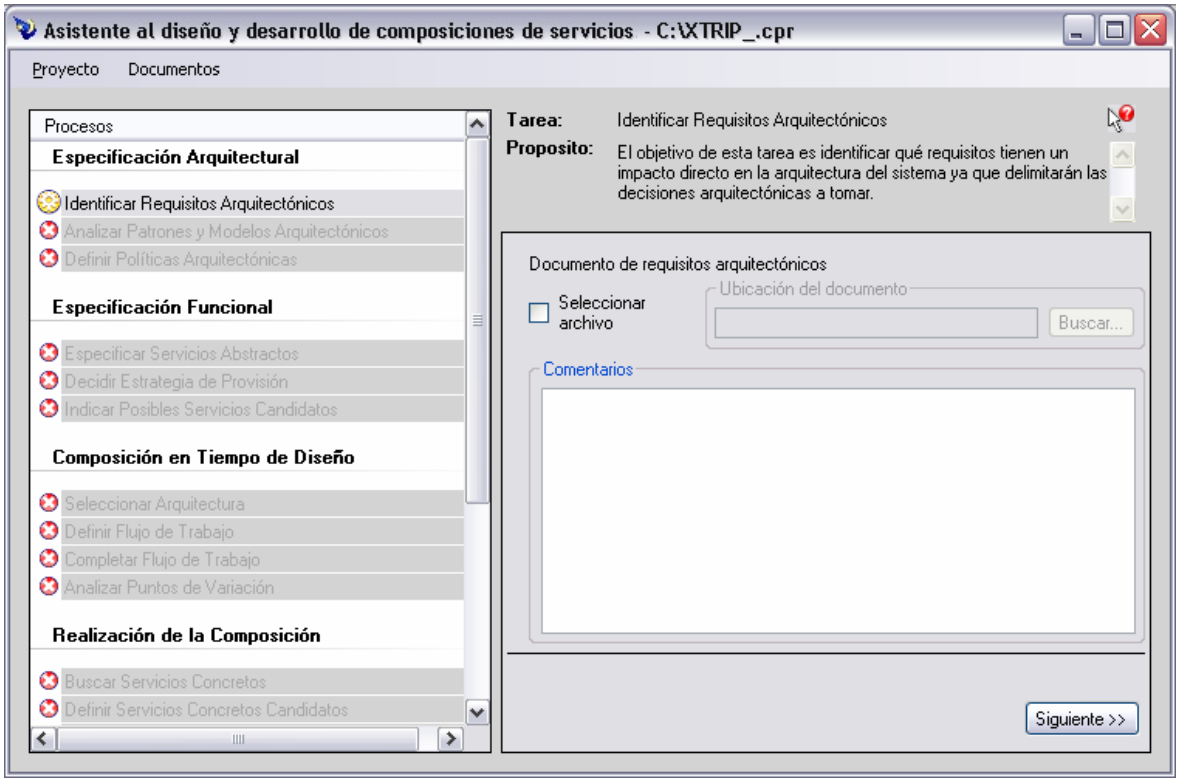


Figura A2-5: Pantalla de la tarea 'Identificar Requisitos Arquitectónicos'

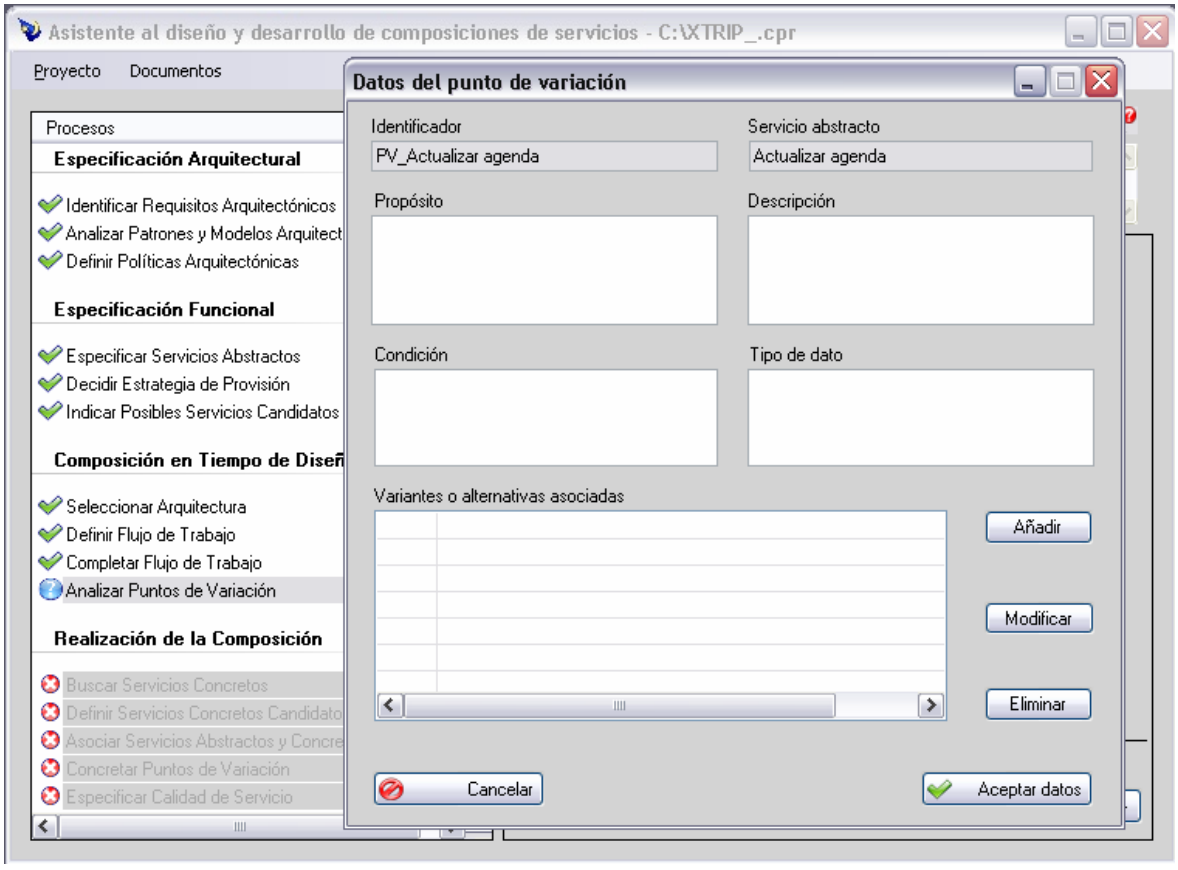


Figura A2-6: Pantalla de la tarea 'Analizar Puntos de Variación'

Si se modifica una tarea previamente realizada, es conveniente volver a realizar las tareas siguientes para verificar que la información proporcionada es correcta o para añadir nuevos elementos relacionados con el cambio.

ANEXO 3

Modelos Ejecutables del Servicio Compuesto XTrip

Este anexo recoge los dos modelos ejecutables asociados al ejemplo del servicio compuesto XTrip representado en el capítulo 6 sobre Validación y Evaluación de la metodología propuesta. Es importante indicar que toda composición de servicios deberá tener asociado, además de los modelos ejecutables, un fichero WSDL que defina el interfaz del servicio compuesto y cómo poder acceder a él.

A3.1 Modelo Ejecutable de la Composición

El siguiente código muestra el modelo ejecutable de la composición del servicio XTrip en formato XML y siguiendo la notación del lenguaje WS-BPEL versión 1.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="XTrip" suppressJoinFailure="yes"
   targetNamespace="uri:xtripasynch.secse.eng.it"
   xmlns:tns="uri:xtripasynch.secse.eng.it"
   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
   xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
   xmlns:xnav="uri:xnavigation.secse.eng.it" xmlns:xag="uri:xagenda.secse.eng.it"
   xmlns:txnav="uri:types.xnavigation.secse.eng.it"
   xmlns:txag="uri:types.xagenda.secse.eng.it"
   xmlns:txtrip="uri:types.xtrip.secse.eng.it"
   xmlns:tpc_int="http://www.csapi.org/wsdl/parlayx/third_party_call/v2_0/
```

```
     interface"
   xmlns:ttpc_local="http://www.csapi.org/schema/parlayx/third_party_call/v2_0/
     local"
   xmlns:ttpc_common="http://www.csapi.org/schema/parlayx/common/v2_0"
   xmlns:tpc="http://www.csapi.org/wsdl/parlayx/third_party_call/v2_0/service"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!--=================================================================-->
  <!--PARTNERLINKS                                                    -->
  <!--List of services participating in this BPEL process            -->
  <!--=================================================================-->
  <partnerLinks>
    <!--The 'client' role represents the requester of this service.-->
    <partnerLink name="client" partnerLinkType="tns:XTRIP" myRole="XTRIPProvider"
        partnerRole="XTRIPRequestor"/>
    <partnerLink name="XNavigation" partnerLinkType="xnav:XNavigation"
        partnerRole="XNavProvider"/>
    <partnerLink name="XAgenda" partnerLinkType="xag:XAgenda"
        partnerRole="XAgProvider"/>
    <partnerLink name="ParlayTPC" partnerLinkType="tpc_int:ParlayTPC"
        partnerRole="TPCProvider"/>
  </partnerLinks>

  <!--=================================================================-->
  <!--VARIABLES                                                       -->
  <!--List of messages and XML documents used within this BPEL process -->
  <!--=================================================================-->
  <variables>
    <!--Reference to the message passed as input during initiation-->
    <variable name="XTRIPinput" messageType="tns:XTRIPRequestMessage"/>
    <!--Reference to the message that will be returned to the requester-->
    <variable name="navigationInput" messageType="xnav:findDurationRequest"/>
    <variable name="tripDuration" messageType="xnav:findDurationResponse"/>
    <variable name="responseMessage" messageType="tns:XTRIPResponseMessage"/>
    <variable name="checkConflictInput" messageType="xag:checkConflictRequest"/>
    <variable name="conflictResponse" messageType="xag:checkConflictResponse"/>
    <variable name="confirmCommInput"messageType="xag:confirmCommitmentRequest"/>
    <variable name="confirmCommOutput" messageType=
        "xag:confirmCommitmentResponse"/>
    <variable name="tpcRequest" messageType=
         "tpc_int:ThirdPartyCall_makeCallRequest"/>
    <variable name="tpcResponse" messageType=
         "tpc_int:ThirdPartyCall_makeCallResponse"/>
  </variables>

  <!--=================================================================-->
  <!--ORCHESTRATION LOGIC                                             -->
  <!--Set of activities coordinating the flow of messages across the  -->
  <!--services integrated within this business process               -->
  <!--=================================================================-->
  <sequence name="main">
    <!--Receive input from requester.
        Note: This maps to operation defined in XTRIP.wsdl-->
    <receive name="receiveInput" partnerLink="client" portType="tns:xtripPT"
        operation="process" createInstance="yes" variable="XTRIPinput"/>
    <!--Generate reply to synchronous request-->
    <assign name="assignment_forCalling_XNavigation">
      <copy>
        <from variable="XTRIPinput" part="payloadRequest"
            query="/txtrip:XTRIPRequest/txtrip:destination"/>
        <to variable="navigationInput" part="Destination1"
            query="/txnav:Destination1"/>
      </copy>
    </assign>
    <invoke name="Calcular_Ruta" partnerLink="XNavigation"
```

```
        portType="xnav:xnavigationPT" operation="routeService"
        inputVariable="navigationInput" outputVariable="tripDuration"/>
    <assign name="assignment_toCheckConflict">
      <copy>
        <from variable="tripDuration" part="Duration" query="/txnav:Duration"/>
        <to variable="checkConflictInput" part="duration1"
             query="/txag:duration1"/>
        <from variable="XTRIPinput" part="payloadRequest"
             query="/txtrip:XTRIPRequest/txtrip:agenda"/>
        <to variable="navigationInput" part="Agenda" query="/txnav:Agenda"/>
      </copy>
    </assign>
    <invoke name="Comprobar_Agenda" inputVariable="checkConflictInput"
        outputVariable="conflictResponse" partnerLink="XAgenda"
        portType="xag:xagendaPT" operation="checkConflict"/>
    <switch name="isThereConflict">
      <case condition="bpws:getVariableData('conflictResponse',
          'response','/txag:response') = 'true'">
        <sequence>
          <assign name="assignTelephoneNumber">
            <copy>
              <from>
                <ttpc_local:makeCall>
                  <ttpc_local:callingParty/>
                  <ttpc_local:calledParty/>
                </ttpc_local:makeCall>
              </from>
              <to variable="tpcRequest" part="parameters"
                   query="/ttpc_local:makeCall"/>
            </copy>
            <copy>
              <from expression="concat('tel:', bpws:getVariableData(
                   'XTRIPinput','payloadRequest','/txtrip:XTRIPRequest
                   /txtrip:callingParty'))"/>
              <to variable="tpcRequest" part="parameters"
                   query="/ttpc_local:makeCall/ttpc_local:callingParty"/>
            </copy>
            <copy>
              <from expression="concat('tel:',bpws:getVariableData(
                   'XTRIPinput','payloadRequest','/txtrip:XTRIPRequest
                    /txtrip:calledParty'))"/>
              <to variable="tpcRequest" part="parameters"
                    query="/ttpc_local:makeCall/ttpc_local:calledParty"/>
            </copy>
          </assign>
          <scope name="Scope_thirdPartyCall_interactions">
            <sequence name="sequence_MakeCall_getCallInformation">
              <invoke name="Realizar_Llamada" inputVariable="tpcRequest"
                  partnerLink="ParlayTPC" operation="makeCall"
                  outputVariable="tpcResponse" portType="tpc_int:
                  ThirdPartyCall"/>
            </sequence>
          </scope>
    <invoke name="asynchResponse" partnerLink="client" operation=
       "sendProcessResult" portType="tns:xtripPT_CallBack" inputVariable=
       "responseMessage"/>
  </sequence>
</process>
```

A3.2 Modelo Ejecutable de la Variabilidad

El siguiente código muestra el modelo ejecutable de la variabilidad del servicio compuesto XTrip en formato XML y siguiendo la notación de las reglas ECA. Con el fin de simplificar la lectura de este modelo se han simplificado aquellas partes que consistían en importar clases, enfatizando la descripción de las diferentes reglas que definen los puntos de variación y las alternativas identificadas durante el diseño y desarrollo de la composición de servicios.

```
<?xml version="1.0"?>
<rule-set name="XTRIP_Rule" xmlns=http://drools.org/rules
          xmlns:java="http://drools.org/semantics/java"
          xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
          xs:schemaLocation="http://drools.org/rules rules.xsd
          http://drools.org/semantics/java java.xsd">

<!-- Import the Java Objects that we refer to in our rules -->
<java:import>java.lang.String</java:import>

<!-- violation codes -->
<!-- invoke input variable: types-->
<java:import>
       org.csapi.www.schema.parlayx.xagenda.v2_0.local.CheckAgendaDocument
</java:import>
<java:import>
       org.csapi.www.schema.parlayx.third_party_call.v2_0.local.MakeCallDocument
</java:import>

<!—import event payloads -->

<!-- Utility class to create action requests  -->

<!-- Utility class to reason on process info and binding preferences data -->

<!-- data of the data model -->

<!-- A Java (Utility) function we reference in our rules-->
<java:functions>

     public CheckAgendaDocument getCheckAgendaInput(ActivityBindingEventDocument
                               eventPayload) throws Exception {
       String xml = eventPayload.getActivityBindingEvent().getInput().xmlText();
       CheckAgendaDocument input = CheckAgendaDocument .Factory.parse(xml);
       System.out.println("Input: "     + input.xmlText());
       return input;
     }

     public MakeCallDocument getMakeCallInput(ActivityBindingEventDocument
                             eventPayload) throws Exception {
       String xml = eventPayload.getActivityBindingEvent().getInput().xmlText();
       MakeCallDocument input = MakeCallDocument.Factory.parse(xml);
       System.out.println("Input: " + input.xmlText());
       return input;
     }

</java:functions>
```

```
<!-- ****************Local rules Calcular_Ruta activity ******************* -->
<rule name="RULE1_Calcular_Ruta: used bv default service - not in blacklist">
       <parameter identifier="activityBindingEvent">
          <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
              .secse.ActivityBindingEventDocument</class>
       </parameter>
       <java:condition>
          activityBindingEvent.getActivityBindingEvent().getActivityName().equals
              ("Calcular_Ruta")
       </java:condition>
       <java:condition>
         !ProcessInfo.isEndpointInBlackList("http://localhost:8080/axis/services/
              mappoint", bindingpreferences)
       </java:condition>
       <java:consequence>
          System.out.println("creating binding");
          Binding b= new Binding();
          b.setActivityName(activityBindingEvent.getActivityBindingEvent().
               getActivityName());
          b.setEndpoint("http://localhost:8080/axis/services/mappoint");
          b.setServicespec("http://localhost:8080/serviceSpec_rep/signatureespec
              mappoint.xml");
          System.out.println("updating binding");
          ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
          System.out.println("updating proxy binding");
          ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);
          System.out.println("building activitybindingresultevent");
          ActivityBindingResultEventDocument actionRequest=ActionRequest.
          createActivityBindingResult(activityBindingEvent.getActivityBinding
          Event().getActivityName(),RuleEngineResultCodes.ABR_EVENT_RESULT_
              SERVICEBOUND);
          System.out.println("sendExternalEvent");
          externalEventList.addExternalEvent(actionRequest);
       </java:consequence>
</rule>

<rule name="RULE2_Calcular_Ruta: - functional POSTCONDITIONVIOLATION to
            Localbinding">
       <parameter identifier="postconditionViolationEvent">
          <class>v20.postconditionViolationEvent.events.datamodel.composition
              language.secse.PostconditionViolationEventDocument</class>
       </parameter>
       <java:condition>
          postconditionViolationEvent.getPostconditionViolationEvent().
              getActivityName().equals("Calcular_Ruta")
       </java:condition>
       <java:condition>
          postconditionViolationEvent.getPostconditionViolationEvent().
              getViolation().equals(ViolationCodes.POSTVE_EVENT_VIOLATION_
              GENERICVIOLATION)
       </java:condition>
       <java:consequence>
          System.out.println("set phase");
          ProcessInfo.setPhase(ProxyPhaseCodes.POSTINVOCATION,postcondition
              ViolationEvent.getPostconditionViolationEvent().getActivityName(),
              processinfo);

          System.out.println("update blacklist");
          ProcessInfo.blacklistCurrentBinding(postconditionViolationEvent.
              getPostconditionViolationEvent().getActivityName(),processinfo,
              bindingpreferences);
          System.out.println("build localdiscoverybindingevent");
          LocalDiscoveryBindingEventDocument actionRequest=ActionRequest.
              createLocalDiscoveryBinding(postconditionViolationEvent.
              getPostconditionViolationEvent().getActivityName(),
```

```
            bindingpreferences, processinfo);
         System.out.println("sendExternalEvent");
         externalEventList.addExternalEvent(actionRequest);
      </java:consequence>
</rule>

<rule name="RULE3_Calcular_Ruta: local binding but mappoint is in blacklist">
      <parameter identifier="activityBindingEvent">
         <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
             .secse.ActivityBindingEventDocument</class>
      </parameter>
      <java:condition>
         activityBindingEvent.getActivityBindingEvent().getActivityName().
             equals("Calcular_Ruta")
      </java:condition>
      <java:condition>
         ProcessInfo.isEndpointInBlackList("http://localhost:8080/axis/services
             /mappoint", bindingpreferences)
      </java:condition>
      <java:consequence>
         System.out.println("creating binding");
         Binding b= new Binding();
         b.setActivityName(activityBindingEvent.getActivityBindingEvent()
            .getActivityName());
         b.setEndpoint("http://localhost:8080/axis/services/viamichelin");
         b.setServicespec("http://localhost:8080/serviceSpec_rep/signaturespec
            viamichelin.xml");

            System.out.println("updating binding");
            ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
            System.out.println("updating proxy binding");
            ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);

            System.out.println("building activitybindingresultevent");
            ActivityBindingResultEventDocument actionRequest=ActionRequest.
                createActivityBindingResult(activityBindingEvent.getActivity
                BindingEvent().getActivityName(),RuleEngineResultCodes.
                ABR_EVENT_RESULT_SERVICEBOUND);
            System.out.println("sendExternalEvent");
            externalEventList.addExternalEvent(actionRequest);
       </java:consequence>
</rule>
<!-- ****************END Local rules Calcular_Ruta activity ***************** -->

<!-- ****************Local rules Comprobar_Agenda activity ***************** -->
<rule name="Rule1_Comprobar_Agenda: Check for conflicts. Using Microsoft">
      <parameter identifier="activityBindingEvent">
         <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
             .secse.ActivityBindingEventDocument</class>
      </parameter>
      <java:condition>
         activityBindingEvent.getActivityBindingEvent().getActivityName().
             equals("checkConflict")
      </java:condition>
      <java:condition>
          getCheckAgendaInput(activityBindingEvent).getSystem().equals
             ("Microsoft")
      </java:condition>
      <java:consequence>
          System.out.println("creating binding");
         Binding b= new Binding();
         b.setActivityName(activityBindingEvent.getActivityBindingEvent().
             getActivityName());
         b.setEndpoint("http://localhost:8080/axis/services/XAgendaMicrosoft");
         b.setServicespec("http://localhost:8080/serviceSpec_rep/servicespec
```

```
                XAgendaMicrosoft.xml");

            System.out.println("updating binding");
            ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
            System.out.println("updating proxy binding");
            ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);
             System.out.println("building activitybindingresultevent");
            ActivityBindingResultEventDocument actionRequest=ActionRequest.
             createActivityBindingResult(activityBindingEvent.getActivityBinding
                 Event().getActivityName(),RuleEngineResultCodes.
                 ABR_EVENT_RESULT_SERVICEBOUND);

            System.out.println("sendExternalEvent");
            externalEventList.addExternalEvent(actionRequest);
       </java:consequence>
</rule>

<rule name="Rule2_Comprobar_Agenda: Check for conflicts. Using Google">
       <parameter identifier="activityBindingEvent">
          <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
              .secse.ActivityBindingEventDocument</class>
       </parameter>
       <java:condition>
          activityBindingEvent.getActivityBindingEvent().getActivityName().
              equals("checkConflict")
       </java:condition>
       <java:condition>
           getCheckAgendaInput(activityBindingEvent).getSystem().equals
               ("Google")
       </java:condition>
       <java:consequence>
           System.out.println("creating binding");
          Binding b= new Binding();
          b.setActivityName(activityBindingEvent.getActivityBindingEvent().
              getActivityName());
          b.setEndpoint("http://localhost:8080/axis/services/XAgendaGoogle");
          b.setServicespec("http://localhost:8080/serviceSpec_rep/servicespec
              XAgendaGoogle.xml");

          System.out.println("updating binding");
          ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
          System.out.println("updating proxy binding");
          ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);
           System.out.println("building activitybindingresultevent");
          ActivityBindingResultEventDocument actionRequest=ActionRequest.
           createActivityBindingResult(activityBindingEvent.getActivityBinding
              Event().getActivityName(),RuleEngineResultCodes.
              ABR_EVENT_RESULT_SERVICEBOUND);

          System.out.println("sendExternalEvent");
          externalEventList.addExternalEvent(actionRequest);
       </java:consequence>
</rule>
<!-- ***************END Local rules Comprobar_Agenda activity *************** -->

<!-- ****************Local rules Actualizar_Agenda activity ***************** -->
<rule name="Rule1_Actualizar_Agenda: Strong dependency with Comprobar_Agenda">
       <parameter identifier="activityBindingEvent">
          <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
              .secse.ActivityBindingEventDocument</class>
       </parameter>
       <java:condition>
          activityBindingEvent.getActivityBindingEvent().getActivityName().
             equals("Actualizar_Agenda")
       </java:condition>
```

```
      <java:consequence>
         System.out.println("resolve dependencies");
         String depactname=ProcessInfo.resolveStrongDependency(
             activityBindingEventgetActivityBindingEvent().getActivityName(),
             processinfo);

         System.out.println("get binding of related activity");
         Binding b=ProcessInfo.getBindingOfActivity(depactname,processinfo);
         b.setActivityName(activityBindingEvent.getActivityBindingEvent().
             getActivityName());
         System.out.println("update binding");
         ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
         System.out.println("updating proxy binding");
         ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);

         System.out.println("build activitybindingresultevent");
         ActivityBindingResultEventDocument actionRequest=ActionRequest.
         createActivityBindingResult(activityBindingEvent.getActivityBinding
             Event().getActivityName(),RuleEngineResultCodes.
             ABR_EVENT_RESULT_SERVICEBOUND);

         System.out.println("sendExternalEvent");
         externalEventList.addExternalEvent(actionRequest);
      </java:consequence>
</rule>
<!-- ***************END Local rules Actualizar_Agenda activity ************** -->

<!-- *****************Local rules Realizar_Llamada activity ***************** -->
<rule name="Rule1_Realizar_Llamada: Binding request to kRecall">
      <parameter identifier="activityBindingEvent">
         <class>v20.activityBindingEvent.events.datamodel.compositionlanguage
             .secse.ActivityBindingEventDocument</class>
      </parameter>
      <java:condition>
         activityBindingEvent.getActivityBindingEvent().getActivityName()
             .equals("Realizar_Llamada ")
      </java:condition>
      <java:condition>
         getMakeCallInput(activityBindingEvent).getMakeCall().getCalled
             Party().substring(1,3).equals("39")
      </java:condition>
      <java:consequence>
         System.out.println("creating binding");
         Binding b= new Binding();
         b.setActivityName(activityBindingEvent.getActivityBindingEvent().
             getActivityName());
         b.setEndpoint("http://163.162.173.70/WsGateway/services/ppxSipTpc");
         b.setServicespec("("http://localhost:8080/serviceSpec_rep/servicespec
             ThirdPartyCallTLAB.xml ");

         System.out.println("updating binding");
         ProcessInfo.updateBindingInfo_forActivity(b,processinfo);
         System.out.println("updating proxy binding");
         ProcessInfo.updateProxyBinding(pid,idsla,b,processinfo);
         System.out.println("building activitybindingresultevent");
         ActivityBindingResultEventDocument actionRequest=ActionRequest.
         createActivityBindingResult(activityBindingEvent.getActivityBinding
             Event().getActivityName(),RuleEngineResultCodes.
             ABR_EVENT_RESULT_SERVICEBOUND);

         System.out.println("sendExternalEvent");
         externalEventList.addExternalEvent(actionRequest);
      </java:consequence>
</rule>
```

```
<!-- rule 2 about dynamically discover concrere services is embedded into the
rule engine component only to test RSD invocation -->

<!-- **************END Local rules Realizar_Llamada activity *************** -->
```